Deliverable D13.1 (DJ2.1.1) Specialised Applications' Support Utilising OpenFlow/SDN

Deliverable D13.1 (DJ2.1.1)

Contractual Date:	30-11-2014
Actual Date:	02-03-2015
Grant Agreement No.:	605243
Activity:	JRA2
Task Item:	T1
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	PSNC
Document Code:	GN3PLUS-14-1233-26
Authors:	Christos Argyropoulos (GRNET/ICCS), Buelent Arslan (FAU), Jose Aznar (i2CAT), Kurt Baumann (SWITCH), Krzysztof Dombek (PSNC), Eduard Escalona (i2CAT), Dani Guija (i2CAT), Eduardo Jacob (EHU), Artur Juszczyk (PSNC), Jiri Melnikov (CESNET), Alaitz Mendiola (EHU), Susanne Naegele-Jackson (FAU), Łukasz Ogrodowczyk (PSNC), Dušan Pajin (AMRES), Damian Parniewicz (PSNC), Ronald van der Pol (SURFnet), Milosz Przywecki (PSNC)

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

Abstract

This document reports on activities carried out in JRA2 T1 to support advanced applications utilising OpenFlow/SDN. An update from the support of SA2-operated GÉANT OpenFlow Facility is provided and issues related to SDN cloud support, multi-domain SDN, SDN monitoring, SDN security and SDN applications are discussed. Description of proof-of-concept of solutions for security applications, monitoring, multi-domain, circuit-oriented SDN and SDNaps are provided.

Table of Contents

Executive Summary

Software-Defined Networking has recently experienced a huge interest in the networking industry and academia. Network programmability creates new possibilities for the provisioning of new, advanced network services. This document summarises results of GN3plus Joint Research Activity 2, Task 1 (JRA2T1). JRA2 T1 focused on investigation of how new SDN capabilities can support requirements of the specialised applications. The work was focused on:

- Multi-domain SDN
- Monitoring in OpenFlow networks
- Using OpenFlow to enhance security in the network
- Support for clouds
- SDN applications.

The sections of this document describe the results in each of the above-mentioned areas.

Section 2 reports on support of the SA2-operated GÉANT OpenFlow Facility (GOFF). It lists the problems discussed in collaboration with SA2. A separate section is related to enhancing the GOFF with the monitoring slice, which is used to support the operation of the facility and to detect failures. Responsibilities of GÉANT NOC, SA2/JRA2 GOFF provider and experimenter are listed.

Section 3 proposes the use of OpenFlow to enhance the security in the network. Two scenarios are described: traffic duplication and redirection, which can be used for the traffic analysis and redirection of the complete traffic flow, which can be used to filter the traffic. For both scenarios, a detailed description is provided and a proposal for the implementation is given.

Section 4 provides an overview of the monitoring capabilities of the SDN/OpenFlow. It begins with a short description of the flow-based monitoring capabilities. Then it continues with a proposal for a complete solution for generating NetFlow data, based on the OpenFlow statistics. An overview of Business solutions for SDN monitoring is also given, to provide a broader view on monitoring possibilities in SDN. Finally, the monitoring capabilities of the selected OpenFlow switches are described, and OAM/OVS extensions implemented as part of the discussion of JRA2T1 work.

There are also a number of appendices providing supplementary information:

- Appendix A details the results of an NREN and user community survey on clouds, SDN and NFV, based on the question: What can the network do for the clouds?
- Appendix B provides an overview of business solutions for SDN monitoring.
- Appendix C discusses the optional OpenFlow features, as well as features for monitoring and statistics available from vendors.
- Appendix D introduces SDN for clouds, testbeds and campus networks, discusses solutions for multi-domain SDN and provides a description of three use cases.
- Appendix E describes SDNapps as generic network functionalities running on top of the SDN/OpenFlow network, as well as discusses the relationship between SDNapps and NFV.

1 Introduction

Software-Defined Networking (SDN) and OpenFlow are both relatively new, but are rapidly becoming more widespread as they mature. A large number of conference papers and presentations related to SDN and OpenFlow show that academic and research communities are especially interested in exploring possibilities of using OpenFlow as a technology to allow control of the network at the flow level. At the same time, SDN solutions development also greatly influence the development of other technologies such as Network Functions Virtualization (NFV). NFV is another emerging technology, heavily supported by the network operators, that allows virtualisation of network functions into reusable modules. A number of efforts are focused on the standardisation of SDN and NFV. The Open Networking Foundation [ONF] is developing OpenFlow standards, and SDN is also being defined by Software-Defined Networking Research Group (SDNRG) at IRTF [SDNRG], as well as by ITU-T (see recommendation Y.3300: Framework of software-defined networking [Y3300]). Network Functions Virtualisation is being defined by ETSI [ETSINFV], and at the time of writing this document (October 2014), there are already eight standards related to the NFV framework, use cases, interfaces, security and available proof of concepts.

Simultaneously, a number of SDN tools have been developed: the most important from the JRA2T1 point of view are SDN controllers and virtualisation tools (e.g. FlowVisor [FV], flowspace firewall [FSFW], OpenVirteX [OVX]). After short analysis of the features and capabilities of the controllers (which is not covered in this document) the JRA2T1 team selected two controllers for building proof-of-concept prototypes: Python-based Ryu [RYU] and Java-based OpenDaylight [ODL]. Both controllers support the most recent OpenFlow protocol versions. Ryu is a controller initially developed by NTT, while OpenDaylight is a collaborative project supported by the Linux Foundation [LINF].

The work within JRA2T1 has focused on a number of different topics, the results of which are discussed in the following sections. After the handover of the GÉANT OpenFlow Facility (GOFF) to SA2, JRA2T1 members provided the necessary support to the SA2 team. The different requirements from Open Call projects also resulted in a number of questions related to the potential enhancement of the GOFF capabilities. JRA2T1 contributed to necessary analysis of the requirements and, where possible, proposed solutions to enhance the functionality of GOFF.

Apart from supporting the GOFF, the JRA2T1 team investigated a number of issues related to using SDN/OpenFlow to support flow-based networking. Since SDN technologies are relatively new, there are areas that are not covered by standards (e.g. multi-domain SDN) or that need further investigation, such as the possibilities of supporting existing services and tools with the new capabilities offered by SDN networks. In particular, cloud-related requirements in GÉANT and the NREN environment have been assessed, and multi-domain SDN possibilities, both for the provisioning of circuits across multiple OpenFlow domains as well as for the creation of multi-domain SDN slices, have been analysed. The OpenFlow capabilities for strengthening the security in the network through traffic redirection and duplication using OpenFlow have also been investigated. The OpenFlow monitoring capabilities have also been reviewed, and a proposal for creating NetFlow data based on OpenFlow flow statistics has been developed. The shortcomings of OAM capabilities of the Open vSwitch have been analysed, and a solution has been proposed.

2 Support of SA2-Operated GOFF

JRA2 Task 1 has acted as the Subject Matter Expert (SME) for the GÉANT OpenFlow Facility (GOFF), consulting SA2 Task 3 for advanced architectural and operational issues.

A representative list of operations and technical issues that were handled by JRA2 Task 1 include:

- The introduction of multi-kernel support for GOFF XEN-based virtual machines (VM).
- Work-around solutions for Open vSwitch bugs and insufficiencies (e.g. OpenFlow port assignment).
- Investigation of FlowVisor failures and troubleshooting (e.g. XML-configuration crash, Java machine memory management, connection problems with OCF management plane, cleaning system logs).
- Resource management (memory consumption, CPU, HDD).
- Log files management.
- Management plane connectivity problems troubleshooting (OXAD crashes).
- Firewalling inconsistencies troubleshooting.
- Direct OpenFlow rule injection to Open vSwitches, without OpenFlow controller intervention, used for the GOFF data-plane monitoring.
- Ethernet link/port assignment to Open vSwitches.
- Investigation of FlowVisor flowspace policy implementation (e.g. OpenFlow NORMAL action, finding and presentation of alternatives to NORMAL action).
- Solving problem with duplicated entries in Optin Manager's Experiment table (delete the duplicated entries, train people on how to solve this problem if appear again).
- Assist on what actions can or cannot be performed on the GOFF.
- Assist on how to bypass FlowVisor by using another controller (discuss on how to configure OVSs, discuss on how to ensure the connectivity between the controller and the OVSs).

2.1 GOFF Data-plane Monitoring

The GOFF is a testbed service currently in place for GÉANT users in heavy use by GÉANT Open Call projects. The following section describes how the different sites of the OpenFlow facility are being monitored by the GOFF operations team, as a result of the monitoring infrastructure slice designed and implemented by JRA2T1.

2.1.1 Operation Levels

The GÉANT OpenFlow Facility consists of three levels of operation. In each operational level, a different set of administration groups are authorised and are able to take actions. The three levels of operations can be considered horizontal due to the fact that each higher level uses the services provided by the underlying level in order to serve its users.

The three levels of operation (see Figure 2.1) and the corresponding responsibilities can be categorised with the following order, starting from the most basic (lower) level:

Infrastructure Provider (GÉANT NOC)

- Cabling
- Power Supply
- Air Conditioning
- Physical Installation/Maintenance
- IPv4/IPv6 Network Connectivity (for the control and management plane)
- IPv4/IPv6 Address Space Allocation
- Domain Name System
- Perimeter Firewalling
- L2 data plane connectivity

OpenFlow Facility Provider (GN3plus SA2/JRA2)

- Operating System Administration (XEN & OVS servers)
- Open vSwitch Administration
- OpenFlow Proxy Controller (FlowVisor) Administration
- XEN Virtual Machine Administration
- G-OCF Software
- Experimenters Support
- Experimenters Registration

Experimenter

- VM actions (create, delete, start, shutdown)
- OF controller operations
- OF topology (create, edit, delete)

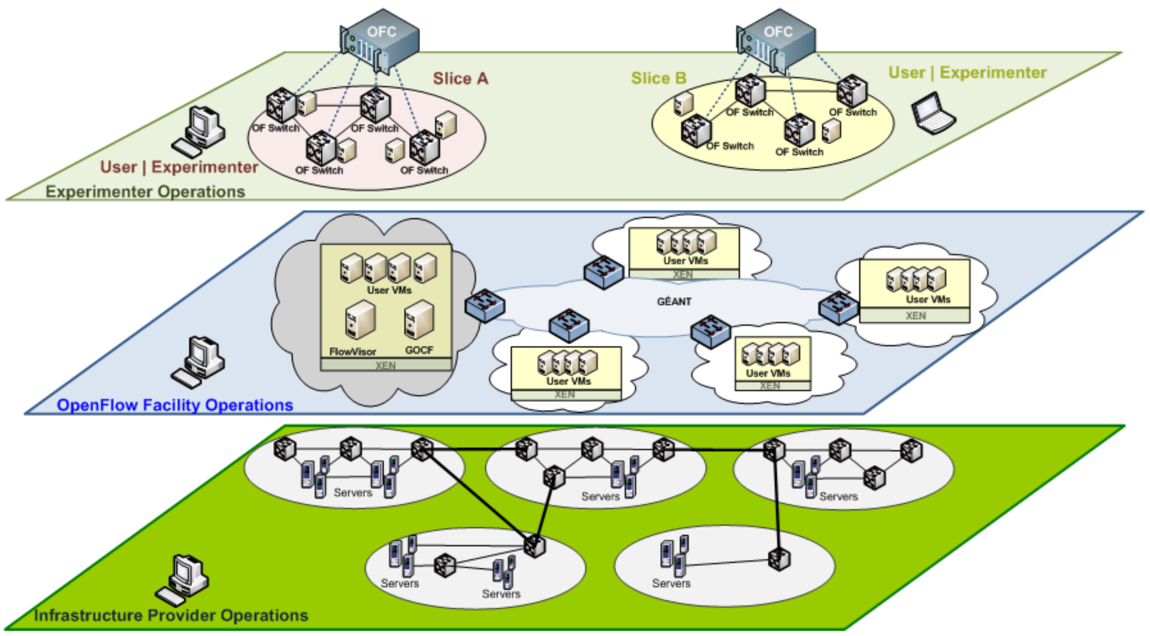


Figure 2.1: GÉANT OpenFlow Facility levels of Operation

2.1.2 Network Monitoring

The GÉANT network's NOC monitors the network connectivity (i.e. the point-to-point Layer 2 MPLS VPNs - pseudowires between Juniper MX boxes) provided by GÉANT and also the Layer 3 GÉANT services which are used by the control and management plane of the OpenFlow Facility (i.e. IPv4/IPv6 Network Connectivity, Firewalling).

Data plane operations and maintenance responsibilities are inevitably divided between Infrastructure Provider (GÉANT NOC) and OpenFlow Facility provider (GN3plus SA2/JRA2). By taking a closer look in Figure 2.2, the installation of the OpenFlow software switches (Open vSwitch - OVS) inside physical servers and their direct network connections with the XEN hypervisors (back-to-back Ethernet cabling) act as Software Defined Networking (SDN) components of the data plane, controlled by the OpenFlow control plane (FlowVisor proxy controller). The data plane connectivity among the experimenters' VMs require the concurrent normal operation of the GÉANT Layer 2 MPLS VPNs and OpenFlow components that are used as SDN-enablers.

Taking into consideration the complex character of the data plane, an OpenFlow Facility provider (GN3plus SA2/JRA2) decided to create a special- purpose user slice that would be used for monitoring purposes by checking the end-to-end connectivity of the GÉANT OpenFlow Facility hypervisors among PoPs. The overall data plane (the entire set of links) connectivity service that is provided to the users' VMs can be easily checked by the OpenFlow Facility provider (GN3plus SA2/JRA2) through automated scripts that use ICMP protocol-based management tools (i.e. traceroute, ping) which trigger email alerts to the administration team in case of packet loss on a link.

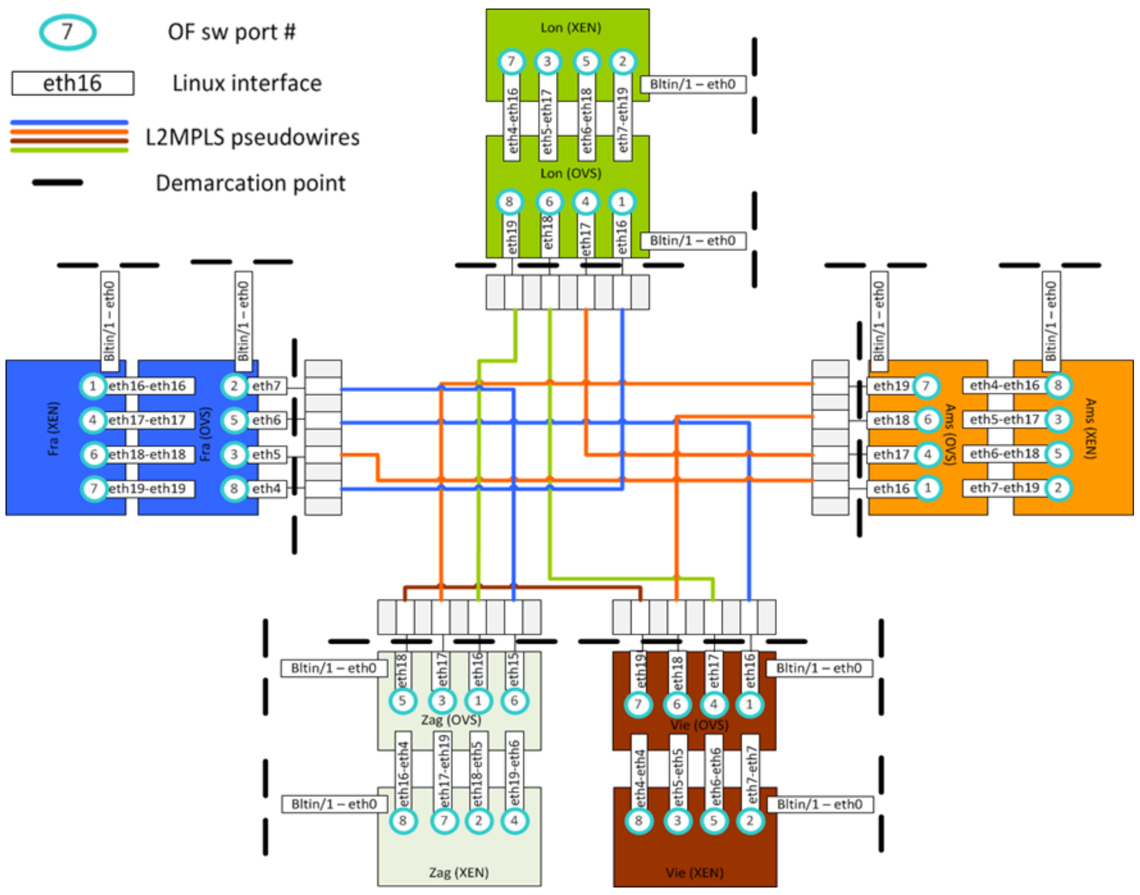


Figure 2.2: GN3plus - DANTE demarcation points for the GÉANT OpenFlow Facility defining maintenance responsibilities

The creation of an infrastructure slice that would be capable of monitoring the full mesh, data-plane topology among PoPs and the OpenFlow switches' normal operation should be autonomously defined without depending on the OpenFlow control plane components, such as FlowVisor proxy controller and OpenFlow controllers on top of the FlowVisor. Hence, OpenFlow forwarding rules were manually injected to the OpenFlow switches in order to implement the forwarding logic that is required for the end-to-end connectivity checks. That way, the persistent OpenFlow rules inside the Open vSwitches are manipulating the packets used for monitoring purposes, without requiring the participation of the OpenFlow control plane for the flow-based forwarding process. Thus, in case of failure at the control plane, the monitoring slice remains unaffected.

A summary of the required steps for the GOFF monitoring slice creation follows.

- Creation of a new slice to the GOFF.
- Reservation of compute resources (VMs) in each PoP.
- Allocation of the appropriate flowspace (VLANs) that would be used explicitly for monitoring purposes.
- VLAN logical interfaces creation inside the VMs.
- Manual persistent flow rule injection (and manipulation) to the Open vSwitches that implement the forwarding logic.

3 OpenFlow Traffic Security Redirection Solutions

In a typical network, critical and/or suspected network traffic is redirected to the security devices for detailed analysis against security attacks (using Intrusion Detection and Prevention Systems) and/or to enforce some security policies (through the use of Firewall or Web security proxy). Currently, traffic redirection is achieved using legacy networking equipment. With the introduction of the OpenFlow protocol, new possibilities for more granular control of traffic are emerging and it can be a very powerful tool both for traffic redirection and traffic filtering purposes. Use of the OpenFlow-enabled networks provide new capabilities to introduce security features directly in the network and to support and interoperate with existing solutions (e.g. by facilitating traffic analysis through delivery of specific flows to network security appliances).

There is extensive and ongoing research in the OpenFlow community that covers these topics. This includes CloudWatcher [CLWATCH], a framework that automatically detours network packets to be inspected by pre-installed network security devices, SE-Floodlight [SEFLOOD], which is the implementation of an OpenFlow security mediation service, including detection and blocking of botnets in the OpenFlow networks. Part of the forthcoming OpenDaylight Project release also includes a system for attack detection and traffic diversion, Defense4All, which is based purely on monitoring and control capabilities exposed by OpenDaylight [DEFENSE4ALL].

This section provides a summary of the OpenFlow traffic security solutions investigated in JRA2T1. Full documentation, along with code examples and testing results, can be found in Section 3.

3.1 Redirect Scenarios

Two use-case scenarios have been proposed to easily and effectively redirect specific traffic flows that need to be analysed by the security devices:

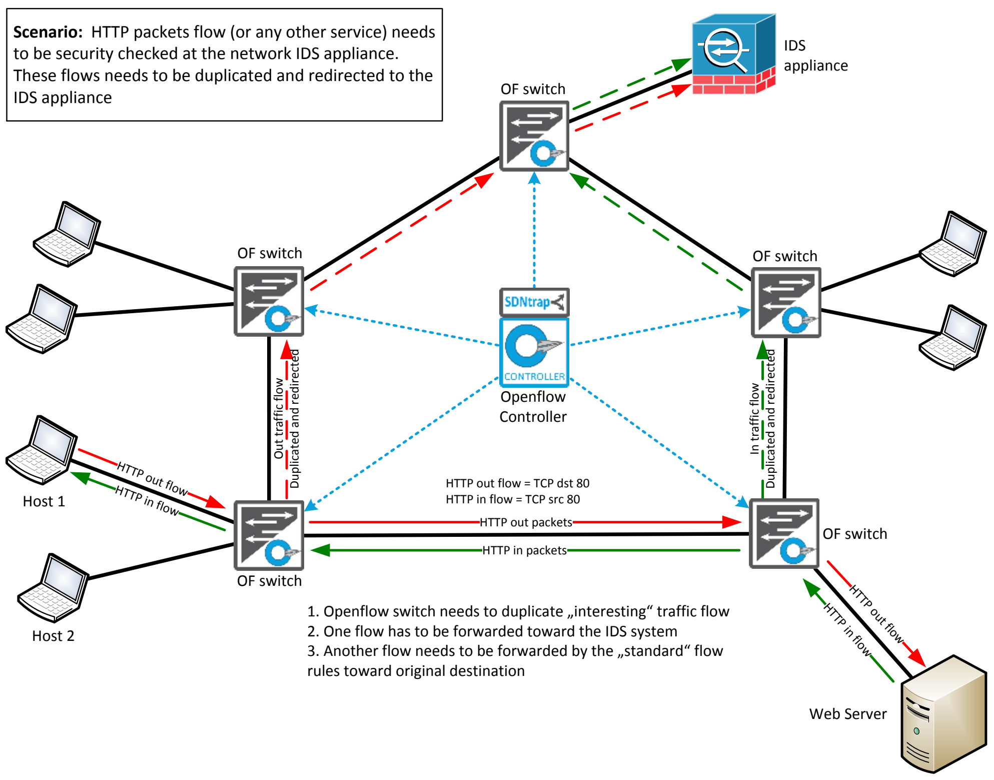
- **Duplication and redirection of the traffic** – This approach is usually used for IDS security devices or other kind of traffic analysis where legacy networking equipment uses traffic mirroring technology. This can be also useful for traffic accounting purposes, or for external devices/software creating NetFlow statistics.

- **Redirection of the complete traffic flow** – This approach is usually used for firewall devices or transparent Web security appliances (IPS security appliances). In legacy networking, these devices need to be located on the traffic path, or "policy-based routing" is used to redirect traffic from the regular routing path.

Both scenarios have an arbitrary network topology from a campus or data-centre network, where a typically large number of hosts or servers have access to resources in the local network or at the Internet. An OpenFlow-enabled network is considered in the "converged state" where an OpenFlow controller has already configured appropriate flow tables of OpenFlow switches.

3.1.1 Traffic Duplication and Redirection

In the first scenario, an IDS security appliance is introduced into the network to provide security analysis of the network traffic. The network administrator's requirement is to define specific traffic flows of interest and to forward them to the IDS appliance. We propose the concept of the "SDN Traffic Redirection Application" (SDNtrap), which sits on top of the OpenFlow controller and will reconfigure the network to forward traffic according to the administrator's requirements.



Scenario: HTTP packets flow (or any other service) needs to be security checked at the network IDS appliance. These flows needs to be duplicated and redirected to the IDS appliance

1. Openflow switch needs to duplicate „interesting" traffic flow
2. One flow has to be forwarded toward the IDS system
3. Another flow needs to be forwarded by the „standard" flow rules toward original destination

Figure 3.1: Traffic duplication and redirection scenario

It is very important to note that under normal conditions, regular traffic-forwarding decisions that are preconfigured in the network must not be changed. SDNtrap will compose and apply only additional flow rules in order to enforce traffic duplication and redirection. In case of detection of dangerous traffic in the network by the IDS, feedback might be given to the controller in order to change some flow rules and drop the dangerous packets.

3.1.1.1 *Proposed Solution for Traffic Duplication and Redirection*

The proposed solution uses features of the OpenFlow specification 1.1 [OF1.1.0] and above, specifically multiple OpenFlow tables and optional "Apply-Actions" instruction. For the successful redirection of the traffic, the SDNtrap application needs the following information, which has to be defined by administrator:

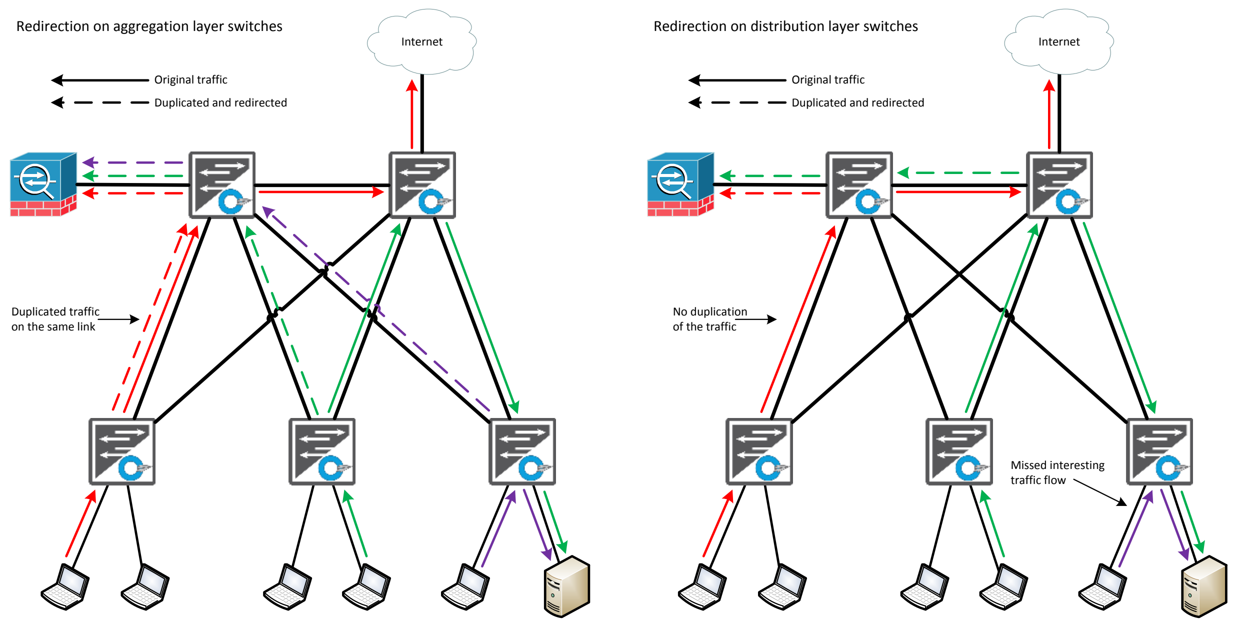
- **Interesting traffic** – Definition of the traffic that has to be checked. It has to contain matching conditions for traffic flows, whose duplicated copies need to be redirected to the security appliance (e.g. source/destination addresses, ports).
- **Location of the security appliance** – point of attachment in the network (noting port and switch).
- **Security label** – network-wide MPLS label or VLAN tag that will be used for labelling interesting traffic.

The proposal for the traffic redirection uses the following concepts and features:

- **Point of duplication and redirection** – It is proposed that the duplication and redirection of the traffic is performed by the OpenFlow switch closest to the source of the traffic. For each traffic flow defined as interesting by the administrator, the SDNtrap determines the closest OpenFlow switch (ingress switch) to the source, where the security redirection flow rules will be applied.
- **Usage of multiple flow tables** – The first OpenFlow Table in the pipeline processing of the OpenFlow switch is dedicated for the security redirection. All other rules are then located in the remaining flow tables, so they can be processed after security redirection rules.
- **Matching flow rules for interesting traffic** – Matching part of the OpenFlow security redirection rules will be carried out according to the definition of the interesting traffic. This will also include an ingress port matching.
- **Actions for matched traffic** – The action part of the security flow rules should duplicate and redirect interesting traffic flow and label it for further efficient forwarding.
- **Forwarding of the interesting traffic** – Further forwarding of the duplicated and redirected interesting traffic through the network should be done solely based on the security label.

3.1.1.2 *Point of Duplication and Redirection*

The duplication and redirection of the traffic is carried out on the OpenFlow switch that is closest to the source of the traffic. For each traffic flow defined as interesting by the administrator, SDNtrap determines the closest OpenFlow switch to the source, where the security redirection flow rules will be applied. In some cases, depending on the topology, this approach will lead to increased network traffic (redundant traffic flows), because regular and redirected traffic could follow up the same path toward its regular destination and security device. Algorithms for the more optimal routing and the point of duplication and redirection can be found in the paper CloudWatcher [CLWATCH] where they are analysed in more detail. In the real network-use case, it will be important to have a separate SDN application for the routing policy decisions (which does not need to be shortest path based, but for example user policy based) and the separate for the traffic redirection. This is related to our objective that the SDNtrap system should not change and influence any flow rules already installed in the network. Also, it is worth adding that the networks usually have some kind of layered or hierarchical structure, where network devices, in this case OpenFlow switches, have different roles (e.g. access/aggregation, distribution, core, etc.). It is also important to understand on the device upon which it is appropriate to have function of the security redirection.



Redirection on aggregation layer switches
Internet
Original traffic
Duplicated and redirected
Duplicated traffic on the same link
Redirection on distribution layer switches
Internet
Original traffic
Duplicated and redirected
No duplication of the traffic
Missed interesting traffic flow

Figure 3.2: Traffic duplication and redirection points in the network

For each definition of traffic flow that needs to be checked (flow matching rules), SDNtrap should identify the source of the traffic (i.e. IP address or prefix, MAC address, etc.), the closest OpenFlow switch and its port toward the source. The closest OpenFlow switch can be identified from the information available on an OpenFlow controller, for example network topology information and/or already configured flow tables on OpenFlow switches. If the interesting traffic definition does not identify specific source (for example matches all HTTP and DNS traffic), than SDNtrap should reconfigure all OpenFlow edge switches and their edge ports (ports toward hosts or servers) to match that traffic.

3.1.1.3 *Usage of Multiple OpenFlow Tables*

To be able to duplicate and redirect traffic and not to change or influence regular traffic-forwarding decisions already in place, multiple OpenFlow tables should be used (as shown in Figure 3.3). The first OpenFlow Table in the pipeline processing of the OpenFlow switch should be dedicated for the security redirection rules and will have OpenFlow rules that will match interesting traffic defined by user. All other rules should be placed or moved to other OpenFlow tables, so they can be processed after security redirection rules. In the initial state, the first table should contain the Table-Miss action, which directs the packet to a subsequent table. Alternatively, 'match all' with the lowest priority and Goto-Table instruction can be set in order to match any traffic that is not matched as interesting from the SDNtrap point of view. Processing the packet in the first table also provides the possibility to manage the traffic based on the original source/destination addresses (which can be altered in subsequent tables).

