Grant Agreement No. 619572

COSIGN

Combining Optics and SDN In next Generation data centre Networks

Programme: Information and Communication Technologies

Funding scheme: Collaborative Project – Large-Scale Integrating Project

Deliverable D1.3

Comparative analysis of control plane alternatives

Due date of deliverable: December 2014
Actual submission date: January 16, 2015

Start date of project: January 1, 2014 Duration: 36 months

Lead contractor for this deliverable:
NXW, Giada Landi

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Executive Summary

This deliverable presents some potential solutions for SDN based control planes to be adopted in COSIGN Data Centre networks. Starting from the outcomes reported in deliverable D1.1, each control plane requirement is further analyzed and mapped to a specific layer of a typical SDN architecture, identifying the associated technical functionalities and a set of Key Performance Indicators for the evaluation of the COSIGN solution. As a second step, the SDN architectures and protocols defined by the main standardization bodies currently active in this area are reviewed to identify the most suitable concepts and approaches to be applied in COSIGN environments and, where needed, the gaps to meet the most challenging requirements. Finally, the main open source SDN controller platforms are analyzed and compared from an architectural and functional point of view, in order to provide an input for the selection of the COSIGN reference SDN controller. This selection will be performed in work package 3, where the architectural considerations will be evaluated together with the software aspects. A specific section is dedicated to the network virtualization techniques. In fact this aspect constitutes one of the main functionalities of a network control plane and orchestration platform for Data Centre environments and a key area where COSIGN solutions will bring innovation. For these reasons, this analysis covers not only open source tools and applications, but also industrial products from a variety of vendors, with the objective to understand the current market trends and the main gaps for driving our research effort and architecture design.

Document Information

Status and Version:	V1.1	
Date of Issue:	December 28th, 2014	
Dissemination level:		
Author(s):	**Name**	**Partner**
	Giada Landi	NXW
	Giacomo Bernini	NXW
	Nicola Ciulli	NXW
	Gino Carrozzo	NXW
	Bingli Guo	UNIVBRIS
	George Saridis	UNIVBRIS
	Shuping Peng	UNIVBRIS
	Reza Nejabati	UNIVBRIS
	Georgios Zervas	UNIVBRIS
	Dimitra Simeonidou	UNIVBRIS
	Katherine Barabash	IBM
	Anna Levin	IBM
	Yaniv Ben-Itzhak	IBM
	Sergey Guenender	IBM
	Fernando Agraz	UPC
	Salvatore Spadaro	UPC
	Albert Pagès	UPC
	José Ignacio Aznar	i2CAT
	Amaia Legarrea	i2CAT
	Eduard Escalona	i2CAT
	Josep Batallé	i2CAT
	Andrea Cervera	i2CAT
	Michael Galili	DTU
	Valerija Kamchevska	DTU
	Cosmin Caba	DTU
	Michael Berger	DTU
	Sarah Ruepp	DTU
	Alessandro Predieri	IRT
	Matteo Biancani	IRT
Edited by:	Giada Landi	NXW
Checked by :	Sarah Ruepp	DTU

Table of Contents

1 Introduction

This deliverable presents and evaluates some potential solutions for Software Defined Networking (SDN) based control planes to be adopted in COSIGN Data Centre Networks (DCNs), providing some initial considerations about the COSIGN network Control Plane (CP) and Data Centre (DC) orchestration platform as input for work package 3 and 4 activities.

This work starts from the set of control plane requirements defined in deliverable D1.1. Each requirement is mapped to a specific layer of a typical SDN architecture, i.e. the core functions or the interfaces of the SDN controllers, the drivers to interact with the data plane or the applications running on top of the controllers. The analysis identifies the technical functionalities to be supported at these layers to meet the associated requirements and the main innovations that would enable to differentiate the COSIGN approach from the current DCN platforms. Moreover, a set of Key Performance Indicators (KPIs) is defined for the evaluation of the COSIGN solution.

