11-07-2018

Deliverable D8.9 GÉANT Testbeds Service 6.0

Deliverable D8.9

Contractual Date:	30-06-2018
Actual Date:	11-07-2018
Grant Agreement No.:	731122
Work Package/Activity:	8/JRA2
Task Item:	Task 3
Nature of Deliverable:	OTHER demo
Dissemination Level:	PU (Public)
Lead Partner:	DFN (FAU)
Document ID:	GN4-2-18-484750
Authors:	J. Sobieski (NORDUNET), S. Naegele-Jackson (DFN/FAU)

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
From May 2016 – June 2018, the development of the GÉANT Testbeds Service (GTS) in the GN4-2 project has now evolved into version 6.0, which is transitioning into a production service. The following document describes the GTS service elements, the service benefits, its user community, technology and architecture, as well as the operational aspects and the next version roadmap.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

The GÉANT Testbeds Service (GTS) has been put in place to offer researchers virtual environments where they can rapidly develop technological innovations without having to go through lengthy processes to put their testbed network in place across Europe. A researcher running an experiment can simply build a test topology by selecting the required resources from a graphical user interface. The system then sets up an automatically provisioned and isolated virtual network for the user and the tests and experiments can begin.

During GN4-2 GTS has evolved from version 3.1 to its latest software release of version 6.0. In contrast to previous GTS versions that were immediately made available to users within the GTS pilot environment by the GTS development team, this new release of GTS version 6.0 will be deployed by the GÉANT operations team for the first time and will be deployed as an official GÉANT production service in Q4 of 2018. During this transition period from pilot service to production service, new operations and support teams, service metrics and service policies will be identified.

This document describes *the software release of GTS version 6.0* that will be the basis for the service transitioning into a production service. The main new feature of GTS version 6.0 is the "Drag'n'DrED editor" that allows users to develop network topologies not by developing code, as in previous versions, but by using their mouse to select network objects and link them by dragging the mouse from one resource object to the next. The corresponding code is automatically generated for the user in the background. The new version also offers the selection of resource attributes and location parameters via drop down menus. Such visual editing will facilitate the introduction of novice users to GTS and will greatly shorten the required learning phase.

1 Introduction

GTS is a revolutionary service in GÉANT that allows users to set up wide-area virtual networks via a web-based, automatic provisioning system. The virtual networks are built within minutes over the underlying physical infrastructure, and thus allow the researcher to carry out experiments with great flexibility and enable rapid prototyping and innovation. The individual network environments are isolated from other users and are assembled following the researcher's topology requirements. Currently, a researcher can select GTS virtual machines (VMs), virtual links (VCs), Virtual Software Defined Networking Switch Instances (VSIs) and Bare Metal Servers (BMSs) as network resources in the virtual network. The architecture of GTS is extensible and can be expanded any time to offer new types of resources. A user can also connect external facilities to such automatically provisioned environments or use GTS as connecting fabric between distributed labs. This feature of geolocation is also what distinguishes GTS from cloud services and is the reason why these virtual environments are called networks – because the researchers can place these resources where they need them across Europe.

This document describes the GTS service and its underlying features, as they evolved, up until version 6.0 (Section 1). Section 2 outlines the GTS user community and how users benefit from the service. Section 3 provides a detailed look at the GTS framework and its underlying Generalised Virtualisation Model (see also deliverable *D8.1 Integrated Services Framework and Network Services Development Roadmap*) [D8.1]. Section 3 is followed by a description of GTS service operation and in section 5 the reader finds a brief outlook on the upcoming developments of the service as planned for GTS v7.0. The document ends with a conclusion in section 6.

1.1 The GÉANT Testbeds Service (GTS)

The development of the GÉANT Testbeds Service (GTS) [NAE18, HAZ14] started in April 2013, when the need for such a service became evident and it was decided to start development in the GÉANT project [GEA18]. Researchers in the GÉANT community greatly benefit from the GTS service platform, as it allows them to define individual virtual network environments that can be compiled and built in seconds by the user himself, using an automated provisioning system. Each virtual network is isolated from other testbeds and services in the network so that critical(e.g., security related) and untested new applications can easily be conducted in the experiment without having to worry about interference from other users or interfering with other users. This is one of the strengths of this service: It allows flexible experimentation with adaptation of the underlying network topology. Another special feature of GTS is that these automatically provisioned, virtual networks are not simulations, but are run over dedicated parts of the physical infrastructure so that researchers can obtain realistic analysis on network behaviour and network traffic. As the GTS physical infrastructure is spread across Europe, the service allows set-up of virtual networks in a Wide Area Network

environment. Currently, GTS has eight Points of Distribution (PODs) in Amsterdam, Bratislava, Hamburg, London, Madrid, Milan, Paris and Prague.

To build a virtual network in GTS, the user first describes the topology of the network and what resources it should contain. This is done by means of a document, which can then be uploaded through a web interface (GUI). The server-side resource manager agent (a software component) receives this document, checks it for syntax and availability of requested resources. If all resources are available and the request can be granted, the resource manager reserves the resources and returns identifiers of the network components to the user so that he can activate and control the resources in his testbed.

The document mentioned above for describing a virtual network includes Domain Specific Language (DSL) code [DSL] based on Groovy (an object-oriented language for the Java platform) [GRO18]. For advanced users, this has the advantage that very complex and large networks can be constructed with iterations in a very quick and easy manner. In addition to this baseline construction process, GTS version 6.0 includes a web-based Drag'n'DrED (drag and drop editor) GUI for easy testbed creation, which is especially beneficial to GTS newcomers, as this editor allows a user to use pull down menus for resource selection and configuration, and then enables the researcher to click on the resource objects, drag and connect them to other resources on a canvas. While the researcher is designing his or her testbed using the mouse, the associated DSL code is automatically created in the background. This feature was demonstrated at TNC18 in June 2018.

This innovative GTS service has been deployed and is currently available for general user access as a GÉANT pilot in GTSv5.0 – maturing the usability and functionality aspects, while simultaneously developing expertise within the GÉANT operations and service management teams in design and operational support aspects of virtualised services. This report describes the release of the GTS Software Suite in version 6.0 which will be deployed into the GTS production service in Q4 2018. With the launch of version 6.0 in Q4, GTS will be transitioned into a fully supported production service offered by GÉANT (no longer a "pilot") and its deployment will be completed by the GÉANT Operations team.

1.2 GTS v6.0 Service Description

The user accesses the service via the GTS website [GTS18] and logs in. To build a network, the user formally describes all resource components and the topology in a DSL document and submits it. The resource manager agent then parses the document and allocates the requested resources to the new network. The user then receives all necessary resource ID information so that s/he can control the network via the GTS Graphical User Interface (GUI) and API primitives (Figure 1.1).

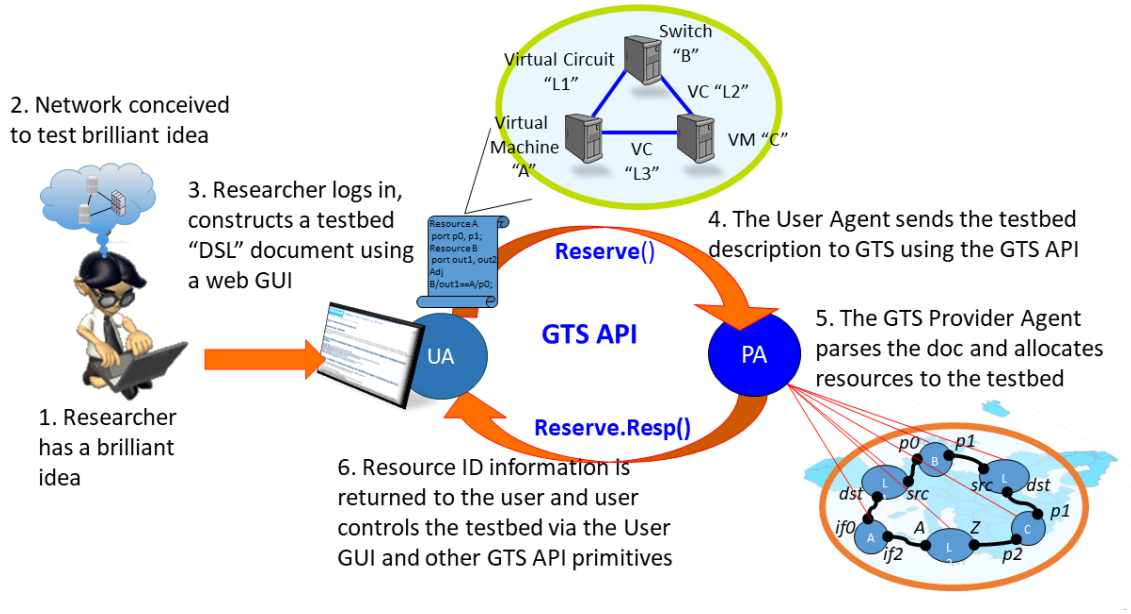

Figure 1.1: Using the GTS service

The service architecture treats all networks as graphs and all network components are generalised, virtualised resources that have explicitly defined data flow ports. The network topology is defined by adjacency relations that specify which ports are connected to one another. Each network is represented by a graph model using three basic constructs: resources, ports and adjacencies [FAR14].

In principle, any type, or "class", of resource can be made available in the service as long as the appropriate Resource Control Agent (RCA) [D6.1] software has been developed to properly configure and control the resource (see Figure 1.2). Such a resource control agent must be able to define() and undefine() the resource class, i.e. install a resource into the network project's library; it must also have a control primitive to reserve() and release() a specific instance of the class, and primitives that place that instance into service with activate() and deactivate() control statements. Finally, each resource also needs a primitive to query() its state.