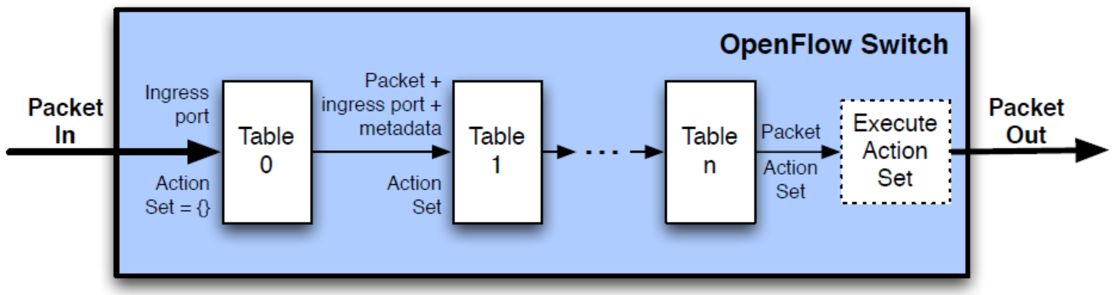


Figure 3.3: Multiple OpenFlow tables processing by the OpenFlow switch

3.1.1.4 *Matching Flow Rules for Interesting Traffic*

The matching part of the OpenFlow security redirection rules will be carried out according to the definition of the interesting traffic. Additionally, these rules should include an ingress port matching, so that only traffic that is coming from edge ports that connects hosts can be duplicated and redirected. In this way, the possibility of duplicating and redirecting same traffic flows on multiple OpenFlow switches in the network is avoided.

3.1.1.5 *Actions for Matched Traffic*

The action part of the security flow rules will consist of multiple instructions and actions:

1. **Instruction Apply-Actions** immediately apply a defined Action List. This instruction is specified in OpenFlow Specification 1.1 [OF1.1.0] and from OpenFlow 1.2 to 1.4, it is specified as an optional instruction, which means it can be optionally supported by a switch. The Actions List for this instruction should be as follows:
   1.1. **Label interesting flow**: done by the push-MPLS or push-VLAN actions, which label redirected flow with a 'security' MPLS label or VLAN tag. Labelling of the redirected packets enables the efficient forwarding through the network toward the security appliance. The objective is to label the duplicated and redirected packet that is destined for the security appliance, so the rest of the OpenFlow switches in the network can forward this packet solely based on that label. Our proposal is to use network-wide allocated labels for each security appliance that is used in the network. The label is then realised as a network-wide MPLS label allocated for this purpose, VLAN tag (number) or even specifically allocated destination MAC address. This action is realised with OpenFlow "push-MPLS" action or "push-VLAN" action.
   1.2. **Duplicate and redirect interesting flow**: action Output is used to forward duplicated and "labelled" packet toward security appliance on the appropriate OpenFlow switch port.
   1.3. **Remove label:** pop-MPLS or pop-VLAN actions change redirected / modified packet back to the original packet, so it can be used for regular forwarding rules in next flow tables.
2. **Instruction GoTo-Table** to continue processing an original packet by the regular traffic flow rules in next or subsequent flow tables. This action is used to forward a regular packet toward its real destination.

In case that the packet doesn't match any flow rule in the security redirection table, the packet should also be processed further by other flow tables in the OpenFlow switch. This is realised with the Table-Miss entry in the security redirection table that should point to the next table in the pipeline, as shown in Figure 3.4, below.

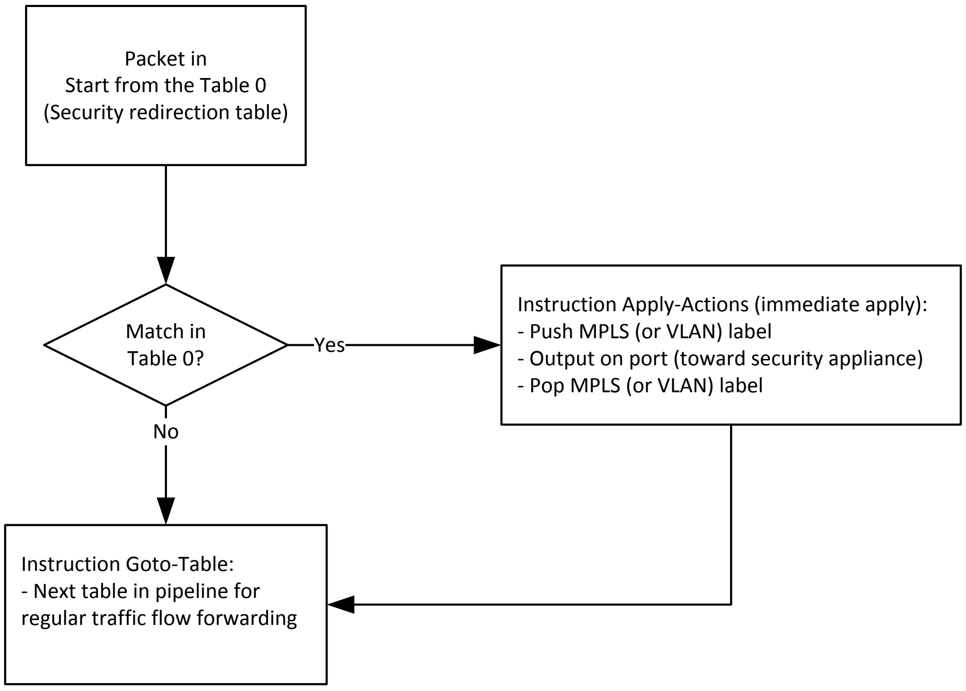

Packet in
Start from the Table 0
(Security redirection table)
Match in
Table 0?
Yes
No
Instruction Apply-Actions (immediate apply):
- Push MPLS (or VLAN) label
- Output on port (toward security appliance)
- Pop MPLS (or VLAN) label
Instruction Goto-Table:
- Next table in pipeline for
regular traffic flow forwarding

Figure 3.4: Processing the packets in OpenFlow “Table 0”

3.1.1.6 *Forwarding of the Interesting Traffic*

Further forwarding of the duplicated and redirected interesting traffic through the network is carried out solely based on the security MPLS label or VLAN tag. This approach is very efficient and scalable, because a large number of redirected flows can be forwarded according to the single OpenFlow rule matching the security label on all intermediate OpenFlow switches. It also provides full control of traffic forwarding independent of the underlying vendor implementations of routing and forwarding mechanisms (such as ERSPAN or PSAMP). The requirement for this approach is that the path for the redirected traffic flow from each OpenFlow switch to security appliance needs to be determined. These paths can be created manually by the administrator, can be calculated based on algorithms (e.g. shortest path) or can be determined/calculated by OpenFlow controller or other SDN applications. As previously stated, path calculation algorithms are out of the scope of this work. When the paths from each OpenFlow switch toward the security appliance are determined, SDNtrap should install flow rules on all intermediate OpenFlow switches that will forward labelled packets. There are a number of other important issues related to this OpenFlow rule:

- **The highest priority rule:** This rule should be installed as the “first” OpenFlow rule (i.e. with the highest priority) in the security redirection table (Table 0) of the OpenFlow switches. The reason is to avoid additional duplication and redirection of the already redirected traffic on intermediate switches on the path from first OpenFlow switch to the security appliance.
- **Matching conditions:** As a precaution from the looping packets or insertion of fake packets from hosts, besides matching the security MPLS label, the OpenFlow rule should have an additional matching condition that matches downstream ports connected to other OpenFlow switches in the network. An additional rule can be created that drops packets with the allocated MPLS label coming from all other ports (edge ports that connect hosts).

- **Egress switch:** The OpenFlow switch closest to the security appliance should have a flow rule that matches specifically matches allocated MPLS label, with additional action for popping the label (OpenFlow *Pop* action) and forwarding flows to security appliance. This additional action ensures that security appliance will receive original flow, without MPLS label, on its tap interface.

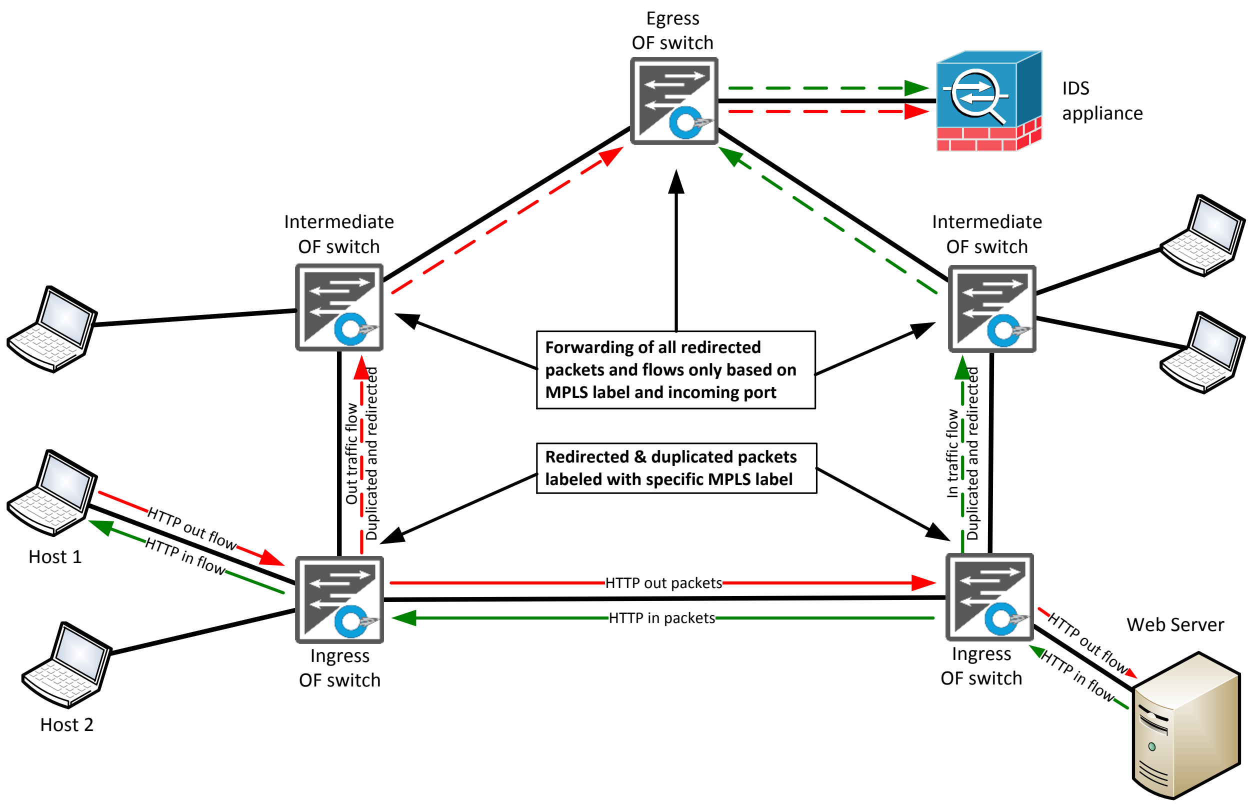


Figure 3.5: Forwarding of the redirected packets based on the MPLS label

3.1.1.7 *Conclusion*

The proposed solution is using features of the OpenFlow specification 1.1 [OF1.1.0] and above, specifically, multiple OpenFlow tables and the optional Apply-Actions instruction. The assumption of this solution is that forwarding in the network would be mostly realised with flow rules preinstalled on the OpenFlow switches by the OpenFlow controller, so a minority of the traffic flows will be redirected to the OpenFlow controller for decisions (or unknown traffic will be dropped). The SDNtrap application shares the information with OpenFlow controller or other SDN application for calculation/creation of forwarding path from every OpenFlow switch to the security appliance and for identification of the OpenFlow switches closest to the source of the interesting traffic.

The SDNtrap concept is designed to be as transparent for the existing traffic as possible, although it requires reservation of Table 0 for security purposes, so the controller cannot use it for applying ‘normal’ flow rules. This concept is also scalable as forwarding security-redirected traffic on intermediate OpenFlow switches is solely based on a single rule-matching, predefined-security MPLS label or VLAN tag. Further steps may include additional communication between the security appliance and SDNtrap in order to filter undesired traffic.

3.1.2 Redirection of the Complete Traffic Flow

In the second scenario, Firewall or IPS security appliance is introduced into the network for the purpose of the security analysis and manipulation or filtering of the network traffic. The network administrator has a requirement is to define specific traffic flows of interest and to redirect and forward them to the security appliance. In Figure 3.6, an OpenFlow-enabled network in the "converged state" is considered, where the OpenFlow controller has already configured appropriate flow tables of OpenFlow switches.

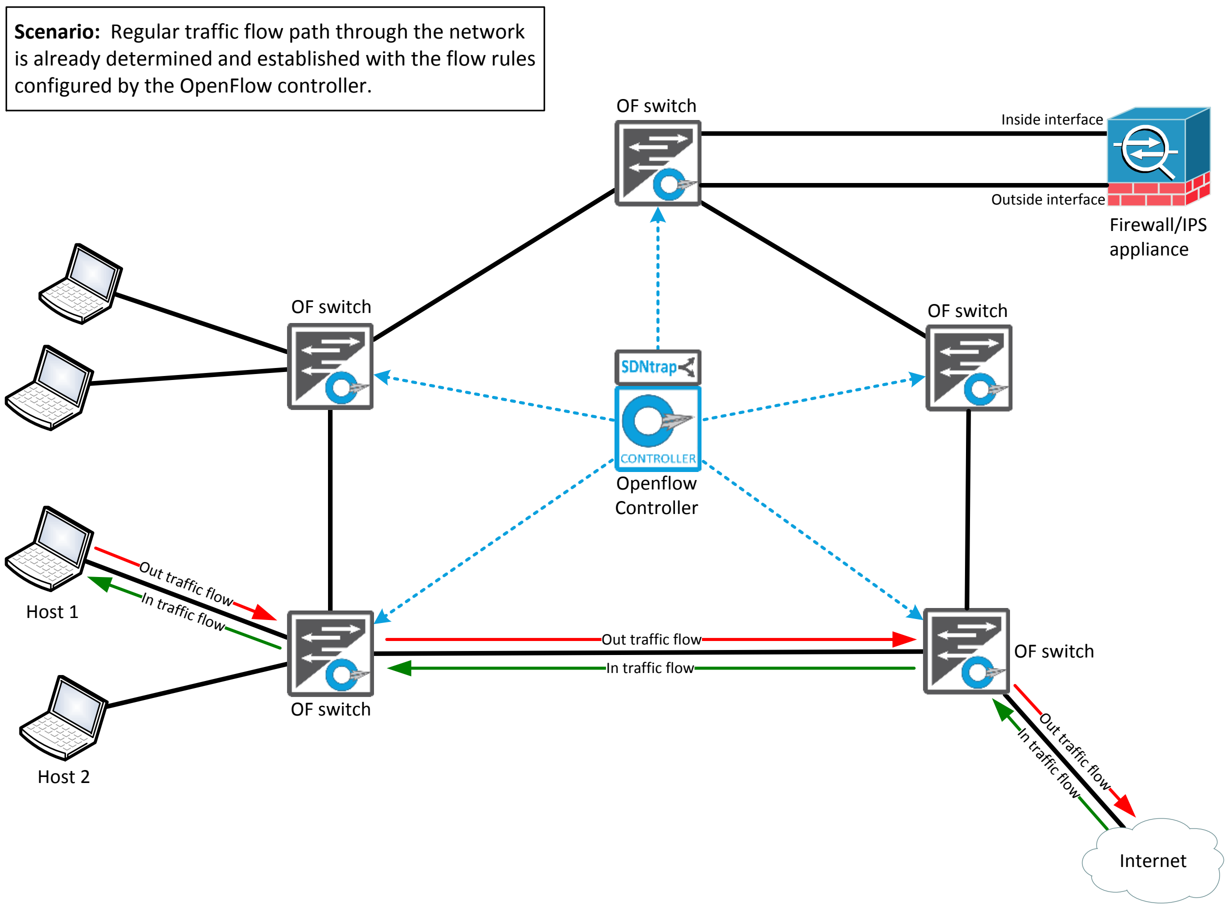


Figure 3.6 Redirection of the complete traffic flow scenario (regular traffic flow)

It is very important to note that under normal conditions, regular traffic-forwarding decisions preconfigured in the network must not be changed. These OpenFlow rules can be configured by other OpenFlow controller applications (e.g. routing application) or administrator, so it is important not to change OpenFlow forwarding rules already configured on the switches. SDNtrap should only compose and apply additional flow rules in order to enforce redirection of the complete traffic flow.

A scenario is illustrated in

Figure 3.7 where the configured interesting traffic flow is redirected toward the security appliance. A simple scenario can be considered where Firewall or IPS devices have two interfaces labelled as Inside and Outside interface and traffic is forwarded between those two interfaces, according to the security policy configured on the security appliance. It is important to note that one direction of the traffic will be redirected to enter the Inside interface of the security appliance, and after processing, will be forwarded on the Outside interface according to the security policy. The other direction of the traffic flow will be redirected to the Outside interface, and after processing will be forwarded on the Inside interface of the security appliance.

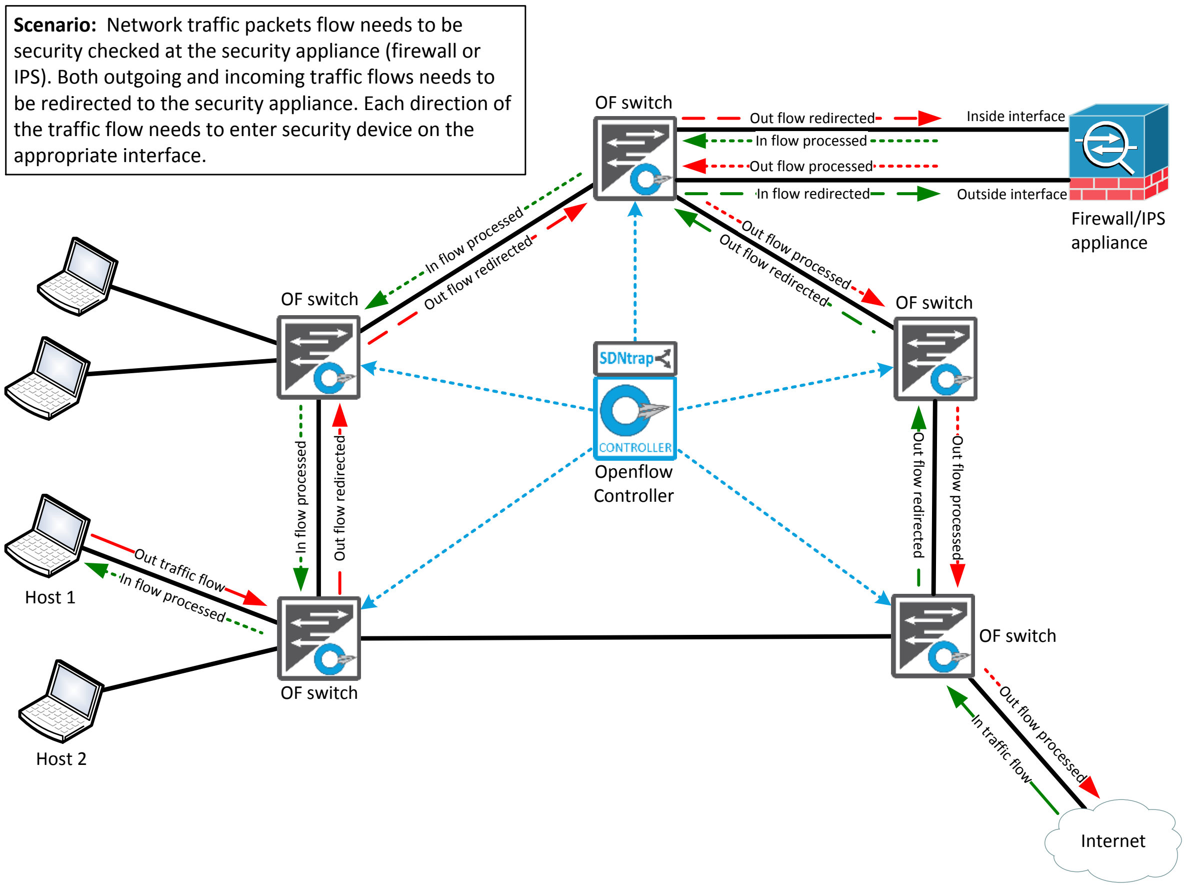


Figure 3.7 Redirection of the complete traffic flow scenario (regular traffic flow)

3.1.2.1 *Proposed Solution for Redirection of the Complete Traffic Flow*

The proposed solution uses features of the OpenFlow Specification 1.1 and above, specifically multiple OpenFlow tables and optional “Apply-Actions” instruction. For the successful redirection of traffic, the proposed solution needs the following information to be defined by an administrator:

- **Interesting traffic** – Definition of the traffic that has to be redirected. It should contain matching conditions for traffic flows (e.g. source/destination addresses, ports), together with the specification of the security appliance interface (*Inside* or *Outside*), where this traffic needs to be forwarded.
- **Location of the security appliance interfaces** – point of attachment in the network (port and switch) for both *Inside* and *Outside* interfaces.

- **Security labels** – network-wide MPLS labels or VLAN tags that will be used for labelling interesting traffic. In this scenario, two labels or tags are used, one for *Inside* and one for *Outside* interface of the security appliance.

The proposal for a complete traffic-redirection solution uses the same concepts and features as the previous solution for the security duplication and redirection of the traffic (see Section 3.1.1.1). In the following sections, relevant differences between the solution for redirection of the complete traffic flow and duplication and redirection of the traffic will be described.

3.1.2.2 *Point of Duplication and Redirection*

It is proposed that the traffic redirection should be carried out on the OpenFlow switch closest to the source of the traffic. For each traffic flow defined as interesting by the administrator, the SDNtrap should determine the closest OpenFlow switch to the source, where the security redirection flow rules will be applied.

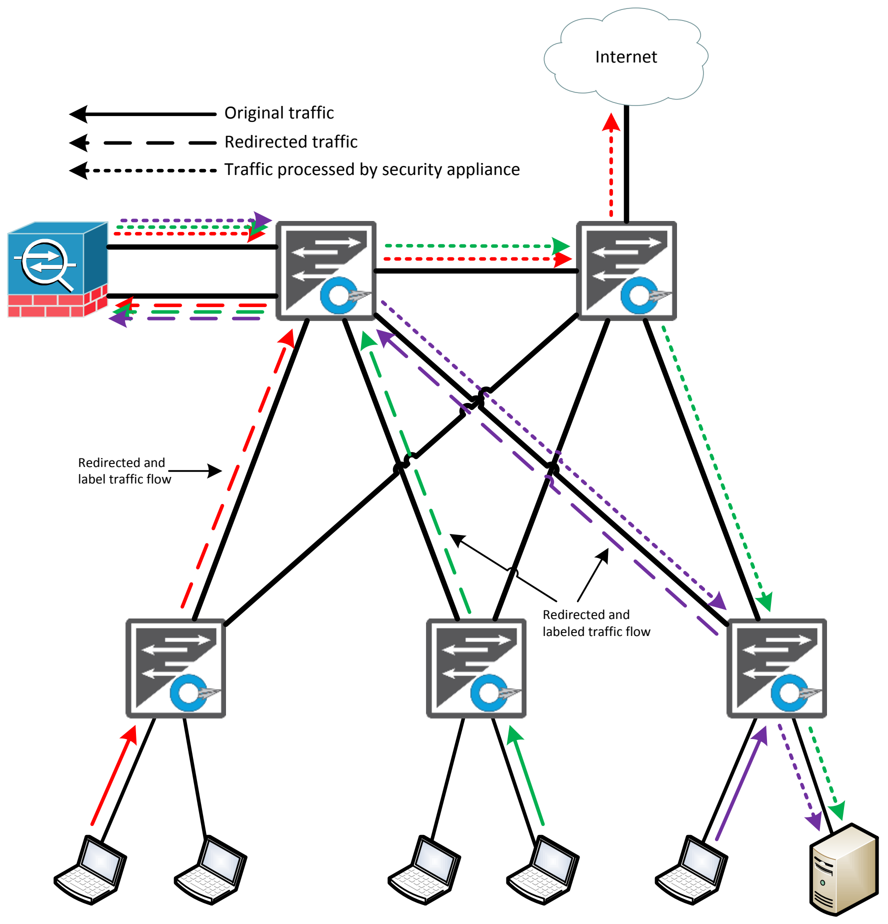

Internet
Original traffic
Redirected traffic
Traffic processed by security appliance
Redirected and label traffic flow
Redirected and labeled traffic flow

Figure 3.8: Complete traffic redirection of different traffic flows

3.1.2.3 *Usage of Multiple Flow Tables*

This scenario does not require duplication of traffic flows, so the solution does not actually need multiple tables. However, for this solution, the security redirection rules must be processed before “regular” traffic forwarding rules. This can be accomplished by using multiple flow tables, as described in the first scenario (see Section 3.1.1), but it can be also accomplished by using higher priority for security redirection rules. If the solution prioritises security redirection rules over regular forwarding rules, then the security redirection rules will be matched and processed before regular rules. In case there is no rule match among higher priority security rules, the OpenFlow switch continues matching regular rules with lower priority. With this approach only one flow table is used, so this solution is compatible with OpenFlow specification 1.0 [OF1.0.0]. If the solution uses multiple tables, then it will need OpenFlow switches that are compatible with the OpenFlow Specification 1.1. Additionally, there should be Table-Miss entry in the security redirection table, as explained in the previous scenario.

3.1.2.4 *Matching Flow Rules for Interesting Traffic*

The matching part of the OpenFlow security redirection rules will be carried out according to the definition of the interesting traffic. Additionally, it is recommended that these rules include an ingress port matching, so that only traffic coming from edge ports that connects hosts can labelled and redirected. This avoids the possibility of labelling and redirecting the same traffic flows on multiple OpenFlow switches in the network. In this scenario, traffic flow is matched and traffic redirected to two different interfaces on the security appliance, Inside and Outside. For this reason, when defining matching rules for interesting traffic, the administrator should specify a security appliance interface where this traffic should be redirected.

SDNtrap application can introduce a simple convention for the case of bidirectional traffic flows: that the "first" specified address is behind the Inside interface and that the "second" specified address is behind the Outside interface. In that case, two rules can be created:

- Traffic flow from source "first" address to destination "second" address is redirected to the *Inside* interface.
- Traffic flow from source "second" address to destination "first" address is redirected to the *Outside* interface.

3.1.2.5 *Actions for Matched Traffic*

The action part of the security flow rules will consist of the following instructions and actions:

1. Instruction Write-Actions (merge actions to the Action Set). This instruction is included in OpenFlow Specification 1.0 and from OpenFlow Specification 1.2 to 1.4, is a required instruction, which means it must be supported by OpenFlow switch. The actions for this instruction should be as follows:

   1.1. **Label interesting flow**: This is done by the Push-Tag action, which labels the redirected flow with a 'security' MPLS label or VLAN tag. It is used for labelling of the redirected packets for the efficient forwarding through the network toward the security appliance, in the same way as done in the first scenario. The objective is to label the redirected packet that is destined for the security appliance, so the rest of the OpenFlow switches in the network can forward this packet solely based on that label. Our proposal is to use network-wide allocated labels for each interface of a security appliance that is used in the network. Proposed usage of network-wide MPLS labels allocated for this purpose, but this can be realised with specifically allocated VLAN tags (number) or even specifically allocated destination MAC addresses.

   1.2. **Redirect interesting flow**: action Output to forward packet toward security appliance. This is used to forward a labelled packet toward the security appliance on the appropriate OpenFlow switch port.

Because the action part of the proposed security redirection rules does not have a GoTo-Table instruction, these actions will be executed immediately after matching the rule and redirecting traffic. It is important to note that in the case of using MPLS labels for labelling of the redirected traffic flows, OpenFlow switches compatible with the OpenFlow specification 1.1 will be needed. In the case of using VLAN tags to label the redirected traffic flows, OpenFlow switches compatible with the OpenFlow specification 1.0 will be sufficient.

3.1.2.6 *Forwarding of the Interesting Traffic*

Further forwarding of the duplicated and redirected interesting traffic through the network is carried out solely based on the security MPLS label or VLAN tag, as described in the first scenario.

Flow rules in single flow table example			
In port	**Match condition**	**Priority**	**Actions**
from downstream switch	SDNtrap MPLS label for *Inside*	65535	fwd toward Firewall *Inside* interface
from downstream switch	SDNtrap MPLS label for *Outside*	65535	fwd toward Firewall *Outside* interface
any port	SDNtrap MPLS label for *Inside*	65534	drop
any port	SDNtrap MPLS label for *Outside*	65534	drop
edge port	sec interesting traffic	65000	push MPLS *Inside,* fwd to Firewall *Inside*
edge port	sec interesting traffic	65000	push MPLS *Outside,* fwd to Firewall *Outside*
..	…	…	…
regular rule	regular rule	32768	regular rule
regular rule	regular rule	32768	regular rule
…	…	…	…

Figure 3.9 Example of the flow table 0 of an OpenFlow switch

Figure 3.9 is an example of the OpenFlow switch flow table in the case that a single flow table is used for this scenario. With the highest priority of 65535 are the rules for forwarding already redirected and labelled traffic flows coming from downstream switches. Priority 65534 is used for rules that should drop any packets labelled with allocated security MPLS labels coming from all other interfaces where they are not supposed to come, avoiding looping or malicious packets. For other security redirection rules, priority of 65000 is used. All other regular forwarding traffic rules are using priority 32768.

3.2 Conclusion

In this scenario, there is no duplication of the redirected traffic, but the same traffic flow is redirected toward the security appliance. For that reason, the proposed redirection on ingress OpenFlow switches will not result in the possible duplication of the traffic and congestion on upstream interfaces.

The solution for this scenario can have two possible implementations:

- An implementation compatible with OpenFlow Specification 1.0 that uses a single flow table and VLAN tags for labelling of redirected traffic.

- An implementation compatible with OpenFlow Specification 1.1 that uses multiple flow tables and/or MPLS labels for labelling of redirected traffic, since both of these features are not supported in OpenFlow 1.0.

4 Monitoring

4.1 Introduction to Monitoring in OpenFlow and SDN Environments

Software Defined Networking (SDN) infrastructures introduce several new challenges to monitoring processes as well as to operations, administration and management (OAM). One such challenge is that applications are no longer tied to dedicated physical resources and with new levels of abstraction monitoring and OAM cannot be limited to physical infrastructures, but must also consider various layers and individual applications with their traffic flows. In contrast to traditional networking, another major challenge (as well as an opportunity) for SDN-based networks is that monitoring and OAM information can be used as direct feedback to the controller to automatically change network behaviour and adjust flow control based on the retrieved information.

The following sections look at monitoring in SDN and OpenFlow environments in detail: Section 4.2 starts out with an investigation of flow-based monitoring in OpenFlow environments. Section 4.3 provides solutions for flow monitoring by exporting relevant information over legacy NetFlow/IPFIX protocols. The interested reader can find more information in 5Appendix B 'Overview of Business Solutions for SDN Monitoring' and 5Appendix C 'OpenFlow Vendor Overview: Optional OpenFlow features and Features for Monitoring and Statistics'.