As a second step, this deliverable reports a summary of the SDN architectures and the operational or management protocols defined by the main standardization bodies currently active in this area. The objective is to identify the most suitable concepts and approaches to be applied in the optical DCN environments and, where needed, the gaps to meet the most challenging COSIGN requirements.

Finally, the main open source SDN controller platforms are analyzed and compared from an architectural and functional point of view, e.g. in terms of supported southbound protocols, flexibility to extend the internal core functions or the external interfaces and interaction with the main cloud orchestration platform. This analysis will provide an input for the selection of the COSIGN reference SDN controller that will be performed in work package 3. Here, the architectural considerations resulting from this deliverable will be mixed with the evaluation of the software aspects, e.g. in terms of maturity of the code, potential of the open-source community and software design for controller platform and SDN applications.

An entire section is dedicated to the network virtualization techniques. In fact this aspect constitutes one of the main functionalities of a network control plane and orchestration platform for Data Centre environments and a key area where COSIGN solutions will bring innovation. For these reasons, this analysis covers not only open source tools and applications, but also industrial products from a variety of vendors, with the objective to understand the current market trends and the main gaps for driving our research effort and architecture design.

1.1 Reference Material

1.1.1 Reference Documents

[1]	COSING FP7 Collaborative Project Grant Agreement Annex I - "Description of Work"

1.1.2 Acronyms and Abbreviations

Most frequently used acronyms in the Deliverable are listed below. Additional acronyms can be specified and used throughout the text.

ACI	Application Centric Infrastructure
A-CPI	Application to Controller Plane Interface
API	Application Programming Interface
APIC	Application Policy Infrastructure Controller
AVS	Application Virtual Switch
CP	Control Plane
DC	Data Centre
DCN	Data Centre Network
D-CPI	Data Plane to Controller Plane Interface
DPCF	Data Plane Control Function

DHCP	Dynamic Host Configuration Protocol
EPG	EndPoint Group
EPR	EndPoint Registry
ETSI	European Telecommunications Standards Institute
FRM	Flow Resource Manager
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
KVM	Kernel-based Virtual Machine
LACP	Link Aggregation Configuration Protocol
LAG	Link Aggregation Groups
LAN	Local Area Network
NDM	Negotiable Datapath Model
NE	Network Element
NVGRE	Network Virtualization using Generic Routing Encapsulation
OF	Open Flow
ONF	Open Networking Foundation
OSS	Open Source Software
OVS	Open Virtual Switch
OVSDB	Open vSwitch Database Management Protocol
PE	Policy Element
PR	Policy Repository
RDB	Resource Data Base
REST	Representational state transfer
RFC	Request For Comments
RSTP	Rapid Spanning Tree Protocol
SDN	Software Defined Networking
SFC	Service Function Chain
SOAP	Simple Object Access Protocol
SSH	Secure SHell
STP	Spanning Tree Protocol
TLS	Transport Layer Security
VE	Virtual Environments
VLAN	Virtual LAN
VM	Virtual Machine
VNF	Virtual Network Function
VTN	Virtual Tenant Network
VXLAN	Virtual Extensible LAN

1.2 Document History

Version	Date	Authors	Comment
00		See the list of authors	ToC
01			First draft
02			Second draft
06	17 Dec. 2014		Pre-final version

2 COSIGN control plane requirements and KPIs

Deliverable D1.1 [COSIGN-D11] presented an initial set of control plane requirements for COSIGN DCNs, defined starting from the use-cases identified as main driver scenarios. In order to achieve a solid and consistent definition of COSIGN control plane architecture, those requirements need to be taken into account with particular attention to the novel functionalities and features that may impact the different layers and components of the SDN-based DCN. For this reason, this section reports the list of requirements from D1.1 where, for each of them, identifies the functions to be supported at the associated SDN platform layer (i.e. the core functions or the interfaces of the SDN controllers, the drivers to interact with the data plane or the applications running on top of the controllers) and the main innovations that would be able to differentiate the COSIGN approach from the current DCN platforms. A list of Key Performance Indicators (KPIs) is also provided, as driver for the evaluation of the COSIGN solution. For each requirement, impact and novelty have been evaluated in a range of three possible values ("low", "medium" and "high"). As summarized in the graphs of Figure 2-1, about the 50% of the requirements can be classified in the medium and high categories for both the fields.