The main software components that define the service at this time are: an RCA-OS, a resource control agent to manage virtual machines with OpenStack [OST18]; an RCA-BMS for bare metal servers; an RCA-VC for virtual circuits; and an RCA-OFX to handle OpenFlow [ONF12] switch fabrics. The Resource Manager (RM) constitutes the core of the software – the "server-side" - and is responsible for the distribution of resources, but also manages projects and users. The RM also manages an Internet Access Gateway created for each project so that users can access their GTS virtual network securely via a public IP address and a point-to-point private VPN link connection. Another main software component is the web GUI, which is the interface for the user to define, reserve, activate, deactivate and release resources. The GUI also provides a link for the user to access each activated virtual host for configuration.

The RM relays reservation, activation, deactivation and release requests to the appropriate Resource Control Agents. The RCA-OS, for example, can reserve a VM in a future schedule; for activation of a VM it will translate the common GTS API primitives into OpenStack specific commands, and then contact OpenStack to either activate a new VM or reactivate a VM (in the case of an existing VM). A virtual machine is deactivated by the RCA-OS with an OpenStack suspend() statement and is delete()'d

when it is supposed to be released. A new VM is created with OpenStack Neutron (create network) OpenStack Nova (create server: Flavor, Image and VNC console); and ports are connected with command line interface tool, *virsh* [virsh].

The RM could also be viewed as the resource controller for "composite" resources. A testbed is essentially just a high-level resource object that contains other resource objects, i.e. a testbed is a "composite" resource. Composite resources can be defined for many different applications, and the user's testbed is just an instance of a composite resource defined in the DSL. A composite resource may also contain other composite resources. Such hierarchical construction allows for effective modularisation of sub-assemblies or custom testbed components, which can then be duplicated to produce large more complex networks or service graphs.

The GTS development has resulted in a Generic Virtualisation Model (GVM) [D6.1] that defines the common set of API "primitives", defined above, to be used to manage all GTS resources through their respective lifecycles. The RCA modules translate these common GTS primitives into the southbound hardware or infrastructure-specific command sequences necessary to have an effect on the GTS function in other third-party codes. Thus, for instance, RCA-OS could easily be adapted to function with an underlying VMware-based facility as an alternative to the OpenStack infrastructure. This resource abstraction (Figure 1.2) means that GTS resources are technology agnostic and makes the GVM services highly adaptable to a wide range of existing resources or future requirements.

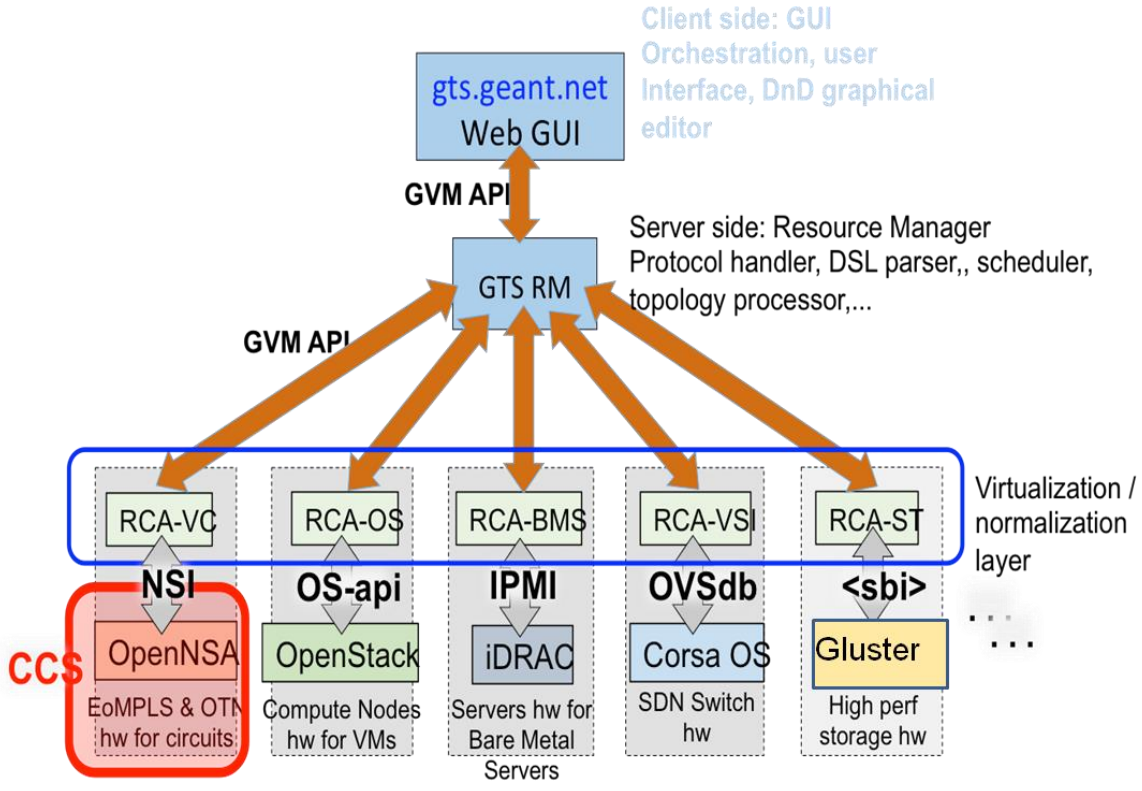

Figure 1.2: Resource abstraction

The RCA-VC is responsible for handling all link (virtual circuit) resources, i.e. virtual links are reserved, activated, deactivated and released with underlying OpenNSA [ONSA18] software via the NSI standards interface for provisioning multi-domain connections. These links/connections are able to extend across multiple service provider domains to link resources distributed across multiple service providers into a single, integrated and insulated, user-defined virtual network environment. GTS calls these environments "testbeds".

Users that would like to work with OpenFlow can currently book (first come first served) virtual SDN switch resources provisioned over a CORSA DP-2100 hardware switch [COR18, CZI16], and the RCA-VSI will manage such fully virtualised switching instances (VSIs). The RCA-VSI also supports software-based SDN switching objects with Open VSwitch (OVS). This is an excellent example of GTS adaptability, as service providers or researchers wishing to deploy an SDN-capable GTS facility can do so with standard commodity servers and then upgrade that infrastructure to hardware SDN switches that can seamlessly enhance the VSIs to support 100Gbps throughput.

Also noteworthy is the VSI capabilities of GTS. As a fully virtualised resource model, the user can define VSIs with multiple ports - even exceeding the number of physical ports on the device itself. Each virtual VSI port is mapped to virtual circuit information (not physical ports), which allows the virtual model to support both high-degree switching elements and very low-degree switching elements (of two or even just one port). This does not affect the throughput of the VSI, and the VSI is provisioned to support the maximum capacity required. .

The user executes all control primitives via the GUI. Up to and including GTSv5.0 a user had to set up a network topology by submitting a DSL file. With GTSv6.0, however, a Drag'n'DrED (drag and drop editor) is available that allows users also to build a network via a drag and drop interface, where the corresponding DSL file is automatically created and submitted for processing by the editor application. This will make it easier for beginning researchers to develop DSL code. Advanced users may still find it more accommodating to write/edit DSL code files themselves, where advanced features of Groovy, e.g. programmable iterations, can be used to quickly assemble the most complex topologies.

At present, the hardware for the data plane resources of the service (VMs, VSIs, VCs, BMSs) is distributed across Europe at eight Points of Distribution (PODs), with the same type of hardware infrastructure available at each location (although the total capacity may vary across locations). The hardware components (servers, switches) are off-the-shelf equipment. These PODs communicate with and are managed by a set of software agents and user support functions that run on a set of centralised servers called the Central Server Facility (CSF). PODs also incorporate control plane access to/from the Central Server Facility (CSF). The primary GTS CSF is located in Prague and contains four servers and the VMs needed to host the GTS core service agents. With the deployment of the released version 6.0 in Q4 2018, GTS will have established a secondary CSF in London to provide a backup CSF and a means to transition smoothly from one GTS version to the next.

GÉANT Lambdas are provisioned between the GTS Core PoPs, over which the Virtual Circuits can be created. These capacity from (or even infrastructure used by) these lambdas can be shared with other services (such as the IP service) when provisioning requirements allow. GÉANT Consolidated Connection Service (CCS) software (OpenNSA) decides when this is possible, based on each circuit's service parameters [ONSA18].

For GÉANT, the best effort links (without bandwidth or performance guarantees) are shared over the Best Effort MPLS backbone of GÉANT, while QoS enforced circuits (see CCS above) are provisioned across a dedicated set of lambdas. Users can request and drive circuits up to 10 Gbps.

1.3 Technical Evolution of GTS in GN4-2

At the beginning of GN4-2 GTS was running version 3.1.1 (a minor update to v3.1), which offered basic service of virtual machines, virtual circuits based on 1Gbps connectivity via Juniper MX-80 routers on the data plane, and a rudimentary OpenFlow switching instance that could be reserved on HP 5900 OpenFlow switches.

When version 4.0 became available in summer 2017, it introduced major technical advances that users had requested over the previous years, such as 10 GE capabilities, new hardware resources called Bare Metal Servers (BMSs), and Virtual Switch Instances (VSIs) over Corsa DP2100 hardware, which replaced the previously deployed HP switch hardware and offered fully virtualised switch instances.

Version 5.0 GTS introduced a new, two-step registration process, where first-time users register via the GUI, fill in a form to provide their project information and then can either join an existing project or can become the project-owners of a new project [GUIDE]. The project owner, in turn, can add additional members to his or her project and manage user access. The new project/user registration is the only "human in the loop" process in GTS – a request for a new project must be approved by the GTS Service Management team. Once this is done – typically in less than 24 hours, the project owner is able to add new users to his/her project without further actions from the GÉANT service management team. Newly registered users receive an email to verify the activation of their accounts.

Another new feature of GTS version 5.0 was the introduction of a restful API, which is now open to application designers and system integrators and provides access to perform testbed actions (such as: reservations, activations, deactivations, releases and queries) via API calls (just like via the GUI) [API]. Therefore, user interaction with the GTS Resource Manager can be accomplished natively from user applications without the GTS GUI. This means that other possible user interfaces – or other orchestrators or GUIs – can interact with the GTS virtualised services to construct virtual networks.