4.2 Flow-based Monitoring in OpenFlow Environments

Traditional monitoring solutions often rely on NetFlow/IPFIX or sFlow (sampled Flow)-based traffic analysis. NetFlow [CIS-2014] was originally proposed by Cisco and offers IP traffic statistics collected on a router interface such as source IP and destination IP, class of service attributes, protocols, bandwidth utilisation or peak usage times that allow a network administrator to determine causes of congestion. NetFlow was superseded by the Internet Protocol Flow Information eXport (IPFIX), as described in RFC 5101 [RFC-5101] and RFC 5102 [RFC-5102]. With NetFlow/IPFIX based traffic analysis the IP flow information that was collected by a NetFlow-enabled router is sent to an external server where the collected information is analysed and interpreted. sFlow is a similar sampling technology that is supported by a large consortium of vendors producing network components and is especially suitable for high-speed networks, as it offers random sampling [SFL-2014]. In contrast to passive monitoring with the Simple Network Management Protocol (SNMP) [RFC-3410], both NetFlow and sFlow allow further insight on application-related details [PAT-2010].

As SDN controllers need to make routing decisions for flow control based on current network conditions, they can certainly benefit from NetFlow or sFlow data analysis [PLI-2013a]. The following section focuses on how SNMP/NetFlow/IPFIX/sFlow-based monitoring can be used in OpenFlow environments.

4.2.1 Monitoring with sFlow

Just like NetFlow sFlow [SFL-2004] is a mechanism for monitoring that does not rely on network probes, but allows the network administrator to analyse traffic based on flows. sFlow was first defined in RFC 3176 [RFC-

3176] and is capable of randomly sampling one packet out of a configurable number of packets on an interface, whereas NetFlow captures accurate total byte readings between hosts [REE-2008].

In NetFlow and IPFIX protocols, flow records are built on network flows that have the same attributes, such as ingress interface, source and destination IP, source and destination TCP/UDP port and IP ToS [PLI-2013b]. During processing, these packet fields are extracted and a hash function is computed over these fields to allow the lookup of an already existing entry for a flow in the flow cache. Existing flow records are brought up to date and new flows records are started for new flows. Periodically or after a timeout, the flow records are sent to the flow collector for traffic analysis. In NetFlow, these flow records represent the number of active connections between hosts [POW-2012].

sFlow, on the other hand, simply samples packet headers and sends this information for analysis. Without this need for building flow records for active connections and flushing flow caches, there is considerably less delay involved when compared to NetFlow, as the monitoring information on sampled packets is immediately available for analysis. The sampled information in sFlow is also not limited to the first 1200 bytes of a packet as in NetFlow and with high traffic rates and frequent sampling settings, the sFlow record rate for the sFlow collector can become large.

Figure 4.1 describes the two methods of sampling that are offered in sFlow:

- In Flow-based sampling, an sFlow-enabled port samples the packet statistics and sends it to the collector.
- In counter-based sampling, sFlow uses a polling function that periodically obtains standard interface counters for network analysis from its sFlow agents in the switches.

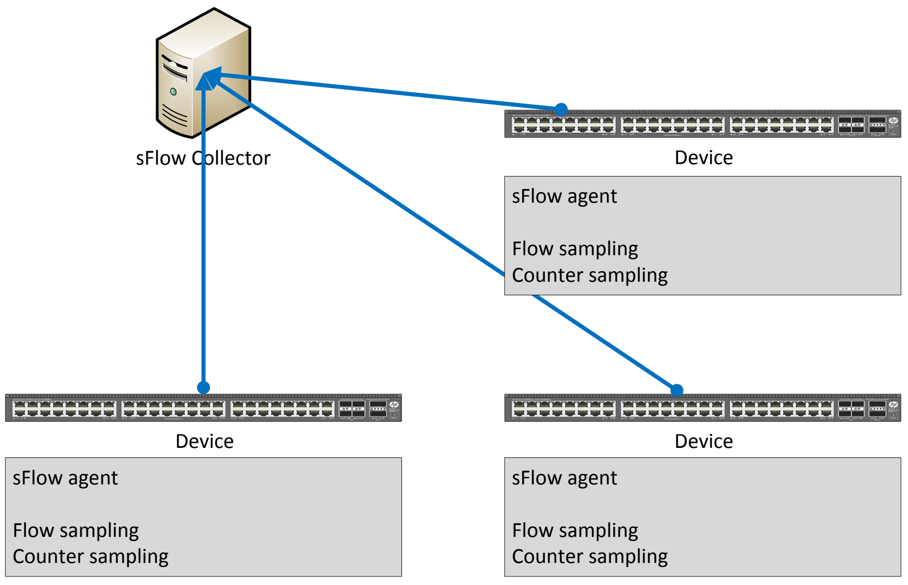


Figure 4.1 Flow-based and counter-based sampling in sFlow; sFlow agents are embedded in network components and capture packet samples and send these sFlow datagrams to the sFlow collector [REA-2013].

4.2.2 sFlow vs. NetFlow/IPFIX Monitoring in SDN

For an OpenFlow or SDN controller to be able to effectively take action upon feedback from monitoring it is important that the delay of the different components involved in the feedback process is kept to a minimum. For example, if a controller is to react to a disturbance in the system, several different types of delays must be taken into account [PLI-2013c], including:

- Measurement and monitoring systems need time to identify a problem.
- There may be a need for a planning delay to determine what type of action is to be taken.
- A configuration delay may be required to put the appropriate action in place.
- Propagation delay may occur while new route changes are taking effect and are propagated through the system.

Monitoring with NetFlow/IPFIX also introduces significant measurement delay, because measurements are not reported until the connection ends, which makes the length of the delay proportional to the duration of the connection [PLI-2011]. The packet sampling technique of sFlow, on the other hand, introduces significantly less delay, as the sampled packets are immediately available for traffic analysis and therefore allow for the rapid detection of large flows [PLI-2013b]. For this reason, sFlow is being used in anomaly detection in SDN environments [GIO-2014], although its sampling-based mechanism offers less accuracy if not enough packets are being sampled [PLI-2009].

4.3 Flow Monitoring in OpenFlow Environment Using NetFlow/IPFIX

4.3.1 Introduction

One of the most important parts of network monitoring is the monitoring of traffic structure regarding source and destination of the traffic, applications, content and other parameters comprising network traffic. Today's predominant technology for such monitoring is flow monitoring using NetFlow.

There are number of network devices that support NetFlow technology, as well as number of applications that collect and analyse NetFlow exported data. These technologies have been used by a number of service providers and enterprises as a valuable asset in identification and understanding of traffic usage by applications and users.

This part of the JRA2 Task 1 activity aimed to investigate the potential of flow monitoring using NetFlow/IPFIX protocols in OpenFlow environment. Research has investigated how to use NetFlow technology, whether it can be used as is or it has to be extended, comparison of OpenFlow match fields and NetFlow supported fields.

The main idea of the research was to monitor traffic flows in the OpenFlow network and to export relevant monitoring data over the well-known NetFlow/IPFIX protocols.

The result of the research is the proposal of the OpenFlow to NetFlow (OF2NF) application to be implemented on top of the OpenFlow controller to be used in both a reactive and proactive OpenFlow environment.

4.3.2 History of NetFlow

NetFlow was originally developed as a packet-switching technology for Cisco routers, with Cisco IOS software implementation for the Cisco 7000, 7200 and 7500 [NF-wiki]. The NetFlow switching technology, also known as fast switching, was created to improve routing performance.

The first packet of a traffic flow would be sent to the CPU to determine the result of the forwarding process (e.g. routing table lookup, ACL lookup) and then to create a NetFlow switching record. The NetFlow switching record would be used for all subsequent packets of the same flow for faster forwarding, until the expiration of the flow. The NetFlow switching record was also known as a route cache record, by caching a result of the routing lookup. Once the router already has a NetFlow cache records, it was only needed to count the bytes and packets of the forwarded flows and to export them to the monitoring server. The protocol to export this information was named NetFlow.

NetFlow switching technology for packet forwarding on routers was replaced by Cisco Express Forwarding (CEF), but NetFlow as flow monitoring technology was retained and further developed.

Further information on NetFlow can be found in the more detailed document Flow monitoring in OpenFlow environment using NetFlow/IPFIX protocols [RFC-5101] NetFlow specifications [NFv5] [NFv9] and IPFIX specifications [IPFIX] [IPFIX-e].

4.3.3 OpenFlow Capabilities Regarding Flow Monitoring

4.3.3.1 *OpenFlow Overview*

OpenFlow is the first standard communications interface defined between control and forwarding layers of SDN architecture [ONF-WP]. OpenFlow allows direct access to and manipulation of the forwarding plane of network devices, such as switches and routers. Manipulation of the forwarding plane of network devices is done by the software of the OpenFlow controller.

OpenFlow uses the concept of flows to identify network traffic based on pre-defined match rules that can be statically or dynamically programmed by the OpenFlow controller. Match rules are defined by matching fields from various network protocols such as Ethernet/802.1Q, IPv4/IPv6, TCP/UDP, MPLS, etc.

The OpenFlow protocol is implemented on both sides of the interface between network devices and the OpenFlow controller.

4.3.3.2 *OpenFlow Network Design Regarding Flow Installation*

One of the fundamental difficulties inherent in the design of an OpenFlow production network is the traffic forwarding process and flow rules installation by OpenFlow controller. Three approaches to this are identified below [NS-blog]:

- **Reactive** – The OpenFlow switch does not have any flow rules configured in its flow tables, except the “fallback” rule that will forward the packet to the OpenFlow Controller. When the first packet of the traffic flow comes into the switch, it will forward the packet to the OpenFlow controller for the decision regarding this flow. The OpenFlow controller will make the decision regarding that flow and install an appropriate matching flow rule with appropriate actions onto the switch. The switch will forward all subsequent packets of the same flow according to the rule installed by the OpenFlow controller.

- **Proactive** – The OpenFlow controller will not wait for the specific traffic flows or types of traffic to appear in the network, but will install appropriate flow rules onto switches with their forwarding actions, covering all the traffic that can appear in the network in advance.
- **Hybrid** – The hybrid design is supposed to combine the advantages of the previous two approaches and should provide a flexibility and control of reactive approach for particular traffic, while preserving low-latency forwarding and resource-saving advantages of the proactive approach for the rest of the network traffic.

4.3.3.3 *OpenFlow Capabilities Regarding Flow Monitoring*

This section identifies important OpenFlow capabilities that can be used for the purpose of effective flow monitoring and accounting.

Flow tables

OpenFlow switch Flow Tables consist of the following entries:

- **Match fields –** key fields used to define a flow.
- **Priority –** important for the appropriate prioritisation and matching of the counting flows.
- **Counters –** providing flow statistics about the packets and bytes matched by the flow.
- **Instructions –** actions for flow processing and forwarding.
- **Timeouts –** idle and hard timeout for flow expiration, very important for flow export
- **Cookie –** data value chosen by the controller that can be used to filter flow statistics, modification or deletion. It is not used when processing packets.

Counters

According to the OpenFlow Specification 1.4, counters are maintained for each flow table, flow entry, port, queue, group, group bucket, meter and meter band [OF1.4.0]. Counters are either required or optional, depending on the specification. Counters that are important for flow monitoring purpose are "per flow entry" counters:

- **Received Packets –** 64-bit counter – Optional
- **Received Bytes –** 64-bit counter – Optional
- **Duration** (in seconds) **–** 32-bit counter – Required
- **Duration** (in nanoseconds) **–** 32 bit counter – Optional

Per-flow entry counters for bytes and packets are still optional.

Flow Expiry and Removal

Flow rules can be removed from flow tables in three ways: by switch flow expiry mechanism, by the request of the controller, and by the switch eviction mechanism.

An OpenFlow switch runs a flow-expiry mechanism, which is controlled by configuration of flow rules. Each flow rule has two associated timeouts associated:

- **Idle timeout** – Flow rule will be removed from the flow table if it doesn't match any packet during the idle timeout period.
- **Hard timeout** – Flow rule will be removed from the flow table after expiration of the hard timeout, regardless of how many packets it has matched.

Both timeouts need to have non-zero values in order to be active.

It is important to note that these OpenFlow timeouts are very similar to the NetFlow timeouts.

Messages

Flow Removed Message

When a flow rule is removed by the OpenFlow switch, it has to check if the flow rule has the OFPFF_SEND_FLOW_REM flag. If this flag is set, the switch must send the flow-removed message to the controller. The OpenFlow Flow Removed message structure can be found in [OF1.4.0].

The important part of the Flow Removed message are Bytes and Packets counter fields that represent statistics related to the removed flow rule and that can be used for flow monitoring. Additionally, the structure of the Flow Removed message matches the Flow Rule containing values of the matching fields and information about flow duration, which is also important for monitoring purposes, table ID and cookies, which can be used to filter messages from flows of interest.

Individual Flow Statistics Message

Among a number of messages that are used in OpenFlow protocol for different communication between OpenFlow controller and switches, there is a message used for getting individual flow statistics that is a Multipart message from the Controller-to-Switch group of messages.

Multipart messages are used to send requests or replies that potentially carry a large amount of data that would not always fit in a single OpenFlow message, (which is limited to 64KB). They are primarily used to request statistics or state information from the switch. More information about the messages can be found in OpenFlow specification [OF1.4.0]

4.3.4 Scenario for Flow Monitoring of the OpenFlow Network Using NetFlow/IPFIX

Although use of NetFlow and OpenFlow in today's networks is different by nature (the former is used for flow monitoring and the latter for programming flow forwarding), similarities can be found between the original usage of NetFlow switching and OpenFlow in a reactive approach described in the previous section. Both of these technologies has resource intensive processing on the first packet of the traffic flow, installation of the cache record / flow rule matching subsequent packets of the traffic flow and forwarding according to the decision/action.

On the other side, today's networks use NetFlow as a flow monitoring technology and protocol, while OpenFlow is used as a traffic forwarding technology and protocol.

This section describes the main idea how to use NetFlow/IPFIX protocols for flow monitoring of the OpenFlow enabled network. The objective is to use existing flow collectors and flow analysing applications for monitoring of the OpenFlow environment.

The research assumes an environment where OpenFlow switches do not support NetFlow/IPFIX export of flow statistics. In this situation, the objective is that the OpenFlow controller (or controller's application) collects flow statistics from OpenFlow switches and to export them over NetFlow or IPFIX protocols to the flow analysing applications. Continuous collection of flow statistics for security or accounting purpose for all or the part of the network traffic (which should be defined by user) is under consideration. Figure 4.2 represents the proposed scenario.

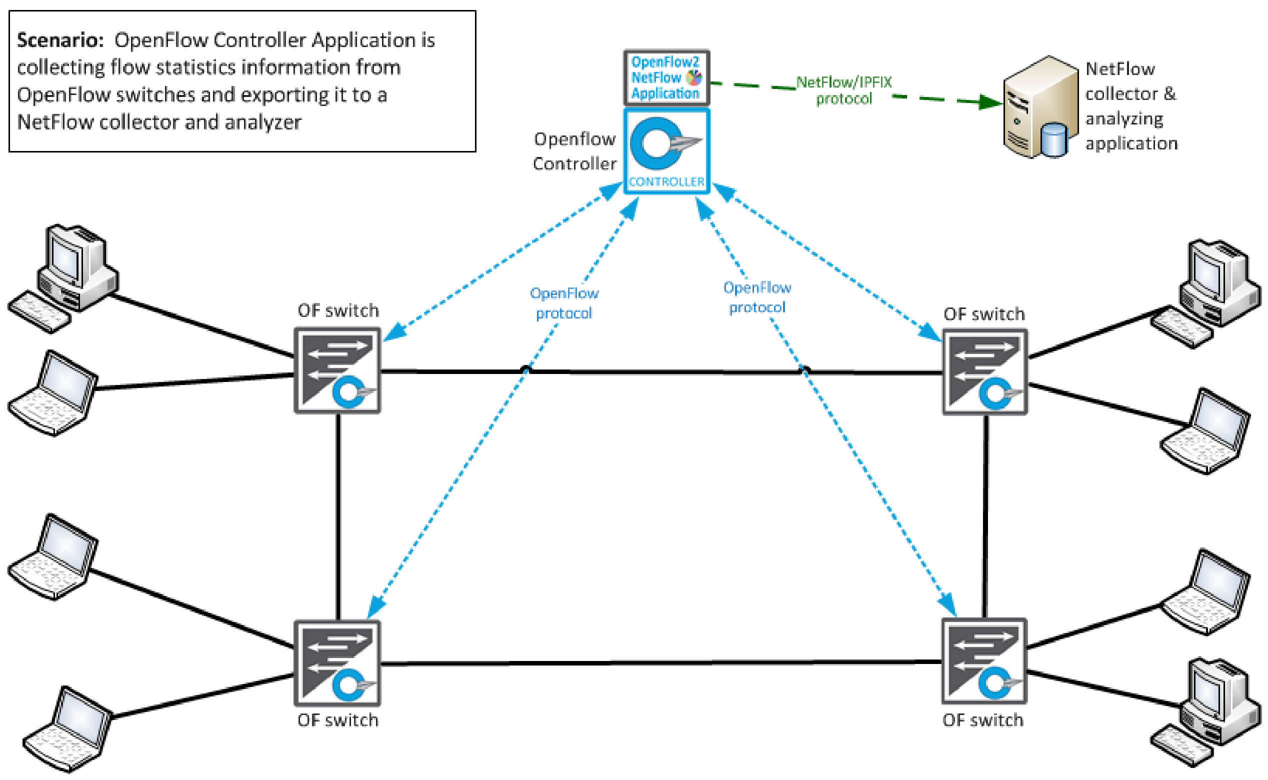

Scenario: OpenFlow Controller Application is collecting flow statistics information from OpenFlow switches and exporting it to a NetFlow collector and analyzer
OpenFlow2 NetFlow Application
Openflow Controller
CONTROLLER
NetFlow/IPFIX protocol
NetFlow collector & analyzing application
OpenFlow protocol
OpenFlow protocol
OF switch
OF switch
OF switch
OF switch

Figure 4.2: Scenario demonstrating the collection if flow statistics over OpenFlow and exporting it over NetFlow/IPFIX

This research can be divided in two problems:

- Collecting flow statistics from switch flow tables by OpenFlow controller.
- Exporting flow statistics from OpenFlow controller over NetFlow or IPFIX protocols.

4.3.4.1 *Collection of flow statistics over OpenFlow*

After detailed analysis of available solutions for collection of the flow statistics over OpenFlow protocol, this research proposes the use of the Flow Removed message. Use of this message has a very similar paradigm to the NetFlow export mechanism and is better suited to the collection of the OpenFlow statistics that will be sent over NetFlow protocol than use of the OpenFlow Individual Flow Statistics message. More information can be found in Appendix C.

4.3.5 OF2NF Application Concept

The concept of the OpenFlow application that collects OpenFlow flow statistics and exports it over the NetFlow protocol will be described in this section. This application concept is called OpenFlow to NetFlow (OF2NF).

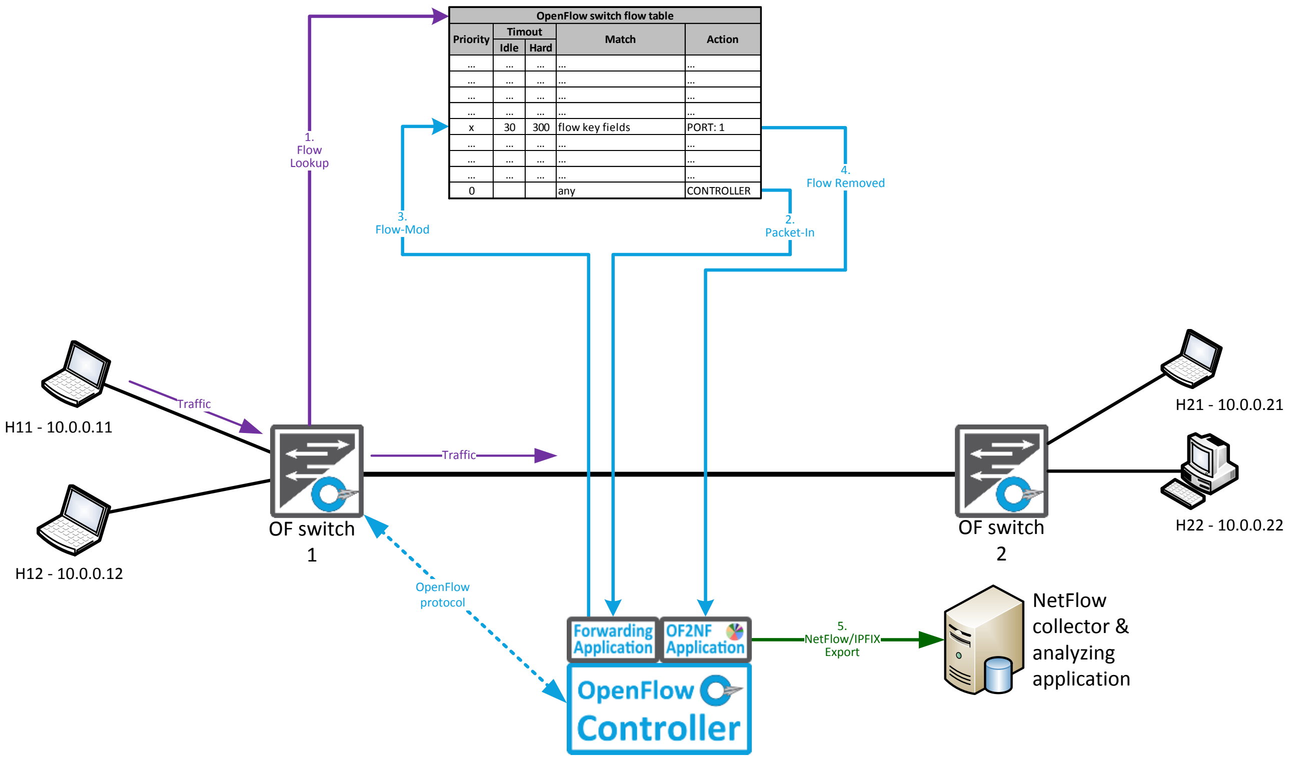
The important objectives and assumptions of the OF2NF applications are:

- The application should be part of an OpenFlow controller or “sit” on top of an OpenFlow controller, either using its API or its northbound interface.
- Use of OpenFlow *Multiple Tables* functionality, which is compatible with OpenFlow Specification 1.1 and above. The support of this functionality is needed by the OpenFlow controller and switches.
- For simplicity, any stateful behaviour or any data exchange with other controllers’ applications should be avoided.
- There are requirements regarding the support of certain OpenFlow features that the OpenFlow controller or controllers’ “Forwarding” application need to support.
- The applications can be deployed in the OpenFlow network environment that uses Reactive or Proactive/Hybrid design where OpenFlow network design influence the design and feature of the OF2NF application.

4.3.5.1 *OF2NF in Reactive OpenFlow Environment Scenario*

The first and simpler scenario for the OF2NF application is in a reactive OpenFlow environment, as shown in Figure 4.3.

In the reactive environment, for each new flow, OpenFlow switches forward packet to the OpenFlow controller (*Packet-In* message) where the packet is analysed and the decision is made about forwarding of the packet. The OpenFlow controller or its “Forwarding” application installs the appropriate flow rule that matches the flow and executes the actions according to the decision (*Flow-Mod* message). When the flow timeout is reached (either by an idle-timeout or hard-timeout) flow is removed from switch flow table and flow removed message is sent to the OpenFlow controller. This message is handled by the OF2NF application and exported to the NetFlow collector over NetFlow protocol.



OpenFlow switch flow table

Priority	Timout		Match	Action
	Idle	Hard		
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
x	30	300	flow key fields	PORT: 1
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
0			any	CONTROLLER

Figure 4.3: OF2NF application in the reactive OpenFlow environment

In this scenario, there are certain requirements for the “Forwarding” application:

- For all flows that are installed, idle-timeout and hard-timeout should be configurable.
- For all flows that are installed, the flow-removed flag should be configurable.

It is also important to have configurable flow-matching fields in the Forwarding application, which will define granularity of the flows. The minimum of the matching fields that should be detected and used/installed by the “Forwarding” application should be key fields of the NetFlow defined flow. It would also be possible to include additional header fields such as MAC addresses, VLAN IDs, MPLS labels and other fields defined by OpenFlow, which will enrich information available to the NetFlow application. Lesser-matching fields rather than key fields may also be included, but this will limit available information for NetFlow analysis, which is contrary to the objectives of this task. It is also possible that some NetFlow application would not work correctly if the key flow fields are not available or equal zero.

4.3.5.2 *OF2NF in a Proactive or Hybrid OpenFlow Environment Scenario*

The second and more advanced scenario for the OF2NF application is in a proactive OpenFlow environment, which can be also applied to a hybrid OpenFlow environment. In the proactive environment, all flow rules are predetermined and installed on OpenFlow switches before actual traffic comes to the switch. More details about this scenario can be found in Section 4.3.4.

4.3.6 Export of Flow Statistics from OpenFlow Controller Over NetFlow/IPFIX

When the OF2NF application receives the flow statistics information, it needs to map and convert it to the appropriate format used by the NetFlow or IPFIX protocols. Some of the flow matching fields and information can be directly mapped to the NetFlow/IPFIX fields, some fields require simple manipulation to be adjusted to the NetFlow/IPFIX format, and some fields are not available from OpenFlow statistics. Table 4.1 summarises the analysis and mapping of NetFlow/IPFIX and OpenFlow fields. It represents how the NetFlow/IPFIX fields are populated and from which OpenFlow available information. The column "*Details*" marks proposed solutions by this research, where "proposed, N/A" means that the proposed solution is not available in current implementation of the OpenFlow specification. More details can be found in the document: Flow monitoring in OpenFlow environment using NetFlow/IPFIX protocols [RFC-5101].

Mapping between NetFlow or IPFIX fields and information available from OpenFlow				
NetFlow/IPFIX Fields			How information is obtained from OpenFlow	
ID	NetFlow Fields	IPFIX Fields	OpenFlow information	Details
1	IN_BYTES	octetDeltaCount	Byte counter from OF Flow Removed message	
2	IN_PKTS	packetDeltaCount	Packet counter from OF Flow Removed message	
4	PROTOCOL	protocolIdentifier	OFPXMT_OFB_IP_PROTO match field	
5	SRC_TOS	ipClassOfService	Combining OFPXMT_OFB_IP_DSCP and OFPXMT_OFB_IP_ECN match fields	
7	L4_SRC_PORT	sourceTransportPort	OFPXMT_OFB_TCP_SRC or OFPXMT_OFB_UDP_SRC depending on the L4 protocol	
8	IPV4_SRC_ADDR	sourceIPv4Address	OFPXMT_OFB_IPV4_SRC match field	
9	SRC_MASK	sourceIPv4PrefixLength	mask from the OFPXMT_OFB_IPV4_SRC match field	
10	INPUT_SNMP	ingressInterface	OFPXMT_OFB_IN_PORT match field	
11	L4_DST_PORT	destinationTransportPort	OFPXMT_OFB_TCP_DST or OFPXMT_OFB_UDP_DST depending on the L4 protocol. In case of ICMP = (256 * OFPXMT_OFB_ICMPV4_TYPE + OFPXMT_OFB_ICMPV4_CODE)	
12	IPV4_DST_ADDR	destinationIPv4Address	OFPXMT_OFB_IPV4_DST match field	
13	DST_MASK	destinationIPv4PrefixLength	mask from the OFPXMT_OFB_IPV4_DST match field	
14	OUTPUT_SNMP	egressInterface	Information could be retrieved from OpenFlow Cookie or OUTPUT action struct	proposed
15	IPV4_NEXT_HOP	ipNextHopIPv4Address	If it is possible to get topology information it can represent Next Hop OpenFlow switch	proposed, N/A
21	LAST_SWITCHED	flowEndSysUpTime	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
22	FIRST_SWITCHED	flowStartSysUpTime	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
27	IPV6_SRC_ADDR	sourceIPv6Address	OFPXMT_OFB_IPV6_SRC match field	
28	IPV6_DST_ADDR	destinationIPv6Address	OFPXMT_OFB_IPV6_DST match field	
29	IPV6_SRC_MASK	sourceIPv6PrefixLength	mask from the OFPXMT_OFB_IPV6_SRC match field	
30	IPV6_DST_MASK	destinationIPv6PrefixLength	mask from the OFPXMT_OFB_IPV6_DST match field	
31	IPV6_FLOW_LABEL	flowLabelIPv6	OFPXMT_OFB_IPV6_FLABEL match field	
32	ICMP_TYPE	icmpTypeCodeIPv4	Calculated as 256 * OFPXMT_OFB_ICMPV4_TYPE + OFPXMT_OFB_ICMPV4_CODE	
36	FLOW_ACTIVE_TIMEOUT	flowActiveTimeout	Can be exported from the OF2NF application if HARD timeout for configured flows is known	
37	FLOW_INACTIVE_TIMEOUT	flowIdleTimeout	Can be exported from the OF2NF application if IDLE timeout for configured flows is known	
55	DST_TOS	postIpClassOfService	Could be possible to obtain if *Flow Removed* message contain action struct	proposed, N/A
56	IN_SRC_MAC	sourceMacAddress	OFPXMT_OFB_ETH_SRC match field	
57	OUT_DST_MAC	postDestinationMacAddress	Could be possible to obtain if *Flow Removed* message contain action struct	proposed, N/A
58	SRC_VLAN	vlanId	OFPXMT_OFB_VLAN_VID match field	
59	DST_VLAN	postVlanId	Could be possible to obtain if *Flow Removed* message contain action struct	proposed, N/A
60	IP_PROTOCOL_VERSION	ipVersion	Can be concluded from the matching fields and populated accordingly	
64	IPV6_OPTION_HEADERS	ipv6ExtensionHeaders	OFPXMT_OFB_IPV6_EXTHDR	
70	MPLS_LABEL_1	mplsTopLabelStackSection	Combining OFPXMT_OFB_MPLS_LABEL, OFPXMT_OFB_MPLS_TC and OFPXMT_OFB_MPLS_BOS	
80	IN_DST_MAC	destinationMacAddress	OFPXMT_OFB_ETH_DST match field	
81	OUT_SRC_MAC	postSourceMacAddress	Could be possible to obtain if *Flow Removed* message contain action struct	proposed, N/A
130		exporterIPv4Address	Can be populated with the IPv4 address of the OpenFlow switch that has sent *Flow Removed*	proposed
131		exporterIPv6Address	Can be populated with the IPv6 address of the OpenFlow switch that has sent *Flow Removed*	proposed
136		flowEndReason	Can be populated from the *Reason* field from *Flow Removed* message	
139		icmpTypeCodeIPv6	Calculated as 256 * OFPXMT_OFB_ICMPV6_TYPE + OFPXMT_OFB_ICMPV6_CODE	
150		flowStartSeconds	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
151		flowEndSeconds	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
152		flowStartMilliseconds	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
153		flowEndMilliseconds	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
154	Not Available in NetFlow	flowStartMicroseconds	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
155		flowEndMicroseconds	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
156		flowStartNanoseconds	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
157		flowEndNanoseconds	Estimated from the Flow Duration field from Flow Removed message	proposed estimation
161		flowDurationMilliseconds	from flow *Duration* field in *Flow Removed* message	
162		flowDurationMicroseconds	from flow *Duration* field in *Flow Removed* message	
243		dot1qVlanId	OFPXMT_OFB_VLAN_VID	
244		dot1qPriority	OFPXMT_OFB_VLAN_PCP	
256		ethernetType	OFPXMT_OFB_ETH_TYPE	

Table 4.1: Populating NetFlow/IPFIX fields with OpenFlow information

4.3.7 Prototypes of OF2NF for Ryu Controller

For the purpose of creating OF2NF proof-of-concept (PoC), simple OF2NF application prototypes are created for the Ryu controller [RYU].