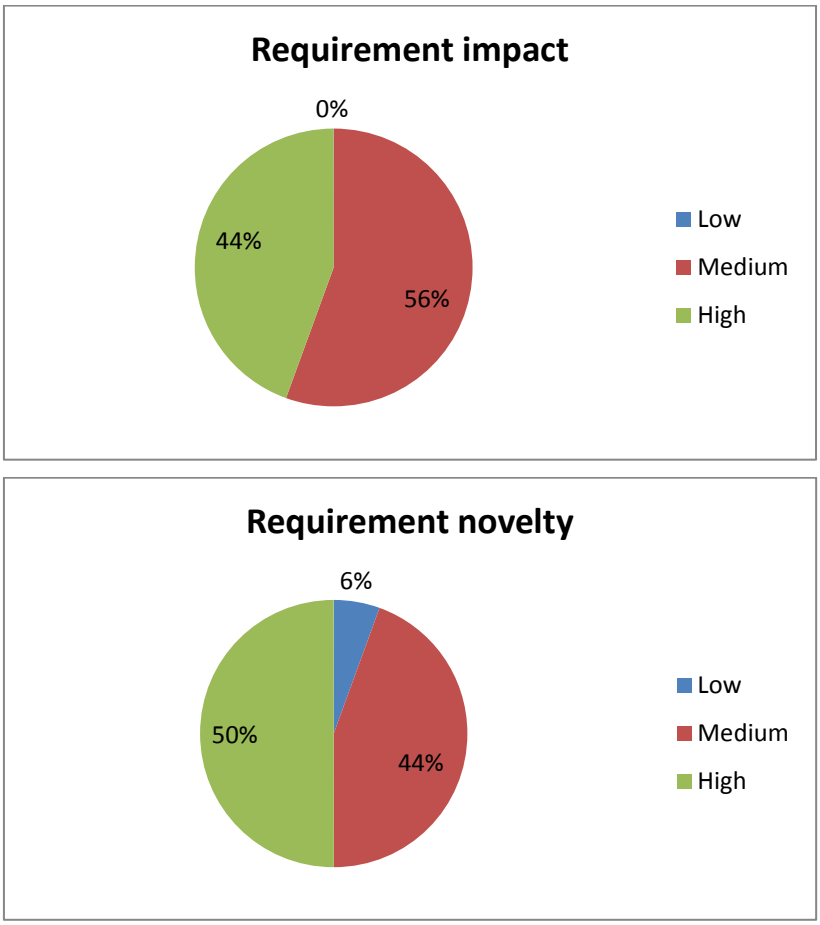


Figure 2-1 : Impact and novelty of COSIGN requirements.

R-CP-01 Description	**Automated provisioning of intra-DC connectivity** The COSIGN CP must provide mechanisms for an automated DCN configuration to establish and destroy intra-DC on-demand connectivity services (triggered by external requests) or in support of internal procedures (e.g. re-optimization of DCN resource allocation, connectivity restoration, etc.).
KPIs	• Connection setup time. • Connection tear-down time.
Impact	Medium. The granularity of connection provisioning must be handled considering the technologies available at the data plane.
Novelty	Medium. The novelty relies in the SDN approach applied to the DCN, where the provisioning of the DCN resources is controlled through the SDN controller(s) and in the automation of the provisioning.
Associated Functionalities	Connection provisioning service.
SDN controller Layer	SDN controller: - Network application. - Application to Controller Plane Interface (A-CPI).