GTS v6.0 offers a new Drag and Drop Editor ("Drag'n'DrED") via the GTS GUI, which makes it possible to create a network environment using the mouse and clicking on resource objects that can then be arranged and linked like on a canvas. Drop-down menus allow the selection of certain attributes that can then be associated with the resources, such as the selection of certain hardware properties (CPU, RAM size for VMs, certain images for operating systems, location parameters for resources, etc.) (Figure 1.3). The corresponding DSL code is automatically generated in the background for the user to save, use or adjust, as necessary (Figure 1.4). The code can be made visible in parallel to editing the testbed topology with a simple mouse-over each object. This makes it very easy and convenient, especially for beginning users, to generate correct DSL code [Drag'n'DrED].

Figure 1.3: Drag'n'DrED editor and dropdown menus

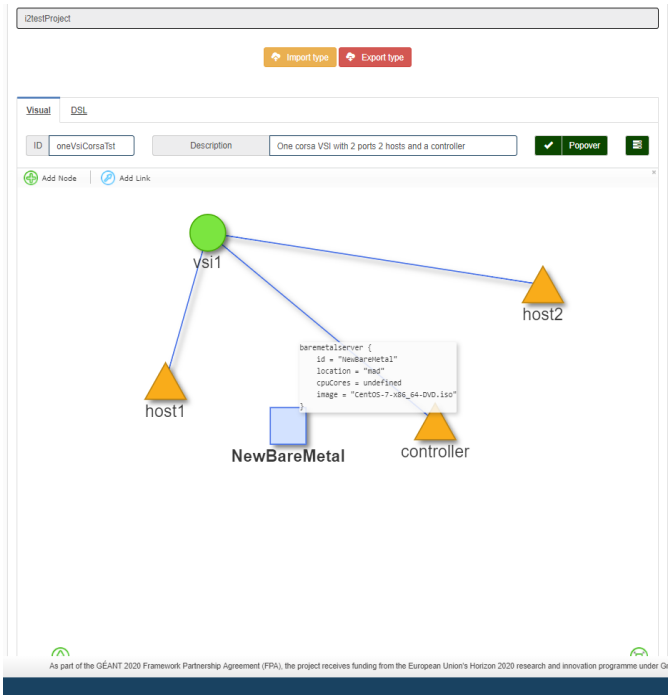

Figure 1.4: Mouse-over and automatic DSL generation

An overview of GTS version evolution in GN4-2 is listed in Table1.1 below:

GTS v.3 (May 2016)	New in GTS v.4 (July 2017)	New in GTS v.5 (March 2018)	New in GTS v.6 (July 2016)
1 GE data plane MX-80	10 GE data plane MX-960	Restful public API	Drag'n'DrED editor with automatic DSL code creation
Switch instances (semi virtualized) based on HP 5900 hardware	Fully virtual switch instances based on CORSA DP2100 hardware	Two-step registration process	
Virtual machines Virtual circuits	Bare Metal Servers	New user role of project owner	Drop down selection menus of attributes

Table 1.1: Overview of GTS evolution in GN4-2

2 Service Benefits and User Communities

With such a flexible way to build European-wide, virtual network topologies in a timescale measured in minutes, GTS is not only an interesting tool for research, but there are also many operational application scenarios where GTS can benefit network administrators and network providers. With GTS and its geolocation, it is possible, for example, to test scalability and network behaviour within a WAN context. Promising new networking models can be deployed safely and evaluated “at scale”, i.e. in a real, widely distributed topology and 100Gbps line rate. Because the GTS software does not get between the user application and the hardware on which the software runs, even real-time services can be staged and their performance accurately characterised.

In addition, it is also fitting to consider aspects in connection with Quality of Service (QoS) or Queuing Mechanisms. Such customised virtual networks are also especially suitable for investigations in connection with: network security, DDOS attacks, penetration tests and VM-based, honey-net deployments [FIL17]. Known attacks and their mitigation can be targeted, investigated and their associated defense efforts can be evaluated.

2.1 Service Benefits

The pilot service has been very well accepted by researchers, as it promotes and enables rapid prototyping. Several National Research and Education Networks (NRENs) (e.g. HEAnet, NORDUnet, DFN, CESNET and soon, RENAM as well as vendors such as Ciena [CIE18] have seen the need to establish this type of service, either as part of their research efforts or within their respective national footprints (Figure 2.1).

This is also promoted by GTS’s multi-domain capability – a federation mechanism that allows experiments to scale across multiple GTS deployments, e.g. with the GTS deployment in CESNET. With multi-domain capability of the service, researchers have access to additional resources beyond what is offered in their local domains, while the complexity of the multi-domain composition of a virtual network stays transparent.

For example, if a user requests a number of OpenFlow switch instances that exceed the number of switching infrastructures locally available, the resource manager will connect an external GTS service to fulfil the request. There is a config file for the resource manager to see where other services exist, with a pointer to their resource managers. By mapping user requests to different domains according to inter-domain authorisation policies, networks requiring a very large number of resources can be supported. Such multi-domain capability is planned as part of GTS version 6.0 deployment in Q4 of 2018.

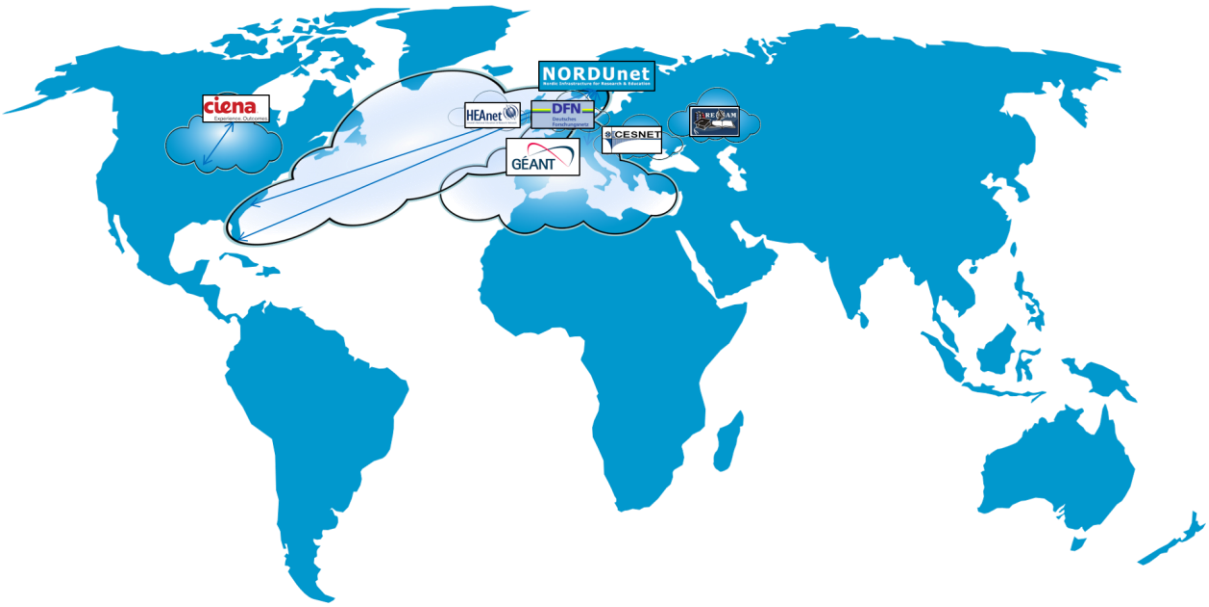

Figure 2.1: GTS deployments

With multi-domain capability and ability to use external domain interfaces (see below) GTS can act as a Software Defined eXchange "SDX" [MAM14, BHA14] point that is capable of providing a wide-range of multi-species resources and services, such as data transport circuits, switching/forwarding elements, and computational facilities with storage facilities, as well as support SDN control principles.

NRENs setting up a GTS service are also realising the benefit of using GTS to flexibly provision virtual networks from a network provider's point of view. It allows a provider to make use of existing hardware resources in the most efficient way and offers network flexibility and adaptation on demand. The fact that GTS also provides isolation between these virtual networks supports security and enables a network provider to easily set up customised environments for different applications.

The GÉANT Testbeds Service and similar services deployed by collaborating NRENs in Europe are examples of the GVM in practice. A service provider can field a GVM-compliant service with commonly requested resource classes, which integrates with a range of common hardware. In addition, the set of resource classes can be extended or more-tightly defined without changing the general model itself. The provider's engineering team only needs to plan the scope and scale of their domain's GVM-compliant service and configure the software to reflect their specific physical infrastructure to get started.

However, the GVM model in itself does not make a successful or useful service. The nature of the resources defined within the service, the quality of the virtualisation layer software – the Resource Control Agents – that creates and manages those resources, and the infrastructure available on which to realise those resources are equally, if not more important than the Generalised Virtualisation Model itself. Implementation and architecture specifics are often confused. Section 3 describes the GVM architecture and then the rest of the document addresses aspects of GTSv6.0 implementation in GÉANT.

In the GÉANT Testbeds Service it became clear over time that users wanted to connect additional facilities (e.g. external laboratories or large conference events such as the Open Networking Summit (ONS15 or ACM SIGCOMM 2015 in London as referenced in [ONS15a], [OS15b]) to their network slice

and that there needed to be a way to link such external facilities to the GTS service environment that are explicitly isolated from external networks. Therefore an "External Domain" resource was defined [SOB15] so that arrangements could be made to provision a connection from the remote target facility to a GTS- edge port using bandwidth from an out-of-band provisioning system – perhaps manual configuration.

