14-12-2018

Deliverable D8.8 Integrated Services Framework and Network Services Development Roadmap – Follow-Up

Deliverable D8.8

Contractual Date:	30-06-2018
Actual Date:	14-12-2018
Grant Agreement No.:	731122
Work Package/Activity:	8/JRA2
Task Item:	Tasks 1, 2, 3 and 4
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	NORDUnet
Document ID:	GN4-2-18-55F811
Authors:	Jerry Sobieski (NORDUnet), Ivana Golub (PSNC)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

Abstract

The Integrated Services Framework (ISF) proposes a set of generic, fundamental service classes that are needed in the emerging network ecosystem and defines a common service model that allows such services to invoke one another and to build incrementally more complex services. It incorporates work carried out in GN4-2 JRA2 Network Services Development Tasks 1–4. This deliverable outlines the principles underlying the ISF and the roadmaps for each of the Tasks involved.

Table of Contents

Executive Summary 1

1 Introduction 3

2 Network Services Evolution 4
- 2.1 Evolution Principles 4
  - 2.1.1 Composability 4
  - 2.1.2 Orchestration 5
  - 2.1.3 Abstraction and Virtualisation 5
- 2.2 Service Building Blocks 7
  - 2.2.1 Service Models 7
  - 2.2.2 Basic Infrastructure Service Objects 9
  - 2.2.3 Composite Services 11
  - 2.2.4 Management Operations 15
- 2.3 Defining Network Objects and Network Functions 16
- 2.4 Service Composition – Responsibility and Verification 17

3 Network Services Development Roadmap 20
- 3.1 Consolidated Connection Services (Task 1) 20
- 3.2 Service Provider Architecture (Task 2) 21
- 3.3 GÉANT Testbeds Service (Task 3) 23
- 3.4 Performance Verification and Monitoring (Task 4) 24

4 Related Work/Supporting Standards 26
- 4.1 Open Grid Forum 26
- 4.2 ETSI NFV 27
- 4.3 TM Forum 29

5 Conclusions 30

Appendix A Task 1 Roadmap for Consolidation of Point-to-Point and Point-to-Multi-point Services 31
- A.1 Work Plan 31
  - A.1.1 CCS v1.0 32
  - A.1.2 CCS v2.0 32
  - A.1.3 CCS v3.0 33
  - A.1.4 ELINE Project 34
  - A.1.5 OpenNSA 34

A.1.6 QoS-Aware Implementation 34
A.1.7 GÉANT Lambda and Multi-Point Services 35
Appendix B Task 2 Roadmap for Service Provider Architecture 36
Appendix C Task 3 Roadmap for GÉANT Testbeds Service (GTS) 39
Appendix D Task 4 Roadmap for Performance Verification and Monitoring (PVM) 42
References 44
Glossary 47

Table of Figures

Figure 2.1: Modular approach to service design and orchestration 9
Figure 2.2: Layered resource virtualisation model 10
Figure 2.3: GTS resource taxonomy – available now (dark green), in development (light green) and envisioned (white) 11
Figure 2.4: Service decomposition to atomic service objects: recursive “tree” segmentation of end-to-end EPL connection service 13
Figure 2.5: Composable services model 13
Figure 2.6: GVM / NSI virtual resource lifecycle 16
Figure 4.1: Common features of GTS/GVM object types and NSI CS v2 27
Figure 4.1: ETSI NFV reference architecture and JRA2 solutions: preliminary mapping 28
Figure B.1: SPA component view 38

Executive Summary

Networking services are evolving rapidly. Conventional monolithic services are being overtaken by requirements for tailored capabilities ranging from global application-specific networks to custom packet filtering applied to individual traffic flows. The ability to address and deliver flexible, bespoke services in a secure and scalable manner requires a new way of thinking about how service providers design and manage their services. Virtualisation technologies and agility in service delivery are essential. Defining basic network services or functions and assembling them into more complex and bespoke services is gaining support as the most promising approach. This approach is being explored by GÉANT in order to utilise advanced capabilities of the network infrastructure and, at the same time, deliver customisable services and control to the users.

An integrated services framework means that services are able to interoperate with one another: internally, within the service provider's own ecosystem of services, and externally, to provide these services to other users/customers. For the evolution of an integrated GÉANT services framework, several principles are key, including: virtualisation, service abstraction, agnosticism with regard to physical and connectivity technologies, object-oriented design and composition, automation wherever possible, and orchestration.

This Integrated Services Framework (ISF) has drawn on work from previous Future Internet research programmes, such as FIRE and GENI, and has leveraged experience gained in the design, development and deployment of the GÉANT Testbeds Service (especially in the area of virtualisation and orchestration). It also builds on innovative work from the R&E networking community, such as the Network Service Interface (NSI) standard, and on experience gained from implementing emerging standards such as the TM Forum ZOOM architecture [TMFZOOM1, TMFZOOM2] and advanced performance verification tools and techniques. The Framework impacts and informs the work of four GN4-2 Joint Research Activity 2 Network Services Development Tasks: T1 Consolidated Connection Services (CCS), T2 Service Provider Architecture (SPA), T3 GÉANT Testbeds Service (GTS), and T4 Performance Verification and Monitoring (PVM).

Many of the ISF concepts and capabilities described in this document have been incorporated into the GÉANT Testbeds Service (GTS) and Consolidated Connection Service (CCS). As GN4-2 draws to a close, they are available and demonstrable in GTS v6 and CCS v3, which are due to enter full GÉANT production service in January 2019. However, there are still ISF-informed features yet to be implemented in GTS (e.g. transparent multi-domain operation, active modification) or that still require substantial design and coding, or that require an associated infrastructure upgrade (e.g. policy engine, operational migration/grooming).

In addition to the TMF ZOOM architecture, standards referenced by this work include TMF Frameworx [TMFFx] in relation to OSS/BSS processes, TMF SID for the information framework

[TMFSID], the ETSI NFV architecture [ETSI NFV] regarding service modelling, orchestration and function chaining, and the IETF YANG data model for network topologies [IETF RFC 8345]. Many of these specifications have been evolving in parallel with GÉANT development efforts or even, given the protracted timeline of standards evolution, several years after GÉANT Tasks have already designed, developed and deployed advanced services that incorporate these concepts; in the latter case, the industry specifications tend to be congruent with GÉANT pre-standards development.

This document presents the principles underpinning a proposed GÉANT services' evolution as well as the roadmaps of JRA2 Network Services Development for Task 1 to 4. It follows up *Deliverable D8.1 Proposed Integrated Services Framework and Network Services Development Roadmap* [D8.1].

1 Introduction

The rate of change in networking requirements is accelerating. These changes are not just in the transmission technologies, but extend to the application layer, where network services must be much more flexible to support the emerging e-infrastructure requirements. Some of the key drivers include the explosion in cloud services, the Internet of Things, Big Data, and mobility requirements. Whereas R&E networks have traditionally been able to focus on high performance, this is no longer a bastion against such growing user requirements. The rapid expansion in the user base, increased security and privacy requirements, new areas of integrated networking (e.g. 5GPPP), coupled with constant budget pressure, are pushing network services to find a newer, more agile, scalable and responsive means of delivering network services. Even the scope of the term "network services" is expanding to include capabilities deployed within the network that have traditionally been considered end points or non-network or not part of the high-performance core. Incremental enhancements and upgrades with continuous adjustments to hardware and services to address this evolution have resulted in complexity in service delivery. Taking into account also the recent developments in virtualisation, an evolved architectural approach is needed for next-generation services.

The growing consensus is that future services will not comprise large, static and manually supported infrastructure, but will evolve to a "conceptual plane" where they are largely driven by software – where services can be conceived, designed, validated and deployed as individualised service instances using a so-called "glass cockpit". These services can be bespoke-designed to meet customer requirements using software-based orchestration tools, and deployed quickly by intelligent automated-provisioning agents. This architectural proposition requires providers, including GÉANT, to refactor their services, identifying basic functions and services, which can then be combined like building blocks to create more sophisticated services. In addition, this service evolution involves functions and services integration that allows common cross-function processes, reduces duplication, enables modularity (which helps manage complexity) and enhances reliability and supportability, all of which increase responsiveness and service speed.

This document is a proposal for GÉANT network services evolution, and follows up *Deliverable D8.1 Proposed Integrated Services Framework and Network Services Development Roadmap* [D8.1]. It presents the key concepts and principles of such an evolution in Section 2. Section 3 presents an overview roadmap of the associated developments undertaken in JRA2 Tasks 1–4. Section 4 provides insight into relevant best practices and standards. Finally, Section 5 consolidates key points. Further details of the roadmaps for Tasks 1 to 4 are included as Appendices.

2 Network Services Evolution

As indicated in the Introduction, network services are evolving. Service providers must offer more than just fixed transport or a single infrastructure that supports a single, fixed, forwarding protocol. Network service providers still perform the traditional mainstay function of basic transmission of data between points, but they must also offer users the option to customise their network service environments, to define networks that most effectively address their specific objectives. Flexible design and deployment of bespoke services to meet specific customer needs, and doing so in a dependable, deterministic manner, are increasingly a user requirement. Of particular interest to network service providers is a reliable means of continuously introducing and supporting these innovations within their infrastructures such that they can function safely and properly without interfering with each other or compromising existing production services. “Composable” services – services comprising multiple component virtualised functional and/or infrastructure resources assembled into more complex and bespoke services – offer a compelling approach to meeting these requirements. Together with the related principles of virtualisation, service function chaining and orchestration, composable services are gaining traction within the advanced networking community as a promising means to deliver network evolution.

The notion of disaggregated services – the separation and disassembling of conventional networking products or infrastructure into commodity hardware and independently configurable software components – is already an emerging reality. Services are rapidly trending towards software-driven dependencies – i.e. service providers will require more software developers and system administrators than traditional network engineers in order to maintain and enhance the services they provide.

2.1 Evolution Principles

2.1.1 Composability

In the new environment, services are treated as objects. Large-scale conventional services are iteratively decomposed into ever more basic subservices. Repeating this process across the existing services portfolio, and combining the resulting service objects with new services requirements anticipated to emerge in the near future, will distil a set of basic services objects or functions that can then be manipulated like building blocks to construct – or to “compose” – new custom services. The resulting composability of new services is very powerful. Templates can be defined that allow the specification of hierarchically more complex or sophisticated or bespoke services. These template service definitions can be stored and reused as necessary to replicate services for different customers with minimal errors, or shared among collaborating service providers (which is especially

important to the R&E community.) The resulting composable services can be modelled accurately and validated, deployed efficiently across the physical infrastructure by automated intelligent mapping agents. Modularisation of services manages complexity and improves supportability, a particular requirement with regard to increasingly software-driven services. Composition allows the creation of flexible new service offerings, and instantiation of new services is simplified. At the same time, decomposition modelling provides the framework to validate and analyse the performance of these services.

2.1.2 Orchestration

Orchestration is a key aspect of composable services architecture. Orchestration is the process by which these basic service objects can be combined together to deliver the comprehensive service required. This can occur within a single service provider to describe a particular customer's requirements on, say, an access port, or it can be a more global process where basic services from many otherwise independent service providers are combined to meet a requirement that can only be solved with a multi-domain service. In this document, orchestration is defined as the process of combining multiple services – potentially from multiple service providers – to create an end-to-end service that meets a specific user requirement. The concept of book-ahead scheduling of resources (the service objects) as included as a part of orchestration.

It should be explicitly noted that the Integrated Services Framework delivers service objects, networking "resources" that can perform certain tasks or functions. It is these resources that will actually perform the network data plane processing of traffic to meet the user's requirements. This entails a conceptual shift with regard to what the service provider provides, i.e. no longer the service, as such, but instances of these service functions – the objects or (virtualised) resources that, when combined and instantiated appropriately across the infrastructure, perform the service, the networking operations on the traffic, the user requires.

2.1.3 Abstraction and Virtualisation

The evolution of network services requires clear analysis and understanding of the principles of abstraction and virtualisation, i.e. disaggregating the conceptual service capability from the underlying physical infrastructure or implementation that may be used to realise the services. This design methodology is technology agnostic, i.e. it does not matter what physical or connectivity technologies (hardware or software) are used to implement and realise the service object; the resultant service instance is measured against what it was defined to do, not how that function is implemented, and so is not dependent upon specific products or technologies. This allows collaborating organisations to define common services that all can offer, without dictating to those organisations specifically how, or which products must be used, to realise these services. The watchword is: "Think globally, engineer locally." Provided the common service definitions are clear and their behaviour is implemented rigorously, it should not matter what particular product or technology a service provider elects to use or how they engineer their infrastructure. This allows interoperation without violating autonomy.

Virtualisation is the abstraction of a service object to define what that service does, or how it behaves, independently from the underlying physical implementation of that service capability. It

decouples the *service object* from the specific physical infrastructure, platform, or technology upon which it may be modelled, or upon which it is realised. By virtualising a physical object, a virtual instance can be mapped to any underlying physical context that can deliver the defined behavioural characteristics.

In the context of emerging network virtualisation, there are virtualised infrastructure analogues such as virtual machines, virtual circuits, virtual switches and routers, virtual storage, etc. Virtualisation also extends to functional capabilities that may traditionally have been integrated as features in particular hardware, such as encapsulation, encryption, policing/shaping, filtering, etc. (see also the ETSI NFV specification, as discussed in Section 4.2).

By delivering services in the form of virtualised service objects, the behaviour of those service objects can be defined independently of any particular physical platform or implementation technology. Service providers are then able to factor large traditionally monolithic and increasingly complex services into a set of common basic subservices, or "atomic" services, that can then be used as building blocks to compose redefined, improved or customised services, as described in Section 2.1.1. These common atomic services are simpler to develop and validate, and can be assembled, or composed, into more sophisticated services in a manner that extends reliability, availability and maintainability in a predictable and deterministic fashion to the composed service construct. Further, because these virtual service objects are decoupled from specific hardware, the services can be implemented using different technologies, or different vendors and different products.

Virtual objects can be grouped together and their data flow topology explicitly defined to create composable services or virtualised *composite* service objects. These composite objects can themselves be manipulated, or "orchestrated", to create more sophisticated and customised services.

A rigorous virtualisation model efficiently bounds a service object's behaviour, i.e. it defines and constrains the object's ability to interact with other objects, and it prescribes the behaviour – particularly in terms of performance – that is expected for that object. This both isolates the object, to prevent it from interfering with other service objects, and simultaneously insulates it from interferences from other objects. This isolation and insulation of individual service objects extends to composite objects, and thus to higher-level service environments. The predictable and deterministic extension of atomic service objects' properties and behaviour, such as insulation and isolation, to larger, complex composite objects of which they are components, is a core tenet of composable services. It allows large-scale composite objects to be modelled in a similar manner to the basic atomic object. Insulation and isolation do not prevent separate service instances or environments from communicating or interacting with one another, but do require that such communication or interaction be explicitly defined in both service objects.