Table 1 - R-CP-01 Automated provisioning of intra-DC connectivity

R-CP-02 Description	**Support for customizable network services** The COSIGN CP must provide mechanisms to provide intra-DC connectivity services compliant with a variety of user-level constraints, including QoS parameters (e.g. bandwidth), timing constraints and service resilience guarantees.
KPIs	Supported service constraints.
Impact	High. Requires a service interface at the A-CPI powerful enough to specify the desired constraints and control plane mechanisms to support them at the provisioning level (e.g. mechanisms for QoS configuration, network service scheduling, recovery, etc.).
Novelty	Medium. Current architectures are able to provide pre-defined and basic network services.
Associated Functionalities	Connection provisioning service; • Additional services as required by the specific constraints (scheduling, recovery, etc.); • Correlated services like extended inventory services or computation algorithms dedicated to QoS-enabled connections, able to operated on scheduled resources or to compute disjoint routes, etc.
SDN controller Layer	SDN controller: mainly network applications and A-CPI, but also core services.

Table 2 - R-CP-02 Support for customizable network services

R-CP-03 Description	**Dynamic Support for multiple automated connectivity paradigms** In order to support a variety of virtual infrastructure topologies, the COSIGN CP must provide mechanisms to establish point-to-point, point-to-multipoint or anycast connectivity, in unidirectional or bi-directional, symmetric or asymmetric mode.
KPIs	• Connection setup time for P2P, P2MP and anycast connections • Efficiency of network resource utilization
Impact	Medium. Requires a service interface at the A-CPI able to specify multiple connectivity models and control plane mechanisms to support them at the provisioning level.
Novelty	High. Current architectures provide pre-defined and basic network services.
Associated Functionalities	Connection provisioning service + correlated services (computation algorithms for P2MP or anycast services).
SDN controller Layer	SDN controller: - Network applications. - A-CPI.

Table 3 - R-CP-03 Dynamic Support for multiple automated connectivity paradigms

R-CP-04 Description	**Multi-layer operation of COSIGN data plane optical technologies** The COSIGN CP must be able to efficiently operate the heterogeneous optical technologies adopted at the COSIGN DCN data plane. This means it must be able to handle their different constraints (even through resource virtualization), as well as the matching among the resource granularities offered for each specific technology. Multi-layer mechanisms must be adopted to coordinate the cross-technology resource allocation and maximize the efficiency for the whole DCN utilization, still in compliance with the requirements of the running services. Where possible, the usage of open, standard protocols for the interaction with the underlying physical technologies is preferred, adopting dedicated extensions if required.
KPIs	• Granularity and dynamicity of end-to-end connectivity services. • Efficiency of resource allocation in multi-layer environments (e.g. through the sharing of a connection established at a server-layer among multiple connections at a client-layer with higher granularity).
Impact	Medium. Requires additional internal functionalities within the SDN controller in order to manage the provisioning and re-adaptation of the multi-layer connectivity. At the interface level, it requires the common extensions to manage optical resources at the Data Plane to Controller Plane Interface (D-CPI).
Novelty	High. Current architectures usually provide static network services, without

	exploiting the multi-layer capabilities of the transport technologies.
Associated Functionalities	Optical resource virtualization + multi-layer connection service provisioning + dynamic re-allocation of the resources providing the connectivity at the server-layer.
SDN controller Layer	SDN controller: - South-bound plugins and core services (for resource abstraction). - Resource virtualization service. - Network application for multi-layer service provisioning and dynamic re-allocation. Agents to interact with the optical devices.