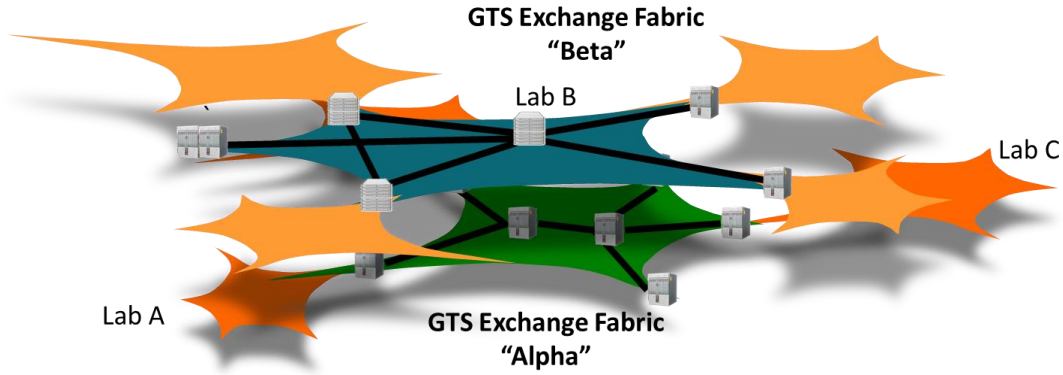

Figure 2.2: External domain connections

This means that on the GTS side, the connection is terminated on a port along the edge of the service domain, and the external facility is presented to the user as just a normal a port on a logical resource that is the external domain (Figure 2.2). With this mechanism in place, a user can allocate this External Domain class within their environment and then define adjacencies connecting that External Domain port with a conventional transport circuit within their environment. This single port for that External Domain resource represents a gateway to that facility and can be used like any other port.

Users working with such isolated virtual environments receive all resources instantiated in their initial state, which means they may need to be configured in a certain way, or user data (i.e. software packages) need to be uploaded. Since GTS provides complete isolation of provisioned network slices between each other and from the outside world, a "hidden", out of band management network is introduced for each project. This network is implicitly created with every new project and is not directly visible (its configuration is the responsibility of the operator of the GVM facility). Then, every relevant resource – usually interactive resources like VMs – within the project is connected to this network and provided with an automatically configured management access.

In order to enable access into this management network from the outside world, the Internet Access Gateway (IAGW) entity was introduced. Generally, the IAGW is a VM, providing services, such as: a VPN server, network attached storage, DHCP, and Internet gateway (default gateway; NAT in case of IPv4) for the private management network.

The exact implementation of the IAGW is out of the scope of GTS and is the responsibility of the facility administrator. However, it is a practical usability requirement of a virtualisation service and as such, a standard feature in GTS software. To connect an instantiated resource to the management network is the responsibility of individual RCAs and depends on the resource implementation in the physical infrastructure. (In future versions of GTS, the IAGW will be treated as a first-class composite resource

and the user will be required to explicitly include the IAGW if they require it and to indicate how they want it deployed within their virtual topology).

The user benefits of GTSv6.0 are summarised, as follows:

- GTS enables deployment of virtual networks with a web-based system in minutes and thus supports rapid prototyping and innovation.
- GTS allows the automatic provisioning of isolated virtual networks.
- GTS supports a wide range of technologies.
- GTS offers network flexibility with an abstraction of the hardware layer.
- GTS is scalable and can integrate any new resource type (see Resource Control Agents in Architecture Section 3.1).
- GTS allows a provider to make use of existing hardware resources in the most efficient way (as the automated network provisioning only takes minutes to complete) and offers network flexibility and adaptation on demand.
- With its external domain capability, GTS can serve as binding fabric between labs, institutions or conference locations for long-term project durations as well as for temporary, additional resource requirements.
- GTS offers easy programmability and software defined infrastructures.
- The geolocation of resources allows long distance performance tests and scalability testing within a WAN context.
- GTS offers code-based topology descriptions to create complex infrastructures with simple iterations.
- GTS also provides visual editing of topologies for beginning users with automatic code-based descriptions.
- Multi-domain capability of GTS (multiple instances of GTS) allows users to also benefit from other GTS deployments in case resources are not available locally.
- GTS provides an Internet Access Gateway for users to upload / download software.
- GTS is suitable for realistic experiments as virtual networks are set up over real hardware (no simulation or emulation).
- GTS enables production-remote policy enforcement, including application of the owner's policy to network traffic transiting the circuit.
- GTS is capable of offering resources, such as containerised service elements with network service functions, such as policers, shapers, rate limiters, encrypters, filters, etc.

2.2 GTS User Communities

The following user groups have evolved as typical users of the GÉANT Testbeds Service:

- Researchers who need a network infrastructure for a research project and would like to be able to start on their research using this infrastructure as quickly as possible without having to go through an extended and tedious setup.
    - this includes scientists who are part of the GÉANT project, who, by using this flexible service, have the opportunity to rapidly test and deploy innovative developments [DEPLOYMENT REFS].
- Computer science and networking students and graduate students who need a network environment where they can test and deploy prototypical developments for their research.
- Professors and teachers who would like to engage their students in hands-on training seminars and demonstrate the practical handling of networks beyond a simulation. GTS has already been used in classroom settings, including, at the University of Amsterdam (UVA), Netherlands, the University of Aalborg, Denmark, the Technical University of Denmark (DTU) in Copenhagen, Denmark, and the University of Göttingen, Germany.
    - By subscribing to GTS, Computer Science departments no longer need to support campus computer/network labs – the administrative support, capital investment, maintenance, and technical overhead can be absorbed by the GTS service and provided much more cost effectively.
- Network researchers who would like to perform network evaluations over long distances (for example, for performance measurements) that cannot be run in a lab and require realistic network conditions over real hardware.
- Research projects in need of a virtual network environment for demonstration in a conference/workshop setting that extends beyond their lab environment and offers additional resources.
- GN4-2 project members or network administrators who would like to test security concepts or run security modules (for DDOS mitigation, etc.) that they could not or would not want to test within their production networks.
- Network administrators who would like to evaluate innovative new services, to refine and mature new, experimental services at full scale in an isolated and insulated environment that eliminates risk to other production services.
- Research projects with partners who have their own lab environments in different places and would like to use GTS as a connecting fabric – with novel behaviour/policy - between their partners' locations.
- Production remote policy enforcement (where trans-oceanic circuits can be terminated in a foreign Open Exchange Point and the remote owner can request a virtual switching instance

at the OXP to front-end their circuit to inspect and apply the owner's policy to traffic transiting the circuit.

In order to stay in touch with its user base, GTS is regularly presented to these different communities at various national, European-wide and global events. A list of these dissemination efforts can be found in the Appendix.

3 Architecture

The Generalized Virtualization Model that GTS is based on works with abstract entities called "resources". Resources are user-facing service objects (a class or module that performs an action) defined by the service provider according to the user's base requirements. The definition of the term "user" in this case is broad. A user is any agent that requests resources from the GVM.

The first step in creating the "Service" is the definition of the resource classes. These resources are the service objects, and the service provided to the "user" is the instantiation of the requested resources, with the data flow arranged as requested.

The emergent functionality of this virtual, functional service graph is the result of the prescribed resource class behaviour, the use-specific parameterisation of the resources, and the data flows processed by these service graphs.

Different resource classes can be defined at the virtualisation layer, representing certain types of physical resources and software components called Resource Control Agents (RCAs), which are responsible for instantiating these abstract resource classes in the physical infrastructure. A resource class includes a synopsis attribute, which describes in a human readable way its functional essence. The class contains a set of parameters and primitives which are relevant for the certain type of physical resource. Taking PC-compliant Virtual Machines (VMs) as an example, parameters can include: ISO boot image, CPU clock speed, number of CPU cores, memory, operating system, storage capacity, and so on.

Resources can be classified in one of two categories:

- Infrastructure analogues such as virtual machines, virtual switches, or virtual circuits, etc.
- (Software) network functions such as policers, shapers, rate limiters, encrypters, filters, etc.

There is actually very little difference between the two. Network functions are typically implemented as "containers" (very lightweight compute elements similar to VMs) with a specific software function contextualised to run in the container when it is instantiated. On the other hand, a VM, can be considered as a compute element contextualized to run a particular ISO image (operating system). Service definition of resource classes, and their instantiation, is very similar for a wide range of resources.

As prototypes have already been demonstrated, GTS plans to offer containerised service elements in the very near future (as part of GTS v7.0). These computational resources, combined with the link resources and the ability to arrange these elements into a functional service graph, enables GTS v6.0 to construct very largescale virtual environments, such as a European-scale WAN network or very intricate virtual environments designed to process data flows with multiple steps and sub-processes all on one infrastructure platform or within a single PoP location. Therefore, GTS and its Generic Virtualization Model is a promising platform for exploring Network Function Virtualisation at full scale and at full performance.

Instances of GVM resources recognise two, stable states:

- ACTIVE – resource is provisioned and in service.

- RESERVED – resource is instantiated, but not yet in service (configured in the physical infrastructure).

As well as several transition states:

- RESERVING
- ACTIVATING
- DEACTIVATING
- RELEASING – instance is being destroyed.