Virtual service objects can, by definition, be hosted in any hardware context able to deliver the required functional attributes. Thus this virtualisation model can extend to objects not traditionally considered network components. For example, a radio telescope can be virtualised by defining a service object that behaves . . . like a radio telescope. That virtual radio telescope (VRT) will define its attributes and data flow ports in the same manner as other virtual service objects (e.g. a VM or virtual circuit) and can be orchestrated using the same lifecycle primitives and topology description semantics as other virtual network objects. It can thereby be dynamically composed to create a large-scale distributed real-time science facility.

The VRT instance could potentially be mapped to any participating physical facility, i.e. radio telescope or laboratory, as long as the requested attributes for the VRT instance can be met at those locations.

Another non-traditional example is the virtualisation of mobile cellular spectrum, capacity, other wireless access mechanisms or other components of the mobile network to allow new mobile service models or even software-defined radio formats to be instantiated and piloted.

These non-traditional virtualised services can be seamlessly integrated with more conventional virtualised services to realise very advanced and highly flexible new services capabilities and environments.

Virtualisation offers greater potential than simply more flexible network service construction. Its innate ability to define the objects that make up the service environment and to manipulate those objects using a common model and advanced “glass cockpit” orchestration enables the construction of highly sophisticated global service environments that are not currently possible with conventional network engineering and/or traditional services.

2.2 Service Building Blocks

For the design and engineering of evolved services, building blocks such as hierarchical service models, physical and logical resource elements, and control and management operations are required. These are discussed in the following sections.

2.2.1 Service Models

Services should be modelled in terms of what they deliver and/or how they behave, regardless of the technology that may be used to realise them or who is requesting the service. As described above, this is service “abstraction”.

Abstraction allows the service objects or functions to be clearly understood and bounded as to what they do – or what they do not do – before they are mapped to particular hardware, connectivity and other infrastructure technologies. It also enables the services to be technology agnostic in terms of how it is implemented by different providers, which allows different network service providers to implement them with different vendors, or different hardware, connectivity or other infrastructure technologies.

The services offered by a provider employing this framework deliver their service functionality by constructing a service object – sometimes highly bespoke, analogous to an elaborate steam-punk mechanical device – that performs the desired function. For instance, in this future framework, GÉANT builds a service object called an “IP service” for a user to move IP packets from one location to another. In this context, GÉANT produces virtual objects, or resources, that are instantiated and interconnected in the GÉANT infrastructure and placed in service for the user. These objects may be simple, common virtual service objects, such as a virtual circuit or a virtual switch, or they may be

more sophisticated service objects that are composed of many other service objects interconnected to deliver the desired function. An example of the latter might be a multi-point L3 VPN.

It is therefore possible to describe (future) services in terms of this object-based model, and to conceptualise "services" as functions performed by "objects" that the provider allocates to, or on behalf of, a user. The term "user" is relative to the service interface – i.e. the user is the agent requesting the creation of the service, not the entity sending packets to the resulting data plane port. The output of the service is the resulting resource (the virtualised service object, which may be atomic or composite). This abstracted user concept is important because the integrated services are intended to be used to deliver services to conventional external users (customers) as well as to other service agents that may be internal to the service provider and that require these service objects to construct their customer-facing service environment. For instance, an IP service would acquire and place virtualised SDN switches from a virtual switch service, a VM to serve as a controller from a VM service, and data plane and control plane virtual circuits from a VC service. This composite IP service object would then be realised and exist as a high-performance network and IP customers could connect to its ports as normal. The notion of services that service other services is a key feature of the Integrated Services Framework.

Object-based models for virtualised services are still evolving across industry and research communities. JRA2 Task 3 GÉANT Testbeds Service has distilled a Generalised Virtualisation Model (GVM) that covers the virtual objects offered in GTS, their lifecycle and their interaction with each other. GTS "testbeds" (or, more generally, "virtual network environments") comprise many different virtualised objects. These testbeds are actually composite objects composed of the constituent virtual circuits, virtual switches, virtual machines and servers, etc. While the GVM implementation could be one of the most advanced realisations (i.e. with working code) for wide area networking to date, it is certainly not the only possible approach. GVM continues to evolve and still requires community input and thinking. In recent years, proof-of-concept implementations of virtual network functions (VNFs) within GTS using the same orchestration model as the other GTS composites were demonstrated to show NFV capabilities. More broadly, industry specifications continue to mature and warrant further investigation, as NFV and GVM appear to be converging. The Integrated Services Framework should take into consideration approaches used in different industry standards and initiatives, for example, modelling network functions in line with ETSI VNFs.

It should be noted here that GTS and GVM are not currently formally compliant to any of the emerging standards; these standards are as new as GTS, and in many cases even newer. There is very little adoption and deployment as yet in commercial environments and essentially no generalised network virtualisation service in production R&E environments. In this respect, GTS and the virtualisation capabilities it offers as running code today are an invaluable learning opportunity and resource for the R&E community and potentially the industry specification groups (ISGs) as well. GTS is not simply a proof of concept, but is a service that has been in pilot for over four years and will be a full-scale production GÉANT service beginning in 2018. The evolving standards and the running code GTS represents are similar, therefore GTS can provide much in terms of operational experience, engineering experience, service management, business planning, and capabilities/capacity planning that will apply to any integrated services implementation. GTS may or may not be the long-term software base, but for the near term it has no peer. In addition, because it has one foot in the network research community, it is drawing on that experience and insight to develop a platform that network researchers can use while simultaneously developing the platform

that can deliver future services. It has the other foot in the production world, where it must be stable and reliable while informing and tracking standards for both the architecture itself and for operational readiness.

2.2.2 Basic Infrastructure Service Objects

This section identifies basic service objects/functions that are "hardware analogues", i.e. that reflect common conventional infrastructure components. Basic *infrastructure* service objects can fall into one of the following categories:

- **Transport service objects**, which move information, unmodified, from one geographical location to another.
- **Switching and forwarding objects**, which sort information among transport service objects. Ingress information is inspected and processing performed to decide how to direct the information on egress. (These could also be considered a subset of computational objects.)
- **Computational service objects**, which are general-purpose compute objects that translate information according to software function.
- **Storage objects**, which hold information, unmodified, for later retrieval.
- **I/O service objects**, which introduce/extract information into/from the system. For example, a scientific instrument or sensor, or a display.
- **Composite service objects**, which provide the ability to group the above objects into larger objects.

These basic objects (components) can form the base for other, more specific functional objects. For example, a "rate limiter" service object could be implemented using a more general "container" computational object instantiated with software that effects the more specific "rate limiter" service object behaviour.

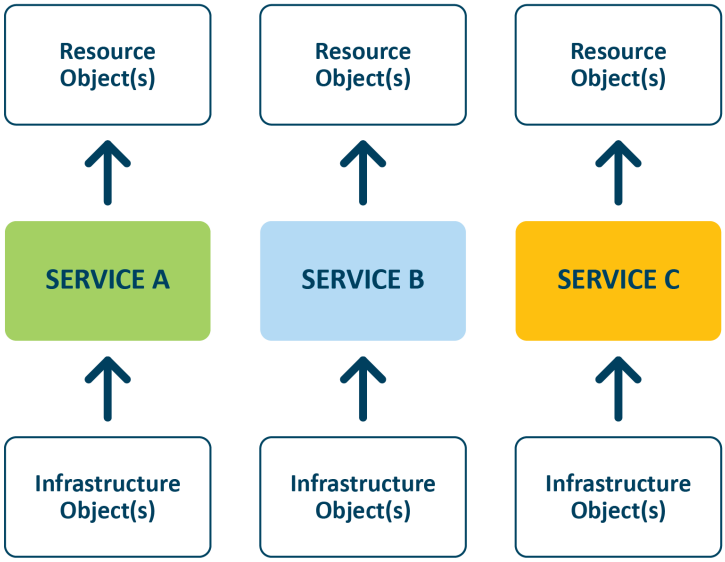

Figure 2.1: Modular approach to service design and orchestration

Several independent services are depicted in Figure 2.1 as a generic software process that takes as input a set of “infrastructure objects”, and produces a user-facing output service object: a “resource object”. This output resource is the object placed in service for the user. These input objects may be real physical infrastructure devices in an inventory pool managed by the service, or may be objects produced by other services. The “user” in this case is not assumed to be a GÉANT customer, but may be any (authorised) requesting agent, even another service. In Figure 2.1, and in a generalised virtual services model, the terms “infrastructure” and “resource” refer to the inputs and outputs, respectively, relative to a particular service, and should not be construed to mean physical infrastructure or the ultimate resource delivered to the end user. This allows virtualised services to be discussed generically, irrespective of where a particular function might reside in a stack or service chain, and without assumption regarding the nature of the input infrastructure object or the output resource object.

This dynamism is particularly relevant to virtualised services where a particular service (e.g. a VM service) may rely on other service(s) (e.g. a bare metal server (BMS) service) to provide the input “infrastructure” they depend upon, presented in Figure 2.2. The underlying BMS infrastructure service is generalised, and can therefore serve many other clients beyond the VM service. The generalised virtualisation model enables this dynamic interplay among services and allows the capital investment of base-level services to be applied more efficiently across a wider range of use cases; it also reduces duplication of support functions required for the other services. In addition, it introduces the ability of services to acquire/release infrastructure, as needed, and thus to be more agile and efficient responding to capacity fluctuations.

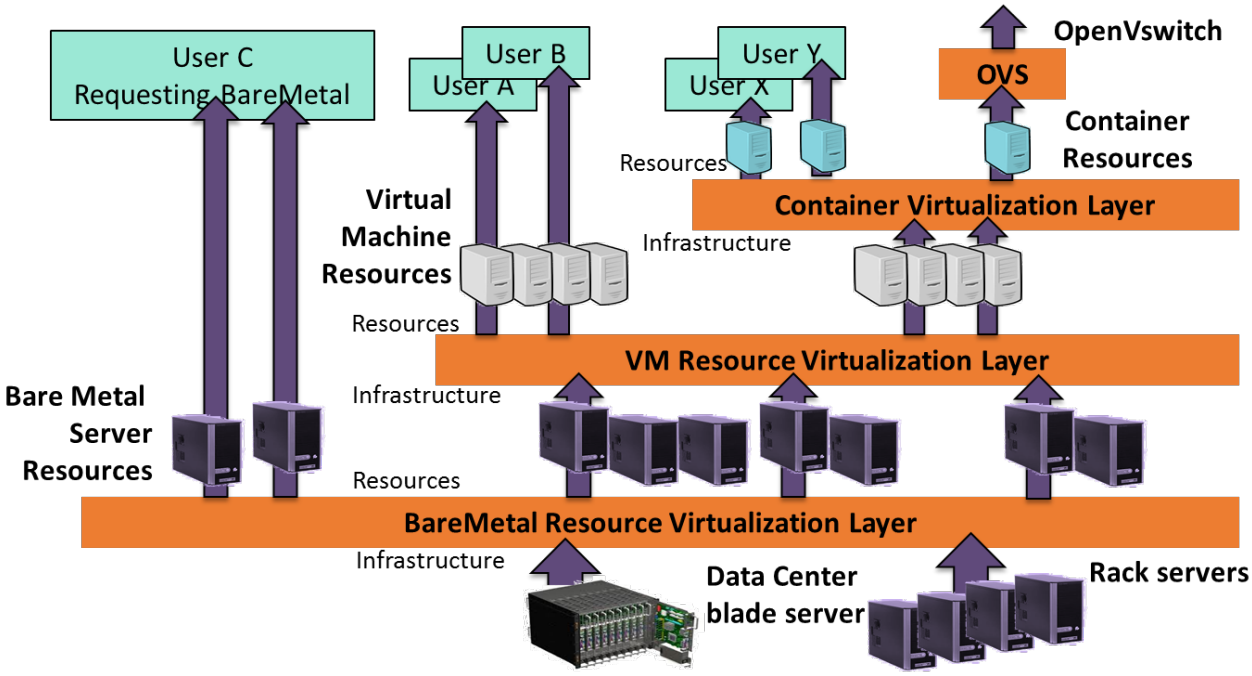

Figure 2.2: Layered resource virtualisation model

Figure 2.2 depicts how services use other services to dynamically acquire/release the (relative) infrastructure components required to produce their respective output service object. It also shows how such service factoring can define services that more efficiently cover many use cases.

Virtualised services offer great potential for responsive dynamic/elastic interoperation and novel service concepts for both users and network service providers. Complexity, however, must be managed in a rigorous, scalable, and deterministic manner.

The model expressed here, where abstracted service objects are defined in a well-bounded manner, are composable within specific rules and are individually verifiable, allows complexity to be effectively managed. It also enables services to be provided, performance to be intelligently analysed, and faulty components to be identified using deterministic, automated processes that extend from base-level atomic services to very complex service composites.

Figure 2.3 shows the specific infrastructure service objects that have been defined for GTS, and indicates whether these resources are available in the current implementation, in development or envisioned for possible development in the future.

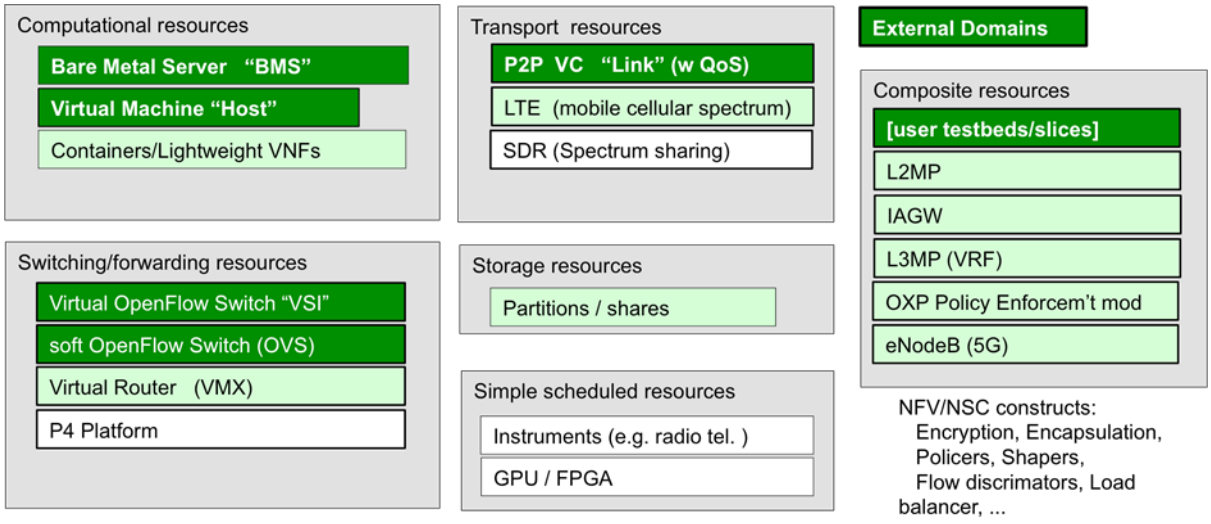

Figure 2.3: GTS resource taxonomy – available now (dark green), in development (light green) and envisioned (white)

2.2.3 Composite Services

As stated in Sections 2.1.3 and 2.2.1, composite services are service objects made up of other virtual service objects, assembled to address specific user needs. An implicit requirement of composite services is the ability to define the data plane relationship among the constituent service objects. The data plane arrangement of the constituent objects, combined with the functional behaviour of those objects, produces the behaviour of the composite service object itself. A "normalisation layer" is necessary to generalise the service interfaces to allow them to be assembled using common lifecycle primitives.