4.3.7.1 *OF2NF proof-of-concept in reactive scenario*

For the reactive scenario proof-of-concept, the following OpenFlow Ryu applications have been created:

- *A forwarding* application that analyses the first packet of the traffic flow and creates appropriate forwarding rules on the OpenFlow switch for forwarding the rest of the network packets in that traffic flow.
- OF2NF application that receives the Flow Removed messages and translates them to appropriate NetFlow/IPFIX messages.

The proof-of-concept has been tested in the Mininet environment where the topology is the same as in Figure 4.3.

The PoC has showed that the OF2NF application is reporting the same bytes and packets counters values as the native NetFlow export from the OVS switches. In addition, PoC has showed that OVS is reporting the first packet of the traffic flow sent from the switch to the OpenFlow controller separately from the rest of the traffic flow. Although the first packet of the flow is returned from the Controller to the switch for forwarding again through the normal pipeline, this first packet is not counted again in the next flow record. More details about PoCs can be found in Appendix B and C.

4.3.7.2 *OF2NF Proof-of-Concept in Proactive Scenario*

For the proactive scenario Proof-of-Concept the following OpenFlow Ryu controller applications has been created:

- *Forwarding* application that is creating proactive forwarding rules that are usually aggregated and forward number of traffic rules. These rules are usually permanent or installed for a longer period of time.
- The OF2NF application in proactive mode matches traffic that should be monitored, creates flow rules for counting bytes and packets of the traffic flows and redirects them to the forwarding flow tables by proactive flow rules.

The PoC has shown that the OF2NF application is reporting the same bytes and packets counter values as the native NetFlow export from the OVS switches.

5 Conclusions

Due to the constantly-changing landscape of SDN solutions, the JRA2T1 results discussed in this document are primarily focused on the GÉANT/NREN use cases, in a landscape where no SDN-capable equipment has yet to be deployed in production within the European NRENs or GÉANT. Although many NRENs are interested in SDN/OpenFlow capabilities [SIEN] [TC2013], the implementation of proposals presented herein are heavily dependent on the adoption of SDN/OpenFlow by GÉANT and the NRENs. The proposed solutions and proof of concept prototypes can be further developed to support production level services in the following areas:

- Cloud support as directly related to the gOCX concept developed by JRA1T2.
- (Connection-oriented) multi-domain SDN – a solution based on NSI to enable end-to-end circuits provisioning.
    - A working proof of concept code has been delivered. This solution may enable an SDN/OpenFlow domain (e.g. campus network) to support Bandwidth on Demand services.
- Security traffic duplication and redirection capabilities using OpenFlow can be used to reinforce security applications in the network.
- Standard NetFlow monitoring based on OpenFlow switches can be possible by using or further development of the PoC proposed by JRA2T1.
- Support for OAM in the network infrastructures based on Open vSwitches can be provided by using OVS code enhanced in JRA2T1.

As part of the work carried out by JRA2T1, important information related to the current SDN/OpenFlow standards and hardware capabilities which may be useful for any future SDN related work has emerged, such as differences in OpenFlow specification support across different hardware. The hardware selected to implement SDN-enabled services should support all the requirements imposed by the specific solution. It is also worth mentioning that proposed solutions may often impose additional requirements on the controller or on the way the flow rules are applied to the switch (e.g. using separate tables or priorities for specific functionalities).

There are a number of appendices to follow that include further information, relevant to OpenFlow/SDN, including:

- The results of an NREN and user community survey on the use of SDN in clouds, testbeds and campus networks
- An overview of business solutions for SDN monitoring.
- OpenFlow features, as well as features for monitoring and statistics available from vendors.
- Multi-domain SDN Technologies for Cloud Computing and Distributed Testbeds, solutions and use-cases for multi-domain SDN
- The concept of SDNapps – generic network functionalities running on top of the SDN/OpenFlow network can be further reused to provide customized solutions to support specific packet forwarding requirements.

The evolution towards programmable networks seems inevitable, and there are already examples of services in RENs using OpenFlow – see [I2AL] [I2VS]. The results and experience, gained within JRA2T1, support service development in SA2, SA3 and SA7 and the overall SDN/OpenFlow adoption in GÉANT in NRENs.

Appendix A Survey on clouds, SDN and NFV

In a collaboration between JRA1T2 and SA7, a survey within the NREN and user community focusing on SDN/NFV and Clouds was elaborated [SURVEY]. The questionnaire primarily identifies the network requirements based on the question '**What can the network do for the clouds?'** This includes information about items/ideas focused more technically on a "SDN/NFV framework" [SDN] [NFV], the network set-up, virtualisation and processes.

The questionnaire was divided into four main sections (A–D):

Section A: The community section shows the affiliation/segment of clouds, the activities/research efforts using clouds and the target audience, the end-user population consuming (new) cloud services, and the influence of cloud computing on their organisations.

Survey Signees: Institute of Computer Science and Mathematics University of Latvia / HPC Laboratory, Institute for Informatics and automation problems, National Academy of Science of Armenia/ IUCC – Inter-University Computation Centre, Aviv University, Israel / University of Crete / FORTH / UIIP NASB, BASNET-United Institute of Informatics Problems of the National Academy of Sciences of Belarus / SWITCH, GLAN, PetaSolution / PSNC Poznan Supercomputing and Networking Centre / CESNET – e-Infrastructure for science, research and education / AMRES – Academic network Serbia / JSCC RAS – Joint Supercomputer Centre of the Russian Academy of Science / URAN / ETH Zurich / IBM Research / RENAM / Belnet / GRNET

Signees can be divided into three categories:

- National Research and Education network – NREN (8)
- Research organisation and Universities (10)
- Others – Research organisation and NRENs (2)

Affiliation: (N)RENs working on

- **Research**: Cloud computing research, computer science, computational chemistry, supercomputing and HPC, Life=science engineering, mathematics, physics, network and system software development and research, Earth science, art science, and security.
- **Operations:** Operations is also concentrating on its own deployment of cloud services, fulfilment of national roadmaps focusing (academic) ICTs as the whole academic community. Mostly IaaS, PaaS and SaaS as cloud services will be provided, operated and supported on (off) the campus networks.
- **Others:** NRENs are acting as cloud providers for constituents. They offer IaaS in two flavours, elastic one, addressed to the end-users (researchers, students, staff etc.) as Virtual Private Server (VPS) and one tailored to NOCs and project's persistent needs. Further NRENs act as a cloud operator for the R&E community.

Maturity working on clouds:
Achieving maturity on cloud computing can be summarised as Building Cloud Competences (BCC), instantiated by national/international projects/activities, trials and prototyping, standard work, and own deployments at institution level.

SDN/NFV Level: Interpretation:

- 14% of all responses have an EXPERT level of SDN/NFV knowledge (3)
- 29% of all responses have a MATURE level of knowledge (6)
- 19% of all responses have a MODERATE level of knowledge (4)
- 10% of all responses have KNOWLEDGE on SDN/NFV (2)
- 24% of all responses have HEARD about SDN/NFV (5)

Segmentation into categories (beginners, intermediate and advanced) on SDN, and NFV shows that 43% of all responses command ADVANCED and MATURE levels of knowledge. It is assumed that this population is actually working on research topics in projects, trials, and prototyping to increase their maturity on cloud computing, BCC. From the rest (57%), 34% identify as beginners (KNOWLEDGE, HEARD) or have knowledge on basic levels, which means starting with this topic or have plans for future steps, but (probably) do not really know how to cope with SDN/NFV. The remaining 23% is in between the first two groups, and has moderate levels of expertise. Thus it is assumed that 66% (EXPERT, MATURE) of the signees would be familiar with SDN and NFV as researchers and engineers. This allows contact to be maintained with this group is at the advanced level, and are able to integrate their expertise in future plans. Furthermore, beginners need coaching, education and support on SDN and NFV by the GÉANT community – community-building is a key subject.

Section B: Regarding Cloud Applications, information is collected about consuming and/or promoting of cloud service models or plans of (new) additional cloud services during the next 12 months, and identifying the software frameworks that will support the delivery/orchestration process.

One of the key questions of this paragraph is about which kind of cloud services (models) respondents would offer, and/or consume. The feedback is as expected: 57% of all responses provide offerings or consume cloud services, such as IaaS and PaaS. Only 10% have offerings/services on SaaS, and 13% are providing/consuming other services. Other services are mostly a mix, e.g. VMs with Linux or Windows OSes, firewalling, separate subnet for research projects, big data services or providing/consuming individual services locally or on specific HPCaaS. In conclusion, however, the main focus is on IaaS and PaaS, which should be introduced globally when providing offering cloud services within GÉANT.

The question about existing cloud service portfolios and future plans of new services during the next 12 months shows a number of trends:

- Extension/redesign of existing cloud service portfolios such as SaaS, PaaS, an HPCaaS.
- The idea of planning the role of support unit for researchers. The aim here is to support institutions by optimising, consulting, migration provisioning and administration.
- Offering special/individual services to the end-users – for example, having a GTS in place for video streaming and in-network caching experiments.
- Having scientific software as a service is in a wider scope – for example, licencing of scientific software.

- Cloud services on demand, e.g. VMs and GUIs. This implies orchestration, aggregation and distribution of resources is part of a wider scope, therefore ongoing migration to OpenStack as an orchestrator is in focus.
- To provide strong cloud services, e.g. ownCloud (the academic Dropbox) [OWNCLOUD] or IaaS on OpenNebula [OPENNEBULA].
- Having plans to implement or build up clouds on commercial products in the next two years.
- Thinking about a disaster recovery concept by providing Backup as a Service.
- Providing Cloud Storage to the GÉANT community, e.g. Synnefo Pythos [SYNNEFOPYTHOS].
- Having a concept of a Cloud Federation, IaaS cloud and Satellite image processing.

Offerings, including plans for cloud services, as shown, have a very wide scope, including: infrastructure, software, platform, orchestration of complex infrastructures, disaster recovery capabilities and looking for federations on cloud computing, which implies trust services supported by trustworthy organisations within GÉANT.

Section C: From a network perspective, there were questions related to experience and scenarios in inter-cloud computing, network configuration statically/dynamically, degree of virtualisation thus planning, using SDN/NFV frameworks also requirements to the GÉANT network providing a cloud architecture, which helped to assess Cloud Computing Network Readiness into GÉANT.

Respondents were first asked about accessing cloud services in general. Most respondents use commodity Internet as well as dedicated connection. Users of university campuses and users within NRENs mostly use a dedicated connection, users outside the campus network use commodity Internet, in some cases with VPN. Usually there are usually no significant limitations with cloud access. Limited bandwidth was noted in 20% of answers, one NREN has the problem with insufficient IPv4 address space and one with missing cloud-based applications for specific use.

The next question was focused on details from inter-cloud computing (federation of infrastructures). Approximately 60% of respondents have the experience with inter-cloud computing. Mentioned was multi-site OpenStack deployment, BonFIRE facility (several testbeds connected through VPN), OCX and MS Azure (self-developed control software). Special section was dedicated to monitoring the network, where NRENs have many different needs. Most important seems to be a monitoring of network utilisation per subjects (user / service / VM), but auditing other resources is also needed – VMs lifecycle, services status, security monitoring and others.

Many NRENs are connecting multiple sites and shares resources between them. Technologies like NFS storage (mirroring, sharing), cluster database or computing power are common. Two specific solutions were mentioned: FedCloud and Percona cluster.

As the SDN is still not very familiar, about 80% of respondents have statically configured network, mostly because it is simple, stable and proven solution. 20% use both statically and dynamically configured network. Dynamically configured in experimental cases or if interfaces and paths are automatically setup by cloud manager. For dynamic configuration, classic approach as VLANs and VPNs was mentioned, but also some more advanced technology like GÉANT BoD, VirtualWall, OpenStack Neutron (Open vSwitch with ML2 plugin).

To ensure traffic isolation between tenants, NRENs still mostly use VLAN technology. Exact results are: 75% VLAN, 15% VXLAN, 15% GRE, some of NRENs use also technologies like MAC address space separation, proxy ARP or MPLS. As tools for configuration are used OpenStack Neutron, ML2, OVS, FlowVisor, Cisco Boxes and OpenNaaS/OpenVirteX above Floodlight.

The QoS requirements were thought to be an important question, but about 50% of respondents don't have any specific requirements. In some cases, the guaranteed bandwidth is needed.

It was of interest to see how far the NRENs were with SDN implementation in their networks. Most of the respondents are planning/considering experimental operation of SDN services based on the some of the OpenSource implementation: 30% are waiting for SND implementation in routers and other network devices or taking SDN capabilities into account while making network equipment upgrades.

With SDN support comes SDN apps: more than 60% of respondents are currently developing or planning to develop apps for OpenStack or OCF, but the rest of the respondents do not see the practical usage of SDN apps. No one currently allows users to deploy their own apps, one respondent is considering the possibility and one allows users' SDN Apps for testing purposes only (not for deployment).

So at the end of the section, how should the GÉANT network support cloud computing? Mainly by stable and high-quality IP network, but also in terms of SDN availability in the GÉANT network, by providing an advanced networking services and supporting GTS. Central cloud services brokerage is a good approach. Such services may leverage efforts to host cloud services within different NRENs' networks or may integrate national cloud initiatives under a collaborative umbrella.

Section D: The paragraph "Additional requirements to the network" does include more detailed, specific questions, not introduced in Section B, and C, focusing on network provision of cloud services. Concentrating on SDN, NFV and vendor-agnostic approaches, it is also important to ask for specs on (network) applications, e.g. SDNapps or virtualisation on network functions, etc.

Questions tried to gather impressions on how cloud services are currently provisioned, as well as for the models for their deployment including network configuration.

Most NRENs would prefer hybrid clouds, in which public and private cloud resources are interconnected to enhance the value of cloud services using, for instance, private clouds for storing sensitive data, and public ones to allow offloading peak demands. NRENs, however, would like to see unified mechanisms for operating both and its integration should be simple for them to achieve. It is important, however, that NRENs have cloud services as part of their portfolios, since users are already demanding them.

The processes for configuring and provisioning services are not yet fully automated. Some NRENs use platforms that provide a certain level of automation, such as Okeanos or Openstack, but as a common note, all NRENs see automation as a much-desired feature.

Of utmost importance is security, ranging from end-host/OS/application to data and personal information protection, to network security, etc. Among the security features that would need to be supported are handling compromised customer domains, prevention of DDoS attacks, ensuring information privacy (not compromising user's data), prevent misuse of VMs, isolation, prevent VM hacking or general VM management to detect outdated, forgotten and vulnerable VMs. Current approaches include a variety of models, from distributed or centralised firewalling, network traffic filtering and NetFlow sampled or unsampled.

As for the network side of cloud services, NRENs use different topology models such as QoS equidistant core IaaS networks, separated I/O cloud networks or redundant networks. Topology and network technologies, however, usually depend on the service (e.g. mobile or optical for HPC). In general, they just require a well-connected and robust topology using their own network, GÉANT, and others such as Internet2, as well as commercial networks (even if they see that in GÉANT it is difficult to openly peer with commercial networks). In general, these technologies currently use traditional routing protocols (OSPF, BGP, IS-IS) and VLAN-based virtual networks to interconnect VMs.

Regarding IPv6, it is already supported in the majority of the NRENs networks as a dual-stack and IPv6 is encouraged also for end users. This support is still not considered through SDN, but SDN features should be IP-version agnostic.

NRENs see Open Source platforms (such as Open Daylight for network and Openstack for cloud) of special interest for the research community as in general they provide easy development and deployment. However, there is concern about security aspects that are not mature in open source solutions. Besides, integration with closed systems can also be achieved but through open APIs.

Finally, NRENs have been asked their views on a possible collaboration among themselves to provide cloud services. They point out that the first step would be to exchange knowledge and align requirements, followed by a cost-benefit analysis regarding the different possible models (brokerage, own clouds, federation). Federated is commonly seen as a possibility but some aspects would need to be well thought through, such as mechanisms for federation of credentials, for instance, eduGAIN for ubiquitous access, AAI integration, interoperability of resources, to have a common services portfolio and manage and plan capacity and SLAs. However, there are concerns that a fully federated environment might be difficult and complex to implement.

Appendix B Overview of Business Solutions for SDN Monitoring

In order to be able to analyse traffic from a variety of different applications and have a number of tools in place, it is typically necessary to not only use monitoring or mirroring ports on switches that can be connected to a monitoring network, but to also incorporate additional network optical or copper taps at different places in the network topology. An SDN controller can then be used to create an out-of-band, overlay monitoring network where traffic collected from these taps and monitoring ports is directed to the appropriate tool ports [HOG-2013]. This way the SDN approach can actually serve as a solution for creating a packet monitoring system.

Big Switch Networks offers such an SDN-based monitoring architecture, where monitoring applications called Big Taps are used to collect and filter traffic at any place in the network and can be programmed to send the filtered traffic to network monitoring or security tools. All configuration and programming of taps is prepared and carried out using the Big Tap Controller Software [BIG-2013].

ExtraHop-Arista also offers a solution for SDN monitoring, which provides a special focus on real-time performance and application workloads via the ExtraHop API and ExtraHop's Context and Correlation Engine (CCE) workloads of specific hosts, applications, clusters, storage and databases, etc. can be monitored [EXT-2013].

Microsoft also offers a tap-based solution that uses an OpenFlow Network to monitor traffic. DEMon (Distributed Ethernet Monitoring) works with low-cost switches and an OpenFlow controller that are employed for traffic analysis and monitoring in its data centres [MCG-2013].

Cisco also offers an SDN-based approach, which uses OpenFlow along with the Cisco eXtensible Network Controller (XNC) and an XNC Monitor Manager Solution [CIS-2014a]. The Monitor Manager aggregates data from network taps and is capable of linking monitoring devices directly to the points in the network fabric that are responsible for managing the monitored packets.

As creating overlay monitoring networks with SDN offers a lot of flexibility for network monitoring and traffic analysis, it must be expected that SDN-based solutions will continue to play an important role in network measurements.

Appendix C OpenFlow Vendor Overview: Optional OpenFlow features and Features for Monitoring and Statistics

HP ProCurve 5900 OpenFlow switch

The HP ProCurve 5900 OpenFlow switch works according to OpenFlow Specification 1.3.1 and the following section will describe its features for monitoring and its capabilities to keep statistics. As an added quality, the switch has the potential to not only be used as an OpenFlow switch as a whole, but also to have several OpenFlow instances defined that can be used independently from one another and with different controllers in place. Each such OpenFlow instance is associated with one or more VLANs, and the forwarding of packets only takes effect in the VLANs associated with an instance. The OpenFlow switch is capable of supporting up to 64 controllers, and 4096 different VLANs can be defined. It is possible to assign individual ports to these VLANs; a port forwards packets for a VLAN only after it is assigned to the VLAN [HP-2013].

For monitoring and traffic analysis, the switch allows:

- Configuration of SNMPv1, SNMPv2 and SNMPv3 parameters, logging and notifications.
- Configuration of RMON (Remote Network Monitoring) as an extension to SNMP (the statistics group of RMON samples traffic data for Ethernet interfaces and collects the data in the Ethernet statistics table (ethernetStatsTable). The parameters include number of collisions, CRC, alignment errors, number of undersize or oversize packets, number of broadcasts, number of multicasts, number of bytes received, and number of packets received).
- Configuration of Network Quality Analyzer (NQA) for performance measurements and QoS evaluations (for various operation types such as voice, path jitter, UDP jitter, etc.) and for threshold monitoring where trap messages are sent to the network management station (NMS) when thresholds are exceeded or violated.
- Configuration of ICMP echo operations to verify the reachability of a device.
- Configuration of port mirroring to copy packets that pass through an interface port to send them to a traffic analysis device for further processing.
- Configuration of sFlow to collect interface counter and packet information; the sFlow agents sends UDP datagrams to the sFlow collector where the data is analysed.
- Configuration of the Embedded Automation Architecture (EAA), which is a framework that allows the definition of monitoring events and associated follow-up actions. When it comes to defining monitoring policies, the following restrictions apply: monitoring policies can be defined for specific OpenFlow instances, but each monitoring policy can only contain one monitoring process event.
- Configuration of the Network Configuration Protocol (NETCONF) that allows the collection of statistics and provides filtering capabilities.

The following tables list switch features and capabilities, as well as features related to monitoring and the collection of statistics.

Feature	Comment
Virtual switch	Maximum number of OpenFlow instances that are supported on one physical switch is 128.
Forward normal support	Hybrid mode supported (both OpenFlow and normal processing).
Forward flood support	Forward to an OpenFlow enabled physical port ALL, CONTROLLER, NORMAL and FLOOD is supported.
Enqueue support	Enqueue action is not supported.
Modify fields support	OpenFlow flow table of type MAC-IP allows modifying destination MAC address, modifying source MAC address, modifying the VLAN and specifying the output port.
IP src/dst lookup for ARP	Not supported.
STP support	STP is supported.
Output to multiple ports (multiple output actions)	Multiple output actions is with group table Group table is capped to 32 per instance and capped to 1024 in total.
Multiple controllers support	Is supported but not in the same VLAN/instance, i.e. the switch supports OpenFlow instances with individual VLANs where each instance can have one controller; can support up to 64 controllers in that way.
Emergency mode behaviour	The switch is able to send information carrying ‘dying-gasp’ events in critical events, such as power failure; switch also has connection interruption modes that can be set to determine actions when connection to controller is lost: 1. Secure mode (switch keeps forwarding traffic based on flow tables and does not delete unexpired flow entries); 2. Standalone mode (switch performs normal forwarding and flow entries are not deleted).
Number of flow entries	Can be configured; at most 65535 flow entries can be used.
OF-CONFIG support	OpenFlow Configuration support after reboot
Flow rate	Traffic policing, generic traffic shaping and rate limiting is supported; rate limits control the rate of inbound/outbound traffic and specify the maximum rate for sending or receiving packets (including critical packets) of a physical interface.

Features for monitoring and statistics:

Feature	Comment
Counters	Interface-,OSPF-, MSDP message-, FSPF-, NQA reaction-, flow table-counters are supported; Counters are maintained for each flow table, flow entry, port, queue, group, group bucket, meter and meter band.
Maximum number of logs	Maximum number depends on the logs.
Maximum number of notifications	Supported maximum number depends on the notifications.
SNMP MIBs	Supports SNMPv1, SNMPv2c and SNMPv3. MIBs support changing with SNMP software versions.
Support for OpenFlow statistics	port statistics, flow statistics, table statistics, group statistics.
Support for statistics related message types	OFPT_FLOW_MOD, OFPT_PORT_MOD, OFPT_GROUP_MOD, OFPT_METER_MOD

HP ProCurve 3500 OpenFlow switch

Feature	Comment
Virtual switch	Every VLAN can be used as its own instance with its own independent OpenFlow configuration and controller.
Forward normal support	Supported; normal non-OpenFlow VLANs can also be used at the same time for normal traffic to be forwarded without OpenFlow management.

Forward flood support	Supported in software.
Enqueue support	Not supported (HP QoS extensions are available).
Modify fields support	Modify eth src/eth dst/ipv4 src/ipv4 dst/tcp srcport/tcp dstport are NOT supported; modify ipv4 ToS is supported.
IP src/dst lookup for ARP	IP src/IP dst fields are matched.
STP support	OpenFlow is confined to the switch spanning tree and does not allow full interaction with the switch spanning tree.
Output to multiple ports (multiple output actions)	Supported and processed in switch software.
Multiple controllers support	Supported with OF 1.3.
Emergency mode behaviour	Emergency flow cache not supported.
Number of flow entries	Group tables for multiple flow entries supported, total number of groups in switch is 1024, total number of groups per OpenFlow instance is 32, 65535 VLAN entries.
OF-CONFIG support	not supported
Flow rate	Per-flow rate-limiting is possible, i.e. the rate of packets running through a switch can be controlled. Per-flow rate-limiters associate an arbitrary number of flows with a rate-limiter. Any number of flows can be mapped to a rate-limiter, regardless of src/dst ports. The use of rate-limiters requires a version of ovs-ofctl, which includes HP QoS extension.

Features for monitoring and statistics:

Feature	**Comment**
Counters	Per flow counters: received packets are maintained correctly; received bytes are NOT maintained correctly; duration(sec) and duration(nsec) is maintained by software.
Maximum number of logs	Alert logs available for various alert types, event logs.
Maximum number of notifications	Information not available.
SNMP MIBs	Supported.
Support for OpenFlow statistics	Full statistics are not available when a rule is executed in hardware (byte_count will not be available, statistics are updated every 7 seconds); full statistics available for flows switched in software; message statistics for OpenFlow instances; port statistics per instance, group statistics, meter statistics.
Support for statistics related message types	Filtering of display information is supported: filters for dest-ip, dest-mac, dest-port, ip-protocol, ip-tos-bits, source-ip, source-mac, source-port, vlan-id, vlan-priority.

Extreme Networks Summit X460 switch

The Extreme Networks Summit X460 Switch uses ExtremeXOS 15.5, which supports the listed optional features in the following way [EXT-2014]:

Feature	Comment
Virtual switch	No
Forward normal support	Hybrid switch operation is possible, i.e. on the same switch standard non-OpenFlow-enabled ports can coexist with OpenFlow enabled ports. The OpenFlow functionality is enabled at the VLAN level, which means that all ports that are assigned to that OpenFlow VLAN only process OpenFlow flows associated with that OpenFlow based VLAN. Ports in other normal VLANs that are not OpenFlow enabled process traffic like standard Ethernet ports. The same port on a switch can support OpenFlow based as well as non-OpenFlow based VLANs.
Forward flood support	OpenFlow actions 'Forward ALL' and 'Forward Flood' are not implemented.
Enqueue support	EXtremeXOS offers a simple enqueue action for forwarding a packet through a queue attached to a port. The queue can be assigned a QoS profile for simple QoS support with this mechanism. A controller may also query information and statistics on such a QoS profile.
Modify fields support	ExtremeXOS from version 15.4 and higher allows VLAN ID editing functions (add, strip, modify) and also allows source and destination MAC modify actions.
IP src/dst lookup for ARP	There is conditional support for IPv4 source address and IPv4 destination address matching in ARP packets. It is currently being investigated by the company [EXT-2014, p. 49].
STP support	Spanning Tree (802.1d domains); the maximum number of 802.1d domains per port is 1. The maximum number of STP protected VLANs is 600.
Output to multiple ports (multiple output actions)	In EXtremeXOS flow table entries forward a packet to one physical port. OpenFlow actions 'Forward ALL' and 'Forward Flood' are not implemented.
Multiple controllers support	Multiple OpenFlow controllers are supported and can be configured to increase availability. It is possible to create controller clusters to be represented by a single IP address where the switch treats this cluster as a single controller, but it is also possible to assign multiple IP addresses to a controller cluster. The switch then connects to the primary and secondary controller at the same time and allows controllers to manage failover, i.e. both controllers are active and provide controller redundancy.
Emergency mode behaviour	There is no emergency flow table available; ExtremeXOS supports only one physical table and ingress table. ExtremeXOS offers an 'open fail' mode where existing flows are kept after connectivity to the controller is lost (this is in contrast to the 'secure-fail' of OpenFlow 1.0 where all flows are removed when connectivity to the controller is lost).
Number of flow entries	The OpenFlow table size is limited by the number of ACLs that the switch supports (platformdependent table sizes). The Summit X460 supports 2048 ingress and 256 egress Access lists (meters). The maximum number of MAC addresses in the FDB is 32768; the maximum number of FDB (blackhole entries) is 32000 [EXT-2014a].
OF-CONFIG support	No
Flow rate	The rate-limit for Packet-in packets sent from the switch to the controller is set to 1000 packets per second as default with a range that can be set from 100 to 2147483647. A burst-size can also be set in connection with rate-limit.

Features for monitoring and statistics

Feature	Comment
Counters	The L2 switching hardware does not count packets or bytes for each entry, however the single wide-key ACL per OpenFlow VLAN provides summary counts. Counters are maintained per-table, per-flow, and per-queue [EXT-2014].
Maximum number of logs	16 logs can be created at a time.
Maximum number of notifications	A maximum number of 16000 notifications can be logged.
SNMP MIBs	Y.1731 Compliant Performance Monitoring SNMP MIBs.

Juniper MX80/240 OpenFlow switches

Juniper offers OpenFlow version 1.0 support for switches of their EX and MX series running Junos OS. For each Junos OS release for OpenFlow support a certain matching OpenFlow software package must be installed [JUN-2014a]. Currently Juniper offers OpenFlow support for their EX4550 and EX9200 Ethernet switches as well as for their MX80, MX240, MX480 and MX960 routers.

Feature	Comment
Virtual switch	Only one virtual switch is supported
Forward normal support	OFPP_NORMAL is supported according to specification Hybrid operation (having traffic on the same port in two VLANs – one processed by OpenFlow and other one sent to the traditional forwarding path) is supported according to specification.
Forward flood support	OpenFlow actions 'OFPP_FLOOD' and 'OFPP_ALL' are supported. OFPP_OUTPUT, OFPP_IN_PORT and OFPP_CONTROLLER are not supported.
Enqueue support	OFPAT_ENQUEUE is not supported
Match fields support	dl_src, dl_dst, dl_vlan, dl_vlan_pcp, dl_type, nw_tos, nw_proto, nw_src, nw_dst, tp_src, tp_dst are supported. OFPC_ARP_MATCH_IP is not supported
Modify fields support	Only OFPAT_SET_VLAN_ID, OFPAT_STRIP_VLAN are supported Other fields cannot be set.
IP src/dst lookup for ARP	nw_proto (IP Protocol or lower 8 bits of ARP opcode) is supported. OFPC_ARP_MATCH_IP is not supported.
STP support	Not supported.
Output to multiple ports (multiple output actions)	No documentation found.
Multiple controllers support	One active OpenFlow controller is supported on each virtual switch (only one virtual switch can be created).
Emergency mode behaviour	Is not supported; if the switch loses connection to the controller then flow entries are deleted.
Number of flow entries	Each OpenFlow interface can have one or more flow entries.
OF-CONFIG support	Not supported.
Flow rate	No documentation found.
Number of tables	1

Features for monitoring and statistics

Feature	**Comment**
Counters	Number of flows, number of packets, number of groups, number of buckets, number of packets, number of bytes
Maximum number of logs	Information not available
Maximum number of notifications	Information not available
SNMP MIBs	information not available
Support for OpenFlow statistics	Junos OS on MX Series Switches supports the following OF statistics: OFPST_DESC, OFPST_FLOW, OFPST_TABLE, OFPST_AGGREGATE, OFPST_PORT, OFPST_QUEUE
Support for statistics related message types	OFPC_FLOW_STATS, OFPC_TABLE_STATS, OFPC_PORT_STATS, OFPC_QUEUE_STATS, OFPT_PORT_STATUS, OFPT_STATS_REQUEST, OFPT_STATS_REPLY

OVS extensions

OAM is still an immature area in SDN. There is not a broadly accepted or adopted solution yet. However, in a production environment link failure detection and reroute is an essential element of a network. In and OpenFlow based network the offered solutions are limited. The OpenFlow protocol supports the OFPPC_PORT_DOWN and OFPPS_LINK_DOWN messages. When a switch detects that a port or directly connected link is down, it can signal this event to the controller. The controller can act and e.g. insert flow rules to route around the failure. But this is a slow process. In production networks a faster recovery is often required.