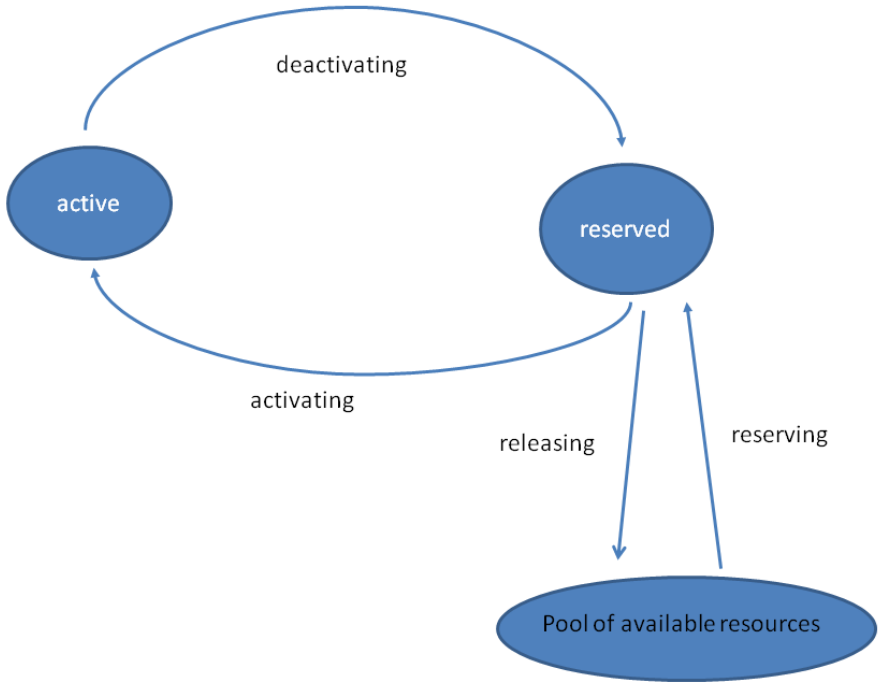

Figure 3.3: GVM states and transitions

Every resource class must implement the following five basic control primitives, which move the resource through the states of its lifecycle:

- Reserve() – instantiate the resource, allocate physical infrastructure components
- Activate() – configure the hardware to realize the resource, move from RESERVED state to ACTIVE state
- Deactivate() – move from ACTIVE state to RESERVED state
- Release() – destroy the instance
- Query() – get information about the actual state of the resource.

Depending on the actual essence or function of the particular resource, these five primitives will require administrative bookkeeping and will result in quite different southbound interface activities to accomplish the primitive function. When a VM is activated, for example, different steps need to be performed, compared to when an Ethernet Virtual Circuit (VC) is activated.

In addition to the atomic resources as explained above, GVM also supports “composite” resources, i.e. resources that contain other resources. Within this hierarchical resource structure, the resources referenced within a composite resource class definition are called “children” resources, and the encompassing composite resource is termed “parent”. The internal structure of a composite resource can contain children resources (with their ports and associated adjacency relations) and is instantiated sequentially in a nested,-recursive fashion.

It is also possible for a composite class to be referenced from within other composite classes, even in a repeated way from more complex structures, which allows the entire user’s virtual network environment to be specified as a composite resource. In other words, a user defines his/her testbed environment (as in the GTS implementation) as a composite resource and then instantiates this top-level resource to create the entire virtual network.

3.1 Functional Components for Network Slicing

A GVM-compliant GTS service has a Provider Agent (PA) that is responsible for managing the infrastructure and the virtualisation services so that the requested resources can be instantiated.

The PA receives the information on what type of resources need to be produced from a Requesting Agent (RA) (also referred to as User Agent (UA), which could be any type of software agent, but it is most often assumed to be a graphical user interface agent – perhaps a web-based graphical user interface that assists the interactive human user to define and manage his/her own virtual resources.

The Provider Agent consists of several components: the core Resource Manager (RM), the Resource Control Agents (RCAs) and the Resource Database (RDB). The core Resource Manager (RM) sits behind the URL representing the GTS service (e.g. gts.geant.net for the GÉANT service). The RM interprets primitives received from User Agents and, after authenticating the UserID credentials, forwards the request to an appropriate Resource Control Agent (RCA) for processing. The GVM API describes the interactions that occur between the RA and the PA. The most basic exchange takes place when the Requesting Agent issues one of the five API lifecycle primitive requests to the Provider Agent.

The Resource Control Agents implement these API primitives for each resource class and convert the GVM API to the infrastructure-specific actions needed to allocate, activate, deactivate, query and release each resource class. In other words, for each resource class that is introduced, there must be an RCA module that translates generalised semantics of the GVM API to the appropriate infrastructure interactions. A well-designed RCA for a particular class should be able to support multiple types of underlying infrastructure by simply having different southbound interfaces for each type of hardware infrastructure. A user should only see the generalised resource model of the requested resource, and should not be exposed to the underlying mechanisms needed to deliver that resource to the user’s environment(s) (Figure 3.1).

To establish a resource instance, the RCAs create a resource record in the RDB and allocate and reserve the infrastructure components that will be needed later to realise the resource instance. At activation time, the RCA knows how to actually provision the allocated infrastructure components in order to construct the instance and place it in service.

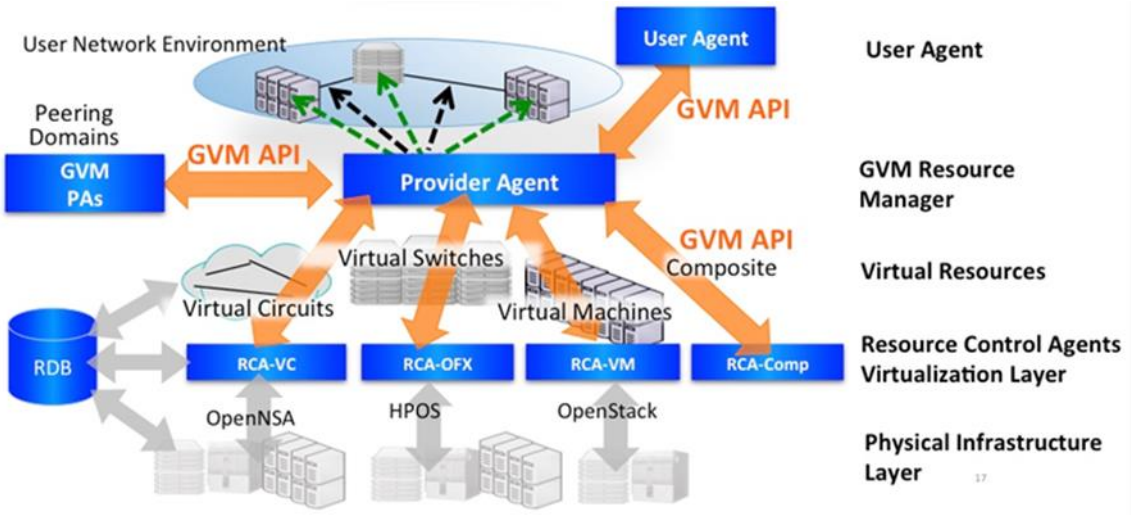

Figure 3.1: Overview of functional components and operational model

The Resource Database holds the authoritative state of the GTS service. The entries in the RDB list projects, users, class templates and resource instances, state, infrastructure, policies, reservations, etc. for the service. The RDB should be implemented as a persistent and highly reliable data store, since it describes both user resource state and state of the infrastructure supporting those user services. Restart and recovery from unexpected failures should always be possible by inspecting the RDB and comparing it to the state recovered from the infrastructure.

Key information structures that the RDB contains are: a Project Table, a Class Template Table, a Resource Tree, a Port Table and Infrastructure Tables (Figure 3.2).

The Project Table delivers all information that define a Project created within a service domain. Projects are equivalent to “accounts” and may contain a list of users authorised to reserve and manage resources attached to this project, a list of class templates defined under this project, and a list of resource instances that have been reserved.

The Resource Tree (RT) represents all the resources instantiated in a single virtual testbed or network slice. Each RT is rooted under a Project and is constructed during the recursive reservation process. Intermediate nodes stand for composite resources in the tree, and each atomic resource instance is represented as a leaf node in the tree. Specific attributes for a resource as provided in the Reserve() request are stored with the associated entry in the Resource Tree.

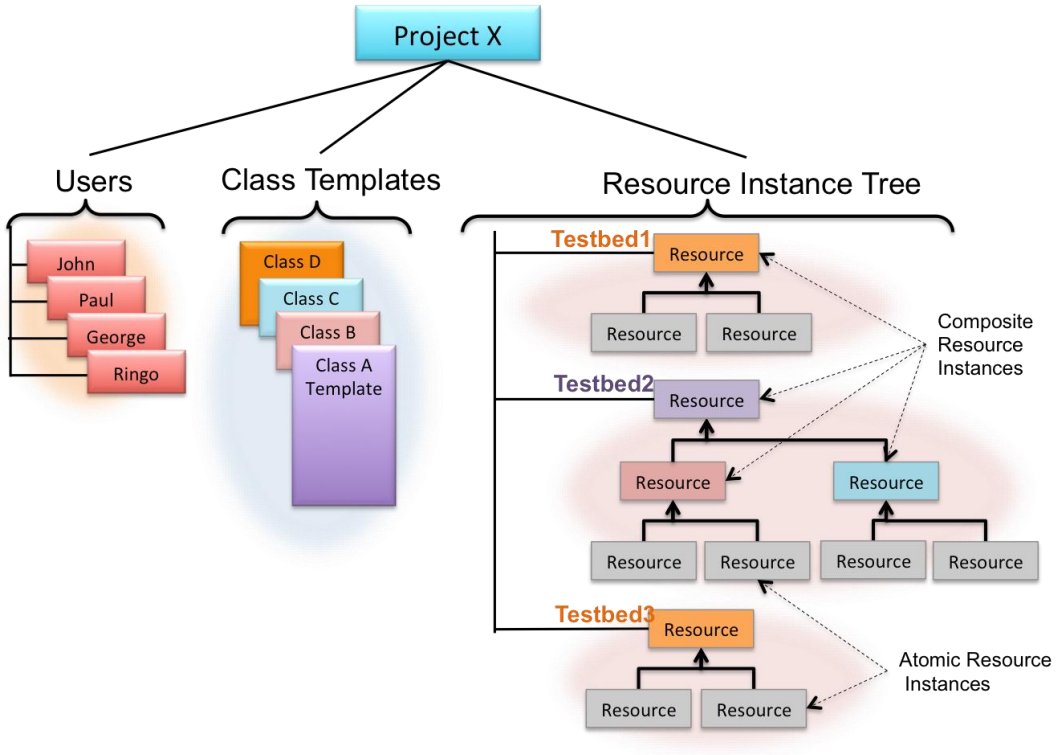

Figure 3.2: Projects, users, class templates and resource trees in GVM

For a user to instantiate a resource, it is necessary to first define the resource class. The Define() primitive sends a class description to the Provider Agent. The Resource Manager then parses the description and if the description is complete and valid, inserts the class into the Class Template Table of the Project.