This normalisation layer provides a single common interface among all of these virtual objects and the object-specific services that create and manage them. In terms of GVM, the normalisation layer defines a common model of a virtual object, and the basic lifecycle such objects go through. For instance, when a virtual circuit is needed by a user, the user issues a Reserve() request to the VC service which will then find, allocate and reserve the infrastructure facilities needed to meet the request. Similarly, a Reserve() request with a different resource type – say, an SDN switch – can be sent to the white box service. For both, the user can then issue requests to Activate(), Query(), Deactivate() or Release(). While these lifecycle primitives are common to all the services, underneath each service the primitives interoperate with other underlying software or other technology. For instance, Activate() for the VM service might be translated into an OpenStack API to spin up a VM on a server in Paris, or another service provider might have to use VMWare underneath. Likewise, the VC service might use OpenNSA underneath, and the Activate() primitive would wrapper an NSI Provision() primitive. Thus these GVM lifecycle primitives apply to all GVM virtual objects and the primitives, along with the respective service's underlying southbound interoperating code, make up the "normalisation layer".

The ability to functionally and recursively decompose high-level service requirements into a set of subservices whose service objects can be re-assembled to produce the desired high-level functionality is the basis for composable services. The resulting service object, composed of these constituent subservice objects, is a composite service object.

Within the existing JRA2 services, the GÉANT Testbeds Service (GTS) has implemented a basic composable service architecture that enables GTS to construct extensive composite structures from basic virtualised infrastructure objects. These composite service objects are, in fact, the "testbeds" created by GTS.

An example of this modular approach to the creation of more complex services is depicted in Figure 2.4. One or more virtualised service objects can be combined, using composability rules, to assemble the multi-domain resource object – an Ethernet-framed connection object – which is provisioned and placed in service for the user.

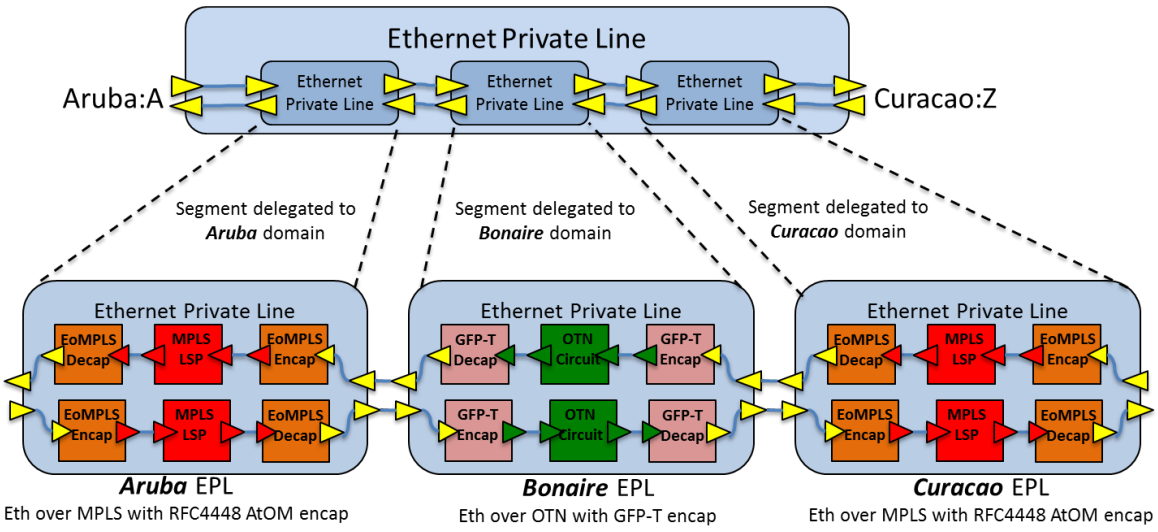

Figure 2.4: Service decomposition to atomic service objects: recursive “tree” segmentation of end-to-end EPL connection service

It should also be noted that a rigorous and deterministic composability model is essential to address the Fault, Configuration, Accounting, Performance and Security (FCAPS) requirements of real networks. Simply stated, FCAPS characteristics of the atomic service objects are defined such that those characteristics will extend and combine into the composite service object in the same manner as the service objects themselves are combined to create the composite.

Another example of an integrated service model is when services request services. Several services may be incorporated into a single, abstracted computational platform – the latter decides which specific type of object to produce, based upon the user’s specified requirements.

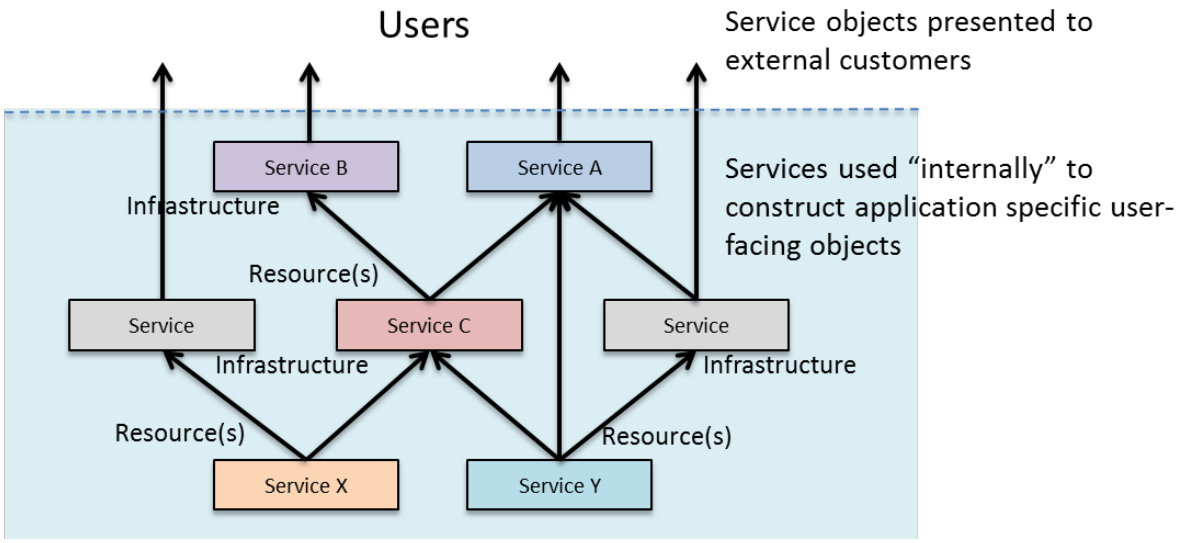

Figure 2.5: Composable services model

The general form of these “services requesting services” implicitly means that, as mentioned above in Section 2.2.1, the “user” of a service is no longer an actual human or organisational entity, but is very likely to be an agent performing some other higher-layer service. As shown in Figure 2.5, the resource object created by the lower-layer service becomes the “infrastructure” object(s) utilised by the higher-level service.

It should be explicitly stated here that, in general, services do not discriminate as to whether the client is an external “customer” or another internal service agent. A common historic practice among service providers is to “silo” (that is, to separate and isolate) their services. This is done intentionally, so that services are not dependent (or are only minimally so) upon other services offered by the provider or contracted from other providers – ostensibly to minimise the risk that a failure or flaw in one service could affect the availability, security, stability, etc. of other services. This common practice creates a significant degree of replication within the service provider and, arguably, if all of the services are stated to be “five nines”, i.e. 99.999% available, it is debateable as to its actual value. Regardless of its rationale and value, the result for the Integrated Services Framework and this document is that the function of “calling services” is not simply an external orchestration notion of a

customer requesting various services from different providers, but refers to internal services invoking one another to compose or construct a bespoke, complex service instance for the client. And that client may be another internal service agent.

This standard model of a network service has interesting implications. Given that the common, semantic lifecycle primitives used to interact with these services describe the requesting agents' required resources, and that the resulting resource object can be input to other service(s) as infrastructure, it follows that the form used to describe the requested resource can also be used to describe the infrastructure for these services. This means that there is no finite set of user-facing services – every service faces a "user", and the "user-facing" service objects are also the "infrastructure" objects that make up the inventory of some other service.

Composite construction, hierarchical construction, object-oriented construction, etc. are common paradigms in software engineering and modern programming languages. These concepts have been applied to many other areas, such as VLSI design, DSP applications, and even software development as a means to manage complexity. In the context of ISF, their meanings are as follows:

- Composite construction denotes grouping. A composite resource is a class or type of resource that contains other virtual (children) resources and indicates how they are interconnected at the data plane. For example, one could define a composite "Policy Enforcement Block" (PEB) that applies user-defined policy to a transoceanic 100 GE circuit terminating at an Open eXchange Point, the purpose being to allow the owner of the circuit to validate IP addresses that are allowed to use it (perhaps rejecting commercial-to-commercial IP flows). The PEB would provide an SDN switch and an ONOS controller VM. The target circuit would be cross-connected to the SDN switch, and the SDN switch then cross-connected to peering networks. The PEB resource could then be manipulated as a single object – reserved, activated, queried, etc. These object-class definitions can be assigned parameters such that they can be dynamically sized to suit specific needs – the number of potential peers, for example, or the size of the forwarding information base required for the IP filters, etc.

- Hierarchical construction is the ability for a composite class object to contain (or reference) other composite class objects. For instance, a "Datacentre" class may contain a number of "DC_Rack" objects, and the "DC_Rack" object may itself be a composite that references a number of "Bare Metal Server" objects connected with a "Multi-Point-Learning-Bridge", the latter being another composite built out of SDN switch element(s), controllers and circuits. The resulting instantiation creates a hierarchical resource tree that is conceptually analogous to a hierarchical file system, a directory of directories.

- Object-oriented construction is conceptually more about how a class of object inherits the characteristics of other classes – even as the instantiation does not necessarily result in multiple atomic objects. For instance, an "ONOScontroller" class object and an "ODLcontroller" class object are both built upon an "UbuntuHost" class object, the last being defined as using a "StockVM" object. The "StockVM" object has two use attributes: ISOimage, and, say, Context. The user reserves an ONOScontroller instance called OC1. The ONOScontroller class invokes an UbuntuHost class and specifies a Context=/user/bin/ONOS. The UbuntuHost class invokes a StockVM instance with the attribute of ISOimage="ubuntu14.4.iso" and the Context value that was set by the ONOScontroller class.

In this example, a single VM is instantiated, but it is defined based upon a number of other more atomic and common objects. Thus the OC1 instance has overlaid the default settings of the more basic objects, and unmodified objects in the more basic classes are inherited by the more specific class instances. This allows new objects to incorporate the changes of the base ancestor as new instances are created without needing to modify all of the more specific classes.

A final observation about object-oriented construction concerns the lifecycle of these resource objects. This lifecycle is described in more detail in Section 2.2.4 below, but in essence all resources conform to a single simple lifecycle state model. The functional primitives that Reserve() or Release() these resources, or Activate() or Deactivate() them, or otherwise transition the resources through their lifecycle, are common to all resource classes. However, in practice, these primitives are quite different depending upon the class being referenced. It should be obvious that an Activate() function to turn up a VM resource will require substantially different actions than an Activate() function to set up a virtual circuit resource. The master resource manager determines which version of Reserve(), Activate(), etc. should be invoked. This so-called "polymorphism" of the resource control primitives is hidden from the user – the user or requesting agent only needs to worry about what stage of the lifecycle they want the resource to enter; the specific form of the Reserve() or Activate() primitive is handled internally.

The model described for creating composite services can be linked to such recursion in service modelling, as presented in [TMFSID], the ETSI NFV Management and Orchestration [MANO], IETF draft on Hierarchical Service Function Chaining [HSFC], and in the GTS Architecture [GTS].

2.2.4 Management Operations

A set of operations for participating services and resources is also required. Such operations can follow the lifecycle primitives as described in the Generalised Virtualisation Model [GTS VIRTUALISATION] upon which the GTS is based, or they could follow the TMF Open API specifications, such as those of the Activation and Configuration API [TMF_Open], or Management and Orchestration operations as defined by the ETSI NFV specifications [ETSI NFV SPEC]. Other models may also be implemented, such as a Slice-based Federated Architecture (SFA).

The operational requirements include:

- **Allocation** of the necessary infrastructure to support a resource instance of the type requested with the specified attributes. This can include temporal scheduling as well, and is often called the "Reservation" phase.

- **Activation** of reserved resources by configuring the infrastructure and then placing the resource into service for the user and passing control of the resource to the user.

- **Deactivation** of a resource, i.e. taking it out of service while retaining the reservation, as might be required for maintenance operations.

- **Release** of the resource instance and all allocated infrastructure back to the available pool.

- **Querying** a resource for the current state or attributes associated with the resource instance.

Resources can then be managed through their lifecycle from initial inception until they are no longer of use. These basic lifecycle management operations are implemented in the OGF NSI Connection Service v2.0 standard for circuit provisioning [OGF-NSI-CSv2] and correspond directly to the GVM primitives as well (which were originally based upon NSI concepts), as shown in Figure 2.6.

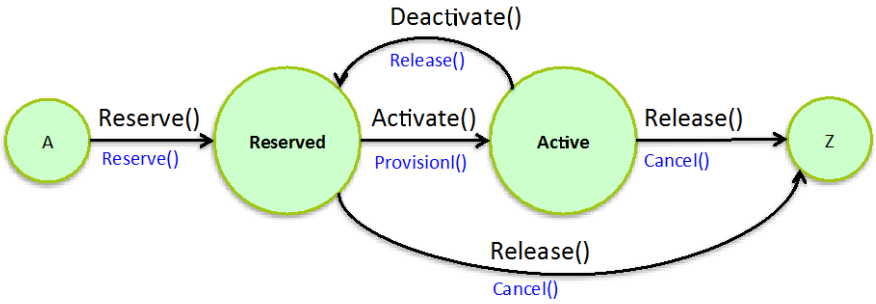

Figure 2.6: GVM / NSI virtual resource lifecycle

Note: The names of the GVM and NSI primitives sometimes differ, but their semantics – what they mean and do – are perfectly congruent.