Table 4 - R-CP-04 Multi-layer operation of COSIGN data plane optical technologies

R-CP-05 Description	**DCN resource usage optimization** The COSIGN CP must provide mechanisms to provide intra-DC connectivity services compliant with a variety of operator-level constraints, including load-balancing strategies, energy efficiency, paths with minimum cost, etc.
KPIs	• Number of supported objective functions for resource optimization. • Time required by the computation algorithms to select the most efficient set of resources with the given constraints.
Impact	Medium. It requires the implementation of suitable optimization algorithms within the SDN controller. The parameters to be used as input for these algorithms needs to be notified at the D-CPI level when related to the transport technologies (e.g. power consumption parameters) or configured through management interfaces when related to operator policies (e.g. "cost" of a link between two network nodes).
Novelty	High. The optimization of the resource usage allows the operator to increase the overall utilization of its Data Centre (e.g. to accommodate more services or users) while reducing the operating costs.
Associated Functionalities	Computation algorithms and policy management within the SDN controller.
SDN controller Layer	SDN controller: - network application for resource selection. - network application for policy management. - A-CPI (for policy configuration). - D-CPI (support of resource parameters to be used as input for resource selection algorithms).

Table 5 - R-CP-05 DCN Resource usage optimization

R-CP-06 Description	**Elastic intra-DC connectivity** The COSIGN CP must be able to dynamically scale-up and scale-down the capacity of the established connections, in support of the elastic features of the cloud services. The decisions about connectivity upgrade/downgrade may be taken internally at the CP layer or coordinated by upper-layer entities (e.g. at the orchestration level), but always in compliance with the established SLAs.
KPIs	• Time required to modify an existing connection. • Time required to detect an inefficient condition in the resource allocation for a given service (at the CP layer) and initiate a modification procedure.
Impact	Medium. It requires the support of modification requests at the A-CPI level and internal function within the SDN controller to support the modification of existing connections.
Novelty	High. This feature allows for applying the same elasticity concepts commonly used for the IT resources in a cloud environments to the network domain, with great benefits in terms of service dynamicity and automatic adaptation to the user requirements.
Associated Functionalities	Modification of connection services on-demand or based on internal triggers.
SDN controller Layer	SDN controller: - A-CPI. - Network applications.

Table 6 - R-CP-06 DCN Elastic intra-DC connectivity

R-CP-07 Description	**Network monitoring** The COSIGN CP must be able to provide monitoring functionalities for network resource usage, network service performance and faults detections.
KPIs	• Time required to detect failures. • Statistics on the DCN resources usage and performance.
Impact	Medium. It requires the support of monitoring information at the D-CPI level. It requires the support of monitoring information at the management interface level.
Novelty	Medium. This feature allows to promptly react to any failure that could happen in the DCN as well as it allows to apply policies (e.g. load balancing) to better adapt to the user requirements while maximizing the utilization of the network resources. The main novelty relies on the usage of the SDN approach.
Associated Functionalities	Monitoring services and statistics on the DCN resources usage.
SDN controller Layer	SDN controller: - D-CPI.

	- Management.

Table 7 - R-CP-07 Network monitoring

R-CP-08 Description	**Programmable APIs** The main configuration options and functions (e.g. service provisioning and tear-down, modification, queries for monitoring data) offered by the COSIGN CP must be exported through programmable APIs (e.g. based on the REST paradigms), with different levels of capabilities depending on authorization profiles. These APIs should allow exposing some (limited) functionalities directly to the users and enable an easy integration with the overall DC control and management platform.
KPIs	Set of configuration options and functions exported supported by the programmable APIs.
Impact	High. It requires the support of: - programmable APIs for users to perform provisioning and modification actions; - requests triggered by the users at the A-CPI level.
Novelty	High. Current architectures do not allow exporting network statistics and information to users.
Associated Functionalities	Exposition of DCN performance to end users.
SDN controller Layer	SDN controller: - A-CPI. - Network applications.