The class description must include port specifications that define how data can ingress or egress a resource instance of that class. As an example, a host class may allow an instance to have 1 to 16 ports. A link class might only allow exactly two ports – a source (src) port and a destination (dst) port.

For the network topology in the virtual environment to be outlined, it is necessary to indicate which ports on each resource instance are connected or adjacent to other ports on other resource instances. Adjacencies are unordered pairs of 2-tuples

<resource_a>.<port_x>=<resource_b>.<port_y>

indicating the ports are connected to one another. Using the adjacency characteristics, it is possible to specify the topology of the resources independently of the resource instances themselves. As the adjacencies represent relationships (not data-plane links or circuits) they only define how ports are connected to one another.

For each resource instance in the Resource Tree there is a link to a Port Table, which holds a list of ports defined for that resource instance along with associated port attributes. Port and resource entries in the tables are linked bi-directionally for adjacency relations to be established.

3.2 Virtualisation versus Performance

One of the main advantages and objectives of the Generalized Virtualization Model is that it provides a way of creating a fully dynamic virtual network environment, or network *slice*, without limiting performance. Although "virtual" is often associated with a simulated, or emulated, environment unable to match the performance of real native hardware, in the context of GVM, virtualisation simply means defining the behaviour of an object such that the object is not tied to a particular piece of hardware or a particular technology This allows the GVM model to realize the object on any underlying infrastructure as long as it meets the performance guarantees of the resource class. When it makes sense for performance reasons to instantiate a virtual resource on dedicated hardware, the resource allocator will do that. If it makes sense to allocate a virtual resource in software it will do that.

The user experiences a performance level dependent on the resource types and on the user's experiment or requirements, which may vary from one application to another. How the performance is mapped to the infrastructure is the role of the service provider. This is "technology agnostic" i.e. the performance needed is specified by the user, and the service provider allocates an object that has that performance. If it is a virtual switch running natively on a hardware SDN switch or whether it is instantiated as a software switch on a server – as long as it meets the user-specified performance, the user neither cares nor sees how it is implemented.

Guaranteed performance, and the ability to enforce this within the underlying infrastructure is a critical aspect of GTS and underlying GVM and GTS. In order to provide the performance that user A requires, user's B, C, and D cannot be allowed to usurp A's allocated guaranteed resource. In order to protect each resource from other potentially interfering resources or services, all services must be provisioned to respect their individual limits. This often requires hardware support for policing mechanisms (such as routers policing a flow at an interface to avoid exceeding a certain rate), whether they are virtual circuits in question, virtual machines, or virtual functions. Ethernet framed transport services are particularly sensitive to network congestion – which is often caused by queuing due to transient burst convergence in the network. Therefore, in order to support both the Quality of Service attributes such as capacity or jitter constraints virtual circuits must be engineered end-to-end to support appropriate policing, shaping, and burstiness. Some transport technologies do this as a matter of fact – in particular, OTN (and SONET/SDFH), and many fibre-delineated transport technologies, etc. Due to its asynchronous and variable length framing technology, Ethernet is unable to do this well without specialised hardware. GÉANT's core network is being upgraded to support a high-level or performance guarantee, available with the deployment of GTS version 6.0. (Note: for circuit QoS, capability is a function of the circuit service: BoD or GÉANT plus, or the newer Consolidated Connection Services (CCS). CCS will be rolling out QoS capable circuits mid-late 2018. GTS will take advantage of the new circuit capability).

For compute elements, the performance guarantees are based upon virtual CPU allocations configured for a VM (1 VCPU = 1 core.) A user can request a default VM with only 1 core and can also request upgraded flavours of VMs with added CPU cores and increased RAM. After a point, it is better to request a full Bare Metal Server than a performance-guaranteed VM with 8 CPUs.

In the virtual switching, GTS maps the nominal packets-per-second switching requirements of the requested virtual switch to be the sum capacity of the transport links that converge on that resource

[ONSA18]. Alternatively, the user may request a specific capability in the switch even if that is insufficient or overly adequate to process the potential traffic flows.

An empty monitoring instance is automatically set up when a VM is reserved. Monitoring data is added once the VM is instantiated and the actual data is available. Services with performance guarantees are essential for GTS to provide insulated and isolated service environments. These performance guarantees must be upheld in order for researchers or service engineers to verify proper performance or to compare actual performance against a verifiable maximum. Reserving resources with guarantees does consume the infrastructure faster than traditional, best-effort resources. This is acceptable if the users requesting those guarantees understand (by means of accounting and billing)[1] that the facilities are finite and by reserving performance guarantees they are making less infrastructure available to other user.

3.3 Multi-Domain GVM

In multi-domain, GVM-compliant environments[2] the Provider Agent (PA) (a software component) can not only automatically allocate resources from its own "local" inventory by invoking the local RCAs, but can also draw from a pool of foreign resources in other domains. In turn, the Provider Agent or Resource Manager may receive resource reservation requests from outside GVM service domains. This ability for a service provider to find and acquire resources from other service providers vastly simplifies the user's plight by moving responsibility for multi-domain interactions from the requester to the provider.

When a provider must go outside its own network to procure the necessary resources, the home service provider assumes the role of user agent and simply issues a reservation request upstream to one of its peering GTS services. If this (possibly recursive) request is filled by the upstream provider, the requesting provider simply notes the upstream resource ID as the allocated infrastructure and returns a local resource ID to the original user request. This process makes the reservation and provisioning process a single relationship between the resource requester and the provider, known as "One-Stop Shopping".

A resource secured from another Provider Agent is processed like a composite resource containing one single child in the local Resource Tree. As explained above, composite resources are processed recursively, i.e. a Reserve() request is sent to each child in the composite resource. A GVM-compliant service should therefore implement some type of "RCA-Multi-Domain" handler that is invoked for resources to be secured from a downstream service. Such an "RCA-Multi-Domain" represents the user side of the API and processes the downstream search.

Once a downstream resource can be obtained, the intermediate Provider Agent must have two resource records in the local Resource Tree: One record for the resource requested by the upstream Provider Agent and one record entry for the resource reserved by the downstream PA. The intermediate PA can then copy relevant information from the downstream resource record to the

[1] GTS in v6.0 does not have any billing mechanisms in place; there are plans to consider sustainability cost models as part of future work.

[2] If a provider does not use the GVM resource description, the service is not compliant and no guarantees can be given that others can make use of the resources.

upstream record and return a local Provider Resource ID to the upstream requester as if the resource was in fact reserved by the local (intermediate) PA.

4 Service Operations

The GTS service is scheduled to move from its current operational state officially as a Pilot service into a full production environment run by the GÉANT operations team in Q4 2018. This move will be conducted using the same hardware infrastructure already in place for users of GTS v5.0 but comprise the *deployment* of GTS v.6.0 by the operations team (which, until Q4 has only been available as a *software release* in the GTS lab in Prague). Running the service in the full production environment requires the definition of operations and support teams, service metrics, key performance indicators (KPIs) and service policies. This section describes the operational aspects of the service (as part of the service definition) that will be in place for a GTS production service.

4.1 Supporting Infrastructure

The following supporting infrastructure is required for running the GTS service:

- Operation and maintenance of the PODs in Amsterdam, Bratislava, Hamburg, London, Madrid, Milan, Paris, and Prague; this also includes the Central Server Facility (CSF) in Prague and backup CSF facility in London.
- Operation of a monitoring system for checking the health of systems and services and raising alarms.
- Operation of the Open-Source Ticket Request System (OTRS) issue-tracking system for operational issues and user contact.
- Operation of a qualification laboratory, where new versions of GTS development can be tested in connection with other GÉANT-specific network hardware and software to verify coexistence, stability, and reliability. This qualification lab must run independently from a GTS development lab (in order to be able to ensure GTS development is independent from deployment testing) and will serve as an intermediary verification step of a newly developed release or version before any deployment into production.

4.2 Operations and Support Teams

Operations of the GTS service (v6.0 in production) is handled by three teams. The GÉANT Operations team (GOC) is responsible for operating and maintaining the production infrastructure. The GÉANT User Support team provides first-level support to GTS users, and the GTS Developers team further enhances GTS, keeps GTS development going and provides third-level support, if needed.

The teams are described in more detail below

- GÉANT Operations – responsible for management and maintenance of all infrastructure components in the field and also for running the associated backbone. This team will also be supporting users; user contact will be via tickets generated by users via the GTS contact form, which, upon submission, will generate a ticket in the OTRS ticket system of GÉANT Operations;

users can also contact GÉANT Operations directly via the gts-operations@lists.geant.org mailing list.

- Business Development and Service Management - Operationally, GTS business development and service management requires insight into how the service is being used – how many resources are being used, of which type, where, which projects are using them, etc. This operational information is needed in real time and for historical trending data (VMs used in last six months, etc). This is crucial for effective capacity planning. And the same features needed to do this usage reporting are also needed for accounting (and eventual billing for service use in order to make the service sustainable). The Service Management team will therefore monitor resource usage in order to be able to manage and direct the placement of resources onto the infrastructure in order to be able to do load balancing, or to simply most efficiently share the overall facilities appropriately among all users. This requires policy (quotas), and migration and grooming features. The Business Development and Service Management team will also have to continuously introduce GTS to new GTS user communities and based on their feedback, actively take part in design of future roadmaps of the service. This can include requests such as active modification and checkpoint/restart capability, mentioned in Section 5.

- GTS Developers – will translate all feedback and roadmap requirements into their software and version development. This team is also available for detailed technical support and implementation-specific issues.

4.3 Service Policies and Service Metrics

Beyond GTS's initial target market for network research in the R&E community (NRENs), additional use cases can include: early TRL advanced services, custom dedicated networks for specific communities and virtualised services mapped onto the WAN footprint. GTS service policies must take all stakeholders into consideration and support network innovation.