2.3 Defining Network Objects and Network Functions

The JRA2 team has identified additional, functional attributes to map onto basic virtual objects to define more specialised virtual network objects. Applying specific attributes to an object and defining that specification to be a new object can simplify the user's task of building the networks they need. For instance, an Ubuntu host could be created by allocating a VM with the Ubuntu ISO image used to boot the virtual machine. Similarly, a Windows host could be defined by specifying a Windows boot image for a VM. An SDN controller could be created by the user allocating an Ubuntu "host" resource and then installing and running ONOS software [ONOS]. Alternatively, the provider could define a service object (say, an "ONOS-controller" object) that is constructed of a VM host with attributes that specify an ONOS boot image and user-specified attributes such as IP addresses of controlled SDN switch devices. The user sees an ONOS controller – not just a VM with some software. This manner of configuring different types of VMs from the same basic virtual foundation object is often called contextualisation – the parameters or attributes specified with the object request define the context of the instance.

Virtual Network Functions (VNFs) are also included in this context. Functions such as encapsulation/de-encapsulation, encryption/de-encryption, packet filters, rate limiters, etc., can be implemented in virtual containers and can be spun up very quickly and many (possibly hundreds) could be placed on a server, or distributed appropriately across the (virtualised) infrastructure.

There is potential for future work by the R&E community and/or industry collaborators in the area of Network Function Virtualisation and Network Service Chaining (NFV/NSC) that can take advantage of GÉANT's advanced virtual infrastructure management via GTS. The concepts of composability already presented include such logical network service chaining, or data flow topology.

Substantial work in service virtualisation is taking place within ETSI, IETF, TMF and other industry specification groups, and there continues to be a great deal of research work being undertaken in the same area. While there are similarities among these groups in many respects, there is not yet unanimity on standardisation. As an example, the IETF has numerous RFCs describing Network Function Chaining, Service Function Chaining, composite objects, and hierarchical decomposition of services using these concepts to address complexity and various administrative or responsibility roles. RFC 7665 Service Function Chaining Architecture [IETF RFC 7665] is recommended as a starting point. IRTF draft Network Virtualisation Research Challenges as of July 2017 [IRTF_NVRC] identifies areas such as QoS, service composition, multi-domain, network function placement, privacy and security, network slicing, and device virtualisation among the topics needing further research. Likewise for the EU 5GPPP Architecture (v2) [5GPPP_Arch].

JRA2 will continue to monitor the developments within these industry specification groups with a view to bringing the GÉANT tools and services into conformance and interoperability with standards, as they emerge and as appropriate to meet the needs of the GÉANT user community. It should be noted, however, that as of the time of writing, there is no clear consensus among the various ISGs as to which standards will prevail in terms of actual adoption. In parallel, GTS, with its simple Generalised Virtualisation Model, has evolved to arrive at a similar architecture as some of these key standards (e.g. ETSI NFV). Thus the virtualisation model presented by GTS, and now running as a demonstration service in GÉANT, can serve as a platform for exploring and developing the total ecosystem that will be required. This ecosystem requires experience designing customer-facing service functions, appropriate monitoring and automated fault analysis mechanisms, and best common practice design rules for virtualised services. Many of these principles, policies and business strategies can be extrapolated from GTS-based virtualised services even in GTS's pre-standards livery. Moreover, given the investment made and close proximity of the ISG vectors and the GTS/GVM approach, it would be logical to assume that the standards could benefit from the GÉANT experience, and, since the architectures are similar, that the APIs could be unified and brought into conformance at some future point with only minimal additional effort.

2.4 Service Composition – Responsibility and Verification

Composing services in this proposed Integrated Services Framework is expected to happen through orchestration, as in [ETSI-NFV]. "Orchestration" in this context may cover a wide range of activities, from assembling a few virtual resources into a testbed (as in GTS), constructing a service chain (e.g. to distribute web services requests across a set of servers), to coordinating complex workflows. Orchestration becomes even more complex if services in one domain are likely to act as agents that assemble composite services spanning many domains.

The orchestrator can invoke multiple service primitives to build a complex service instance on behalf of any user, including other orchestrators. Such a recursive service model is present in NSI [NSI], GVM, ETSI NFV, and TMF SID service models.

In cases when a service software agent must issue a service primitive to an agent that does not conform to a common model for service composition and orchestration, all that is required is the construction of wrappers that convert the common service/resource lifecycle primitives to appropriate command sequences in the old system. This is how GTS incorporates OpenStack and OpenNSA software into its service. This common semantic layer is sometimes called a "normalisation layer" (discussed above in Section 2.2.3) in that there is a common layer in which all interfaces and interactions take place in accordance with one shared service model.

A service provider only knows about, and is only responsible for, the services it delivers. It is not responsible for other services, upstream or downstream – indeed, it is questionable whether a "local" service agent will know if any such upstream or downstream services exist. If it is asserted that a "parent" service is responsible for delivering a working service to its requester, then the parent need only verify that its children service objects are functioning properly. If the root orchestrator agent acts accordingly to verify that its children all conform to service specifications, and each child does likewise recursively, then the proper functioning of the entire service complex can be assured. The behaviour of a root service object, even a sophisticated composite, can be determined and verified by "walking the tree" and combining the properties of the children.

There is an important corollary to the above hierarchical decomposition theory: if a child service object is found to be faulty, then the parent should also be considered to be faulty. A performance analysis of the parent performance would result in a performance analysis of each child. Similarly, it is the child service's responsibility to correct the fault.

As a result, there needs to be a protocol to a) ask that a child service instance be verified, b) notify a child service – or, more accurately, notify the *provider* of a child service instance – that it has been found to be faulty, c) notify a parent service that this service object has detected an internal fault, and d) notify a parent service that the fault has been corrected. The hierarchical and multi-domain composition and realisation of services means that the constituent subservices must be assumed to be opaque, i.e. their internal structure is not visible. Only external interfaces and expected performance attributes are known, and only measurements of subservice performance taken by an external agent(s) are reliable or trustworthy. Thus composability is a prerequisite for deterministic automated analysis and verification of these services.

In order to verify such opaque service objects, there must be an architecture that places test points at the edge of each service object and allows that service object to participate in a deterministic performance verification process. The protocol required will manage the process recursively to identify failing components and to notify the appropriate agents of the results. The appropriate agents are not necessarily the "user" but more likely to be the responsible service provider with a suspect service object. This Performance Verification and Monitoring (PVM) processing is the subject of work currently being developed as part of JRA2 Task 4.

The architecture and protocol are essential to the FCAPS aspects of hierarchical decomposition and verification. Any authorised agent, for instance, a high-level orchestrator, or an OSS/BSS process, can leverage this capability to conduct automated fault analysis. It should be noted that such hierarchical authorised decomposition of the fault analysis process is critical to global scaling of PVM across multiple and often remote administrative domains where direct operational trust relations are less likely to exist.

The end-to-end scenario is relative to each service object – not relative to some global absolute end point(s). That said, end-to-end service management as traditionally understood and as expected by the user is essential. However, it is not done by a single omniscient service provider. For example, in a composed virtual circuit instance – such as provisioned by the OGF Network Service Interface protocol – the end-to-end range is limited to the range each service provider is given in their respective segment as they each build a portion of the whole end-to-end transport path. Thus each service provider in a concatenated series of segments is only aware of their own segment and any children they generate within it in order to fulfil their segment "end to end". Indeed, in the decomposition of a composite service, a service provider is not even aware of whether any upstream or downstream concatenated segments exist. As described above, the performance of the entire montage of concatenated segments is the emergent performance of all of the constituent segments combined. The point here is that composability limits the responsibility of each service provider, while holding each service provider rigorously accountable for the proper performance of their piece of the total composite service. This model works whether it is applied to QoS on a segmented intercontinental circuit, or for a functional service graph of virtualised computational network functions.

It should be noted that (sub)service selection is often done at different levels – each service object being the product of a particular agent (human or computer) that does its own service decomposition and provisioning, and might do so differently from another agent or orchestrator. It is often the case that the process of service selection and specification within a composite service will take place on multiple levels, depending on the scoping of each domain agent. This means that an orchestrator is not a global agent. As it segments or decomposes the service processing, it acts as a requester to other service providers to reserve their respective service components. Those service providers may, in turn, have an intelligent service decomposition agent, i.e. an orchestrator, to further decompose the request. Thus rather than being a global master agent, orchestrators are simply schedulers and planners working with service providers that may also be schedulers and planners. There is no conflict here.

3 Network Services Development Roadmap

Network services evolution, as presented in this document, is already influencing the development of several tasks within JRA2. Many of the principles outlined here were results of work directly related to the GÉANT Testbeds Service, as it was recognised that the best way to build widely distributed and multi-purpose testbed environments, and isolate them from one another, was through an object-oriented approach that incorporated techniques such as virtualisation, service abstraction, composability, rigorous performance management, etc.

Four Tasks in JRA2 (Tasks 1 to 4) are related to the evolution principles already presented: Consolidated Connection Services, Service Provider Architecture, GÉANT Testbeds Service and Performance Verification and Monitoring. Further roadmap details are provided in Appendix A (Task 1), Appendix B (Task 2), Appendix C (Task 3) and Appendix D (Task 4).

3.1 Consolidated Connection Services (Task 1)

The Consolidated Connection Service (CCS) developed in JRA2 Task 1 provides an abstracted service model for connections that harmonises and consolidates existing GÉANT connection services under a single coordinated service development effort and a single integrated, intelligent/automated service provisioning agent, thus creating a fundamental building block of more advanced network services.

The CCS approach is based upon the NSI abstraction of a “connection”: a logical conduit between two locations through which user data is transported, unmodified, from ingress to egress.

This simple, technology-agnostic service abstraction allows various service definitions (or object classes) to be defined, as needed, by the broader user community and implemented locally by each participating NSI service domain. Currently, only the Ethernet-framed transport service is implemented, although this covers a large class of common use cases. The only requirements this service class asserts on the infrastructure is that Ethernet frames be transported, unmodified, from ingress to egress. Such a connection could be implemented over Ethernet infrastructure, or an MPLS infrastructure, or an OTN structure, or even over raw spectrum. The user perceives the defined behaviour of a logical conduit that moves their Ethernet frames from A to Z. This type of abstraction is essential for technology-agnostic virtualisation.

Further, the NSI service model poses a virtualised object-oriented semantic that deals with the service object’s *lifecycle*. Thus a connection is instantiated when it is Reserve()’d and the necessary infrastructure (whatever that may be) is selected, and allocated/scheduled, as necessary. The service instance is Provision()’d (placed in-service to the user) according to the reservation schedule, and it

can be Release()'d and Cancel()'d by user control as well. The reservation process for the requested service instance recursively decomposes the request into appropriate multi-domain path segments and recursively Reserve()'s those segments using NSI protocol [NSI]. This lifecycle model and the interface semantics disaggregate the user-facing service *model* from the underlying per-domain implementation at both the hardware and software/information architecture layer.

The recursive decomposition of the connection request is typically referred to as path computation, path finding, or path aggregation (NSI). However, this path selection and reservation process is also a facet of orchestration. Orchestration is generally taken to mean a broader coordination of heterogeneous services, e.g. NFV, or coordination of network resources with other e-infrastructures (such as work undertaken in JRA1 Task 3), but fundamentally this difference is subjective. The pathfinding function of NSI-based connection services does not conflict with other orchestration functions or approaches. The end-to-end view of the NSI connection object is defined solely by the end points specified by the requesting agent and the requested performance attributes. If that requesting agent is a separate orchestration agent doing its own path planning and reservation, this is allowed within the NSI service model.

The point to note here is that resource selection and reservation may take place at multiple levels that may hide or expose complexity or that have a different "end-to-end" scope. Each service planning agent (orchestrator, PCE, etc.) is able to be as detailed as it is authorised to be or can delegate the more detailed aspects to other appropriate agents. The resulting composite service object will still meet the requested user specifications.

As a starting point, CCS is piloted to deliver connection objects that GTS requires. GTS is a client of CCS. As CCS is an independent service, it can also serve other clients or purposes, and measures are underway to incorporate the GÉANT Bandwidth on Demand (BoD) service capability and the OTN-based Wave service. All three of these services deliver the same abstract connection object: Ethernet-framed connections with varying performance attributes. In addition, CCS is now being integrated with SPA as part of the ELINE project.

CCS/ELINE v1 results should be ready for pilot testing early in CY2018-Q1. A subsequent release (version 2) will provide additional needed functionality, and should be ready for pilot in late CY2018-Q4, with a potential first production capability early in CY2019 [M8.1].

3.2 Service Provider Architecture (Task 2)

As explained in [M8.2], Service Provider Architecture (SPA) in JRA2 Task 2 "is taking a unified approach to the typical working of Communication Service Providers (CSPs). The Task 2 vision is to set up the groundwork and building blocks for the implementation of a holistic framework that will bring together all available services and user/business/operations supporting activities under a single umbrella. In this way, a consistent user (customer)-centric view [CCV] of the GÉANT network and its services can be provided in an automated, self-service manner to all relevant stakeholders."

SPA in JRA2 Task 2 builds common underlying information systems, supports common operations support systems / business support systems (OSS/BSS) processes and is aligned with TM Forum

Frameworx [TMFFx]. The initial services portfolio consists of abstracted service models that are technology agnostic in terms of their physical implementation.

OSS are the systems used to support the daily operations of the service provider (such as network, inventory and service management), while BSS covers business management processes such as billing and customer management [BOSSF]. SPA's focus for the GN4-2 project period is on fulfilment and assurance operations procedures, as defined in TM Forum Frameworx [TMFFx]. These constitute about 30% of the component APIs that comprise the Frameworx architecture. Implementing the remaining APIs and porting existing services to interoperability and then ultimately into full integration will take several years and substantial and ongoing software support. The benefit of this investment will be a well-vetted information architecture serving GÉANT, and forming a comprehensive, unified underlying information base.

The ELINE project is a "pathfinder" project that aims to evaluate empirically the SPA and to understand more clearly the service implementation and transition process. ELINE employs an initial version of SPA, and will migrate an existing GÉANT service to the SPA. In Phase 1 of the ELINE pilot, the Consolidated Connection Service (CCS) will be covered by SPA as the test case service. ELINE will also integrate the service performance verification and monitoring function (PVM) for CCS into the SPA environment. The CCS service class implemented will conform to the MEF Ethernet Private Line service specification [MEF-10.3].

In Phase 2 of the initial pilot, the ELINE business process as defined for the SPA-based CCS will be evaluated with a small set of opt-in use cases. This will provide feedback and tuning of the ELINE service delivery process.

ELINE includes the following aspects of SPA, in accordance with TMF Frameworx [TMFFx]:

- The Resource Inventory – to keep records of the infrastructure elements and the topology.
- The Service Inventory – with information about customer connections (circuits).
- Monitoring of the established circuits.