OpenFlow 1.3 also supports the group table:

Group Identifier	Group Type	Counters	Action Buckets

This table supports four types:

- (required) ALL: execute all buckets (multicast)
- (optional) SELECT: execute 1 bucket (load balancing)
- (required) INDIRECT: execute only bucket (IP next hop)
- (optional) FAST FAILOVER: execute 1st live bucket

The “FAST FAILOVER” type can be used for rerouting. When there are multiple paths provisioned between two OpenFlow switches, the group table can be used to send traffic on a working path. On the sending switch, the group table uses the outgoing ports that are used for paths to the other switch as “action bucket”. A packet that needs to be sent to the other switch is sent on a port that is “live” (working). When this port goes down, another port that *is* working is used to send the traffic. This provides fast failover, but lacks detection of path failures. This is illustrated in the figure below.

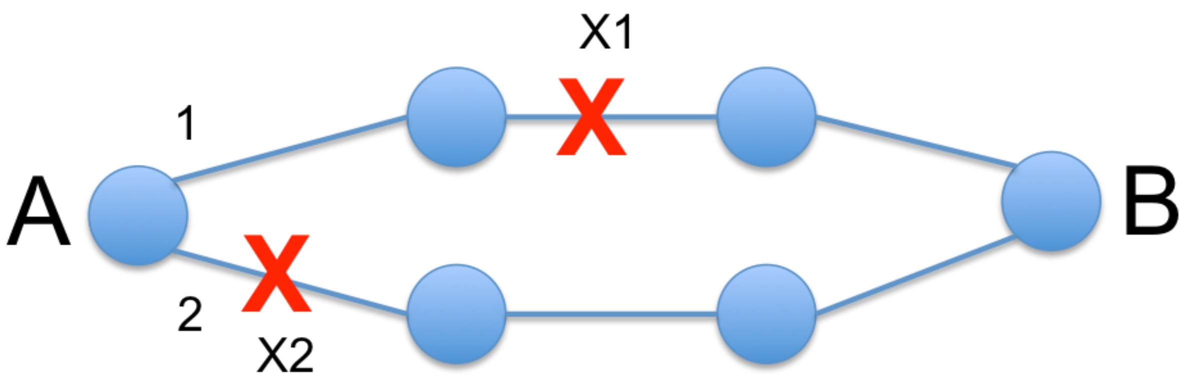


Figure C.1: Illustration of fast failover without path failure detection

When there is a link outage at X2, port 2 of switch A will go DOWN. This can be detected by switch A and traffic can be switched to another port (port 1 in this case). However, when there is a link outage at X1, port 1 of switch A remains UP, and this failure cannot be detected by switch A. A mechanism for path failure detection is required. A protocol that sends periodic hello packets is needed, such as bidirectional Forwarding Detection, RFC 5880 (BFD) or Ethernet OAM (IEEE 802.1ag). Ethernet OAM support is currently being worked on in Open vSwitch.

Ethernet OAM (IEEE 802.1ag) uses Ethernet frames for OAM functionality. It uses an Ethernet type of 0x8902 and supports three message types:

- Continuity Check Messages (CCM)
- Link Trace Message (LTM/LTR)
- Loopback Message (LBM/LBR)

The link trace and loopback messages are similar to IP traceroute and IP ping, but instead of working with IP addresses, Ethernet OAM works with MAC addresses. Each Ethernet interface that supports 802.1ag has a MAC address that can be used for OAM. For example, a loopback messages (LBM) can be sent to the MAC address of an Ethernet switch port and if that switch port is reachable, it sends back a loopback reply message (LBR). The Continuity Check Messages (CCMs) are used to periodically send hello messages on Ethernet broadcast domains and other Ethernet switches use these to monitor reachability of other switches.

802.1ag also has the concept of maintenance domains, maintenance associations and maintenance levels. These are used to build hierarchical OAM domains where there is a separation between monitoring an individual link, an end to end path, a regions within a service provider, the link between customer and service provider, etc.

Open vSwitch (OVS) is a software switch that supports OpenFlow. It is part of the standard Linux kernel and is also used in some hardware OpenFlow switches, like those from Pica8. OVS has limited support for 802.1ag. It only implements CCM, and the maintenance names and levels are fixed. This makes it unsuitable for situations where OVS switches are connected via paths through traditional switches and path failures need to be detected by the OVS switches. Ronald van der Pol has implemented the 802.1ag protocol as Open Source software (https://svn.surfnet.nl/trac/dot1ag-utils/wiki) and this code is being merged in the OVS OAM code. The interoperation with traditional switches will be tested on the SURFnet OpenFlow testbed, that uses OVS based Pica8 switches with Ethernet tunnels over traditional Ciena Ethernet switches.

Develop the Authentication and Authorisation Infrastructure (AAI) services to cope with the challenges of end-user mobility.

Appendix D SDN for Clouds, Testbeds and Campus Networks

D.1 What can the network do for clouds?

Cloud Computing is a widely used distributed concept for provisioning services to the end-users on commercial also academic (network-) platforms. The National Institute of Standards and Technology (NIST)[1] identifies the different deployment models of public-, private- and hybrid clouds, as well as the different service models with Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) being the most widely recognised. [NIST] Service/deployment models of clouds raise the question “What can the network do for clouds?” Software-Defined Networking (SDN) driven by the ONF [ONF] as a Standard Development Organization (SDO) can be exploited to meet specialised requirements.

JRA2T1 ran a GÉANT Symposium session in 2013, addressing the cloud-network interoperability based on SDN. During the session, cloud provisioning requirements were discussed, such as the dynamicity inherent to cloud services provisioning – not only in creating and modifying virtual servers, but also in moving and interconnecting with them. Static provisioning for clouds works, but will not scale to the implementation of dynamicity requirements at large. Another requirement is that of multi-tenancy. Solutions such as VLANs, VXLANs or NVGRE do not fully address fast dynamic changes and extended cloud-network interoperability, such as that which can be achieved by exploiting SDN. Control over the network traffic flows also provides more options for monitoring. Network monitoring data becomes relevant and useful for the cloud.

Regarding control-plane integration, a lightweight approach seems to be optimal. Resources and capabilities should be advertised between both the cloud and the network infrastructures, on an as-needed basis. Overall, the adoption/design/adaptation of stable, production-level specialised controllers for different use cases/applications is required. A closer look at cloud networking requirements and potential solutions by SDN is in order.

As a follow-up to the introductory session at the Symposium, JRA2 T1 ran a network requirements’ survey addressing the NREN community with clouds, SDN and NFV topics in order to estimate the innovation of SDN on “real” use cases. Details about this survey and outcomes are provided in Appendix A.

[1] [NIST]
Public Cloud: *The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider*
Private Cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises
Hypbrid Cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

D.2 Multi-domain SDN Technologies for Cloud Computing and Distributed Testbeds

D.2.1 Slice-oriented vs Connection-Oriented Multi-Domain SDN

The following sections describe the research effort in terms of approach to multi-domain, SDN-based architectures. The Multi-Domain SDN research activity comprises a set of possibilities that has driven this work group to split the study into two, main branches.

The first approach focused on the pure aspects of multi-domain, SDN connectivity services, aiming to enable to end-users with the creation of dedicated circuits interconnecting at least two points in the multi-domain environment, involving SDN/OpenFlow domains. Therefore, this working group has developed the so-called "**Multi-domain, Connection-Oriented Approach**".

The other identified approach aims to provide end-users with sets of federated resources belonging to different domains, so that they can run their experiments (in the case of research institutions end-users), deploy software applications (in the case of software developers and SaaS providers) or simply provide resource infrastructure to their users (as it is the case of IaaS providers) in a multi-domain environment. This alternative will be discussed as the "**Multi-domain, Slice-Connection Approach**".

Broadly speaking, the Connection-Oriented Approach constitutes a particular case of the Slice-Oriented Approach. This is evident, since the Slice-Oriented Approach implies creating slices of resources, as well as establishing the connection between the slices allocated in different domains, i.e. the objective of the Connection-Oriented Approach. Nevertheless, key requirements have forced the research effort to split into these two approaches. For example, one key difference between approaches is related to the overall multi-domain performance. In the case of the Slice-Oriented Approach, it is necessary to keep track of the overall multi-domain network status to ensure that slices are created at the most convenient domains. Thus, the Slice-Oriented Approach proposes an EXTERNAL entity that determines and configures the flow rules, based on previous knowledge of the flow-spaces. In the Connection-Oriented scenario, there is no need to keep control over the flows in a multi-domain network; there is only the need to setup the E2E path based on the SDN controller configuration flows within the SDN resources. The connection approach is a particular case in which the flow-spaces configuration and control is delegated to the INTERNAL SDN controller.

Both approaches will also address the limitations and challenges while integrating legacy (non-SDN) domains into the solution, since this represents a typical situation in common current multi-domain scenarios.

Main objectives of the multi-domain SDN analysis and study comprise:

- Definition of building blocks of SDN Slice-Oriented and Connection-Oriented architectures.
- To design a generic architecture and provide description of the functionality.

- To identify type of use cases for both approaches.
- To provide support and guidance to the SA2 activity in order to facilitate the migration of the multi-domain functionality to the GTS.

D.2.2 Use Cases

D.2.2.1 *Extension of the BoD service to OpenFlow-based campus networks*

In order to satisfy the increasing demand of high-capacity and highly reliable connections, most NRENs offer point-to-point connectivity services to their clients [ESNET] [INTERNET2], including GÉANT [GÉANT]. A vast range of technologies are used to provide this type of services. Researchers trying to conduct a global experiment may need to establish a connection between two distant hosts not directly connected through their local NREN. In these cases, the circuit has to be established across several domains, which requires the use of multi-domain technologies.

Scenario

Considering a generic case in which the SDN campus network is connected to an NREN, which may or may not be SDN based, a user may want to establish a connection between a campus host and a remote host through the local NREN. Taking into consideration that at least two different network domains are involved, a multi-domain technology must be used to establish the end-to-end path. Figure D.2: depicts the scenario.

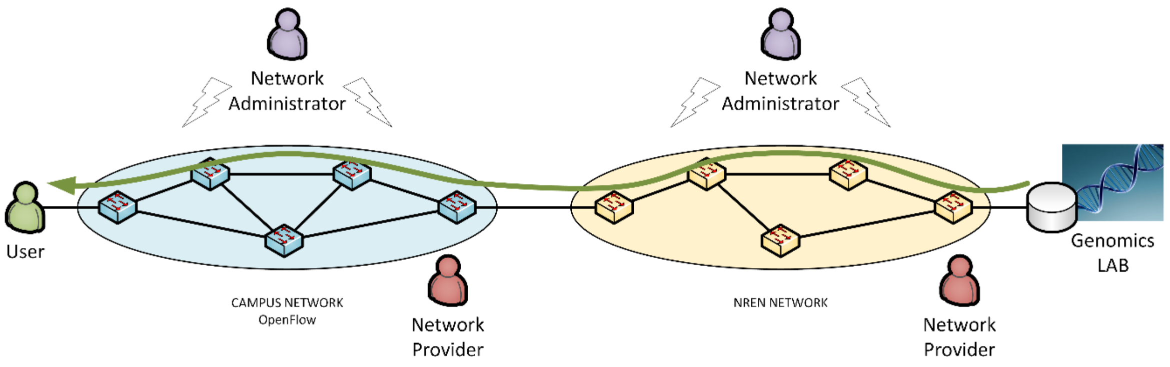


Figure D.2: Extension of the BoD service to OpenFlow-based campus networks

The actors involved in this scenario include:

- *User:* Entity that wants to establish a connection between two hosts of different domains.
- *Network administrator:* Entity in charge of the administration, management, control and operation of the network.
- *Network provider:* Entity that provides the connectivity service to the user.

First, the different *network providers* involved must agree on the type of service to be provided to *users*. They can agree some SDAs (Service Delivery Agreement), for example, provide L2-VLAN circuits with certain guaranteed bandwidth. Second, *network administrators* must configure the network to fulfil the specific requirements of the service that it is going to be provided. Once the service provided to *users* is well defined, users can request connections to their local *network provider*. Since the connection must be established through several domains, the *network provider* must request the connection establishment in the adjacent domains on behalf of the *user.* Once the connection request has been addressed in all the domains involved, the *user* will obtain the requested service.

Non-SDN-Based NRENs

When talking about the integration of OpenFlow with multi-domain technologies, its great flexibility and granularity can arise some compatibility problems. To overcome this possible failure situation, it is essential to have a previous negotiation between the network providers to agree on a common supported service type.

Figure D.3 depicts a particular case where only two domains are involved, a SDN/OpenFlow-based campus network and a non-SDN/OpenFlow NREN. In this case, the network providers of both domains have agreed on providing circuits defined by a VLAN tag, meaning that the user will have a connection established from source to destiny where a specific VLAN tag is used. In Figure D.3, two different connections are depicted, the purple and the red one. As can be seen, each of the connection uses a different VLAN tag, identified by the green and grey paths. Note that if the service is defined by VLAN tags, the users will have to tag the traffic before it enters the network. At the moment of forwarding the traffic, each of the domains will forward the traffic using the mechanism of their choice. Considering that the solution must be technology agnostic, the edge-switches of each domain must be able to adapt the incoming traffic so that it reaches the destination unchanged.

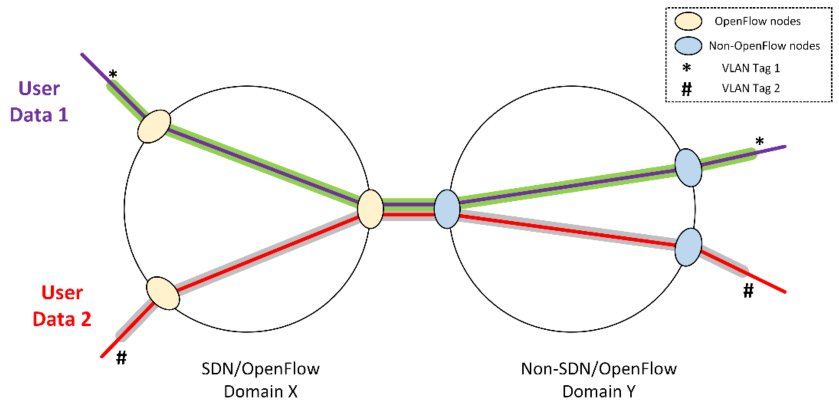
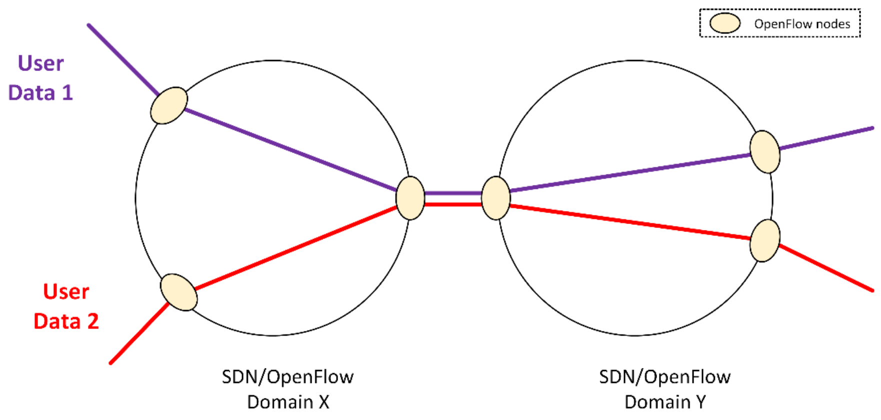


Figure D.3: OpenFlow campus network and non-OpenFlow NREN scenario

SDN-Based NRENs

In this scenario, the NREN is an SDN domain, specifically, an OpenFlow-based domain. As a consequence, because all the domains involve OpenFlow, one of the most significant problems when dealing with multi-domain approaches disappears: the compatibility between technologies. In this case, it is possible to establish a connection across all the domains involved, taking advantage of the flexibility and granularity of OpenFlow. However, there will still be differences between the OpenFlow protocol versions, which must be taken into consideration.

FigureD.4 depicts a particular case where only two OpenFlow domains are involved. In this case, the network providers of both domains have agreed to provide circuits defined by a combination of fields from L2 to L4. In FigureD.4, two different connections are depicted, the purple and the red one. One of the main advantages of using OpenFlow is that services can be identified without tagging the traffic. This way, users are not required to do any modifications to their traffic before it enters the network.



FigureD.4: OpenFlow campus network and OpenFlow NREN scenario

Requirements

Functional Requirements

- Users should be able to request a service through their local network provider.
- Users should be able to obtain end-to-end connectivity services with guaranteed QoS.
- Network providers should be able to negotiate the service characteristics with the network providers of the remaining domains.
- Network should be automatically configured whenever possible.
- Services should be automatically provided when requested by users.
- Services should only be provided after an authentication and authorisation process.
- The required mechanisms needed to guarantee that a service has been successfully removed should be *provided.*

Non-functional Requirements

- The solution must be unique and technology agnostic, valid for any combination of SDN/OpenFlow domains and non-SDN/OpenFlow domains.
- It should make the most out of SDN/OpenFlow granularity and flexibility.

D.2.2.2 *Slice-oriented multi-domain SDN use case: Collaborative research experimentation*

Overall Description

A group of three research institutions aims to run a collaborative experiment that will result in a heavy workload i.e. significant amount of resources on their DC facilities, the real-time interchange of big-data and consequently, tight multi-domain connectivity requirements across different domains to connect research institutions based on SDN technology. Computing and storage resources can be either owned by research institutions or the NRENs providing connectivity to the GÉANT backbone and to other research institutions. The research institutions' requirement is for a testbed (network resources and connectivity between resources) besides computing and storage resources for experimentation purposes.

Scenario

Figure D.5 depicts the use case: three research institutions are connected to their respective NRENs. Research Institutions 2 and 3 own their DC resources to run the collaborative experiments (i.e. they host their own storage and computation resources). The other possibility is for the NREN to act as the connectivity service provider as well as an IaaS provider, providing Research Institution 1 with the required DC resources. In this use case, DC facilities are also based on SDN technology (OVS switches) so that E2E multi-domain connectivity and service provisioning services should also comprise hosting facilities. Apart from SDN domains, the scenario also shows a non-SDN domain (MPLS technology instead) that should be integrated in the E2E connectivity service.

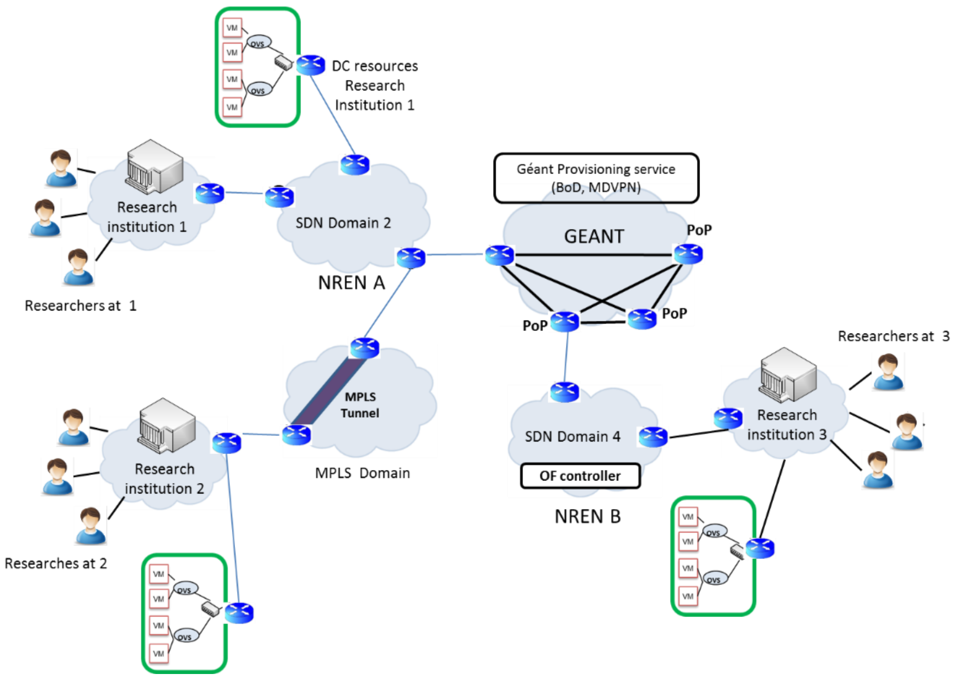


Figure D.5: Multi-domain SDN scenario overview for collaborative research experiments.

Actors and Associated Roles

Research Institutions

- Service Consumers
- Physical Infrastructure (IT) Providers. (In the case of research institutions 2 and 3).
- Virtual Infrastructure (IT) Providers. (In the case of research institutions 2 and 3).

NRENs and MPLS domain NOC

- Physical Infrastructure (Network and IT) Providers.
- Virtual Infrastructure (Network and IT) Providers. (In the case of research institution 1).
- Virtual Operators (Network and IT). (In the case of research institution 1).

GÉANT NoC

- Physical Infrastructure (Network) Providers.
- Virtual Infrastructure (Network) Providers.
- Virtual Operators (Network).

Functional Requirements

Service Level Requirements

Name	Requirement Description
GÉANT provisioning services	On-demand and/or automated deployment and provisioning of customisable GÉANT connectivity services. It should be possible to specify BW allocation to ensure proper QoS to support data interchange among the different research institutions.
Elasticity	Self-service, on-demand and/or automated scale-up and scale-down elasticity features.
Privacy, service security	Experimental data must be guaranteed privacy and security, without which the resources and connectivity among them will not be reliable.
Seamless Network Resources Provisioning across NRENs	NRENs should allocate network resources allowing dynamic provisioning and configuration to react upon unexpected QoS degradation.
Network Ctrl and Mngmt interfaces	It will be required to enable interfaces for the management of the connectivity service across the E2E path. The interfaces should allow the configuration of network resources and virtual appliances, independently of other services.
DC management platform and interfaces	Due to the SDN nature of DC facilities, a Cloud management platform and the corresponding interfaces must be adopted to (i) be managed by a Cloud administrator and (ii) interact with the network connectivity service. OpenStack constitutes a DC platform example.
Multi-tenant isolation	Each online instance deployed in the E2E path should be isolated from the rest of the workloads, in terms of data isolation, management isolation, and performance.
Support of Big Data	The experiment may make use of Big Data. Big Data should be supported within the DCs hosting the research facilities, since it has an impact on capacity, latency, access, security, cost and flexibility.

Control Plane Requirements

Name	Requirement Description
Automated provisioning of intra-domains connectivity	Every domain involved in the use case must provide mechanisms for an automated intra-domain configuration to enable CRUD connectivity services on-demand (triggered by external requests). The same applies to intra-DC configuration regarding the facilities involved in the service.
Data Centre Network (DCN) resource-usage optimisation	It is required that the control plane solution provides the mechanisms to support intra-DC connectivity services compliant with a variety of operator-level constraints, including load-balancing strategies, energy efficiency, paths with minimum cost, etc.
Programmable APIs	Main connectivity and resource provisioning configuration options offered by the control plane must be exported through programmable APIs (e.g. based on the REST paradigms), with different levels of capabilities depending on authorisation profiles. These APIs should allow exposing some (limited) functionality directly to the users and enable an easy integration with the overall DC control and management platform.
Network service resilience	The network and resources provisioning service across SDN domains, should detect network failures and support service restoration procedures or the possibility to trigger asynchronous failure alerts to notify upper layers (e.g. to the management/orchestration level) to enable recovery.
Easy interoperability with existing cloud management platforms.	Easy integration with Cloud platforms is fundamental to ensure E2E connectivity and provisioning of resources. Thus, the solution control plane should include mechanisms (e.g. APIs based on the REST concept and HTTP protocol) to enable an easy integration with existing orchestration systems and cloud management platforms from where the hosting resources are provided.
Integration with non-SDN connectivity services	The control plane should also incorporate mechanisms to interact with legacy non-SDN domains while providing E2E connectivity and resources.
Virtualization and abstraction	Resource sharing across domains is fundamental to provide E2E connectivity services and resources in an efficient and scalable way. Thus, resource virtualisation and abstraction procedures are fundamental to achieve this requirement.
Control-plane scalability	Computation and storage resources (in terms of servers and devices to be managed) as well as the expected huge number of traffic flows among servers should not affect control plane operations.
Scheduled network connectivity support	It is necessary to support of scheduled connectivity between intra-DC resources or in support of connectivity among resources located in different DCs. This is key for the success of the use case. Suitable synchronization procedures must be provided to coordinate the enforcement of the overall scheduled actions across the different DC resources.

Management Plane and Orchestration Requirements

Name	Requirement Description
Network resources management	Network equipment management, maintenance, administration, configuration, and performance monitoring.
Network management	Provisioning and management of network paths and circuits. Network paths must enable setting, monitoring, and enforcing the QoS and QoE, such as bandwidth, latency, amount of redundant paths, etc.
E2E service management	Provisioning and maintenance the of the E2E connectivity network forcing the traffic through the service enforcement points (appliances), elasticity of the network and load balancing between the service instances.
Inter-domain topology awareness	Topology abstraction awareness is fundamental to be able to properly manage and provide with resources and E2E connectivity service in an optimised way.
Virtualization management.	Virtualisation management of resources enables resource slicing and provisioning in this multi-domain, SDN-based use case.
Resources planning (tool)	Prior to network and DC resources provisioning, resource reservations at the network level should be planned and automatically updated to guarantee resource availability and optimise the global DCN utilisation among the shared physical resources.
Workload awareness	Orchestrated resources management at DC facilities must be capable of receiving the workload hints or requests specifying network-related behaviours (connectivity, latency, path redundancy…) and to enforce these requirements through the DCN controls and management tools.

Non-Functional Requirements

Name	Requirement Description
Usability	Flexible, multi-domain service operation without an impact on service when adding or removing network elements, domains or configuring network parameters. Backward compatibility in case of implemented changes outside of the user interface. User-friendly interface.
Reliability	Service reliability is a vital, non-functional requirement, especially in multi-domain scenarios in which several resource, network and services administrators may collaborate to achieve the E2E connectivity service.
Performance	E2E service performance is essential to the use case.
Efficiency	Network efficiency can be measured by mapping QoS requirements onto key performance indicators while providing the service.
Computation and storage location sensitivity	Computation and storage location sensitivity: there are many reasons to consider the physical location of the data placed in the cloud as key for businesses (e.g. the performance experienced by the users to access the data). Depending on the nature of the data (medical financial, etc.) there are also strict legislated requirements to take into account.

D.2.2.3 *Cloud Support Use Case: SDN-Enabled GÉANT Open Cloud Exchange*

Cloud computing is an emerging topic, which has great momentum in both private and public sectors. Reasons for that are mostly economically driven, increasing capex/opex[2] and the requirement for smart aggregation and provisioning of cloud resources and services. Cloud support in GN3plus is a cross-activity initiative between JRA1T2, focusing on architecture of cloud services, JRA2T1 is investigating an SDN framework for supporting cloud computing initiatives, and SA7 is designing and providing cloud services on top of the architecture. This use case focuses on gOCX and its SDN extension.

Overview

The GÉANT Open Cloud Exchange (gOCX)[GOCX] is an JRA1T2 initiative. It enables cloud services provisioning between the private sector, represented by public Cloud Service Providers (pCSPs) and the public sector, the (N)RENs by the academic CSPs (aCSPs).

The gOCX service is focused on network layers (0), 1, 2 (and 3), and deals with negotiation, establishing and disseminating connectivity, and (optional) will run on top of a trusted third-party service (TTPs), hosted by a neutral trust organisation, e.g. DANTE (GÉANT). Higher OSI-layers services affecting the network are propagated, for example, by a brokering system.[3]

Purpose

In general, cloud services are provisioned on demand, using the Internet, which copes well enough for most cases/applications. Services such as HD Video streaming, or transferring bulk data within a multi-domain network, however, have special needs for Quality of Services (QoS), e.g. guaranteed bandwidth in E2E or B2B. Thus, the proposed gOCX architecture acts as an "Open Cloud Exchange" over GÉANT that will offer exchange of cloud services via direct connections bringing together academia with CSPs, according to BigBusiness meets BigScience, as Helix Nebula (HNX)[HNX] demonstrates. Furthermore, the gOCX will facilitate interconnectivity over multi- domains between pCSPs and aCSPs, providing multipoint-to-multipoint cloud services, as demonstrated at the TERENA Conference 2014 [GOCXMOV].

Strategy and Objectives of gOCX to SDN–gOCX

Strategically, the conceptual platform OCX acts in a similar manner to an Internet exchange point (IXP). It is a peering mechanism that facilitates the inter-communication over domains for special demands of network resources, including:

- The capability for a direct connection to multiple service providers that can be deployed at different layers (0)1, 2 (and 3).
- Virtual, isolated and secure extensions of the direct connections to the networks of the users, and to their requested services.
- Automatisation using a gOCX inter-/intra connectivity management portal (orchestrator) will allow the (N)REN users' and/or institutions to choose among different offers, to setup their preferred connections

[2] e.g. Infrastructure, HW components etc / e. g FTEs, operations activities
[3] Brokering system: The term "brokering system" stands for an agent/agency to negotiate transactions (services) technically, based on an economical (financial) model between different parties.

and to manage and monitor them. Automatisation allows cloud providers to dynamically setup their offerings on top of inter-/intra network connectivity and to monitor their users' activity, too. Furthermore, an SLA clearing house service can also be provided.

Thus, gOCX provides "connectivity as a service".

SDN/OpenFlow shows potential as the underlying framework that could enable the gOCX service provisioning (see Figure D.6). The aim is to introduce SDN, as a tool to enable scalability and automatisation at the point of negotiating the provisioning. At the same time, SDN/OpenFlow technologies can provide the orchestrator for the aggregation of resources and provisioning of services to the end-users into a/pCSP. In order to achieve this goal, the south- and northbound APIs of the orchestrator must be thoroughly investigated, and the control-/data (forwarding) and orchestration plane (re)designed. Another challenge is the management and monitoring of such an infrastructure. The management plane (not described in this document) will be defined out of band, with access to the (Ethernet) Intelligent Platform Management Interface (IPMI) of the gOCX components. Furthermore, the purely functional details of the SDN–gOCX architecture are to be provided in further detail in subsequent paragraphs of the current section.

The ways in which SDN–gOCX might apply to the optical layer needs further investigation for a multi-domain approach, depending on how the broadcast domain is defined. One example of this is providing "connectivity as a service" on the lightpath from the p/aCSP to the OCX instances, using ROADMs [ROADM].

A first step towards a proof of concept (PoC) is to understand the (N)REN and CSPs' needs/requirements that will be an indicator and a driver for implementing an SDN–gOCX -Infrastructure over GÉANT. With this goal in mind, the questionnaire [CQUE] presented in Appendix A was used. Efforts to contact the major CSPs[4] in order to establish a strong collaboration and define a set of standard connectivity alternatives that will help define an API for creating and managing the direct connection to a CSP are ongoing.