Table 8 - R-CP-08 Programmable APIs

R-CP-09 Description	**Support of network service monitoring and accounting** The COSIGN CP must produce and expose through a suitable management interface a set of monitoring information about DCN resource usage and allocation, as required to support cloud service accounting for various pricing models (e.g. pay-as-you-go or commit model).
KPIs	Co-existence of network services with different pricing models
Impact	Medium. It requires the support of management interfaces between SDN controller and OSS to gather DCN resources and allocation and usage statistics.
Novelty	Medium. This feature allows for gathering information about the network usage to support the various pricing models.
Associated Functionalities	Information about the usage and performance of DCN resources
SDN controller Layer	Orchestration functions and Operation Support Systems (OSS) SDN controller: D-CPI

Table 9 - R-CP-09 Support of network service monitoring and accounting

R-CP-10 Description	**Support of scheduled network connectivity** The COSIGN CP should be able to support scheduled or periodical connectivity services, between intra-DC resources or in support of connectivity among resources located in different DCs. Suitable synchronization procedures, in cooperation with upper layer entities (e.g. at the orchestration level), must be provided to coordinate the enforcement of the overall scheduled actions across the different DC resources. However, in-advance resource reservations at the network level (i.e. without actual configuration) should be planned and automatically updated to guarantee resource availability and optimize the global DCN utilization among the shared physical resources.
KPIs	Time required to automatically provide scheduled connectivity services.
Impact	High. It requires the support of automated configuration of the DCN resources for the scheduled events requiring scheduled connectivity services. It requires the support of cooperation through the A-CPI interface between the SDN controller and the orchestration level.
Novelty	High. This feature allows for achieving a global optimization of the shared DCN resources to provide both scheduled and on-demand connectivity services.
Associated Functionalities	Automated scheduled connection services.
SDN controller Layer	SDN controller: A-CPI. Orchestration level.

Table 10 - R-CP-10 Support of scheduled network connectivity

R-CP-11 Description	**Network service resilience** The COSIGN CP must be able to detect network service failures and, where possible, react in an automated manner through fast connection recovery procedures. Depending on the established SLAs and the original service requests, protection or restoration mechanisms can be applied. When network service restoration procedures are not applicable, asynchronous failure alerts must be produced and notified to the upper layer (e.g. to the orchestration level) to enable recovery escalation strategies. Simple restoration mechanisms can probably be implemented directly in the physical HW (or close to it). Restoration based on SLA is beyond the scope of data plane alone.
KPIs	• Time to automatically recover from a failure is support of the established SLA and original service characteristics. • Time to detect and notify to the orchestration level the failure occurrence.
Impact	Medium. It requires the support of (fast) recovery actions at both the A-CPI and D-CPI level.
Novelty	Medium. The novelty relies on the usage of the SDN approach and the overall orchestration of the DCN resources.

Associated Functionalities	Detection and notification through the D-CPI and A-CPI of failure occurrence.
SDN controller Layer	SDN controller: - D-CPI. - A-CPI. - Network applications.

Table 11 - R-CP-11 Network service resilience

R-CP-12 Description	**Multi-tenant isolation** The COSIGN CP must enforce suitable virtualization mechanisms to enable the sharing of a common physical network infrastructure among fully isolated connectivity services.
KPIs	• Given the DCN resources, the number of fully isolated virtual slices (tenants). • Usage and performance statistics of the DCN resources for each tenant (virtual slice).
Impact	High. It requires the support of virtualization techniques for the efficient allocation of the shared DCN resources.
Novelty	High. This feature allows maximizing the efficient allocation of the shared DCN resources.
Associated Functionalities	Virtualization of DCN resources isolated virtual slices allocation.
SDN controller Layer	SDN controller: - A-CPI. - Network applications.

Table 12 - R-CP-12 Multi-tenant isolation

R-CP-13 Description	**Support of SLA enforcement, runtime monitoring and assessment** The COSIGN CP must be able to support an SLA-driven provisioning and runtime management of intra-DC cloud network services. Moreover, it should provide automated mechanisms to monitor and evaluate the performance of the running services according to the metrics and KPIs specified in the enforced SLA and, when possible and required, automatically react to predicted or detected SLA breaches. Synchronization and cooperation procedures with the upper layer (e.g. the orchestration level) for SLA management should also be provided.
KPIs	• SLAs compliance • User service satisfaction pool
Impact	High. SLAs compliance is key to ensure users' satisfaction. SDN technology enables monitoring mechanisms based on performance metrics retrieved from the underlying resources. The COSIGN architecture may leverage such SDN monitoring to ensure that agreed SLAs are being achieved.