To assess the quality of the GTS service, service usage and associated metrics can be taken into account. Usage metrics and key performance indicators include the availability of the established hardware infrastructure over time (to assess the quality of the service in terms of availability) and also the number of resources used per testbed over time (to assess how useful the resources were to the researcher / experiment). This is not to be confused with general resource exhaustion, which should be tracked as part of a broader picture of service utilisation.

5 Service Roadmap

As this service evolves it is important to stay in touch with the user community to see what features are most useful, what new features are urgently needed and what extensions to the service would be beneficial in numerous experiments. In response, the JRA2 team has organised three GTS Tech+Futures Workshops in order to gain insight into users' requirements and collect feedback in open discussions with the research community in order to find out how this service should progress [M8.3], [NAE16a], [NAE16b]. A fourth such workshop is planned for September 2018.

In the remaining project period the focus will be on transitioning GTS v6.0 into production. The next software version (GTS v7.0) will consider enhancements to the system, such as a checkpoint/restart and migration/grooming capability that will allow for testbeds to be saved and moved or released and be restored and restarted at a later point in time. The future development will also consider active modification of testbeds that will offer the capability to modify a testbed while it is in reserved or active state without having to release any resources. Other important features in v7.0 will include a policy engine to support enhanced resource allocation rules and quotas, simplified installation processes, containers and contextualisation, support for composite libraries, and improved integration with SFA testbeds.

The development of the next version (v7.0) at the design level will start during this project period, while the transition into production of v7.0 is targeted for a future project period.

6 Conclusion

This document provided a detailed description of the GTS release in its improved version 6.0, listing its benefits and user communities. An in-depth look was also offered into the underlying GVM architecture. At the end of the document, service operations and a future roadmap were described. To facilitate rapid prototyping and innovation, researchers must have the facilities in place that allow them to set up network environments where they can conduct research without impacting others with faults or unexpected errant behaviour. Such network environments need to support future network demands and must be setup for large-scale experiments, for tests in high speed terabit per second and beyond networks. Virtual network environments for research must allow for programmability and supporting resources, such as high-performance storage and compute resources as well as tools or service functions that facilitate performance investigations.

While the GTS service already offers some of these capabilities, the service has the potential to expand its international and global reach with exchange points and (cloud) ecosystems (described in Section 2.1) that allow multi-domain experiments with software-defined infrastructures.

The service architecture must remain independent from the service and must enable the introduction of new and upcoming hardware devices into the service. GVM as a formal framework specification should, therefore, continue to evolve and provide the underlying model of how automatic provisioning of flexible and dynamic ecosystems with software abstraction and network function virtualisation can be processed.

Appendix

Presentations to the NREN communities:

- J. Sobieski, Virtualization Services, Nordic CTO Forum, Copenhagen, Denmark, August 2016
- J. Sobieski, Network Function Virtualization using containers in GVS, NORDUnet All Hands, Copenhagen, Denmark, September 2016.
- J. Sobieski, S. Naegele-Jackson, Building user-defined and user-controlled virtual networks for WAN workflows, DI4R2016 (Digital Infrastructures for Research 2016), Sept. 28-30, 2016, Krakow, Poland.
- S. Naegele-Jackson, GÉANT Testbeds Service (GTS), DFN-GTS Workshop, Oct. 27, 2016, Berlin, Germany.
- J. Sobieski, GTS: Virtualized Network Services for Advanced e-Science Environments, Internet2 Network Architecture Meeting, Washington DC, USA, Dec 2016.
- K. Meyer, J. Sobieski, S. Naegele-Jackson, GÉANT TESTBEDS SERVICE – Bringing Researchers and Technology Together, Connect Magazine, Issue 24, 2017.
- J. Sobieski, GÉANT Experimental Facilities – lessons learned, TNC17, Linz, Austria, May 29 - Jun 2, 2017.
- J. Sobieski, GTS: Virtualized Network Services for Advanced e-Science Environments, TNC17, Linz, Austria, June 2017.

Presentations to the GTS research communities:

- J. Sobieski, Virtualization, the Generalized Virtualization Model, and Innovation within GÉANT4, TNC16, Prague, Czech Republic, June 13, 2016.
- J. Sobieski, A Generalized Virtualization Model for Emerging SDN Based Network Services, OpenNFV, Berlin, Germany, June 2016.
- J. Sobieski, Virtualization Rising: Some Emerging Concepts in Advanced Networks, TechEx 2016, Miami, FL, USA, September 26, 2016.
- J. Sobieski, GÉANT Virtualized Network Services for Advanced e-Science Environments, Aalborg University Graduate Seminar, Aalborg, Denmark, October 2016.
- J. Sobieski, GÉANT Testbeds Service: Virtualized Network Services for Advanced e-Science Environments, Supercomputing (SC16), Salt Lake City, USA, Nov. 15, 2016.
- R. Cziva, J. Sobieski, Y. Kumar, High-Performance Virtualized SDN Switches for Experimental Network Testbeds, INDIS WS, SC16, Salt Lake City, USA, Nov. 13, 2016.
- S. Naegele-Jackson, J. Sobieski, J. Gutkowski, M. Hazlinsky, Creating Automated Wide-Area Virtual networks with GTS - Overview and Future Developments, 8th IEEE International Conference on Cloud Computing Technology and Science (CloudComp), 12-15/12/2016, Luxembourg.
- J. Sobieski, GÉANT Testbed Service (GTS) at panel discussion: What's the right testbed for your experiment goals, GENI NICE 2016 and GENI Regional Workshop, Irvine, CA, USA, Dec. 12, 2016.
- J. Sobieski, GTS: Virtualized Network Services for Advanced e-Science Environments, CENIC, Berkeley, CA, USA, Dec. 2016.

- S. Naegele-Jackson, Creating Automated Wide-Area Virtual Networks with GTS, ICN2020 Project Meeting, Jan. 23, 2017, Rome, Italy, ICN2020 project.
- J. Sobieski, S. Naegele-Jackson, 3rd GTS Tech+Futures Workshop, Feb. 28 - March 1, 2017, Utrecht, Netherlands.
- N. Iliuha, GTS Monitoring, 3rd GTS Tech+Futures Workshop, March 1, 2017, Utrecht, Netherlands.
- N. Iliuha, Knowing GÉANT Testbed Service (GTS), Joint workshop IDSI-RENAM, Chisinau, Moldova, March 17, 2017.
- N. Iliuha, Knowing GÉANT Testbed Service (GTS), Joint workshop IMI-RENAM, Chisinau, Moldova, May 17, 2017.
- N. Iliuha, P. Bogatencov, Approaches in Building of Cloud Base Scientific Computing Infrastructure (SCI), JINR 26th Symposium on Nuclear Electronics and Computing (NEC2017), Montenegro, Budva, Becici, 25-29 Sept. 2017.
- J. Sobieski, Tutorial: Introduction to GÉANT Testbed as a Service (GTS), 2nd FED4FIRE+ Engineering Conference, Volos, Greece, Oct. 4-6, 2017.
- S. Naegele-Jackson, Creating Automated Wide-Area Virtual Networks with GTS - Overview and Future Developments, 1st KuVS Fachgespraech "Network Softwarization" - From Research to Application, Tuebingen, Germany, October 12-13, 2017.
- S. Naegele-Jackson, Automatic Provisioning of Virtual Networks with GTS, Supercomputing Conference (SC17), Denver, CO, USA, Nov. 14-16, 2017.
- S. Naegele-Jackson, Automatic Provisioning of Infrastructures with GTS, 9th SDN Workshop, IBM Research Center, Rueschlikon, Switzerland, Dec. 4, 2017.
  S. Naegele-Jackson, Establishing Large-Scale Virtual Infrastructures with the GÉANT Testbeds Service (GTS), Austrian HPC Meeting (AHPC18), Linz, Austria, Feb. 19-21, 2018.

References

[API] See the section Get Information Over API at https://wiki.geant.org/display/gn42jra2/GTS+USER+GUIDE+V+5.0

[AST15] J. Astorga, A. Mendiola, A. Urtasun, E. Jacob, M. Higuero, V. Fuentes, DynPaC: Dynamic and Adaptive Traffic Engineering for SDNs, TNC15, Porto, Portugal, 15-18 June 2015

[AST16] J. Astorga, V. Fuentes, M. Higuero, E. Jacob, A. Mendiola, A. Urtasun, Open Call Deliverable OCH, DS1.1.1, Dynamic Path Computation Framework, Final Report (DynPaC), available from http://geant3plus.archive.geant.net/Resources/Open_Call_deliverables/Documents/DynPaC_final_report.pdf

[BHA14] D. Bhat, N. Riga, M. Zink, Towards Seamless Application Delivery using SW Defined Exchanges, FIDC, Karlskrona, Sweden, 2014.

[BRO15] M. Broadbent, N. Hart, Experiments with GTS at Lancaster University, 1st GTS Tech+Futures Workshop, Copenhagen, Denmark, October 15, 2015, http://services.geant.net/GTS/Resources/Pages/1st_GTS_Workshop.aspx

[CIE18] Ciena, www.ciena.com

[COR18] CORSA Technology, https://www.corsa.com/.

[CZI16] R. Cziva, J. Sobieski, Y. Kumar, High-Performance Virtualised SDN Switches for Experimental Network Testbeds, INDIS WS, SC16, Salt Lake City, USA, Nov. 13th, 2016.

[D6.1] Deliverable D6.1 (DS2.3.1): Architecture Description: Dynamic Virtualised Packet Testbeds Service. https://geant3plus.archive.geant.net/Resources/Media_Library/Documents/D6-1_DS2-3-1_TaaS_v1.0.pdf

[D8.1] D8.1: Integrated Services Framework and Network Services Development Roadmap. https://www.geant.org/Projects/GEANT_Project_GN4/deliverables/D8.1_Integrated-Services-Framework.pdf

[DEPLOYMENT] There are a number of sites that have successfully tested and deployed GTS, please see references MEN14, AST15, BRO15, FEM15, MAI15, MEN15, MOB15, ONS15a, ONS15b, PRE15a, PRE15b, SAL15, VEN15, AST16, HER16a, HER16b, HER16c for more detail.