As explained in [M8.1], together with more details about the ELINE project and the service, ELINE builds upon the existing NSI-compatible code base of the OpenNSA software package, and adapts the three initial SPA components mentioned above to leverage the existing OpenNSA/NSI code base [OpenNSA].

The ELINE project combines standards and best practices from MEF, TMF, OGF, IETF, IEEE, and ITU, and presents a model for future consolidation of GÉANT connection services.

The ongoing work between TM Forum and ETSI, to converge the next-generation service-delivery ecosystem developed by TM Forum with the requirements for NFV imposed by the ETSI standards, is reflected in a suite of evolving specifications [TMF_ETSI_SPEC]. These hybrid network management and information models for virtualisation, for example, Hybrid Infrastructure Platform (HIP), provide example use of the various TM Forum artefacts (e.g., APIs and data models) to manage an infrastructure that consists of virtualised, physical and mixed resources or "Connectivity Patterns for Virtualisation Management". Such efforts have been taken into account by the work carried out in JRA2 T2.

3.3 GÉANT Testbeds Service (Task 3)

GÉANT Testbeds Service (GTS) in JRA2 Task 3 (and previous work undertaken in GN4-1 and GN3) has defined and developed an integrated virtualised service capability based upon a Generalised Virtualisation Model (GVM). GVM was defined as part of the GTS design effort and drew upon prior research work in FEDERICA, NOVI, GEYSERS, OFELIA, GOFF, ExoGENI, DRAGON, MANTICORE, and the NSI standards working group. GVM was positioned as a potential extension and enhancement to the Open Grid Forum Network Service Interface (NSI) service abstraction model. The GTS Task is focused on developing a viable, scalable service model to support a broad range of network research and distributed applications/service development and piloting. GTS's mission was not to develop an SDN-specific testbed, but rather to develop a flexible and reliable capability supporting a wide range of network research in the WAN. A Generalised Virtualisation Model was a necessary step towards this goal and has served as the guiding design principle for the testbeds service to date. The incremental refinement of GVM, and the extensive experience gained in the development and deployment of GTS, have resulted in a virtualised service capability in GÉANT that now informs the integrated services vision described in this document.

As GN4-2 comes to a close, GTS has deployed version 6.1.2 into the field as a pilot service. The service supports a number of resource classes including:

1. Virtual machines (VMs), which can be allocated in a number of "flavours" corresponding to progressively more powerful VM configurations.
2. Bare metal servers (BMSs), which come in two flavours. The larger provides 2 CPUs @ 8 cores, 2 x 10 Gbps NICs, 128 GB memory, and SSD.
3. Virtual circuits (VCs), which can be best effort or performance guaranteed (i.e. have a guaranteed capacity and associated shaping). GTS virtual circuits are currently limited to 10 Gbps, though the service can support 100G if available.
4. Virtual SDN switches, which are mapped to physical OpenFlow hardware or to software-based OpenFlow switches. These objects function much like VMs do on physical server platforms, allowing users to define the virtual switch characteristics while ensuring efficient and transparent sharing of the hardware.
5. External domain objects, which allow the operations support team to manually arrange for a global circuit to appear at a GTS location. The user can then reference the external domain (ED) like any other virtual object and thus link their network slice to that ED.

Other resources planned for version 7 or later:

6. Storage resources, required for large data applications and for migration/grooming functions.
7. Cellular/wireless spectrum resources. These are being developed for GTS to serve mobile research in the metro and campus mobility areas (5G, Internet of Things, etc.).
8. Layer 3 resources such as IP routers, VRFs, etc.
9. Potential other instruments, sensors, platforms (e.g. P4 switch, GPUs/FPGAs), which can be timeshared by reserve/schedule functions of GTS or virtualised for agility/elasticity.

For a graphical summary of the status of GTS resources, see Figure 2.3 on page 11.

GTS supports approximately 50+ independent testbeds (or network "slices") at this time. The largest GTS slice to date is a performance verification and monitoring (PVM) testbed comprising almost 90 components. The GTS effort has produced >100,000 lines of code, and the service incorporates a substantial amount of third-party open source software (such as OpenStack and OpenNSA).

In parallel, the ETSI NFV industry specification group virtualisation model has introduced many of the principles discussed in this document, including a number of atomic services that create virtual objects, such as point-to-point connections, computational elements, storage elements, etc. Recently, this ISG has also introduced groupings (equivalent to GTS composites) and functional service graphs (equivalent to GTS topological/data flow specifications).

The IETF YANG data model for network topologies specification, introduced in 2018 [IETF RFC 8345], is another potential means of describing networks that could be used as input to GTS. The GTS team will explore this specification, and the other RFCs it references, to determine whether they have the potential to bring a greater functionality to the service framework or are lagging behind the GTS development effort or do not suffice for the R&E requirements.

GVM defines a very simple set of service primitives (drawn from the NSI standards) to create virtual objects, describe their data flow topology, and step them through their lifecycle. The GTS software suite provides these primitives in the form of an exposed API that can be called from user GUIs or other software agents to construct the integrated service environment.

GTS v6.1.2, the latest version deployed in the field, includes a visual drag-and-drop network editor, support for deterministic performance (QoS) circuits, a two-step project registration/approval process, a Docker-based installation process, and many minor fixes. GTS v6 will be the first fully supported production version of the service.

At the time of writing, GTS v7.0 is in the planning stage. It will include important provider features such as a policy engine, resource allocation management (quotas), migration and grooming, and simplified/incremental deployment processes. In addition, v7.0 will deliver further user features such as checkpoint/restart, active modification, improved link termination features, enhanced Internet connectivity options, multi-domain one-stop shopping, and advanced DSL and composite features.

More details about the GTS current status and future plans are provided in the Milestone document [M8.3] and *Deliverable D8.9 GÉANT Testbeds Service 6.0* [D8.9].

3.4 Performance Verification and Monitoring (Task 4)

The formal objective of Task 4 Network and Service Monitoring, including Performance Verification and Monitoring (PVM), as defined in [M8.4], is "to develop a generalised, but comprehensive, network monitoring capability that will: measure the performance of GÉANT network services, provide real-time feedback to network operations personnel or users", as well as "determine

whether the network services are performing to specification, and if not, initiate an automated analysis to localise the fault, and notify the appropriate agent(s) to take corrective action."

The PVM Task has defined an architecture for undertaking performance analysis of physical and/or virtual service objects by which such services can be monitored, verified, and problems localised across a global landscape of aggregated services.

The monitoring, verification and problem localisation processes are being integrated with the SPA, such that PVM can be applied generically to any atomic service and recursively, to composite services. Such monitoring and verification must be tailored to each service, for example, a virtual circuit would be monitored and performance verified quite differently than a virtual machine or a testbed or an encryption VNF. PVM is currently addressing connection services as part of the ELINE project. The PVM Task plans to provide monitoring and verification for other services as well, e.g. computational elements, switching elements, etc., focusing on monitoring of key network service performance parameters of a service in production, as defined in Y.1540, Y.1541 and MEF specifications [ITU-1540], [ITU-1541], [MEF-10.3], as presented in the Milestone document [M8.4].

4 Related Work/Supporting Standards

The key work presented here correlates with the work of three standards and/or best-practice-oriented working groups: the Open Grid Forum for the Network Service Interface (NSI) standard and Network Mark-up Language (NML) inter-domain topology descriptions, the European Telecommunication Standards Institute (ETSI) [ETSI] in ETSI NFV work, and TM Forum [TMF] with TMF ZOOM and TMF Frameworx.

4.1 Open Grid Forum

The Open Grid Forum (OGF) has standardised the Network Service Interface (NSI v2) inter-domain connection provisioning protocol and service architecture [NSI]. NSI offers abstracted and technology-agnostic connection services, and its service-oriented lifecycle primitives formed the basis of the virtualised services of GTS.

The OGF Network Mark-up Language (NML) specifications have been broadly accepted within the R&E networking community to describe network topology [NML]. NML provides both a global inter-domain, service-oriented topology consistent with the NSI service abstraction, and a standard means for describing intra-domain infrastructure topology.

The NSI protocol is now deployed in almost thirty R&E networks worldwide, including most Open eXchange Points. GÉANT has been a key participant in the NSI specification and standardisation process, and continues to co-chair the OGF NSI Working Group. Other GÉANT partners (in particular SURFnet, PSNC, NORDUnet, and I2Cat) have been active in both the NSI and the NML definition and standards process.

The lifecycle primitives described in this ISF document – the Reserve()/Release(), Activate()/Deactivate() and Query() functions – correlate directly to the NSI lifecycle primitives, forming a proper superset of them. Indeed, the early development of the GTS lifecycle was based explicitly upon the NSI lifecycle. Figure 2.6 on page 16 shows this correlation, while Figure 4.1 below shows the common features of the GTS/GVM object types (and of this Integrated Services Framework) and NSI Connection Service v2.

As at the time of writing, the Generalised Virtualisation Model defined and implemented in GTS has been proposed to the NSI working group chairs for potential merger to create NSI v3. This proposal is now under consideration.

As of December 2017, the EU's *View on 5G Architecture* document [5GPPP-Arch] recommends NSI as an option for its 5G initiatives. However, the document also states that NSI does not have

virtualisation extensions supporting virtualised services generally. This is both a strong endorsement for NSI and the service architecture, and simultaneously a challenge to move forward aggressively with the NSI/GVM extensions and standardisation.

Thus it is a recommendation of this Integrated Services Framework – particularly as it applies to virtualisation of the integration of 5G and IoT service capabilities (such as virtual network slicing) – that GÉANT should also support the NSI v3 effort.

NSI v2 GVM (Virtualization extensions to NSI)

Connection	Virtual Machines	Virtual SDN Switch	Bare Metal Server	Composite Object	Other...	NSI Service Definition GVM Object Class/Type
Capacity Jitter MTU FER	Location MemSize Cores Ports ISOimage	Location OF version Hard/Soft Controller Addr	Location MemSize Cores ClockSpeed ISOimage	Children Adjacencies Overloaded attributes	Class specific Attributes	Object Attributes
Src, Dst ports	Port{1..n]	Port{1..n]	Port{1..n]	Port{1..n]	Ports	Object data flow

Figure 4.1: Common features of GTS/GVM object types and NSI CS v2

4.2 ETSI NFV

Perhaps the most relevant recent developments around virtualised network services, and NFV in particular, are those offered by the ETSI NFV industry specifications group [ETSI NFV]. This group was established and began working on a reference architecture for Network Function Virtualisation in 2012 (shortly before GTS development was commissioned).

As stated in Section 3.3, GTS was conceived based upon prior research work in a variety of programmes. Development began in 2013, with the goal of delivering user-defined and user-controlled wide-area testbeds for network research and with the intention of supporting a wide range of research topics. ETSI NFV was established specifically to create a coherent approach for the industry development of emerging NFV technologies.

Despite distinct starting points, GTS and ETSI NFV have evolved in convergent directions. The GTS architecture and the ETSI NFV architecture currently overlap in several key functional areas, including virtual infrastructure management (VIM) components and some orchestration aspects (as shown in Figure 4.2), and there has also been convergence of the ETSI architecture with the current GÉANT Testbeds Service architecture with respect to virtualised environments.

In some areas, the GTS implementation and the Generalised Virtualisation Model architecture have been well ahead. For example, the concept of hierarchical composite structures has only been adopted by ETSI NFV since 2017 as “functional service graphs”; GTS has supported composites since 2013. Even today, the ETSI NFV model differentiates infrastructure (hardware analogues such as VMs and servers) from functional elements such as virtual network functions (VNFs). GTS (GVM)

views these as simply different classes of objects that can co-exist in a unified architecture, letting intelligent software agents do the work of differentiating their implementation issues or optimising infrastructure mapping, etc. (In a 2016 demonstration, GTS was shown to support lightweight containerised VNFs and service graphs using the same conceptual network model as the infrastructure analogues.)

Ongoing analysis and comparison is part of JRA2's future work. It is possible that GÉANT could offer test environments for the ETSI NFV efforts. GTS experience and working code might also influence the emerging models, while becoming more interoperable with them. However, it is the recommendation in this document that GÉANT's main goal with development efforts such as GTS should not be to ensure that GTS conforms to the emerging models, but rather that these evolving recommendations reflect and support the experience and vision the R&E community requires and has deployed already in services like GTS. The network research and R&E community should be providing the pre-standards examples. Such pre-standards activities are of great interest to the commercial sector and ISGs, and this is an area to which the R&E community has a unique ability to contribute.

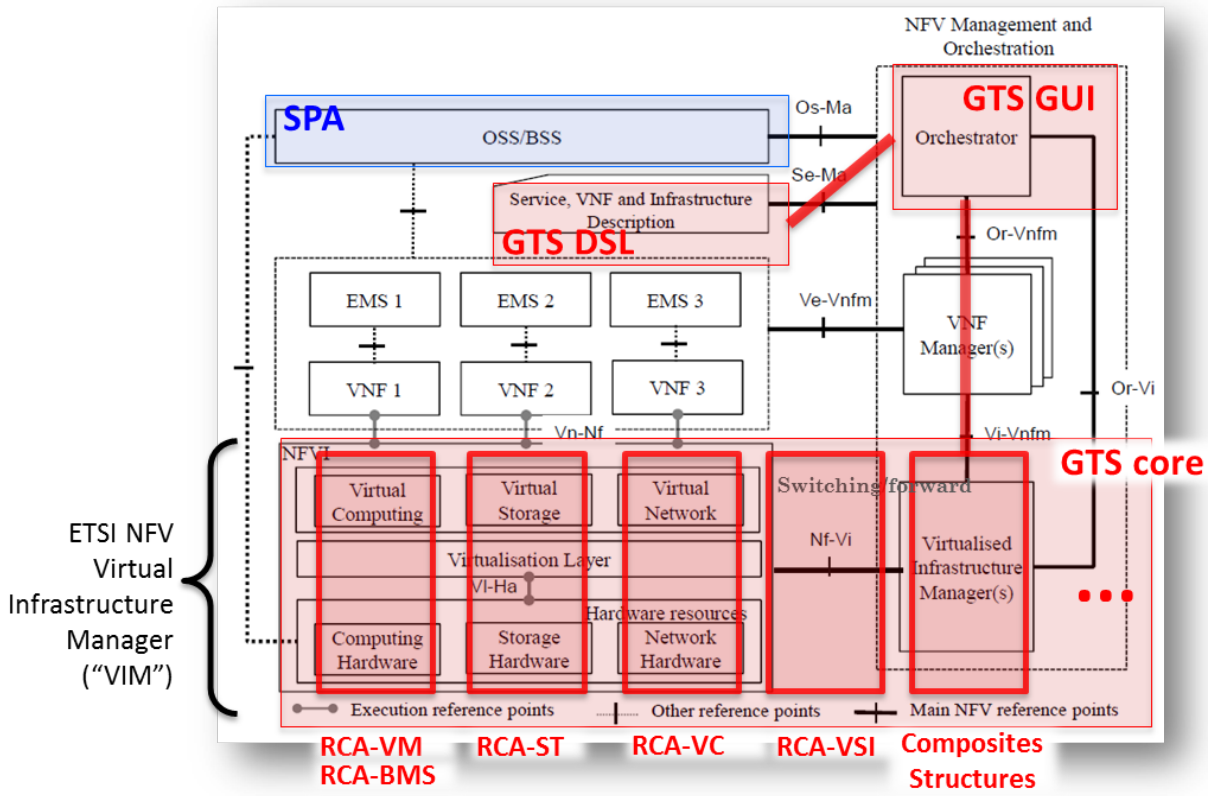

Figure 4.2: ETSI NFV reference architecture and JRA2 solutions: preliminary mapping

The second aspect where the two architectures, ETSI NFV and GTS, correlate is the OSS/BSS. (OSS/BSS was outside the scope of the NFV reference architecture, and hence is represented only as a single small box in the mapping diagram in Figure 4.2; with regard to the overall viability of the architecture, OSS/BSS aspects constitute a much more substantial and important functional area

than the size of the box suggests.) In JRA2, OSS/BSS work is conducted within Task 2, Service Provider Architecture, implementing the ZOOM architecture.