Roles of gOCX Parties

gOCX/parties and users such as the p/aCSPs and the NRENs' end users as trusted third parties will have to interact in a multi-domain environment.

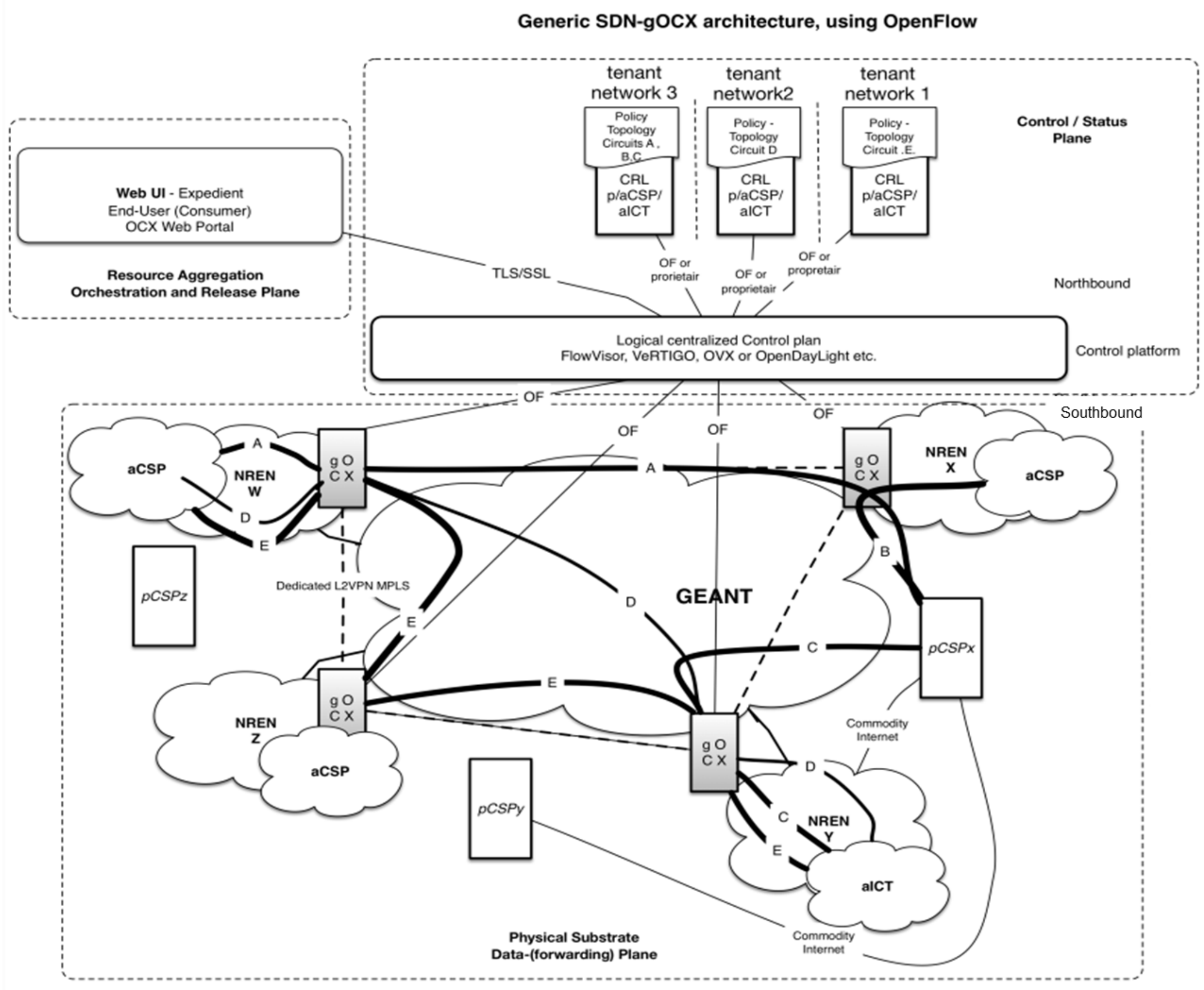
The SDN Capability and Innovation on gOCX

A gOCX is a concept platform designed to enable high performance, multi-site, cloud clusters to work together easily and effectively. In SDN–gOCX (see Figure D.6) connectivity is provided as a service in the form of slices.[5] Of course, a TTP and also a brokering model will be elaborated in further design phases. At the time of writing this report, the gOCX is aligned to the GÉANT productive environment.

A generic topology of an SDN–gOCX infrastructure (see Figure D.6) will be described in the following paragraphs. A minimum number of four PoPs will allow implementation of realistic connectivity as a service scenarios over the production GÉANT network. Three functional planes will be described, the data (forwarding), control/status and orchestration/release plane.

[4] AWS (Demo SC14), CloudSigma (see Demo SC14), Microsoft, IBM

[5] A slice is a logical partition of a physical entity, in this case a slice between p/aCSPs – multipoint to multipoint service provisioning upon the physical substrate, the GÉANT network.



Legend:
- aICT - academic ICT
- gOCX - GEANT Open Cloud Exchange Point, dedicated boxes aligned the PoPs?!
- pCSP: public Cloud Service provider like Amazon, IBM, MS etc.
- aCSP: academic Cloud Service provider like GRnet Okeanos, SWITCH, SURFnet etc
- OF: OpenFlow protocol v. 1.0+ resp. 2.0
- Web UI - Expedient Web User Interface with clearinghouse VT and OptIn Plugin (FOAM)
- A - slice pCSPx to aCSP(NREN W)
- B - slice pCSPx to aCSP(NREN X)
- C - slice pCSPx to aICT(NREN Y)
- D - slice aCSP(NREN W) to aICT(NRENY)
- E - slice aICT(NREN Y) through gOCX(to aCSPZ, NREN Z) to aCSP(NREN W)
- ° OVX - http://tools.onlab.us/ovx.html

System Area Border

Note: Management plane is not visible

Figure D.6: SDN–gOCX, a centralised approach

- ***The data (forwarding) plane*** (see Figure D.6), is the "workhorse" of router/switches and is responsible for parsing packet headers in a high-speed search on ASICs. It manages QoS, filtering, encapsulation, queuing and policy management all on silicon or also called customised ASICs. On the SDN–gOCX entities, the data (forwarding) plane is running on the devices/instances. According to the southbound API in use, e.g. "OpenFlow2.0" [OF2.0] a redesign of the forwarding abstraction is needed.

- ***The control/status plane*** (see Figure D.6) is a logical, centralised platform that interacts both through its south- and the northbound interfaces using the OpenFlow protocol v(x) with the physical substrate of the gOCX infrastructure and the p/aCSPs OpenFlow controllers running on top of the architecture or remotely at CSPs locations. At the time of writing (November 2014), tools used as a logical centralised platform are FlowVisor [FV], VeRTIGO or OpenVirteX (OVX) [OVX], which allow independence from the topology of the physical substrate to extend the network capability by continuing on link and/or node virtualisation scenarios, and further the useful isolation of user-traffic/slices of various experiments. Multi-tenancy is guaranteed in this example. The OpenFlow controllers are in access of the p/aCSPs. This would allow the configuration of slice topologies by policy management or network virtualisation through open northbound APIs. SDN/OpenFlow will guarantee dynamical negotiation, establishing and dissemination of connectivity that can also be defined as network "Flow Space as a Service". Thus, research on OVX and OpenFlow 2.0 would be a future perspective. Using a valid control platform is a challenge, while a first step for harmonising controllers is planned in Q4 2014.

- ***The orchestration/release plane*** (see Figure D.6) covers the end-user access to all authorised cloud services. In fact, there is a Web-UI in place with a clearinghouse that allows setting up a user profile – member of a p/aCSP – for authN/Z the end-users and to set-up/introduce their slices/services. On the orchestration layer, on a broker platform that would take care of aggregation, and distribution of cloud services coupled with their privacy. Experiences in orchestration of slices upon OpenFlow-enabled network components were, for example, collected on the GOFF (GÉANT OpenFlow Facility) with the OFELIA Control Framework (OCF). Approaches based on OpenNaaS [OPENNAAS], orchestration regarding OpenDayLight or an OpenStack implementation with Neutron as NaaS could be adopted.

Requirements of an SDN–gOCX

Functional Requirements

The functional requirements may include, but are not limited to:

- End-users should be able to request/withdraw a service through a common Web-UI.
- Services should be automatically provided when requested by the end-users through the Web-UI
- The SDN architecture has to guarantee traffic isolation using network virtualisation.
- pCSPs/aCSPs/academicICTs should be able to program/configure through northbound APIs remotely or by generic controllers of the SDN-architecture their own network topology (NaaS) and network services with corresponding QoS guarantees.
- The (N)RENs should be able to provide automated support to the pCSPs/aCSPs in establishing connectivity to their peering points.

- pCSPs/aCSPs/academicICT should be able to negotiate the service characteristics among each other by, e.g. policy management through open APIs offered by the SDN network provider, e.g. GÉANT.
- Service provisioning should be only allowed after an authN/Z process – see TTP.
- The required mechanisms to guarantee AA and Accounting or a brokering platform, a "marketplace", in place.

Non-Functional Requirements

The non-functional requirements may include, but are not limited to:

- The solution must be technology agnostic – valid for any combination of (SDN)OpenFlow, Standards and Network virtualisation technologies.
- The conceptual SDN–gOCX architecture should benefit from OpenFlow granularity and flexibility.
- End users consuming cloud services should be invited to access cloud services through the Web-UI should.
- The performance is the key requirement for providing cloud (network) services for guaranteeing QoS.
- Reliability of the SDN–gOCX architecture is essential to implement on top of a market place.

Details of functional, and non-functional requirements to a SDN–gOCX will be elaborated in further design phases. A position paper on SDN–gOCX is planned for April 2015.

Future Perspective

The visibility of gOCX is pursued via demonstrations on appropriate conferences/events. Three types of OCX instances, including: a local OCX(NREN), GÉANT-like open connect and a hybrid OCX (NREN share)[6] are envisaged.

Scalability, reliability, usability and network management, however, require automatisation. The best-placed solution is to introduce SDN as a framework, which allow configurability/programmability of (network) resources, slices (experiments) or authorised Cloud services (brokering) from the end-user perspective. Furthermore, SDN–gOCX is a framework focusing on federation, where NRENs and the private sector can participate in exchange services, "BigBusiness meets BigScience". In order to realise this, a TTP service on a brokering model has to be elaborated.

D.2.3 Connection-Oriented Multi-Domain SDN

Connection-oriented multi-domain SDN is a concept that allows the provision of multi-domain circuits in the multi-domain environment involving OpenFlow domains. OpenFlow doesn't define how to create multi-domain networks, and there are no widely used tools for the provisioning of connections across multiple SDN/OpenFlow domains. Therefore, an investigation of how the NSI protocol could be used to provide connectivity across the

[6] Demonstration at the SC14 Conference

OpenFlow domains to facilitate participation in the BoD service (see use case described in Section D.2.2) has been conducted.

D.2.3.1 *Network Service Framework Overview*

The Network Service Framework (NSF) is an effort of the Open Grid Forum (OGF) that describes network resources as manageable objects and enables the automated provisioning of federated network services. Within the framework, network services are used by applications to monitor, control, interrogate and support the network resources. One of the key services included in the NSF is the Network Service Interface – Connection Service (NSI-CS), which enables the reservation, creation, management and removal of connections that traverse different network domains [NSF].

Architectural Elements

The NSF defines a set of architectural elements that compose the NSI Architecture, which can be applied to every service supported within the framework. All these architectural elements reside on a notional service plane called the NSI Service Plane.

- *NSA (Network Service Agent):* Software agent that implements the NSI protocol to communicate with other NSA. Can take the following roles:
    - *Ultimate Requester Agent (uRA):* First NSA in the request chain. The originator of a service request.
    - *Ultimate Provider Agent (uPA):* Last NSA in the request chain, services the request by coordinating with the local Network Resource Manager (NRM) to manage network resources.
    - *Aggregator (AG):* An NSA that aggregates the responses for its child NSAs and acts as a gateway to other providers.
- *Network Service Interface (NSI):* Provides secure and reliable sessions for service-related communication between two NSAs.
- *Message Transport Layer (MTL):* Provides reliable and secure delivery of messages between NSAs. It is the message delivery mechanism.
- *Coordinator Function:* Provides intelligent message and process coordination.

Furthermore, there is an element outside of the NSI Service Plane.

- *Network Resource Manager (NRM):* Controls and manages the local network resources.

Topology Representation

NSF identifies two different topologies, the inter-domain topology, concerned with the interconnection of the domains and the intra-domain topology, related to the resources within the network. It is worth mentioning that only the inter-domain topology is inside the scope of NSF, which is represented using a schema based on the Network Markup Language (NML) called NSI Topology [NSI-CS].

In order to identify network resources NSF defines the Service Termination Points (STP). An STP is a URN identifier that refers to a network resource capable of terminating an NSI connection, which usually identifies physical or virtual network ports.

An STP is defined as a three-part identifier:

- The *network identifier* points to the domain in which the STP is located.
- The *local identifier* to the specific resource in that domain.
- The *optional label* component allows flexibility in STP definition.

Additional qualification by a labelType and/or labelValue pair can also be used to describe technology-specific attributes of the STP.

Service Definition

The Service Definition instance describes the allowed parameters with optional value restrictions that can be used during the service reservation process. The uRA uses the Service Definition to generate the request message, whereas the uPA uses it to validate the request.

NSI-CS uses the hierarchical XSD schemas to describe the allowed parameters in the generic Service Definition (e.g. P2P Base Service) and XML documents to describe additional service-specific parameters and restrictions.

D.2.3.2 *SDN/OpenFlow Integration in NSF*

One of the most important features of NSI is that it aims to be a technology-agnostic solution, meaning that it is intended to work regardless the underlying transport technology used at the network. This is achieved by means of the Network Resource Manager (NRM), which has been previously introduced.

In the case of OpenFlow domains, NSI-CS can complement OpenFlow with the necessary mechanisms to provide multi-domain mechanism. By using an OpenFlow controller as the NRM of a domain, it is possible not only to obtain multi-domain connectivity services between OpenFlow enabled domains, but also between OpenFlow and non-OpenFlow domains. In Figure D.7, the NSI service plane with the architectural elements introduced in the previous subsection is depicted. It also shows how an OpenFlow controller can be integrated within the NSF as an NRM.

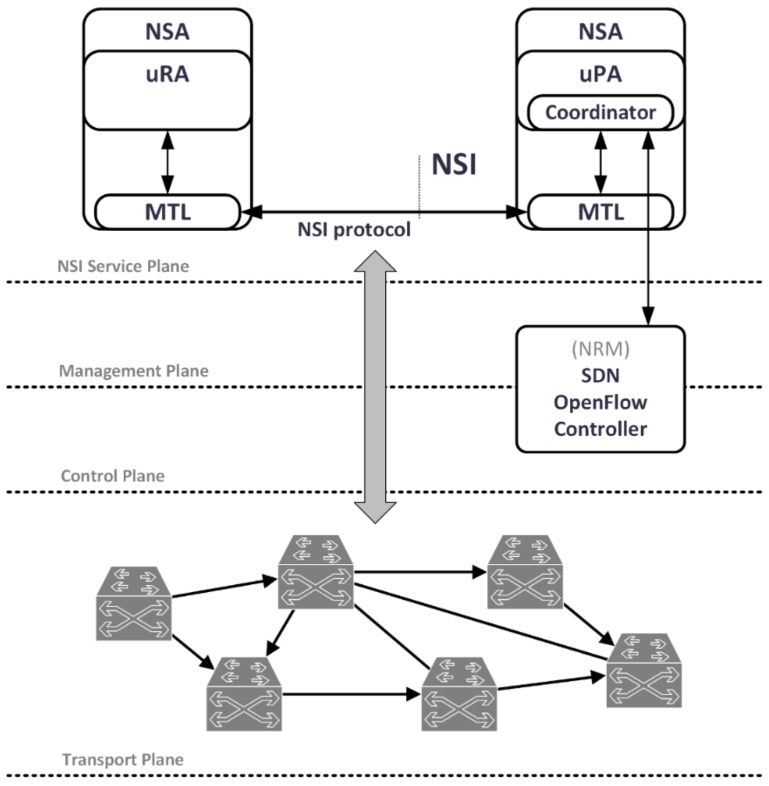


Figure D.7 OpenFlow integration in the NSF Architecture

However, the particularities of OpenFlow, especially its great flexibility and granularity, make the integration with NSI-CS a challenge. In the OpenFlow domains, packet forwarding can be carried out based on a much wider range of parameters than in the traditional forwarding approach. As a consequence, it is possible to have different types of connections within the OpenFlow domain, from pure Layer 1 connections to Layer 4 connections and combinations.

Taking into account that in the NSF the requested service must be supported in all the domains involved, the granularity of OpenFlow imposes some challenges and the optional OpenFlow-specific parameters need to be passed without disabling the possibility to setup a circuit in non-OpenFlow domains. The most recent specification of this service (v2.0) introduced new data model elements that enable the extension of the base functionalities without changing the core elements of the protocol. One of the most important features, from this considerations point of view, is the existence of the 'ANY' attribute in NSI messages, which semantic meaning depends on a defined namespace instead of on pre-defined assumptions [CTv2.0].

By utilising the newest Service Termination Point (STP) definition, it is possible to code into the 'TypeValueType' string attribute [STv2.0] information regarding e.g. a range of VLANs, a Layer 3 IP subnet/range or even the L4 TCP/UDP ports available on the interface. In that way, appropriate STP ports can be chosen to transport network traffic with specific network parameters (characteristics). The schema of 'Reserve' connection request remains untouched, and for a connection reservation, process basic NSI service type (P2P) can be used, providing compatibility with the non-SDN domains [CSP2P].

In the second version of NSI-CS, the definition of the 'Reserve' request message uses the 'serviceType' field (inside the ReservationRequestCriteria object) in order to transport additional (technology- or domain-specific) parameters within the protocol message. SDN/OpenFlow-specific information can be passed in the form of new base service type or by defining a new optional namespace in the request.

- **New service base type**: The protocol defines a base P2P service type that provides a set of properties for multi-domain connections [SDEC]. However, the ability to define additional parameters (placed in the 'ANY' attribute) could make it possible to perform the agreement between the different service domains agents involved. As a consequence, the whole service could be provided to the customers. In order to provide SDN/OpenFlow-specific connections, a new service type can be proposed. It should extend the service base type with Layer 3 or/and Layer 4 fields to enable the setup of more granular, flow-based connections. There is also an opportunity to define a new service with the needed parameters and attributes.
- **New OpenFlow/SDN-specific namespace**: Despite service-specific attributes, a 'Reserve' request possesses a ReservationRequestCriteria object with a property called 'anyAttribute', which has been added in order to cope with any domain/technology-specific extensions without a need of modification of the protocol core and the addition of new service parameters or the whole service. Custom SDN namespace might enable flexible approach to the management of network resources and enable the use of connection-related SDN apps (e.g. custom statistics monitoring, packet inspection, resiliency or load balancing etc.) or suggest a quality of service for the circuit. Domains that do not support this extension(s) should silently ignore them, instead of dropping the whole request. As one of the NSI's basic service types, [SDEC] can still be used in the connection request, circuits via non-SDN domains can be easily established.

D.2.3.3 *Proof of Concept*

A prototype based on the OpenDaylight controller, results of the DynPaC OpenCall[7] project and NSA code developed in the GÉANT project has been built. The prototype uses VLAN tags to create circuits across OpenFlow domains and communicates with other domains using NSI protocol. A demonstration involving three domains: EHU-OEF test domain located in Spain, PSNC test domain located in Poland and one geographically distributed domain created in the GÉANTs testbed is planned for the SuperComputing 2014 (November 2014, in New Orleans).

D.2.4 Multi-Domain Slice-Oriented Approach

A slice-oriented, multi-domain SDN concept is focused on how distributed slices of networking resources can be created and used among many different SDN domains. In this section, different architectural approaches and functional modules are identified that the slice-oriented version of the SDN multi-domain service may involve. A high-level overview of the available SoTA possibilities are provided, to detail specific technology solutions and components, as well as illustrate the need for a multi-domain slice-oriented solution and associated challenges.

[7] http://www.geant.net/opencall/SDN/Pages/DyNPaC.aspx

D.2.4.1 *Introduction*

SDN-based technologies have provided techniques to share and partition networking resources, i.e. “slicing”. For instance, university campuses provide many different, multi-tenant services deployed within their networks: some universities have medical or high-security facilities and must maintain appropriate regulatory compliance. Also, student networking services vary, depending on whether they are on or off campus. Administration offices must also be able to manage the day-to-day activities of the university separated from the academic activity, etc. Companies present a similar case: different departments require different networks to which only the members of each department must have access.

The previous examples constitute intra-domain situations, nevertheless, there are network services that imply multi-domain scenarios. In the case of university campuses, collaborative experimentation platforms are currently being developed to provide resources to experimenters, developers, etc. On the commercial side, companies are also establishing partnerships to set up collaborative networks and share resources across the world, as well as provide international platforms for software development testing, as in the case of FI-PPP XIFI project [XIFI], and FELIX project [FELIX].

Thus, the need of defining SDN-based scenarios comprising heterogeneous domains in order to enable new complex services and applications is becoming clear. In these complex scenarios, ensuring multi-tenancy and isolation at the same time is a must. There is also a requirement to federate domains (or at least establish collaboration agreement among them) and offer services/resources as a single entity to the tenants (taking into account that different domains are currently controlled by different SDN Controllers and Resource managers).

Slicing mechanisms constitute a key and valuable tool that SDN multi-domain frameworks should exploit, since this approach may provide solutions to some of the previously stated challenges and requirements that these types of services impose, especially in terms of service isolation. Therefore, one of the objectives of JRA2T1 aims to research current state-of-art mechanisms and technologies and define a multi-domain SDN framework incorporating the advantages of slicing techniques that enables to address complex provisioning service challenges.

D.2.4.2 *Control and Management Distributed vs Centralised Architecture Approach*

Introduction

One of the key tenets of SDN is the potential advantage it presents in the separation of a network device’s control and data planes. This separation affords a network operator flexibility while designing the control and management planes in a programmatic way.

The shape of the architecture control and management planes go hand in hand with discussions about multi-domain provisioning and connectivity services. Multiple SDN domains may imply several independent SDN control planes (i.e. separated and potentially different SDN controllers, each managed by its own NOC). At the same time, the heterogeneity of the domains entails a variety of control plane and functional blocks for network management. In summary, it is necessary for the different domains to be coordinated. In this section, the different foreseen alternatives are analysed, i.e. distributed, centralised and orchestrated control and management-plane approaches, setting out their advantages and limitations.

Centralised Approach

The SDN multi-domain centralised approach assumes the existence of a single entity (SDN controller) that controls the entire network's resources at the data plane, with each of the network resources running a local management agent. The centralised entity has potential to coordinate a very large number of agents. This centralised manager, typically known as NMS (Network Management System) monitors and administers several network domains, composed of network elements (NEs), known as managed entities. This approach makes extensive use of Northbound and Southbound SDN interfaces of the SDN controller. Figure D.8 shows an overview of the centralised approach, illustrating a manager with a single controller talking to several SDN domains.

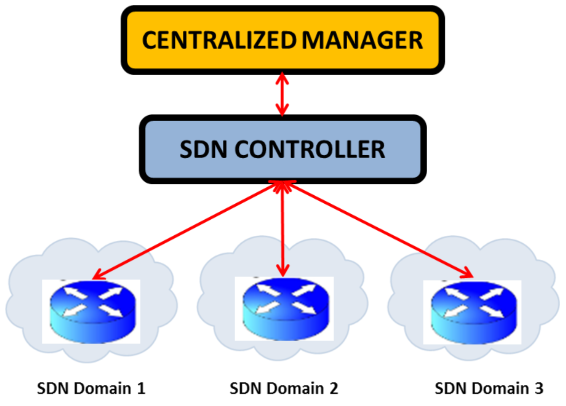


Figure D.8: *Full centralised approach*

The primary advantage of a centralised control plane is the view of the network it can provide to management, service and application layers, as well as the simplification of programmatic control. The centralised approach also simplifies the management of complex flows that are related to specific services and applications traversing several domains.

On the other hand, factors such as the following [SDN] make centralisation of multi-domain control extremely difficult and perhaps undesirable:

- **Scale**: A centrally based controller will support a control session with each managed device. As the scale and volatility of the network increases, updates to an individual element require increases in per-session I/O and processing. Additional features, such as collecting analytics through the channel or performing other management tasks, presents an additional burden. At some point, it makes sense to divide the burden into more manageable pieces.
- **High Availability**: Even if the control-session-scale burden can be handled by a single controller, that controller becomes a single point of failure that would result in the entire network failing. Even if the entire network is configured to operate 'headless', without central control functions for a significant period, at some point other network failures or changes will need interaction with the controller. If the controller has

not been restored by then, this will be a problem. The simplest high availability strategy would allow for an active/standby combination of controllers.

- **Geography**: Within a data centre, almost all elements are geographically close to each other, even if the data centre is many city blocks or many stories tall. Once the controller and controlled element are separated by a metro, state, or national network, transmission delay could begin to affect the efficiency of operation. Greater geographies also increase the risk of partition (separating the controller from the element).
- **Trust**: While thinking on a centralized approach, it is important to take into account the fact that different domains may imply different companies/SMEs/Research institutions, which definitely want to preserve their domain topology and information details. A centralised approach to network control requires a central entity with (at least) a partial view on the domains being controlled, fact to which domain owners may be reluctant due to privacy and security items.

Orchestrated Approach

In a distributed model, the centralised manager is combined with distributed controllers and data planes. The distributed model with a centralised management interface makes it look as if the whole system is being controlled centrally, even though each node is individually managed in the distributed network. In this case, the East-West controller interfaces (among the distributed controllers) play a key role in terms of network control, while propagating the rules to provide with E2E services.

The good news about distributed SDN networks is that they are evolutionary and easy to scale as networks expand, rather than the traditional centralised model of 'rip and replace' (ripping out the old system and replacing it with a new system). It is also easier to apply policies to individual departments for specific applications with this approach [SDN].

Alternatively, convergence and load balancing are important focal points for network operators/designers, and may present limitations while discussing in terms of multi-domain SDN scenarios.

- **Convergence**: One of the components of convergence that might be obvious is the propagation delay of a specific update. This is normally a function of the average distance from the site of first change measured in the number of intervening nodes that have to re-flood/re-advertise the update.
- **Load balancing** is normally applied to equal cost paths or bundled point-to-point circuits, although there are non-equal-cost variants for certain purposes. The actual efficiency of a load balancing algorithm is bounded by both the computation algorithm itself, as well as the potential imbalances in flow size an implementation might encounter. These can result in bin-packing efficiency problems that ultimately lead to limitations in the number of equal cost paths or bundle members supported by the implementation.
- From the implementation view, there is also complexity in **deploying a distributed control networ**k. It may not be possible to get to all of the nodes to create the same path with the same prioritisation – so there are timing and synchronisation issues.

Figure D.9 shows the distributed scheme; still the management is centralised and talks to distributed combined SDN controller each of which controls its own SDN domain.

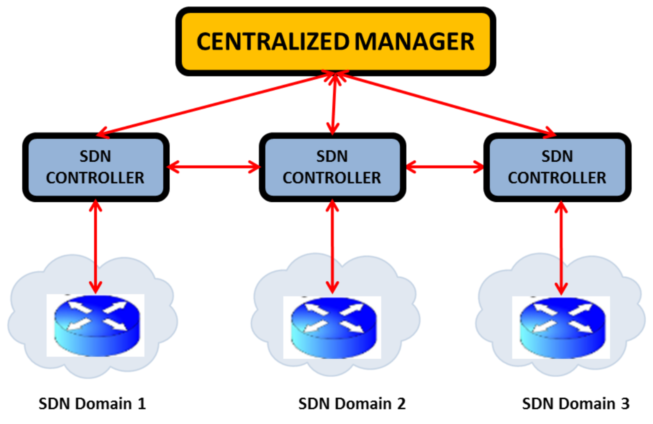


Figure D.9: Distributed approach

D.2.4.3 *Sliced-oriented, multi-domain solution based on OpenNaas management platform*

While addressing the slice-oriented multi-domain approach, several technologies and mechanisms in terms of: virtualisation and slicing (to create the flowspace), intra- and inter-domain connectivity and a multi-domain service orchestration, need to be taken into account.

This section proposes a generic approach, with examples of services and technologies that could match the slice-oriented context.

Virtualisation and slicing technologies: Networks today typically achieve isolation using low-level mechanisms such as VLANs or firewalls. These mechanisms can be used to effectively provide both traffic and physical isolation, but their use requires careful configuration by expert operators and is prone to errors. The creation of flowspaces also entails some specific requirements. Thus, some other alternatives to create flowspaces should be studied.

FlowVisor is the most prominent example of a system that provides isolation in OpenFlow networks. It allows multiple controllers to manage a single network [FV]. Architecturally, FlowVisor is organised as a hypervisor that sits between the controllers and switches, filtering the events going up to controllers and masking the messages going down to switches. In addition, FlowVisor allows an administrator to identify 'slices' of the flowspace using topology and packet header characteristics similar to slices. FlowVisor also supports bandwidth, switch control plane, and controller isolation for slices, using heuristics to estimate the amount of processing power needed to implement each slice.

Another example of a system that provides isolation using a hypervisor-based approach is XNetMon [XNETMON]. The frameworks for virtualising the described OpenFlow networks can also be used to provide various forms of isolation.

VERTIGO also constitutes a potential alternative [VERTIGO]: VERTIGO is able to instantiate generalized Virtual Topologies (VT) including virtual links and virtual ports by exposing different virtual views of the network to different controllers Vertigo could be used as a tool for enabling virtualised overlaid networks.

OpenVirteX, developed at ON.LAB, is another example of the network hypervisor that allows creation of multiple virtual programmable networks on top of a single physical OpenFlow based infrastructure [OVX].

A Flowspace firewall, developed by Indiana University, provides similar functionality to FlowVisor. It provides OpenFlow virtualisation on a per-VLAN, per-port basis and allows rate limiting of control channel messages between the controller and switch. (For more information see [FSFW].)

Intra-domain connectivity service

Each domain owner administers his or her own internal connectivity services. Therefore, while dealing with a scenario composed of OF and non-OF domains it is important to provide with a heterogeneous solution that enables the creation of the flowspaces' internal domain connections and works with established intra-domain connectivity services.

Inter-domain connectivity service:

It is necessary to make use of a stitching approach in order to link the adjacent OF and non-OF domains. One of the preferred options for this partnership is to make use of NSI protocol. A first step to stitch domains may consist of deploying the NSI agents on top of each controller of the SDN/OpenFlow domains. This corresponds to a distributed approach for a Distributed NSI framework, which presents a drawback: it may entail added complexity. Issues derived from control plane propagation latencies and the number of controllers used is typical of distributed frameworks. Distributed approaches imply more abstracted information interchange to preserve privacy. Each NSI Agent should track the controller status, and the status of the other OpenFlow domains and keep this status up to date. An alternative to address this complexity is to coordinate the NSI framework in a centralised way, with NSI plus a management platform acting as orchestrator. A deeper analysis of NSI and other potential candidates will follow up the research effort.