[Drag'n'DrED] Please note: The Drag'n'DrED (editor) is not to be confused with the jFed drag and drop editor developed in the Fed4FIRE+ project that is being adapted to GTS allowing researchers to meld GTS services seamlessly with Slice Federated Architecture (SFA) testbed facilities in labs around Europe [jFed]. (Note, the jFed editor is a standalone application rather than a web GUI, so it is a bit more involved for new users to install and configure. For those that use jFed, however, the newest version will allow them to seamlessly incorporate GTS WAN facilities into their research topology.)

[DSL] Examples of DSL code may be found in Appendix II https://wiki.geant.org/display/gn42jra2/GTS+USER+GUIDE+V+5.0

[FAR14] F. Farina, P. Szegedi, J. Sobieski, GÉANT World Testbed Facility - Federated and distributed Testbeds as a Service facility of GÉANT, FIDC, Karlskrona, Sweden, 2014.

[FED18] FED4Fire, http://www.fed4fire.eu/.

[FEM15] M. Femminella, Testing ARES on the GTS framework: lesson learned and open issues, 1st GTS Tech+Futures WS, Copenhagen, October 2015, https://www.geant.org/Services/Connectivity_and_network/GTS/Documents/MF3.pdf

[FIL17] S. Filiposka, Possibilities for using GTS as a practical tool in higher education courses on networking, GTS workshop, Utrecht, The Netherlands, 28 February & 1 March 2017.

[GEA18] GÉANT, http://www.geant.org/.

[GRO18] GROOVY, http://groovy.codehaus.org/.

[GTS18] The GÉANT Testbeds Service, http://services.geant.net/gts and https://gts.geant.net/login. For v.6.0 code please contact gts-operations@list.geant.org

[GUIDE] https://wiki.geant.org/display/gn42jra2/GTS+USER+GUIDE+V5.0

[HAZ14] Hazlinsky, M.; Farina, F.; Pietrzak, B.; Sobieski, J.; Szegedi, P.: SDNI: The GÉANT testbeds service - virtual network environments for advanced network and applications research, 2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC) , Moscow, Russia, 28-29 Oct. 2014, **DOI:** 10.1109/MoNeTeC.2014.6995585.

[HER16a] S. Hermans, P. de Niet, Docker Overlay Networks: Performance Analysis in high-latency environments, MSc Research project, University of Amsterdam, February 7, 2016, http://rp.delaat.net/2015-2016/p50/report.pdf

[HER16b] S. Hermans, P. de Niet, Docker Overlay Networks: Performance Analysis in high-latency environments, presentation, University of Amsterdam, http://delaat.net/rp/2015-2016/p50/presentation.pdf

[HER16c] S. Hermann's, gtsperf, the files in this repository are used to test the performance of various Docker overlay solutions in an automated manner. Specifically to be used within the GÉANT Testbeds Service (GTS). https://github.com/siemhermans?tab=overview&from=2016-08-01&to=2016-08-31&utf8=%E2%9C%93

[jFed] https://jfed.ilabt.imec.be/

[M8.3] Milestone M8.3, GÉANT Testbeds Service Roadmap, Sept. 2016

[MAI15] Ray Le Maistre, Eurobites: GÉANT Builds SDN Testbed, Light Reading, August 21, 2015, http://www.lightreading.com/carrier-sdn/eurobites-geant-builds-sdn-testbed/d/d-id/717770

[MAM14] J. Mambretti, J. Chen, F. Yeh, Software-Defined Network Exchanges (SDXs): Architecture, Services, Capabilities, and Foundation Technologies, FIDC 2014 in Karlskrona, Sweden, 2014.

[MEN14] A. Mendiola, A. Urtasun, A. Leguina, V. Fuentes, E. Jacob, J. Astorga, University of the Basque Country Michał Balcerkiewicz, Krzysztof Dombek, Artur Juszczyk, Łukasz Ogrodowczyk, Damian Parniewicz, Miłosz Przywecki, PSNC, NSI based multi-domain connection provisioning across OpenFlow domains, Demo at SC14, November 2014, USA, http://geant3plus.archive.geant.net/opencall/SDN/Documents/NSI-CONTEST%20and%20DyNPaC_NSI-OF-demoSC2014.pdf

[MEN15] A. Mendiola, A. Urtasun, V. Fuentes, J. Astorga, M. Higuero, E. Jacob, M. Przywecki, Krzysztof Dombek, A. Juszczyk, D. Parniewicz, Ł. Ogrodowczyk, M. Balcerkiewicz, S. Naegele-Jackson, A solution for connection-oriented multi-domain SDN based on NSI-CS and the DynPaC framework, TNC15, Porto, Portugal, 15-18 June 2015

[MOB15] Mobile Europe, GÉANT, telcos deploy open-source SDN network in Europe, August 21, 2015, http://www.mobileeurope.co.uk/press-wire/geant-telcos-deploy-open-source-sdn-network-in-europe

[NAE16a] S. Naegele-Jackson, J. Sobieski, J. Gutkowski, M. Hazlinsky, Creating Automated Wide-Area Virtual networks with GTS - Overview and Future Developments, 8th IEEE International Conference on Cloud Computing Technology and Science (CloudComp), 12-15/12/2016, Luxembourg. (https://ieeexplore.ieee.org/document/7830745/)

[NAE16b] S. Naegele-Jackson, GÉANT Testbeds Service (GTS), DFN-GTS Workshop, Oct. 27, 2016, Berlin, Germany.

[NAE18] S. Naegele-Jackson, J. Sobieski, Establishing Large-Scale Virtual Infrastructures with the GÉANT Testbeds Service (GTS), AHPC18, Feb. 19-21, 2018, Linz, Austria

[ONF12] Open Networking Foundation (ONF), OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04), June 25, 2012.

[ONS15a] ONOS, Global SDN Deployment Powered by ONOS, https://www.opennetworking.org/images/stories/sdn-solution-showcase/germany2015/ONOS%20-%20Global%20SDN%20deployment%20powered%20by%20ONOS.pdf

[ONS15b] ONOS, Global ONOS and SDN-IP deployment, http://onosproject.org/wp-content/uploads/2015/06/PoC_global-deploy.pdf

[ONSA18] OpenNSA, https://redmine.ogf.org/projects/nsi-wg.

[OST18] OpenStack, https://www.openstack.org/software/.

[PRE15a] L. Prete, Global SDN deployment powered by ONOS, Internet2 TechExchange 2015, Oct. 6, 2015, https://meetings.internet2.edu/media/medialibrary/2015/10/05/20151006-prete-global-deployment-powered-ONOS_jocpVr0.pdf

[PRE15b] L. Prete, Global SDN deployment powered by ONOS, 1st GTS Tech+Futures Workshop, Copenhagen, Denmark, October 15, 2015, http://services.geant.net/GTS/Resources/Pages/1st_GTS_Workshop.aspx

[SAL15] S. Salsano, Pier Luigi Ventre, Francesco Lombardo, Giuseppe Siracusano, Matteo Gerola, Elio Salvadori, Michele Santuari, Mauro Campanella, Luca Prete, Hybrid IP/SDN networking: open implementation and experiment management tools, IEEE Transaction of Network and Service Management - December 2015, https://arxiv.org/ftp/arxiv/papers/1505/1505.03579.pdf

[SOB15] J. Sobieski, S. Naegele-Jackson, B. Pietrzak, M. Hazlinsky, F. Farina and K. Kramaric, GÉANT Testbed Service - External Domain Ports: A demo on multiple domain connectivity, EWSDN, Bilbao, 2015.

[SOB16] J. Sobieski, Virtualization, the Generalized Virtualization Model, and Innovation within GÉANT4, TERENA 2016 (TNC16), Prague, June 13, 2016.

[VEN15] P. L. Ventre, Stefano Salsano, DREAMER and GN4-JRA2 on GTS, 1st GTS Tech+Futures Workshop, Copenhagen, Denmark, October 15, 2015, http://services.geant.net/GTS/Resources/Pages/1st_GTS_Workshop.aspx

[virsh] https://help.ubuntu.com/community/KVM/Virsh

Glossary

API	Application Programming Interface
BMS	Bare Metal Server
CCS	Consolidated Connection Services
CSF	Central Server Facility
DDOS	Distributed Denial of Service
Drag ‘n’DrED	Drag and Drop Editor
DSL	Domain Specific Language
Gbps	Gigabit per second
GE	Gigabit Ethernet
GOC	GÉANT Operations Centre
GTS	GÉANT Testbeds Service
GUI	Graphical User Interface
GVM	Generalised Virtualisation Model
GW	Gateway
IAGW	Internet Access Gateway
jFed	Java-based framework for testbed federation
JRA	Joint Research Activity
KPI	Key Performance Indicator
MDPA	Multi-Domain Provider Agent
NAT	Network Address Translation
NFV	Network Function Virtualisation
NREN	National Research and Education Network
NSI	Network Service Interface
OFX	OpenFlow Switch instance
OTRS	Open-Source Ticket Request System.
OXP	Open eXchange Point
PA	Provider Agent
POD	Point of Distribution
PoP	Point of Presence
QoS	Quality of Service
RCA	Resource Control Agent
RDB	Resource Database
RM	Resource Manager
RT	Resource Tree
QoS	Quality of Service
SFA	Slice Federated Architecture
SDN	Software Defined Networking
SDP	Service Demarcation Point
SDX	Software Defined Exchange
STP	Service Termination Point
UA	User Agent
VC	Virtual Circuit
VM	Virtual Machine
VPN	Virtual Private Network
VSI	Virtual Switch Instance
WAN	Wide Area Network