4.3 TM Forum

The Integrated Services Framework relates to TM Forum specifications in multiple aspects, most important of which are ZOOM architecture [TMFZOOM2], Business Process Framework – eTOM [eTOM, TMFFx] and Information Framework - SID [TMFSID].

As per [TMFZOOM1], TM Forum Zero-touch Orchestration, Operations and Management (ZOOM) "is working to develop best practices to support both the technology and business transformation brought about by the introduction of Network Function Virtualisation (NFV) and Software Defined Networking (SDN)".

In its Business Process Framework, TM Forum provides systematic elaboration of business processes at several levels of complexity. This has already been used as a starting point for the SPA and PVM work. While SPA focuses primarily on fulfilment and assurance, PVM relates to the Service Quality Management process in its search for performance verification.

TM Forum Information Framework "provides standard definitions for all the information that flows through the enterprise and between service providers and their business partners" [TMFSID], thus helping the design, development and implementation of integration interfaces. Decomposition of individual services into atomic abstracted services, and the integration of management methods and functions into the specific, more or less complex customer-facing or internal services, can use TMF SID as a potential source of information for easier composition and decomposition of services. TMF SID is used in the SPA work, as well as in the ELINE project.

The information architecture described by the TMF specifications offers GÉANT a well-vetted model that can provide a foundation for GÉANT's long-term information management strategy.

5 Conclusions

The potential impact of virtualisation and composability techniques on GÉANT's delivery of future network services is enormous. The Integrated Services Framework described in this document leverages virtualisation and dynamic service construction techniques that will allow GÉANT to easily build complex, bespoke services that can meet world-class reliability, availability, flexibility, performance and maintainability objectives. JRA2's development of these services has evolved from the Generalised Virtualisation Model, implemented in the GÉANT Testbeds Service, and the first-hand experience of developing, deploying, and running the GTS service.

Rigorous network service abstractions and virtualisation provide the prerequisite capability for supporting automation, orchestration and interoperability. Virtualisation provides a framework for engineering and operating basic atomic services and subsequently for composition and orchestration of more bespoke and sophisticated services. Moreover, they do this in a highly supportable fashion. This allows services to be tailored to the specific needs of each user, reduces duplication and complexity, and improves the reliability and manageability of the resulting services. In addition, a clear understanding of the internal logical structure of a service and its constituent elements is achieved. This enables flaws to be identified in the service logic, or faults in the service instances themselves – and such rigorous fault detection and resolution leads to greater predictability and availability.

The consolidation work is related to several standardisation and best-practice efforts, including TM Forum ZOOM and Frameworx, the ETSI NFV architecture, Open Grid Forum NSI standards, and MEF service operations specifications. Due to the continuously evolving standards and the constantly changing landscape, the work presented here is ongoing (see also the roadmaps found in the appendices).

Appendix A Task 1 Roadmap for Consolidation of Point-to-Point and Point-to-Multi-point Services

The mission of JRA2 Task 1, Consolidated Connection Services, is to consolidate the many connection-oriented services currently offered by GÉANT into a single, unified service offering. The initial services included are GÉANT Plus, Bandwidth on Demand, GÉANT Testbeds Service virtual circuits, and MDVPN. Other, similar connection-oriented services, such as GÉANT Lambda and GÉANT Alien Wave, are planned to be incorporated into this unified model as the service progresses. The objective is not only to make access to these services more of a one-stop shop, but also to collapse multiple product service concepts and support silos into a single, comprehensive service planning and development function that presents a simplified "connection service" model to the user. The strategy is to abstract these services in order to identify the common user requirements, and allow an intelligent provisioning system and API to map those service attributes to the appropriate underlying technologies and infrastructure elements.

Fully automated delivery of these services is a key objective. This effort includes: GUI development, hardware configuration protocols, NSI protocol, time-based path analysis and reservation, network topology specification/distribution, path finding, federated AAI, policy support, multi-domain interoperability, etc.

A.1 Work Plan

Task 1 of JRA2 is a mostly infrastructure-oriented task. It will cover re-evaluation of all connection services currently offered by the GÉANT network (there are services that are very similar). Special attention will be paid to service implementation in the current GÉANT network infrastructure, the delivery and administration process, and the required level of automation. The results of such effort should be narrowed and a simpler service portfolio maintained. For example, BoD, GÉANT Plus and GTS circuits are the same thing with a different provisioning process. They can be presented as one common service definition of a connection-oriented service.

An important part of Task 1's responsibility will be the provisioning of a service development laboratory in Prague, in cooperation with other JRA2 Tasks. The laboratory is now used for the development of the GÉANT Testbeds Service (software development as well as hardware engineering, integration, and configuration interface testing).

Except for laboratory support, which will be continuing effort, the planned Task 1 roadmap will consist of the following:

A.1.1 CCS v1.0

CCS version 1.0 will focus on the consolidation of point-to-point, Ethernet-framed virtual circuit provisioning. This will establish a Network Service Interface (NSI) standard circuit provisioning capability, using open source OpenNSA software as the code base. These will be best-effort circuits only, with no performance guarantees. (This is the same as existing EoMPLS circuits of BoD and GÉANT Plus.) This initial CCS will assimilate the GÉANT Testbeds Service connection provisioning, the BoD service, and will provide an open and publicly accessible NSI interface that other user agents can use to create "connections" across the GÉANT core.

As part of CCS v1 transition to the GÉANT core routers, Junos Space was utilised to mediate provisioning requests between OpenNSA and the routers (Junos Space is a proprietary configuration management tool from Juniper). This new southbound interface has proved to be unreliable for the rapid provisioning requirements of GTS. Resolution of intermittent timeout issues and provisioning failures of Junos Space has required more time and effort than was initially anticipated. This beta testing included GTS as the primary user application. Further beta testing has continued to resolve the Junos Space interface reliability issues. GÉANT Operations and JRA2 jointly agreed to halt any further debugging of the Junos Space SBI. OpenNSA will be reverting back to the prior (simpler and more reliable) direct SSH interface used in GTS v3. This change was due to occur in early 2018; final testing and improvements took place 2018 Q3. CCS v1 will also begin multi-domain peering with other NSI networks, starting with NetherLight and CESNET. Chain provisioning to direct neighbours is currently enabled in OpenNSA, and NML topology processing enhancements will be added to support more extensive "tree"-style path reservations following the NSI recommendations. Other NSI domains, particularly those at Open eXchange Points, will quickly follow. This requires enabling TLS authentication between NSI agents in peering networks, which will have an impact on GTS and other client agents within the GÉANT ecosystem that do not currently hold digital certificates applicable for TLS authentication. A minor update to deal with authentication has been developed for GTS and in the CESNET NSI network, and tested against NetherLight (P2 Q2), meaning multi-domain peering can begin in earnest.

As of December 2017, CCS v1.0 had been deployed at eight locations in the GÉANT core to support GTS v4.0. This deployment fully implemented GTS virtual circuits as a best-effort Ethernet over MPLS service, with a maximum capacity of 10 GE. Additional PoPs will be supported as the BoD service termination points are migrated to the CCS.

CCS v1.0 is planned to be in official pilot by 1 March 2018 (updated timeline as of December 2017). This pilot is planned to continue until CCS v2 is ready.

A.1.2 CCS v2.0

CCS v2 is now planned to deliver the QoS performance-guaranteed connections, an NSI-enabled circuit-provisioning GUI, and transition of GÉANT Plus circuits to CCS management. The QoS feature is the critical path for CCS version 2.

Quality of Service for Ethernet-framed connection services – or, more broadly, "performance-guaranteed" (PG) connections – are a critical feature to be offered by CCS. Without per-circuit performance management, a service like GTS cannot offer researchers or developers a predictable level of service, making performance analysis impossible and many applications untenable. Further, without rigorous capacity management a service like GTS can easily overwhelm the infrastructure, negatively impacting all the GTS service environments, as well as posing a risk to other services. QoS and performance guarantees are essential for GÉANT and emerging virtualised services.

The MPLS core is currently unable to provide QoS guarantees on MPLS LSPs. As a result, it was decided the needed QoS/performance guarantees would be delivered over the Infinera-switched OTN layer instead of the MPLS core. This has necessitated the development of a new southbound interface in OpenNSA to manage the switched Infinera kit, and requires additional OTN hardware to be purchased and deployed to support the ~100 interfaces that must support PG circuits. (Update: as of Q4 2017, the OTN path planning and timeslot allocation code was complete in OpenNSA, and a southbound interface to Ciena OTN kit was working. The SBI for Infinera was expected in early Q1 2018 in the CCS software, and the additional hardware deployment in Q3 2018. As of P2 Q3, a decision has been made to switch from OTN as a way of providing performance-guaranteed circuits, and the new Juniper MX204 router, which supports HQoS, has been selected for deployment instead. These routers are planned to be deployed in selected GÉANT PoPs in the last quarter of 2018.)

In parallel with the consolidation of the provisioning aspects of these Ethernet-framed, point-to-point circuits, Task 1 will develop or adapt an interactive user/operations interface to establish and monitor standalone circuits. This is mainly to provide an operations tool for the visual monitoring of the status of service instances, and secondarily for direct manual provisioning by operations or conventional users. Upon making this GUI available, CCS, in conjunction with Operations, will begin transitioning GÉANT Plus service instances to full CCS support. (Update: as of Q4 2017, the MEICAN graphical provisioning tool, an open source, NSI-enabled tool developed by the Brazilian NREN RNP, is being evaluated for this role. As of P2 Q3, it was decided that the self-service portal developed by JRA2 T2 for the SPA would be used for the CCS GUI.)

At this point, all Ethernet-framed point-to-point connection services previously covered by BoD, GÉANT Plus, GTS and GÉANT Lambda will be provided by CCS. The CCS software can support GÉANT Open as well. CCS will provide a public, secure, NSI API for other clients to use to request or monitor circuits. Thus, a single service and a single product development team will be responsible for the evolution of "Connection Services", and the features of these services can be managed and provisioned more effectively and efficiently.

A.1.3 CCS v3.0

CCS v3.0 will incorporate enhanced-path selection capability, additional multi-domain peering, and simplified installation, configuration and operations.

The enhanced path computation is needed to support multi-point services and to provide improved multi-domain scheduling, topology processing, and switching point placement for multi-point multi-domain connection services. It is relevant, for instance, to the process required to incrementally grow a multi-point L2 broadcast domain, or L3 forwarding domain (VRF), across domains in Europe and North America. Inefficient layout or periodic rebalancing can result in highly inefficient trees.

The simplified CCS v3 is intended to ease the initial deployment of connection services so that NRENs, campuses, or even labs, can initially deploy CCS services using only commodity hardware (e.g. some available servers running OVS or similar software switches), and can install and configure the service quickly and easily, to minimise the impact on local engineering or operations staff (the goal is less than one hour per server) and to ensure the operational support of the service can be easily integrated into existing environments.

As a facility gains experience, the deployment process will yield more control of the configuration, and as performance requirements eventually increase, the servers can be upgraded to hardware switching elements. The goal is to make virtual circuit services as simple as possible to install, configure and get running to enable services to expand into the last kilometre.

CCS v3.0 is currently estimated to be available in early Q4 2018.

A.1.4 ELINE Project

In June 2017, a joint effort with Task 2 was initiated in order to design and develop the ELINE service (internally referred to as “the ELINE project”). ELINE is a prototype integration of the Consolidated Connection Service capability, using the SPA information architecture and APIs.

The ELINE project, while using a basic CCS v1 service platform, is part of the JRA2 T2 Service Provider Architecture development effort. This first ELINE port of CCS to SPA is a high-level, wrapper-based approach sufficient to evaluate a real service within the SPA context. If this ELINE pathfinder project proves the SPA to be a viable and strategically valuable approach, a more substantive adaptation of CCS to utilise the SPA information base directly will be required. The scope of effort and schedule for such a re-implementation will be addressed as part of the SPA pilot report.

A.1.5 OpenNSA

Task 1 contributes to the OpenNSA open source code base. The OpenNSA has a modular, southbound interface architecture, which makes it easily adaptable to many possible network infrastructures. The NSI connection service specifications are general enough to allow other institutions/networks (member NRENs, other research networks around the world) to adopt it, and using the standardised NSI lifecycle API, allow automated provisioning of multi-domain connection instances. Task 1 can support this deployment in member NRENs by developing OpenNSA backends, as needed, for network equipment used by interested NRENs.

A.1.6 QoS-Aware Implementation

The first QoS-aware CCS implementation was introduced and demonstrated at SC17. The implementation works with the Ciena 6500 platform (OTN switch) and allows a flexible number of 1 Gbps timeslots to be mapped to a connection, as needed, to guarantee capacity and jitter bounds. It was demonstrated together with GTS deployment over Ciena research infrastructure in Ottawa, Canada (called the Ciena Testbeds Service). The southbound interface to cover Infinera OTN transport is work in progress, and will result in the consolidation of the GÉANT Wave service.

Within the GÉANT core network, performance-guaranteed circuit services have been implemented using Ethernet over MPLS [IETF RFC 4448] and hierarchical QoS processing. This required a substantial hardware upgrade in core network elements to support the capacity enforcement schemes necessary for asynchronous packet performance guarantees. As GN4-2 comes to a close, the requisite EoMPLS switches have been deployed in three of the eight GTS service locations (London, Amsterdam and Hamburg). These QoS-capable switches are part of the CCS NSI virtual circuit service topology. The proper traffic behaviours and co-existence for both performance-guaranteed and best-effort circuits were tested and demonstrated at Supercomputing 2018 (November 2018) between London and Hamburg. Work to support QoS at the remaining five GTS sites (Paris, Prague, Milan, Madrid and Bratislava) is scheduled to be completed by early 2019.