Service Orchestrator

The slice-oriented, multi-domain approach will make use of an orchestrator, which enables control, management and coordination of inter-domain, virtualisation and flowspace creation services. To achieve this, the candidate orchestrator may comprise the following elements:

- **Platform layer**: This layer provides re-usable components to manage services. For instance, to make use of a Driver to the OpenFlow controller in each domain.

- **Resource layer**: Specific resource types, capabilities, and devices are added in this layer. Two elements, Resources and Capabilities play a major role. A Resource represents a manageable unit. It can be a switch, a router, a link, a logical router, or even a SDN Network. Capabilities also constitute interfaces to a given Resource functionality. For instance, for the SDN Network resource, a Capability could be the "Network Topology".

- **Intelligence layer**: This layer groups external application-consuming APIs in order to perform complex network operations, thus leveraging network abstraction provided by underlying components. The NSI is one of these services. Each NSI "greenbox" operates on top of each of the SDN Network resources, so that these become "NSI domains".

Thus, the orchestrator has the perspective of the entire domain. One potential orchestrator for the centralised approach could be using OpenNaaS [OPENNAAS] as a management platform system. Besides the orchestration functionality, OpenNaaS includes some other options and functionalities that could be re-usable for the E2E multi-domain connectivity provision. Some examples include:

- REST-APIs to smartly perform the multi-domain service appearance to the user requesting the service, and abstract all the service complexity for the user.
- It may enable certain management rights delegation to the user, e.g. not only to enable to request a multi-domain connection between two end-points, but also to configure certain flow-space options of specific domains to which the user has rights.
- Hybrid scenarios (SDN and non-SDN) could be easily integrated in OpenNaaS.

There are some other alternatives (e.g. OpenDayLight) that will be further analysed.

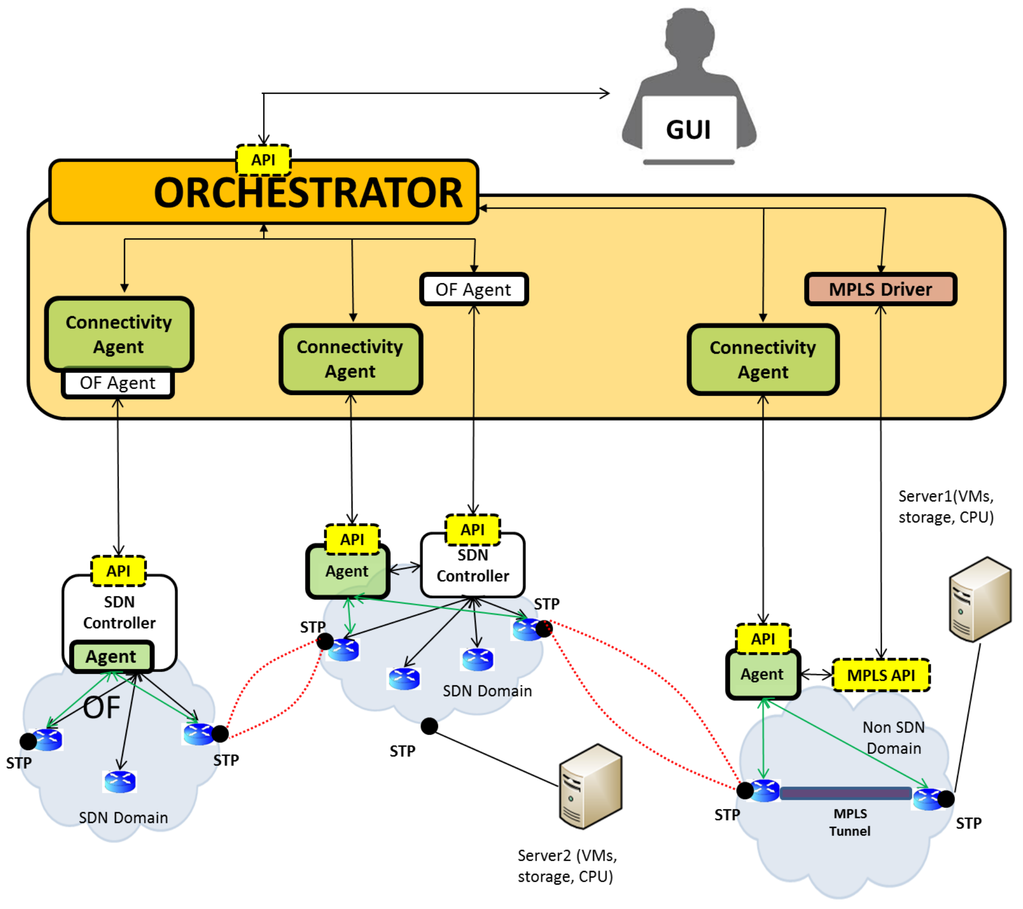


Figure D.10: Sliced-oriented, multi-domain solution approach.

Figure D.10 shows an example of the slice-oriented, multi-domain approach. There are three domains. Two of them are OF domains and there is also an MPLS-based domain. The orchestrator manages and controls the inter-domain connectivity service configuring the border nodes of each domain acting as Service Termination Points (STP). The orchestrator also interacts with OF agents and a MPLS driver to tune the intra-domain connectivity services. Each of the services in the system should expose an interface to the orchestrator so that all the configuration setting can be done. The consumer of the flowspace would control and manage the resources assigned by means of the orchestrator. This offers a GUI with specific, enabled features to guarantee that the user is able to configure the assigned resources according to his/her preferences.

Appendix E SDNapps

E.1 Changing the Rules and Perspective

In SDN, applications own a real opportunity to rule and define the network. Applications are enabled to tap into information that the controller possesses in terms of traffic patterns, application, etc., and to react accordingly to serve whatever the application desires; thus, business applications can operate on an abstraction of the network, leveraging network services and capabilities without being tied to the details of their implementation. SDN makes the network not so much "application-aware" as "application-customised" and applications not so much "network-aware" as "network-capability-aware". Nevertheless, there have been specific issues observed that constitute a real limitation while applying SDN to Layer 4-7 networking and making the best of the features that SDN technology provides with. The current work is focused on Layers 4-7 and is devoted to propose the so-called SDNapps as part of the SDN architecture framework. The overall target consists of introducing a middleware layer to enable SDN with the mechanisms to customise the SDN control layer and configure the forwarding plane “a la carte”, depending on the requirements that on top application exert, overcoming observed limitations.

In the following sections, relevant SDN-based platforms and architectures are presented and the observed limitations towards the SDNapps concept, architecture and benefits. SDNapps do not present a disruptive concept for SDN, but are an added value for stakeholders, while providing their services on top of an SDN-based environment.

E.2 Most Relevant SDN Controller Proposed Frameworks

Starting from previous SDN controllers’ developments, several SDN framework initiatives have been developed. The most relevant are briefly reviewed for its common features below. For further information and architecture details, check appendix document about SDNapps [SDNAPPS].

SDN frameworks	Description
Floodlight	Floodlight is an open free Apache-licensed Java-based OpenFlow Controller and a collection of applications built on top the floodlight controller. These are used to solve different user needs over the network [Flood1]. The design implementation principle of FloodLight is the following one: “Everything is a module”: Modules export services. According to the SDNapp definition provided in Section E.3, both modules and REST applications are SDNapps.
Ryu	Ryu is a component-based, software-defined networking framework, Apache 2.0-licensed and fully written in Python. Besides OpenFlow, it also supports various protocols for managing network devices. Ryu architecture follows the standard SDN architecture. The SDNapps communicate with the Ryu SDN framework by means of well-defined APIs. The control layer includes several

	Ryu apps that are also used as services by the SDN framework, as the “Module Applications” are used in the Floodlight design.
OpenDaylight	OpenDaylight is an open source project with a modular, pluggable, and flexible controller platform at its core. This controller is implemented strictly in software and is contained within its own Java Virtual Machine (JVM). The controller exposes open northbound APIs that are used by applications. OpenDaylight supports the OSGi framework used for applications that will run in the same address space as the controller while the REST (Web-based) API is used for applications that do not run in the same address space (or even necessarily on the same machine) as the controller. The business logic and algorithms reside in the applications.
HP SDN Application Store	HP has created a software development kit (SDK) for the northbound APIs on its SDN controller and the HP SDN App Store, where customers can easily browse for SDN applications and download and install them on HP's controller. The HP SDN SDK provides developers the essential tools to create, test, and validate SDN applications, leveraging HP’s SDN infrastructure and full complement of support services. The HP SDN App Store creates an open marketplace for SDN applications [HPAPPs].

E.3 SDNapps Concept Statement

E.3.1 SDNapps Definition

SDNapps can be described as generic network functionalities running on top of the SDN/OpenFlow network, adapting the control plane to fit operator and carrier requirements in terms of control and management and consequently configuring the forwarding plane “à la carte”. Consequently, SDNapps can be understood as a kind of packetised and modular middleware functions that implement specific functionalities and core services of common interest integrated within these SDN controllers, but also, as controller-like and controller-less applications placed on top of the REST API interface to the SDN framework. While important classes of SDN application make sense integrated with the SDN controller (e.g. network virtualisation, virtual firewalls, etc.), also integrating performance aware SDN applications (e.g. load balancing, DDoS mitigation, traffic marking, multi-tenant performance isolation, etc.) within the analytics platform makes architectural sense [DIFFERENCE1]. Taking a detailed look, a basic SND controller’s core functionalities can be defined as those that enable the controller to learn and communicate with lower layer OpenFlow devices, being aware of the network topology and being able to receive packet-in inputs and install flowmods into the flow rule tables of OpenFlow devices accordingly, plus additional OpenFlow standardised services. Those functionalities could be denoted as basic or core for the controller. Extra functionalities can be added thanks to the deployment SDNapps, or even improve or replace functionalities integrated by the controller. Moreover, an SDN application is a software program designed to perform a task in a software-defined networking (SDN) environment that can replace and expand upon functions that are implemented through firmware in the hardware devices of a conventional network [SDNDEF1]. Thus, instead of having controller-specific functionalities, these should be able to be packetised and externalised, so that depending on the specific needs required by the on-top applications, specific SDNapps could be incorporated. This feature provides flexible, scalable and automation control to the application layer, gaining unprecedented programmability and enabling business to rapidly adapt according to new business needs and requirements.

A top-standardised interface from a controller would ease the use and communication of SDNapps on different SDN controllers or even between apps, but it would also limit SDNapps capabilities to a specification that would be really difficult to achieve. An alternative, feasible solution is a middleware framework capable of communicating with different SDN controllers and SDNapps. This middleware framework would be responsible of managing communication between different SDNapps and behave as an adapter. Furthermore, a middleware framework that integrates Basic Network functions and services with a bunch of SDNapps could act as a controller. Deploying additional SDNapps, the framework would extend its functionalities and value, and would give a new range of opportunities to tailor a custom “controller” shaped for the customers’ needs. An SDN controller is useful because it provides that single point of control to modify the forwarding path in the switches, but it's an enabler for customers to build the applications they need specific to the business problems they are trying to solve. SDNapps enable network automation, multi-tenancy and integration. SDNapps appendix document includes extended information about SDNapps and the proposed framework [SDNAPPS].

E.3.2 Overcoming Barriers of Current Solutions: SDNapps Motivation

SDN Network solution cannot be a ‘rip a replace’ one, but rather an integrated one: SDNapps packaged design contributes to the progressive technologies replacement and migration towards SDN, by providing modular pieces of software, capable of integration in an SDN-based environment, and enabling new SDN-based functionalities in the network environment as well as existing ones in such a way that the legacy technology replacement can be progressively performed by adding functions in the forms of pieces (packages, modules, SDNapps) of software.

Managing control traffic in centralised networks: The combination of service control software and multi-vendor network telemetry illustrates a clear missing link for SDN: management of control traffic in centralised networks. In any SDN model where information is collected to support central oversight of network behaviour, there's a balance between the value of knowing enough and the cost of delivering too much information to that central point. SDNapps enable virtualisation of specific functions, delegating the intelligence, control and management of concrete features. By releasing heavy-load control functions to the controller and migrating them in the form of pluggable/unpluggable SDNapps, the control and management traffic and specific heavy-load computation operations constitute a smart compromise.

E.3.3 SDNapps Framework

The SDNapp framework is defined as a Middleware deployed on the application layer and it is hooked to the northbound interface over a SDN controller that might perform as a proxy. Figure E.11 shows SDNapps' allocation in an SDN-based architecture.

Basic Network Functions: A basic SDN controller, shaped with Basic / Core Network Functions may integrate proxy services, a topology manager to enable awareness of Forwarding plane, a device manager to discover hosts and OpenFlow-connected devices to the network, and a forwarding module able to read an input, compute rule decisions and finally install the proper rules on the OpenFlow devices. The most important SDN controllers can now share a common point on its architecture. Those basic functionalities are there to follow and meet the standardised OpenFlow specification: a set of common functionalities to control and manage an OpenFlow network.

Middleware framework: This provides the platform that enables the build of SDNapps. Such a platform (i) provides a reference point to users making use of already deployed SDNapps (or deploying additional ones) as well as (ii) integrates in the control plane to coordinate SDNapps with the Basic Network Functions of the SDN controller. The proposed middleware framework is OpenNaaS [OPENNAAS]. The middleware also exposes an interface able to incorporate already-developed SDNapps from external repositories. This constitutes a very valuable feature, since the programmatically aspects and services that can be incorporated in SDN-based environments can be easily enlarged. An alternative solution is an integrated development environment, such NetIDE [NetIDE] proposal. Similar to OpenNaaS, NetIDE will deliver a single development environment point to support the whole development of controller apps in a vendor-independent fashion, offering a solution for developers to deal with the current fragmented control plane in OpenFlow networks.

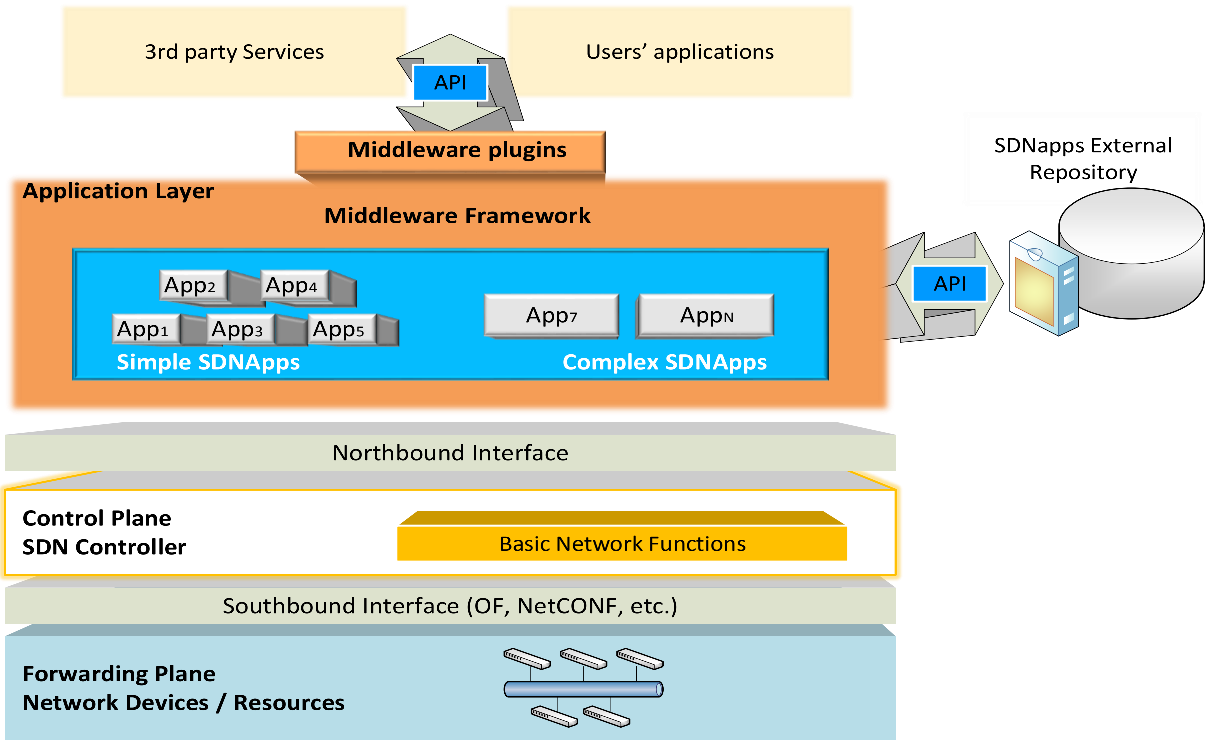


Figure E.11: SDNapps placement in SDN-based architectures.

Simple and complex SDNapps: The main difference between simple and complex SDNapps relies on the following: a simple SDNapp can be defined as an atomic software program designed to provide a service or perform single function, such as a topology mapping service, a basic firewall or a load balancer. Complex SDNapps can also be a module or package of software, even multiple modules or packages, made of many atomic software programs or basic SDNapps. They could be overlaid topology services (based, for instance, on previous topology SDNapp), complex QoS-based routing algorithms, content availability, etc. Or even replace and improve a basic firewall SDNapp extending its functionalities, options and capabilities. Even more, Complex and Simple SDNapps could communicate and share data between them, offering improved functionalities. Thus, SDN based architectures enable the possibility to programme, customise, shape and automate the network control based on the SDNapps.

Middleware plugins: Middleware plugins enable users to consume SDNapps, to programme the network as well as integrate their own applications with the SDNapps in order to develop more advance SDN-based services.

E.3.4 Challenges While Applying SDNapps

The main challenges in applying SDNapps are found in the lack of a standardised northbound interface, or a specification to determine some common points in the northbound API of each SDN controller. While standardising an API for SDN applications has been debated for a long time, an open framework in place of the northbound API is a solution for a lack of northbound standardisation, essentially allowing applications to be part of an SDN environment, and enabling a wide range of different network applications. The current SDN environment is full of controller API implementations, which causes a fragmented SDN environment, where application developers find it difficult to decide where to focus their implementation. This means that an SDNapp is made to work on a single SDN controller, with a complete lack of portability. A great diversity of implementations can be found in SDN deployments, and that is the most important challenge present because it is a strong barrier that hurts the adoption of a viable SDN ecosystem. An open framework solution to act as a Middleware framework would allow the community to work together and cooperate on their effort to increase portability and develop SDNapps for different SDN controllers' implementations, adding new features and API extensions without being tied to a single controller's API.

E.4 SDNapps Key Benefits – Who can benefit from SDNapps?

The most exciting opportunities for SDN are found in the key benefits of the SDN applications that can be built upon its framework. These benefits are described below.

SDNapp key benefits	Description
Costs reduction	A key point of a SDN-enabled network, is that service providers can create any number of software applications that can be deployed over a network controller without any need of specialised hardware. This can cut their opex and capex while promoting better service to the end user by allowing for optimisation with less oversubscription.

SDNapp key benefits	Description
Network Services easy customisation	Network operators and other stakeholders will find many benefits in the ease of customisation of the network services, along with the opportunities of introducing new capabilities into ageing and complex networks. This enables a tailored control and granular programmability to enable specific services on the network.
New monetisation opportunities	SDNapps offer a wide range of monetisation opportunities. Similar to the launch of consumer app stores for mobile devices, SDNapp stores in place, can be made available for networking gear, allowing Software Providers and developers to be able to develop applications that might suit many interests.
Simplified deployment	Thanks to SDN northbound interfaces, deployment of tools and services is simplified with SDNapps. It allows implementation of these capabilities on an SDN network quickly and inexpensively, changing the speed and flexibility of how networks are managed.
Control of networks' resources	Cloud Service Providers and Network Infrastructure vendors will benefit from SDNapps control of/or visibility of underlying networks and resources that enable much simpler provisioning in a multi-vendor virtual environment. SDNapps allow monitoring of network traffic and devices in real time, responding dynamically upon application's changes.
Highly scalable, efficient and manageable network services	SDNapps, much like virtualised network services, deliver highly scalable, efficient and manageable network services in form of protocols, custom logics, and algorithms that are used to program the forwarding plane / control plane.
Improved security	SDN allow Cloud Service Providers and other stakeholders to deploy apps such virtual intrusion detection system or virtual firewall on a network controller. It can collect information about traffic patterns, application data and capacity and enable custom security levels on the network.

E.5 SDNapps Success Use Cases

At the time of writing, a number of stakeholders are developing and providing suites and catalogues of SDN applications. Such catalogues constitute a good point of reference as a starting point, and might be relevant for NRENs and GÉANT community. Some of them relate to Network optimisation through automation. Data centre migration SDN applications are also receiving the attention of the industry due to the great impact that Cloud exerts on the Network. In the following section, authors provide with some early commercial SDN applications and Section E.6 describes the implementation of an SDN application done for this JRA2T1, which provides and manage Real-Time Quality of Service (QoS) through an end-to-end network for real-time applications; this is the QoS Pathfinder SDNapp.

E.5.1 SDNapps Commercial Use Cases

SDN Security Application Example [SDNappex1]: This SDN security application offers protection by looking at DNS queries and avoiding any nefarious URLs. As the user makes various DNS requests, instead of having an appliance looking at these, the switches can intercept them and send them over to the controller. Without adding any new hardware, the controller can run them by the HP TippingPoint IDS database to the URLs.

Another SDNapp security example [SDNappex2]: When it comes to security, SDN can use this information for traffic engineering to direct flows to specific firewalls or IDS/IPS elements, thus helping to align the right security application with the right traffic flow. In addition, separating the logical from the physical aspect of the network allows Layer 4-7 attributes to follow the application as virtual machines migrate to new physical locations.

Content Availability [SDNappex3]: SDN apps built to handle content availability will be able to provision flows in the network based on the type and availability of the content. Before routing requests to servers, SDN applications can check the availability of the content in the content servers. A content-routing SDN application will enable discovery of content in the content servers and provide intelligence on its availability. It can be used to route requests to the correct server where the content resides. Therefore, SDN applications will be able to route requests from websites that generate dynamic content, which are non-cacheable, to a server providing dynamic content rather than a caching server, greatly reducing network latency.

Service Availability [SDNappex3]: SDN applications will also be able to monitor the availability of network services across the entire network before routing data. Using SDN applications, content routing can be designed to perform service-availability checks before provisioning flows to the network switches. Traditionally, network monitoring services only check the availability of Layer 2 or Layer 3 paths. However, in the instance of the content-delivering application being down, this would not be picked up when monitoring only Layer 2 and Layer 3 paths.

E.6 SDNapps Implementation

A proposed SDNapp use case is the QoS Pathfinder [PATHFINDER], an SDNapp based on Network as a Service paradigm (NaaS) to provide and manage Real-Time Quality of Service (QoS) through an end-to-end network for real-time applications, adapting the network's control plane to satisfy the requirements of an application or service thanks to the use of the standardised OpenFlow interface, which supports basic QoS offerings. The application's core is an algorithm called Pathfinder that is responsible for the dynamic demand and on-demand provisioning of network resources, which takes into account network-wide traffic implications as link state or port stats using a controller's counters or a monitoring an SDNapp. A proof of concept of an SDNapp that can work within the Middleware Framework, it has an interface (RESTful API) that enables communication with the Middleware Framework, third-party services and users. SDNapps appendix document offers further details of QoS SDNapp implementation [SDNAPPS].

E.6.1 QoS SDNapp Proof-of-Concept: Pathfinder Implementation

The Pathfinder SDNapp implementation tries to take advantage of the mechanisms provided by the OpenFlow protocol. Pathfinder makes use of the 'slicing' technique (discussed in SectionD.2.4) to provide QoS as BoD from an OpenFlow network. It also makes use of a queuing mechanism to push and keep QoS state in the OpenFlow network that relies on network devices because OpenFlow 1.0 protocol does not provide a queue setting mechanism. The SDN application can be classified as a Complex SDNapp because it consists on a package of various atomic SDN apps responsible for a process function that works together to provide QoS on-demand. Figure E.12 shows the architecture of Pathfinder SDNapp on a SDN architecture.

At the application's core, the Pathfinder algorithm is responsible for the dynamic demand and on-demand provisioning of network resources, which takes into account network-wide traffic implications as link state or port stats using controller's counters or a monitoring SDNapp.

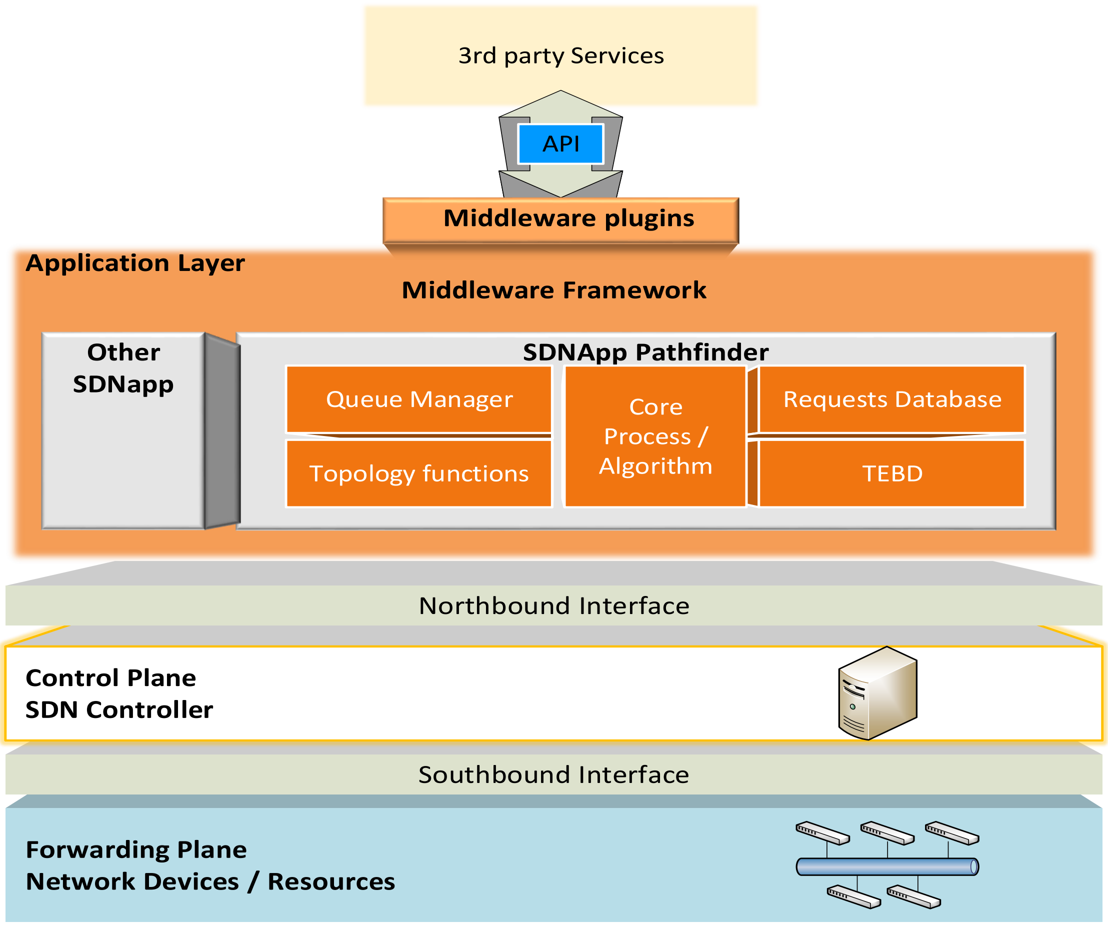


Figure E.12: QoS SDNapp architecture on a SDN environment

Each atomic SDNapp that forms Pathfinder performs one or more functions inside the app workflow. The next steps show a proof-of-concept implementation of a first version of the QoS provisioning SDNapp. The initialisation of the workflow happens on the Core / Main algorithm, where a QoS request is received. In a first approach, the

QoS request is stored on a cache file, where the SDNapp reads its contents. On further versions of the implementation, SDNapp will be able to load QoS requests through the upper API from the requester service or application, where QoS requested constraints are specified to achieve a level of QoE. The QoS request contains critical data such as:

- Single QoS bottleneck attribute: Minimum bandwidth.
- Additional QoS parameters such as packet loss, delay and/or jitter.
- Flow parameters: IP Source, IP Destination, ToS or other OF-matching entry.

The request includes a unique identification code, which is checked on a request log/database. A new ID will start the Pathfinder process to compute a new QoS path. On the other hand, when the ID is found in the database, a re-routing process is started for a given flow, meaning that a QoS path cannot achieve a level of QoS and needs to be changed. Figure E.13 represents this initialisation.

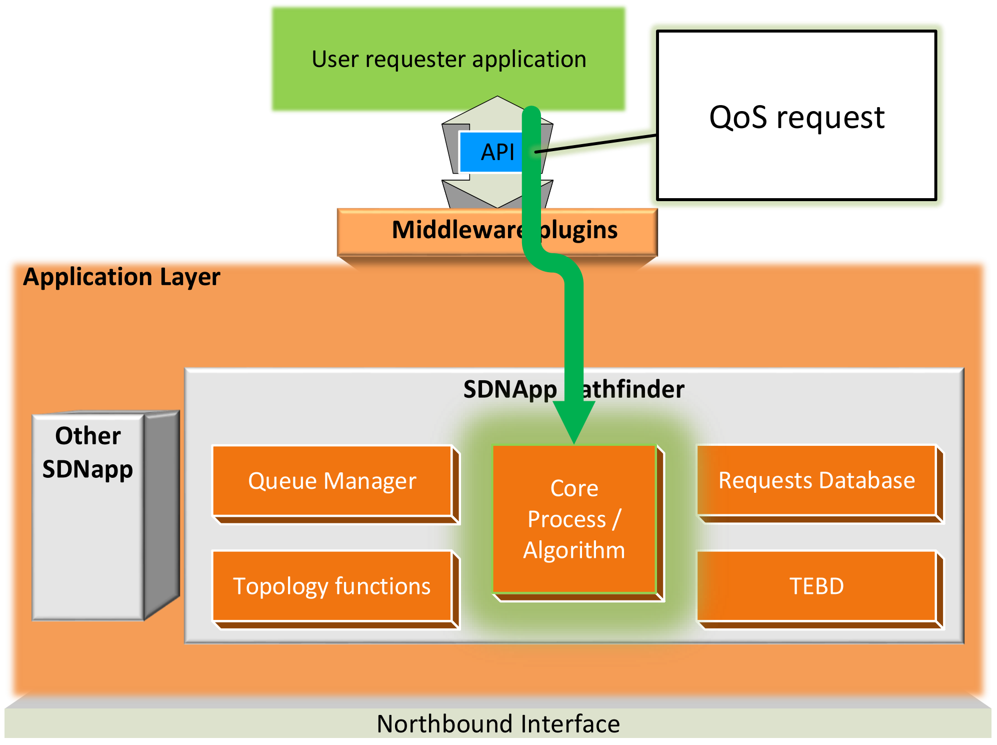


Figure E.13: QoS request reception on SDN application

E.6.2 Pathfinder Real-time Topology Abstraction

The network topology is learnt and built on the Topology Manager by polling the information about network nodes, links and source-destination attachment points from the SDN Controller. Then, a topology graph as a data structure computes all feasible paths meeting the processed request. For additional topology stats, network QoS parameters can be retrieved and evaluated from a Monitor SDNapp if available, e.g. collecting OpenFlow counters through a switch-port monitoring mechanism. Topology statistics are retrieved from OpenFlow counters of each switch, which enables the computation of packet transmission rates, bandwidth, packet errors and packet drop rates. Then these parameters can be shared with Pathfinder to be assigned as link weights to its corresponding edge in the topology graph. On the other hand, for some QoS parameters such delay or jitter, use of third-party applications or hybrid OpenFlow models are necessary.