A.1.7 GÉANT Lambda and Multi-Point Services

Another area to study in T1 is the GÉANT Lambda service [GÉANT_LAMBDA]. As this is a connection service (and in this case also just an Ethernet service over “big pipes”), it should be provisioned using the same tools and protocols as other circuit services. In order to achieve this, T1 will examine the potential to provision such circuits using similar service definitions, as mentioned above. This entire circuit services portfolio should then be presented to users using one common web portal or programmatic API (e.g. NSI).

Multi-point services such as L3VPN are, by their nature, also connection services [L3-VPN]. There are two models that are generally considered for abstracting multi-point services: the first is to treat point-to-point (P2P) connections as a trivial case of a multi-point connection, i.e. a multi-point connection with only two service termination points. The second model treats multi-point connections as a composite construct built up from one or more “switching” objects, each connected by point-to-point connections to the user-designated end points and P2P connections interconnecting intermediate switching points for large spanning trees. This latter model leverages work in optimised spanning tree construction to grow larger, multi-point connections efficiently and incrementally.

The CCS development team expects this composite model to be more flexible and dynamic, particularly in multi-domain global construction and in future NFV environments as it allows explicit placement of switching points, enabling optimised distribution trees and minimal latency. CCS plans to use it to implement multi-point services. The Task 1 team will investigate and propose possible generalised service definitions to cover multi-point services in order to allow their provisioning using NSI protocol and common software elements, such as path computation, virtualised switching and control functions, etc.

Appendix B Task 2 Roadmap for Service Provider Architecture

The Service Provider Architecture Task aims to provide an initial OSS/BSS pilot implementation that sits on top of the virtualised services and resources as represented with the Integrated Services Framework (ISF) and referenced by ETSI Network Functions Virtualisation (NFV). It is composed of loosely coupled microservice-based components that represent different functional components based on the TMF TAM decomposition. All components expose TMF-compliant REST APIs that use a SID data model. SPA focuses on the business/high-level operation-oriented orchestration of the components based on the end-to-end eTOM-based business processes.

The initial SPA v1.0(beta) pilot is implementing the Consolidated Connection Services(CCS) point-to-point Ethernet private line service within the SPA information architecture. This CCS implementation within SPA is designated the ELINE project. The ELINE effort will implement the CCS-based P2P connections using the SPA northbound API. This pilot is scheduled to begin January 2018, and run for 120 days. This pilot has two objectives: a) to determine the actual scope of effort to port a service to SPA, and b) to test the process of business process specification and use.

The features available in the initial pilot focus on migration of the CCS service to function under the control of SPA components and offer service fulfilment to a set of alpha users. These users will provide their feedback on testing functionalities related to the automated creation and termination of ELINE-based circuits using a self-service portal. Based on the information provided from the dynamic service catalogue on the portal, the user chooses a service and activates the related business processes. The system captures user orders and tracks their status, manages the service instance data in the service inventory, and uses the logical resource inventory. For the purposes of interfacing with the T1 CCS ELINE implementation, a TMF-compliant activation and configuration API is used.

The pilot will run until the end of April 2018, with the aim of capturing and documenting the efforts needed to migrate the ELINE service to SPA and the user experience of using the automated system. Any issues identified in terms of design, implementation or performance and scalability will be addressed before continuing with the extension of the solution. It is also presumed that the ELINE pathfinder will expose hidden complexities. The pilot report will inform the future planning for SPA implementation. If the decision is to continue the implementation, the report will enable Task 2 to better plan the buy vs. build analysis, and better estimate the manpower and timeline that will be necessary.

SPA v1.1 development will commence immediately following the conclusion of the 1.0 pilot period (April 2018). Key features of 1.1 include a) integration of the Performance Verification and Monitoring (PVM) module to perform verification testing and monitoring of ELINE service instances, and b) implementing the SPA "feasibility" function. The PVM module (developed by JRA2 T4 PVM) will verify that the circuit is established correctly and functioning as expected after provisioning, but before placing the connection into service for the user. This module will continue to monitor the service object and will trigger alerts and/or notifications when the monitoring functions determine the service object has failed. "Feasibility" translates to path selection and reservation for the CCS point-to-point connection services. This function is currently performed by the OpenNSA agent of CCS as the NSI Reserve() primitive. This function will perform end-to-end reservation of the needed path segments across multiple domains, or may be restricted to path selection only within the local domain.

This work in version SPA 1.1 will provide further in-depth experience addressing service migration to SPA. V1.1 is expected to be in beta by Q3 of 2018, provided there are no substantial challenges with feasibility implementation.

Towards the close of SGA2, as Task 2 prepares for future work, the SPA will be ready for incorporation of a second service that will enable integration tests, focusing on the correctness of the service-agnostic implementation in terms of component interaction and data model. The services to be ported, their sequence, and anticipated timeline will be part of the SPA 2.0 Strategy phase.

The remaining work in GN4-2 should focus on the scalability of the solution, addressing procedures for microservices management and intra/inter security of the distributed system. Also, the main new feature to be added is a policy engine that will provide the means to implement different levels of rights of access. Another important aspect of the initial GN4-2 efforts for SPA is the training of the service management and operations teams so that the process of migrating existing services to SPA can continue. Based on the user experience with SPA v1.x, the addition of other features will be prioritised. These aspects will be resolved by the end of GN4-2, during the SPA v2.0 Strategy phase in Q3 of 2018.

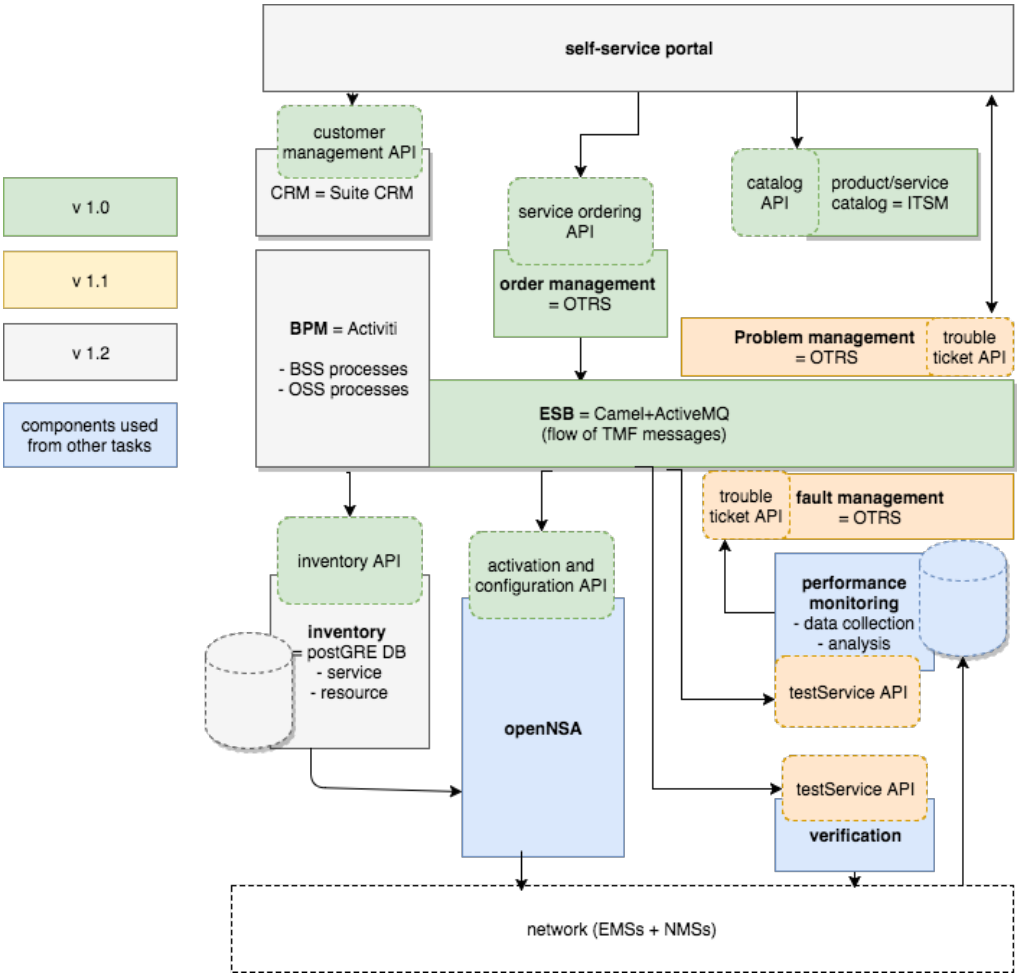

Figure B.1: SPA component view

Appendix C Task 3 Roadmap for GÉANT Testbeds Service (GTS)

GTS began design and development in GN3plus (April 2013) and made its debut in Q3 of 2014 with Version 1 launching in Bratislava, Ljubljana, Amsterdam and London. In 2015, Version 2 was deployed with three expansion sites added in Hamburg, Prague and Milan. In 2016, as GN4-1 was concluded, Version 3 was deployed and two additional sites were added in Madrid and Paris. In July 2017, GTS Version 4 was made available to users in 8 locations (Ljubljana was decommissioned due to backbone limitations). GTSv4 represents a major evolutionary step for GTS, migrating the service from its initial demonstration infrastructure to the GÉANT core MPLS infrastructure, upgrades in interface capacity to 10 Gbps, introduction of advanced and fully virtualised SDN switching resources, and delivering bare metal servers as a major new user resource. In Q4 2017, the GTS development team began completing the operational integration of the new servers, addressing reliability and performance issues associated with the circuit migration to the core routers, and preparing a number of remaining features in upcoming minor releases of Version 4.

After its currently deployed version (GTSv4.1) there will be a follow-up version, **GTSv4.2**. Its main features will be a) QoS/performance guarantees on transport circuits, b) an improved and more scalable Internet Access Gateway (IAGW) functionality for bare metal servers (BMS), and c) multi-domain (MD) environments. The IAGW for each project allows users to access the testbed infrastructures via a public IP address and VPN.

Perhaps the most critical feature of all of GTSv4 is the ability to instantiate GTS “links” that have real and enforced pseudowire-like performance guarantees. With GTSv4, “link” resources are established by requesting virtual circuits from the Consolidated Connection Service (CCS).

As of Q4 2017, CCS started pursuing QoS using OTN transport technology instead of MPLS hardware; this is now expected to be available in Q2 or Q3 2018. The positive control over capacity allocation provided by the QoS becomes a critical component of GTS version 4, now that GTS has an aggregate capacity of some 900+ Gbps available directly to user testbeds. Other minor features will also be introduced with v4.2 and 4.3, such as simplified DSL specifications; enhanced, user-based project administration privileges; and various bug fixes.

The central theme for **GTSv5.0** will be a focus on features needed to promote GTS as a sustainable service of GÉANT. Two important features along this trajectory are 1) checkpoint/restart migration/grooming, and 2) policy engine.

Checkpoint/restart (c/r) allows a user to take a snapshot of their testbed’s state, save that state to persistent storage, then at some later time remap the virtual environment to another equivalent set

of infrastructure, and restart the testbed from the saved snapshot. This feature requires a notification protocol to be developed to inform the testbed application of an impending checkpoint process (in order to allow the application to enter a quiescent state), and a high-performance distributed storage facility to be integrated across the GTS core in order to capture the state and redistribute the state files to their new locations in preparation for restart. The GTSv5 checkpoint/restart feature is expected to require approximately 30–50 TBytes of high-speed storage in each GTS location (about 250-400 TB in total).

A related feature to checkpoint/restart is active modification (AM). AM allows the structure of an active testbed to be modified without requiring a complete release and new reservation of the entire testbed. In order to simplify a highly complex modification process, AM will add or delete only a whole resource instance to/from a composite resource object. (GTS testbeds are constructed as hierarchical assemblies of composite objects, so "testbeds" are, in fact, just the root node in a tree of composite resource objects.) In order to perform these changes to a running distributed environment (i.e. a testbed), the testbed itself must be notified of an impending change and allowed sufficient time to halt or idle the processes on affected components. It is this notification process that is also required for checkpoint/restart.

From the service provider's perspective, checkpoint/restart enables resource migration/grooming (m/g). With m/g, the GTS service provider can save all or part of a running testbed and remap the testbed in order to vacate some set of infrastructure components, or to gather a set of sparsely distributed resources onto a single infrastructure component. These actions may be necessary in order to perform maintenance on hardware, or to free up partially utilised infrastructure for other purposes, or to perform green energy-saving power-down processes, etc.

Testbeds – or, more realistically, virtual networked environments – typically require substantial setup, configuration and tuning over time. This makes users reluctant to release testbed resources when the testbed is not actively being used because the time and effort needed to recreate the runtime environment are substantial. The c/r feature is crucial to support user release of resources since it makes restart an easy process. This allows the user to apply their allotted quota more efficiently over the life of their project and allows the provider's hardware investment to efficiently serve more users before capacity upgrades become necessary.

GTSv5 will also deliver a policy engine feature that will enable service providers to more effectively define the rules and limits of resource allocation across their user base. The policy engine will define actors, roles, functions and privileges associated with authorisation policy, and will define policy check functions, and relevant data object specifications on which the policy checks will be based (e.g. rollup of accumulated resource-hours.)

As reported in *Deliverable D8.9 GÉANT Testbeds Service 6.0* [D8.9], checkpoint/restart, migration/grooming, active modification and the policy engine will now feature in GTSv7.0.

Meanwhile GTSv5, as released, introduced a new, two-step registration process and a restful API that provides access to perform testbed actions (such as reservations, activations, deactivations, releases and queries) via API calls (just like via the GUI) [D8.9].

A further feature planned for GTSv6 was a graphical drag and drop testbed editor. The initial plan was to adapt the jFED tool developed within the Fed4FIRE project to utilise the GVM API [jFED,

FED4FIRE]. Such a graphical editor must be able to edit complex hierarchical composite resource objects, generate the corresponding DSL, and submit that DSL file for Reservation and Activation. The graphical layout metadata must be stored with the DSL description in order to maintain graphical rendering of the testbed from one editing session to another. Further, implementation of active modification (partial manipulation of a testbed service tree), graphical navigation through the testbed topology structure, real-time monitoring, rollback features, etc. result in a complex development and implementation of the graphical editing capability. As reported in [D8.9], the drag and drop editor was included in GTSv6.0. There are two implementations: the iFED standalone editor, and the Drag'n'DrEd web GUI developed specifically for and by GTS.

In the remaining project period the focus will be on transitioning GTSv6.0 into production.