E.6.3 Path Computation

The core algorithm uses path computation algorithms stored in the traffic engineering database. A common process for every request is to apply a graph reduction (Step 1) based on an analysis of network links meeting QoS constraints. Then, the end-to-end connection is checked. Therefore, if connection check succeeded, the core can operate two subsequent processes upon path requirements (Step 2). The core algorithm applies selection criteria whether the request demands a path meeting a single or multiple constraints: if a single constraint is requested, then a path computation algorithm is chosen from the TEBD and applied according the requested constraint. On the other hand, if multiple constraints are requested for a QoS path, then a weight aggregation process starts to apply the according path computation algorithm. The current implementation supports a single QoS constraint based on BoD for a maximum weight path along with minimum hop count.

E.6.4 Path Application

If a feasible QoS path is found, the Pathfinder core communicates with the SDN controller to push the necessary flow rules into network nodes to create a circuit across the network. It makes use of REST API to dynamically create flow rules and push flowmod messages to the controller to enable an end-to-end path between two points on the network. This step will also be responsible to create or dynamically configure port queues to each switch contained in the QoS path. The current implementation makes use of the Floodlight controller, which uses OpenFlow protocol version 1.0. Creation of queue and setting its min/max bandwidth cannot be done from Floodlight controller. All these configurations need to be done out of band and rely on switch capabilities. For Open vSwitch (current software switch utilised for testing), at least two queues are configured on each switch for one QoS flow. One queue, called ‘q0’, is responsible of Best Effort traffic in its port without any forwarding schedule. Instead, another queue, called ‘q1’, is responsible of guarantee a minimum bit-rate for QoS traffic.

Finally, when a QoS path is applied into the network, it is logged in the active QoS paths database along with its included queues managed by the Queue Manager, shown in Figure E.14.

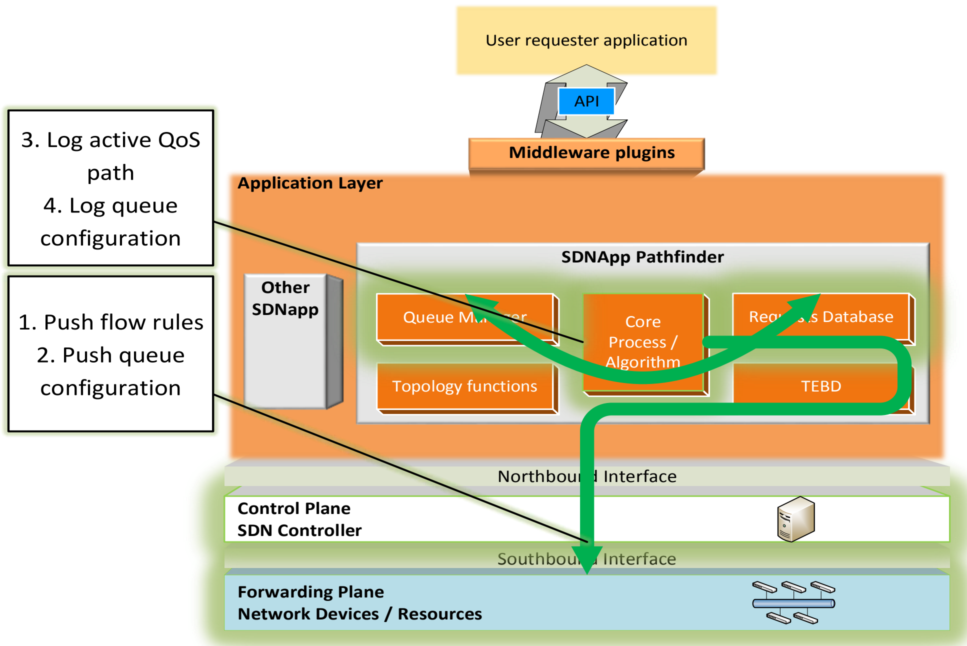


Figure E.14: QoS path application on Forwarding Plane

E.6.5 Implementation Roadmap

At the moment of writing, SDNapp Pathfinder is under development. As a proof of concept, core modules and basic functionalities are implemented to provide QoS provisioning. Code is implemented in Python language and uses JSON notation as data definition language. Current implementation includes next modules:

- Core process Algorithm (Initialisation, Topology graph abstraction, Path Computation, Path Application) module
- TEBD includes path computation algorithms based on NetworkX libraries:
  - BFS/shortest-path algorithm
  - Custom Adapted Yen's k-shortest path (AKSP) algorithm
  - Custom Longest/Heaviest path algorithm
- Topology functions module
- Active paths / Request Database log
- Queue Manager module (for Open vSwitch)

On the other hand, current work includes architecture changes, extended functionalities and support:

- Adapter: Entry point for network data and QoS requests. Responsible for gathering necessary information to process and build proper input for the SDNapp.
- API: SDNapp interface and web service. It will allow checking log data, request information and enable a web service to send remotely QoS requests.

- Computation functions and algorithms module: Critical logic functions and computation algorithms. It will enable Pathfinder to support additional QoS parameters as delay, jitter or packet-loss, from a single constraint to multiple constraints computation. New path computation algorithms include:
    - Path length computation
    - AKLP: Adapted *k*-longest path
    - AKSP: Adapted *k*-shortest path (rework)
    - ALP: Adapted Longest Path
    - Stats aggregated: Multi-constrained Path problem
- Path Pusher / Queue configuration: Path Application process will be decoupled from the main core algorithm. It will enable flow rules configuration from obtained output of path computation. Queue configuration may remain the same.
- GUI: Not defined yet, it may be useful to show topology graph and feasible paths available on network.

Some considerations for future works also include:

OpenFlow 1.3: introduction of meter bands – rate-limiting via meter tables. Use of meter tables instead of queues consists of meter entries that defines per-flow meters. Per-flow meter entries enable OpenFlow to implement various simple QoS operations, such as rate-limiting, and can be combined with a per-port queue mechanism to implement complex QoS frameworks, such as Diff-Serv. A meter measures the rate of packets that are assigned to its corresponding flow and enables rate control of its packets [OF1.3.0].

E.7 SDNapps and NFV

SDNapps are related to the SDN context, with principal elements:

- Separation of control and forwarding functions (Control and Data Planes)
- Centralisation of control (Control Plane)
- Enable the programmability of the behaviour of the network thanks to well-defined interfaces

Network Functions Virtualisation (NFV) are virtualised functions, not necessarily conditioned to an SDN based environment. NFVs are highly complementary to the SDNapps, but not dependent on it (and vice-versa). A NFV can be implemented without any requirement for SDN and any NFV can be achieved using any non-SDN mechanisms, even though both solutions can be combined. The main goal of the NFVs is to relocate the network functions from dedicated appliances to generic servers.

SDNapps can provide not only network services, but also resources or capabilities, while NFV applies to 'low-level' related functions, which are very close to Data Plane (even at the packet level). NFVs may be located on the Control Plane, impacting directly the Data Plane. Instead, SDN operates on the Control Plane to indirectly impact the Forwarding Plane. Approaches that rely on Control and Data Forwarding Planes' separation, as SDN can enhance performance and simplify deployments' compatibility. NFVs can support SDNapps by providing the infrastructure upon which they can be deployed.

Alternatively, SDNapps are more focused on the top of the Control Plane, the Application Level. Moreover, an SDN controller could for instance be consumer of NFVs, e.g.: a NAT virtualized function could be used by the SDN controller which could make its action over the NAT NFV in such a way that the mapping of Private/public addressing could be modified.

Differences exist in the scope of the two approaches, where SDN was initially focused on campus and data centre / cloud networks while NFVs for Service provider networks. The initial applications for the SDN were Cloud orchestration and networking, which were expanded by the SDNapps wide range of applications and functions. However, NFV focuses on network functions such routers, gateways, firewalls, CDN, WAN accelerators, etc.

E.8 Conclusions

SDN can be seen as a Middleware layer that enables applications to be deployed on the northbound side and communicate to the network via the southbound. An SDNapp environment is a networking scenario where applications allow users to adapt network environments to meet their needs, an approach similar to Application-defined networking (ADN) model, which implements software applications to enable dynamic management of network resources. SDNapps offer an increased value to the SDN thanks to many of its key benefits such as costs reductions and new monetisation opportunities, but also for its simplified deployment, control, manageable and customisable network services.

To achieve an SDNapp environment, SDN applications need a point of interconnect between network software and infrastructure through application-program interfaces (APIs), and this point is where SDN is found lacking solutions at present. Most known SDN controllers (NOX, POX, Floodlight, Beacon, Ryu, ODL, etc.) offer their own high-level set of SDN and distributed systems libraries, and their own northbound API, upon which an application ecosystem could be built, but each controller has its own set of interfaces (RPC, RESTful, etc.) and interconnections, and without any common set of abstractions to work from, software developers have no guarantee that their apps will be portable across a wide range of SDN environments, with many controllers on it. Networks need to become agile and deployed services must be delivered and perform optimally across networks' domains without compromising on portability. While there has been some discussion around northbound APIs standardisation, the ONF has clarified that a standard for northbound APIs is not being formalised at this time. ONF believes that no single northbound interface will serve all use cases and environments. Until now, the industry is providing APIs that developers can write to, but those applications cannot be moved between systems. Any standard for northbound APIs may limit and stem innovation and users' implementation.

A potential solution is a Middleware framework that allows applications to be plugged and unplugged according to their needs, rather than adjusting each application and infrastructure repeatedly to get a service running. In addition, a Middleware framework could act as an adapter, where multiple-vendor SDNapps may be able to communicate to a single controller, thanks to an open northbound interface that interconnects the framework to the users. This means that users would build their own applications and deliver features they need. There is a wide range of possibilities that can be addressed by the Middleware framework. Furthermore, this allows customisation of customers' network controllers and services, as shown by the Cyan's Blue Planet SDN Platform, a proprietary framework [CYAN] and OpenNaaS. Finally, progress on the abstraction and API front is needed, and a Middleware framework for an SDNapp ecosystem can help avoid a very difficult application environment in the SDN.

References

[Arbor]	Arbor Networks, Cisco Catalyst 6500 Supervisor 2T/Peakflow SP Interoperability Testing, http://www.arbornetworks.com/component/docman/doc_download/556-arbor-peakflow-sp-cisco-catalyst-6500-supervisor-2t-interoperability-testing
[BIG-2013]	Big Switch Networks, Open SDN for Network Visibility, July 2013, http://www.bigswitch.com/sites/default/files/sdnresources/monitoringfabric.pdf
[CIS-2014]	Cisco IOS NetFlow, http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html
[CIS-2014a]	Cisco Solution Brief, Cisco Extensible Network Controller with Monitor Manager Solution: Increase Network Traffic Visibility, Cisco 2014, http://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/extensible-network-controller-xnc/solution-overview-c22-729753.pdf
[CLWATCH]	Seungwon Shin and Guofei Gu."CloudWatcher: Network Security Monitoring Using OpenFlow in Dynamic Cloud Networks."" Oct 2012. http://people.tamu.edu/~seungwon.shin/Cloudwatcher.pdf"
[CSP2P]	OGF. NSI Connection Services P2P v2.0 WSDL schema. https://code.google.com/p/ogf-nsi-project/source/browse/trunk/ConnectionService/ogf_nsi_services_p2p_v2_0.xsd
[CTv2.0]	OGF. NSI Connection Types v2.0 WSDL schema, https://code.google.com/p/ogf-nsi-project/source/browse/trunk/ConnectionService/ogf_nsi_connection_types_v2_0.xsd
[CYAN]	http://www.cyaninc.com/products/blue-planet-sdn-platform
[CQUE]	JRA2T1 internal document 'Cloud questionnaire', https://intranet.geant.net/JRA2/T1/Documents/Cloud%20support/SDN_Cloud_SupportQuestionary_Brainstorming_final.docx?Web=1
[DEFENSE4ALL]	https://wiki.opendaylight.org/view/Defense4All:Tutorial
[DIFFERENCE1]	http://blog.sflow.com/2013/10/embedding-sdn-applications.html?goback=%2Egde_4359316_member_278199095#%21
[DOVE]	R. Recio, "Distributed Overlay Virtual Ethernet (DOVE) Networks" (2012) Available from http://www.ethernetsummit.com/English/Collaterals/Proceedings/2012/20120222_2-103_Recio.pdf"
[DPKT]	DPKT library https://code.google.com/p/dpkt/
[DREAMER]	DREAMER: https://intranet.geant.net/JRA0/DREAMER/SitePages/Home.aspx
[ESnet]	Energy Sciences Network https://www.es.net/about/
[ETSINFV]	Network Functions Virtualisation website at ETSI, http://www.etsi.org/technologies-clusters/technologies/nfv
[EXT-2013]	www.extrahop.com, ExtraHop-Arista Persistent Monitoring Architecture for SDN, http://www.extrahop.com/wp-content/uploads/2013/08/ExtraHop-Arista-Datasheet-1.pdf
[EXT-2014]	Extreme Networks, Advanced Features, ExtremeXOS 15.5 User Guide, 120936-00, April 2014, San Jose, CA, USA, Advanced_Features.pdf
[EXT-2014a]	Extreme Networks, ExtremeXOS Release Notes, Software Version ExtremeXOS 15.5.2-Patch1-1, 120935-00 Rev 06, August 2014.
[FELIX]	http://www.ict-felix.eu/

[Flood1]	http://www.slideshare.net/patrickhuang712/floodlight-overview-performance-comparison-by-patrick-huang
[FSFW]	Flowspace firewall website, http://globalnoc.iu.edu/sdn/fsfw.html
[FV]	FlowVisor website, https://github.com/OPENNETWORKINGLAB/flowvisor/wiki Also Ali Al-Shabibi and Rob Sherwood. "FlowVisor." https://github.com/OPENNETWORKINGLAB/flowvisor
[GÉANT]	http://www.geant.net/Pages/default.aspx
[GIO-2014]	K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris, Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments, Computer Networks, Volume 62, April 7, 2014, Elsevier, pp. 122-136, http://www.sciencedirect.com/science/article/pii/S1389128613004003
[GOCX]	gOCX, https://intranet.geant.net/JRA1/T2/Documents/draft-gn3plus-ocx-architecture-v08.1.pdf
[GOCXMOV]	gOCX movie, https://intranet.geant.net/JRA1/T2/Documents/Demo_TNC14/JRA1T2_GEANT_ocx.mov
[HNX]	Helix Nebula (HN), http://www.helix-nebula.eu/
[HOG-2013]	Scott Hogg, Using SDN to Create a Packet Monitoring System, NetworkWorld, Dec. 15, 2013, http://www.networkworld.com/article/2226003/cisco-subnet/using-sdn-to-create-a-packet-monitoring-system.html
[HP-2013]	Hewlett Packard Development Company, L.P., HP 5920 & 5900 Switch Series, Network Management and MonitoringConfiguration Guide, Part number: 5998-2900, Software version: ESS2305, Document version: 5W100-20130830
[HP-2012]	Hewlett Packard, HP Switch Software (3500 switches, 3500yl switches), OpenFlow Supplement, Software version K.15.06.5008, February 2012.
[HP-2013]	Hewlett Packard, HP 5920 & 5900 Switch Series OpenFlow Configuration Guide, Part number: 5998-4680, Software version: ESS2305, Document version: 5W100-20130830, 2013.
[HP-2013a]	Hewlett Packard, HP 5920 & 5900 Switch Series Fundamentals Configuration Guide, Part number: 5998-2891, Software version: ESS2305, Document version: 5W100-20130830, 2013.
[HP-2013b]	Hewlett Packard, HP 5920 & 5900 Switch Series ACL and QoS Configuration Guide, Part number: 5998-2897, Software version: ESS2305, Document version: 5W100-20130830, 2013.
[HP-2013c]	Hewlett Packard, HP Switch Software OpenFlow Administrator's Guide K/KA/WB 15.14 (for HP Switch 3500 series), HP Part Number: 5998-4400, Published: October 2013, Edition: 2.
[HP-2014]	Hewlett Packard, HP 5920 & 5900 Switch Series OpenFlow Command Reference, Part number: 5998-4679, Software version: Release 2307, Document version: 6W100 20140108, 2013.
[HPAPPs]	Get Started Developing SDN apps now brochure. http://h17007.www1.hp.com/docs/sdn/HP_SDN-SDK_brochure-1767-4234.pdf
[I2AL]	http://www.internet2.edu/products-services/advanced-networking/layer-2-services/#service-features
[I2VS]	https://www.sdncentral.com/news/internet2-virtualizes-network-backbone/2014/10/
[ICmyNet.Flow]	ICmyNet.Flow http://www.icmynet.com/products/icmynet.flow/overview.html
[Internet2]	http://www.internet2.edu/about-us/
[IPFIX]	IETF IPFIX WG Documents, http://datatracker.ietf.org/wg/ipfix/
[IPFIX-e]	IP Flow Information Export (IPFIX) Entities, IANA, http://www.iana.org/assignments/ipfix/ipfix.xhtml
[ITU-T]	http://www.itu.int/en/ITU-T/Pages/default.aspx

[JUN-2014]	Juniper Technical Documentation, Show OpenFlow Statistics Flows, January 9, 2014, http://www.juniper.net/techpubs/en_US/junos13.3/topics/reference/command-summary/show-openflow-statistics-flows.html.
[JUN-2014a]	OpenFlow support on devices running Junos OS, Juniper TechLibrary, September 29, 2014, http://www.juniper.net/documentation/en_US/release-independent/junos/topics/reference/general/junos-sdn-openflow-supported-platforms.html
[LINF]	Linux Foundation Collaborative Projects, http://collabprojects.linuxfoundation.org
[MCG-2013]	Shamus McGillicuddy, Microsoft Uses OpenFlow SDN for Network Monitoring and Analysis, April 17, 2013, http://searchsdn.techtarget.com/news/2240181908/Microsoft-uses-OpenFlow-SDN-for-network-monitoring-and-analysis
[MngEng]	ManageEngine http://www.manageengine.com/products/netflow/help/installation/setup-cisco-netflow.html
[NETIDE]	http://www.netide.eu/
[NSF]	Roberts, G., Kudoh, T., Monga, I., Sobieski, J., MacAuley, J. and Guok, C., 2014. Network Service Framework v2.0 (draft). Open Grid Forum.
[NSI-CS]	Roberts, G., Kudoh, T., Monga, I., Sobieski, J., MacAuley, J. and Guok, C., 2014. NSI Connection Service Protocol v2.0 (draft). Open Grid Forum.
[NFV]	http://etsi.org/technologies-clusters/technologies/nfv
[NFv5]	NetFlow Export Datagram Format, Cisco Systems, http://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html
[NFv9]	NetFlow Version 9 Flow-Record Format, Cisco Systems, http://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html
[NF-wiki]	NetFlow, Wikipedia: http://en.wikipedia.org/wiki/NetFlow
[NS-blog]	Openflow: Proactive vs Reactive Flows, NetworkStatic, Brent Salisbury's Blog, http://networkstatic.net/openflow-proactive-vs-reactive-flows/
[NIST]	NIST, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
[ODL]	http://www.opendaylight.org/project/technical-overview
[OF1.0.0]	OpenFlow Switch Specification Version 1.0.0, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf
[OF1.0.0]	OpenFlow Switch Specification Version 1.0.0, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf
[OF1.1.0]	OpenFlow Switch Specification Version 1.1.0 https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf
[OF1.3.0]	https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf
[OF1.4.0]	OpenFlow Switch Specification Version 1.4.0, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf
[OF2.0]	P4-Programming Protocol Independent Packet Processor - http://arxiv.org/pdf/1312.1719v3.pdf

[OFSEC]	OpenFlow security traffic redirection solutions, JRA2T1 internal document, https://intranet.geant.net/JRA2/T1/Documents/Security/Openflow_security_traffic_redirection_solutions_final.docx?Web=1
[ONF]	Open Networking Foundation, http://www.opennetworking.org
[ONF1]	Software-Defined Networking: The New Norm for Networks, ONF White paper. April 2012. Available from https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf
[ONF-WP]	ONF White Paper: "Software-Defined Networking: The New Norm for Networks", https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf
[OPENDOVE]	https://wiki.opendaylight.org/images/5/5a/Open_DOVE_for_OpenDaylight_tech_v2.pdf
[OPENFLOW 1.3]	OpenFlow Specification Version 1.3.0, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf`
[OPENNAAS]	http://opennaas.org/
[**OPENNEBULA**]	http://opennebula.org/
[OVX]	OpenVirteX website, http://ovx.onlab.us
[OWNCLOUD]	http://owncloud.org/
[PAT-2010]	Michael Patterson, Comparing SNMP to NetFlow, http://www.lovemytool.com¬/blog/-2010/06/comparing-snmp-to-netflow-by-michael-patterson.html, June 27, 2010.
[PATHFINDER]	Daniel Guija Alcaraz, "Control and management for Real-Time SDN Quality of Service", i2CAT, UPC BarcelonaTech. Available from http://hdl.handle.net/2099.1/22178
[PLI-2009]	Scalability and Accuracy of Packet Sampling, http://blog.sflow.com/2009/05/scalability-and-accuracy-of-packet.htm, May 18, 2009.
[PLI-2011]	Plixer International, delay and Stability, May 22, 2011, http://blog.sflow.com/2011/05/delay-and-stability.html.
[PLI-2013a]	Plixer International, OpenFlow vs. NetFlow, http://www.plixer.com/blog/netflow/openflow-vs-netflow/, January 9, 2013.
[PLI-2013b]	Plixer International, Rapidly detecting large flows, sFlow vs. NetFlow/IPFIX, http://blog.sflow.com/2013/01/rapidly-detecting-large-flows-sflow-vs.html.
[PLI-2013c]	Plixer International, SDN and Delay, January 7, 2013, http://blog.sflow.com/2013/01/sdn-and-delay.html.
[Plix]	Plixer Blog: IP flow-cache timeout active – Are you using it?, http://www.plixer.com/blog/network-problem-resolution/ip-flow-cache-active-timeout-are-you-using-it/
[POW-2012]	Adam Powers, NetFlow vs. sFlow for Network Monitoring and Security: The Final Say], http://www.plixer.com/blog/netflow/netflow-vs-sflow-for-network-monitoring-and-security-the-final-say/, August 27, 2012.
[REA-2013]	Brook Reams, Campus Network Solution, Best Practice-sFlow Monitoring for Brocade Products, May 20, 2013, http://community.brocade.com/t5/Design-Build/Campus-Network-Solution-Best-Practice-sFlow-Monitoring-for/ta-p/36688.
[REE-2008]	Brad Reese, NetFlow or sFlow: Which is the open standard?, Cisco, http://www.networkworld.com/article/2350352/cisco-subnet/netflow-or-sflow--which-is-the-open-standard-.html, January 10, 2008.

[RFC-3176]	P. Phaal, S. Panchen, N. McKee, InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, September 2001, available from http://www.ietf.org/rfc/rfc3176.txt.
[RFC-3410]	J. Case, R. Mundy, D. Partain, B. Stewart, Introduction and Applicability Statements for Internet Standard Management Framework, https://tools.ietf.org/rfc/rfc3410.txt.
[RFC-5101]	B. Claise, Ed., Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information, January 2008, http://tools.ietf.org/html/rfc5101.
[RFC-5102]	J. Quittek, S. Bryant, B. Claise, P. Aitken, J. Meyer, Information Model for IP Flow Information Export, January 2008.
[ROADM]	ROADMs: http://searchtelecom.techtarget.com/definition/ROADM-reconfigurable-optical-add-drop-multiplexer
[RYU]	Ryu controller http://osrg.github.io/ryu/
[RYU1]	http://www.slideshare.net/yamahata/ryu-sdnframeworkupload
[Ryu-Doc]	Ryu controller documentation http://ryu.readthedocs.org/en/latest/
[SDEC]	MacAuley, J., 2013. NSI-CS Service Decoupling. <https://redmine.ogf.org/dmsf_files/13101?download=>
[SDN]	SDN: Software Defined Networks, Chapter 2. Published by O'Reilly Media. August 2013. Also http://www.opennetworking.org/sdn-resources/sdn-definition
[SDNAPPS]	https://intranet.geant.net/JRA2/T1/Documents/Traffic%20Engineering/SDNapps-final.docx?Web=1
[SDNappex1]	http://searchsdn.techtarget.com/news/2240164400/New-HP-SDN-portfolio-Controller-switches-and-network-apps#
[SDNappex2]	http://searchsdn.techtarget.com/tip/How-SDN-applications-will-change-Layer-4-7-network-services#
[SDNappex3]	https://www.sdncentral.com/news/need-sdn-applications-developed/2013/10/
[SDNDEF1]	http://searchsdn.techtarget.com/definition/SDN-application-software-defined-networking-application
[SDNRG]	Software-Defined Networking Research Group (SDNRG), https://irtf.org/sdnrg
[SEFLOOD]	SE-Floodlight on OpenFlowSec.org http://www.OpenFlowsec.org/
[SFL-2004]	Peter Phaal, Marc Lavine, sFlow Version 5, http://sflow.org/sflow_version_5.txt, July 2004. http://www.sflow.org/about/index.php.
[SIEN]	http://meetings.internet2.edu/media/medialibrary/2014/10/20/20141030-przywecki-sdn-in-ren.pdf
[STv2.0]	OGF. NSI Service Types v2.0 WSDL schema.
[SURVEY]	SDN/NFV and Clouds: https://docs.google.com/forms/d/1mljt1sp4ufTV7AQmTSpolhXgN1ftBnIMDs7Yf3a6TqQ/viewform
[SYNNEFOPYTHOS]	https://services.grid.am/astakos/ui/landing
[TC2013]	TERENA Compendium 2013, http://www.terena.org/publications/files/TERENA-Compendium-2013.pdf
[VERTIGO]	R. Doriguzzi Corin, M. Gerola, R. Riggio, F. De Pellegrini, E. Salvadori, "VeRTIGO: network virtualization and beyond", EWSDN 2012.
[XIFI]	XiFi website, https://fi-xifi.eu/home.html

[XNETMON]	Fernandes N.C., Duarte O.C.M.B. XNetMon. A Network Monitor for Securing Virtual Networks.[C].Communications (ICC), 2011 IEEE International Conference on 2011IEEE,265-271
[Y3300]	Y.3300 : Framework of software-defined networking, https://www.itu.int/rec/T-REC-Y.3300-201406-I/en

Glossary

AA	Authentication and Authorisation
aCSP	academic Cloud Service Provider
ACL	Access Control List
ADN	Application Defined Networking
AG	Aggregator
AKLP	Adapted k-longest path
AKSP	Adapted k-shortest path
ALP	Adapted Longest Path
API	Application Programming Interface
ARP	Address Resolution Protocol
B2B	Business to Business
BCC	Building Cloud Competencies
BFD	Bidirectional Forwarding Detection
BoD	Bandwidth on Demand
BW	BandWidth
capex	capital expenditure
CCM	Continuity Check Messages
CEF	Cisco Express Forwarding
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRUD	Create Read Update and Delete
CSP	Cloud Service Provider
DCN	Data Centre Network
DDoS	Distributed Denial of Service
E2E	Exchange to Exchange
EAA	Embedded Automation Architecture
ETSI	European Telecommunications Standards Institute
FDB	Forwarding DataBase
FI-PPP	Future Internet Public Private Partnership
FSPF	Fabric Shortest Path First
FSFW	Flowspace FireWall
FV	FlowVisor
GN3plus	GÉANT Network 3 plus (GN3plus)
GOFF	GÉANT OpenFlow Facility
gOCX	GÉANT Open Cloud Exchange
GTS	Global Task Scheduling
GUI	Graphical User Interface
HD	High Definition
HDD	Hard Disk Drive
HNX	Helix Nebula
HPC	High Performance Computing

HW	HardWare
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
IDS	Inter-Domain Controller
IPS	Intrusion Prevention System
IPv4	Version 4 of the Internet Protocol (StB IETF)
IPv6	Version 6 of the Internet Protocol (StB IETF)
IXP	Internet eXchange Point
IPFIX	Internet Protocol Flow Information Export
IPMI	Intelligent Platform Management Interface
IRTF	Internet Research Task Force
JRA2	Joint Research Activity 2
JVM	Java Virtual Machine
KPI	Key Performance Indicators
LINF	Linux Foundation
MAC	Media Access Control
MPLS	Multi-Protocol Label Switching
MSDP	Multicast Source Discovery Protocol
MTL	Message Transport Layer
MX	Mail eXchanger
NaaS	Network as a Service
NAT	Network Address Translation
NE	Network Element
NETCONF	Network Configuration Protocol
NFV	Network Functions Virtualization
NML	Network Markup Language
NMS	Network Management System
NOC	Network Operations Centre
NQA	Network Quality Analyser
NREN	National Research and Educational Network
NRM	Network Resource Manager
NSA	Network Service Agent
NSF	Network Service Framework
NSI	Network Service Interface
NSI-CS	Network Service Interface-Connection Service
OAM	Operations, Administration and Management
OCF	OFELIA Control Framework
OCX	Optical Cross-Connect
ODL	OpenDaylight
OF	Open Flow
OF2NF	OpenFlow to NetFlow
OGF	Open Grid Forum
ONF	Open Networking Foundation
opex	Operational Expenditure
OSGi	Open Source Gateway Initiative

OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OVS	Open vSwitch
OVX	OpenVirteX
PoC	Proof of Concept
PoP	Point of Presence
P2P	Point to Point
PaaS	Platform as a Service
pCSP	public Cloud Service Provider
QoE	Quality of Experience
QoS	Quality of Service
REN	Research and Educational Network
REST	REpresentational State Transfer
RMON	Remote Network Monitoring
RPC	Remote Procedure Call
SaaS	Software as a Service
SDA	Service Definition Agreement
SDK	Software Development Kit
SDN	Software-Defined Networking
SDNRG	Software-Defined Networking Research Group
SDNtrap	SDN Traffic Redirection Application
SDO	Standard Development Organization
SME	Subject Matter Expert
SoTA	State of the Art
STP	Service Termination Points
JRAnTm	Joint Research Activity n Task m
TCP/UDP	Transmission Control Protocol / User Datagram Protocol
TEBD	Time-Evolving Block Decimation
TTP	Trusted Third-Party Service
UDP	User Datagram Protocol
UI	User Interface
uRA	Ultimate Requester Agent
uPA	Ultimate Provider Agent
URN	Uniform Resource Name
VLAN	Virtual Local Area Network
VM	Virtual Machines
VPS	Virtual Private Server
XML	Extensible Markup Language
XSD	XML Schema Definition