The next software version (GTSv7.0) will consider checkpoint/restart, migration/grooming, active modification and the policy engine, as mentioned above, as well as simplified installation processes, containers and contextualisation, support for composite libraries, and improved integration with SFA testbeds.

Appendix D Task 4 Roadmap for Performance Verification and Monitoring (PVM)

JRA2 Task 4 aims to develop a generalised but comprehensive network monitoring capability that will:

- Measure the performance of GÉANT network services.
- Provide real-time feedback to network operations personnel or users.
- Determine whether those services are performing to spec and, if not, initiate an automated analysis to localise the fault and notify the appropriate agent to take corrective action.

This network performance verification and monitoring (PVM) capability must function within the virtualised network transport service context and support emerging non-traditional service objects, such as virtualised network service functions, and do so in a multi-domain environment.

JRA2 T4 will focus the design of the network PVM capability to support the following types of network services: point-to-point Ethernet-framed connections based on NSI common service definitions (e.g. CCS), Ethernet-over-MPLS and MPLS-based VPNs (multi-point L3 and L2 VPLS and MDVPN), chained service functions, non-standard SDN switched circuits and networks. It will focus on monitoring key network service performance parameters of a service in production as defined in Y.1540 and Y.1541 specifications.

The PVM approach will adapt and use both active monitoring methods, capturing the traffic and metadata (e.g. timestamps, payload signature, etc.) on monitoring zone border points, and near real-time performance analysis from the captured flow data. The amount of captured traffic will be balanced in such a way that system scalability, measurement accuracy and security/privacy are achieved.

Key PVM feature sets and the release roadmap and timeline consist of the following:

- PVM v1.0 – Design and development M1 – M22 (February 2018).

  PVM v1.0 includes the following features: performance verification and fault localisation of basic network services (L2 and L3 point-to-point and multi-point VPNs), per-service instance monitoring and SLA verification, GUI/dashboards for monitoring parameters, integration with the service inventory and service provisioning OSS components.

  Key milestones:

  - Milestone 4.1 – Advanced Service Monitoring/Performance Verification Architecture (ASM/PVA) – Month 9 (January 2017) – completed
  - Milestone 4.2 – Prototype for the measurement of large-scale network infrastructure elements – Month 12 (April 2017) – completed
  - PVM prototype demonstration – Month 13 (May 2017) – completed
- PVM v1.0 – Pilot Transition M22 – M26 (June 2018) – completed

  PVM 1.0 will be deployed in a limited set of locations – specifically those associated with the GTS service footprint. This is due to the availability of hardware consistent with the measurement point placement described in the PVM architecture.

  As reported in *D8.6 Network Monitoring / Performance Verification Architecture Production Service* [D8.6], for the remaining project period the focus is on the transition of PVM v1.0 into production. It is installed, fully operational and tested in the GTS testbed environment, and by the end of 2018 will also be deployed in the GÉANT core network.
- PVM 1.1 includes the rest of the features that are defined by the description of work: integration with other OSS/BSS components, the implementation of 100G capturing hardware for high-speed (100G) link monitoring and monitoring chained services. This version will include a module to be integrated with SPA to address ELINE service verification and initial automated fault analysis/notification.
- PVM v1.1 – Design and development will begin M21 (January 2018) through M29 (September 2018) – ongoing
- PVM v1.1 – Pilot phase M30 – M32 (December 2018). PVM v1.1 is not expected to meet production capabilities until GN4-3.
- PVM v2.0 will go through a strategic review in 2018 Q3 to identify specific features, and is expected to enter development by the end of SGA2. An important feature will be enhanced verification capability for the CCS/ELINE services. CCS/ELINE will be offering Ethernet-framed point-to-point and multi-point services provisioned with both performance guarantees and best effort, across both MPLS and OTN infrastructure, in both a GÉANT-only edge-to-edge mode and in a multi-domain end-to-end mode, and up to 100 Gbps/flow. Support for 100G performance verification, monitoring and fault analysis across all GÉANT CCS end points and legacy IP ports will be part of the strategic requirements of v2.0.

References

[5GPPP-Arch]	EU 5G PPP Architecture Working Group, *View on 5G Architecture*, Version 2.0 18-07-2017 https://5g-ppp.eu/wp-content/uploads/2017/07/5G-PPP-5G-Architecture-White-Paper-2-Summer-2017_For-Public-Consultation.pdf
[BOSSF]	Jithesh Sathyan, *Fundamentals of EMS, NMS, and OSS/BSS*, CRC Press, 2010 Jithesh Sathyahttps://www.crcpress.com/Fundamentals-of-EMS-NMS-and-OSSBSS/Sathyan/p/book/9781420085730n, Fundamentals of EMS, NMS, and OSS/BSS, CRC Press, 2010
[CCV]	*White Paper OSS Systems Need to Evolve From Network Management to Customer Centric Service Management*, Cisco, 2012 http://www.cisco.com/c/dam/en/us/products/collateral/cloud-systems-management/prime-fulfillment/hr_cisco_customer_centric.pdf
[D8.1]	*Deliverable D8.1 Proposed Integrated Services Framework and Network Services Development Roadmap* https://www.geant.org/Projects/GEANT_Project_GN4/deliverables/D8.1_Integrated-Services-Framework.pdf
[D8.6]	*Deliverable D8.6 Network Monitoring / Performance Verification Architecture Production Service* https://www.geant.org/Projects/GEANT_Project_GN4/deliverables/D8.6_Network-Monitoring_Performance-Verification-Architecture-Production-Service.pdf
[D8.9]	*Deliverable D8.9 GÉANT Testbeds Service 6.0* https://www.geant.org/Projects/GEANT_Project_GN4/deliverables/D8.9_GE%CC%81ANT-Testbeds-Service-6.0.pdf
[eTOM]	https://www.tmforum.org/business-process-framework/
[ETSI]	http://www.etsi.org/
[ETSI NFV]	http://www.etsi.org/technologies-clusters/technologies/nfv
[ETSI NFV SPEC]	"Network Functions Virtualisation (NFV); Management and Orchestration" ETSI GS NFV-MAN 001 V1.1.1 (2014–12)
[FED4FIRE]	https://www.fed4fire.eu/
[GÉANT LAMBDA]	https://wiki.geant.org/pages/viewpage.action?pageId=53117504
[GTS]	https://www.geant.org/Services/Connectivity_and_network/GTS/Pages/Resources.aspx
[GTS VIRTUALISATION]	https://www.geant.org/Services/Connectivity_and_network/GTS/Documents/M6-1_Generalised-Virtualisation-Model-v2016_v02.pdf
[HSFC]	D. Dolson, S. Homma, D. Lopez, M. Boucadair, Hierarchial Service Function Chaining https://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjk1aSK2L7YAhXB2KQKHZlIBCsQFggpMAA&url=https%3A%2F%2Ftools.ietf.org%2Fhtml%2Fdraft-ietf-sfc-hierarchical-05&usg=AOvVaw3CkdBOpd_N3PZKyO99uGjT

[IETF RFC 4448] L. Martini, ed., *Encapsulation Methods for Transport of Ethernet over MPLS Networks*, April 2006 https://datatracker.ietf.org/doc/rfc4448/

[IETF RFC 7665] J. Halpern, C. Pignataro, eds., *Service Function Chaining (SFC) Architecture*, October 2015 https://tools.ietf.org/html/rfc7665

[IETF RFC 8345] A. Clemm, J. Medved, R. Varga et al., *A YANG Data Model for Network Topologies*, March 2018 https://tools.ietf.org/html/rfc8345

[IRTF_NVRC] C. J. Bernardos et al., *Network Virtualization Research Challenges*, draft-irtf-nfvrg-gaps-network-virtualization-10, September 2 2018 https://datatracker.ietf.org/doc/draft-irtf-nfvrg-gaps-network-virtualization/

[ITU-1540] "Internet protocol data communication service – IP packet transfer and availability performance parameters, ITU-T Recommendation Y.1540", March 2011.

[ITU-1541] "Network performance objectives for IP-based services", ITU-T Recommendation Y.1541, December 2011.

[jFED] http://jfed.iminds.be/

[L3-VPN] https://www.geant.org/Services/Connectivity_and_network/Pages/VPN_Services.aspx

[M8.1] (Link for GN4-2 project participants) https://intranet.geant.org/gn4/2/Activities/JRA2/Milestones%20Documents/Consolidated%20Connection%20Services%20Roadmap/M8.1_Consolidated-Connection-Services-Roadmap.pdf

[M8.2] (Link for GN4-2 project participants) https://intranet.geant.org/gn4/2/Activities/JRA2/Milestones%20Documents/Service%20Provider%20Architecture%20Roadmap/M8%202_Service-Provider-Architecture-Roadmap.pdf

[M8.3] (Link for GN4-2 project participants) https://intranet.geant.org/gn4/2/Activities/JRA2/Milestones%20Documents/G%C3%89ANT%20Testbeds%20Service%20Roadmap/M8.3_GE%CC%81ANT-Testbeds-Service-Roadmap.pdf

[M8.4] (Link for GN4-2 project participants) https://intranet.geant.org/gn4/2/Activities/JRA2/Milestones%20Documents/Network%20Monitoring%20-%20Performance%20Verification%20Architecture%20Roadmap/M8.4_Network-Monitoring-Performance-Verification-Arch-Roadmap.pdf

[MANO] https://osm.etsi.org/

[MEF-10.3] *MEF Technical Specification* https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_10.3.pdf

[NML] Network Markup Language Working Group https://redmine.ogf.org/projects/nml-wg

[NSI] https://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwiBrZLB8b7YAhWRGuwKHYWFCVkQFggtMAE&url=https%3A%2F%2Fwww.ogf.org%2Fdocuments%2FGFD.212.pdf&usg=AOvVaw0rvM-Y1U5PukD-Jyb-Jo77

[OGF-NSI-CSv2]	https://www.ogf.org/documents/GFD.212.pdf
[ONOS]	https://onosproject.org/
[OpenNSA]	https://github.com/NORDUnet/opennsa
[SOA]	SOA Blueprint – SOA Practitioners Guide http://www.soablueprint.com/practitioners_guide
[TMF]	https://www.tmforum.org/
[TMFFx]	TM Forum Frameworx https://www.tmforum.org/tm-forum-frameworx/
[TMF_ETSI_SPEC]	**"**Hybrid Infrastructure Platform (HIP) Implementation and Deployment Blueprint", TMF070A, Release 17.0.1 , November 2017 'Connectivity Patterns for Virtualization Management', IG1147, Release 16.5.1, March 2017
[TMFSID]	https://www.tmforum.org/information-framework-sid/
[TMFZOOM1]	https://www.tmforum.org/collaboration/zoom-project/
[TMFZOOM2]	https://www.tmforum.org/resources/technical-report-exploratory-report/tr235-zoom-policy-model-and-architecture-snapshot-r14-5-1/

Glossary

5GPPP	5th Generation Infrastructure Public Private Partnership
AAI	Authentication and Authorisation Infrastructure
AM	Active Modification
API	Application Programming Interface
ASM	Advanced Service Monitoring
AtOM	Any Transport over MPLS
BoD	Bandwidth on Demand
BMS	Bare Metal Server
BSS	Business Support System
c/r	checkpoint/restart
CCS	Consolidated Connection Services
CCV	Customer-Centric View
CLI	Command-Line Interface
CPU	Central Processing Unit
CSP	Communication Service Provider
CY	Calendar Year
DC	Datacentre
DSL	Domain Specific Language
DSP	Demand Side Platform
ED	External Domain
EoMPLS	Ethernet over MPLS
EPL	Ethernet Private Line
eTOM	enhanced Telecom Operations Map (TM Forum Business Process Framework)
ETSI	European Telecommunication Standards Institute
FCAPS	Fault, Configuration, Accounting, Performance and Security
FIRE	Future Internet Research and Experimentation
FPGA	Field Programmable Gate Array
GE	Gigabit Ethernet
GENI	Global Environment for Network Innovations
GFP-T	Generic Framing Procedure – Transparent
GN3plus	GÉANT Network 3 plus, a project part-funded from the EC's Seventh Framework Programme under Grant Agreement No.605243
GN4-1	GÉANT Network 4, Phase 1 project part-funded from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No.691567
GN4-2	GÉANT Network 4, Phase 2 project part-funded from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No.731122
GPU	Graphics Processing Unit

GTS	GÉANT Testbeds Service
GUI	Graphical User Interface
GVM	Generalised Virtualisation Model
HIP	Hybrid Infrastructure Platform
I/O	Input/Output
IAGW	Internet Access Gateway
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISG	Industry Specification Group
ITU	International Telecommunication Union
JRA	Joint Research Activity
JRA1	Joint Research Activity 1 Network Infrastructure Evolution
JRA1 Task 3	JRA1 Task 3 Integration with Other e-Infrastructures
JRA2	Joint Research Activity 2 Network Services Development
L*n*	Layer *n* (OSI model)
LAN	Local Area Network
LSP	Label-Switched Path
M	Month
m/g	migration/grooming
MANO	Management and Orchestration
MD	Multi-Domain
MDVPN	Multi-Domain Virtual Private Network
MEF	Metro Ethernet Forum
MPLS	Multi-Protocol Label Switching
NFV	Network Function Virtualisation
NIC	Network Interface Card
NML	Network Mark-up Language
NMS	Network Management System
NREN	National Research and Education Network
NSC	Network Service Chaining
NSI	Network Service Interface
ODL	OpenDaylight
OGF	Open Grid Forum
ONOS	Open Network Operating System
OSS/BSS	Operations Support Systems/Business Support Systems
OTN	Optical Transport Networking
OVS	Open Virtual Switch
P*n*	GN4-2 Project Period *n*
P2P	Point to point
PCE	Path Computation Element
PEB	Policy Enforcement Block
PG	Performance Guaranteed
PoP	Point of Presence
PVA	Performance Verification Architecture
PVM	Performance Verification and Monitoring
Q	Quarter

QoS	Quality of Service
R&E	Research and Education
RCA	Resource Control Agent
RCA-VC	Resource Control Agent Virtual Circuits
REST	Representational State Transfer
RFC	Request for Comments
SBI	Southbound Interface
SDN	Software-Defined Network
SFA	Slice-based Federated Architecture
SGA	Single Grant Agreement
SID	Shared Information Data (TMF Information Framework)
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
SPA	Service Provider Architecture
SSD	Solid-State Drive
SSH	Secure Shell
ST	Storage
T	Task
TAM	Telecoms Application Map (application framework)
TLS	Transport Layer Security
TMF	TM Forum
VC	Virtual Circuit
VIM	Virtual Infrastructure Management
VLSI	Very-large-scale integration
VM	Virtual Machine
VNF	Virtual Network Function
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding
VRT	Virtual Radio Telescope
VSI	Virtual Switch Instance
WAN	Wide-Area Network
ZOOM	Zero-touch Orchestration, Operations and Management