Novelty	Low. SLAs compliance and enforcement is a well-known topic. Adding SDN monitoring capabilities to monitor and evaluate the performance of the running services may add consistency to SLAs achievement, but it is not novel at all.
Associated Functionalities	• CP monitoring module • SLA management module at CP layer
SDN controller Layer	Infrastructure Level: Management and Orchestration layers (consumers of the retrieved information). Infrastructure Level: SDN control. Infrastructure Level: Physical layer (provides with SDN monitoring metric measures).

Table 13 - R-CP-13 Support of SLA enforcement, runtime monitoring and assessment

R-CP-14 Description	**Easy interoperability with existing cloud management platforms for unified DC control.** The COSIGN CP should provide powerful APIs, possibly based on the REST concept and HTTP protocol, to enable an easy integration with existing orchestration systems and cloud management platforms adopted to handle the overall DC management. Where relevant, the usage of open, standard interfaces should be preferred to guarantee an easy interoperability.
KPIs	Joint cloud and network management provisioning tool available.
Impact	Medium. Cloud and Network management services are separated items. Usually IT guys are not experts on Network service provisioning too. Therefore, it would be interesting to create a tool which enables to centralize both management systems
Novelty	High. With the new advancement of cloud and network management tools, the possibility to achieve such interoperability by exposing cloud management platforms APIs has arisen.
Associated Functionalities	Joint cloud and Network management services tool.
SDN controller Layer	Infrastructure Level: Management and Orchestration layers Infrastructure Level: SDN control Layer.

Table 14 - R-CP-14 Easy interoperability with existing cloud management platforms for unified DC control

R-CP-15 Description	**Integration with external connectivity services (inter-DC)** The COSIGN CP must be able to configure the intra-DC network to efficiently support also the traffic generated among computing resources located in different data centres. This traffic could have various characteristics and requirements, since it could belong to running cloud applications distributed across different sites or it could be generated by management procedures, e.g. for inter-DC content replication. Mechanisms and interfaces for integrated management of intra-DC and inter-DC connectivity should be supported, with reference to different deployment and business models (e.g. inter-DC connectivity offered by an

	external provider or by the same administrative entity that manages the DC infrastructure). Open standard interfaces and protocols should be preferred where applicable to enable and simplify the (multi-domain) interoperability.
KPIs	• E2E network service orchestration tool available. • E2E inter-domain Network service delivery with guaranteed QoS.
Impact	High. This feature may enable to control the intra-DC network connectivity and provisioning service but also to coordinate with other connectivity services managed by other NOCs, and agree on specific SLAs to guarantee the support of multi-domain services.
Novelty	Medium. Not directly, but Operator agreements and SLAs are carried out to support the traffic from each other's clients. In this case, it is not proposed to only propose to support network connectivity, but preforming such support according to the SLAs of the DC network services.
Associated Functionalities	
SDN controller Layer	Infrastructure Level: Management and Orchestration layers SDN controller: A-CPI or East-West interfaces

Table 15 - R-CP-15 Integration with external connectivity services (inter-DC)

R-CP-16 Description	**Dynamic DCN reconfiguration for optimization strategies** The COSIGN CP should support the automated re-planning of already established network services. The automated re-planning will bring the possibility to autonomously re-adapt the resource allocation to the dynamicity of the real-time DC loads, according to global re-optimization criteria. However, DCN re-configuration procedures must avoid any disruption of the existing services.
KPIs	• Reduction of the total number of SLAs violations indicator. • Enhancement of the QoS monitored metrics. • Queue congestion reduction at network resources. • Better DC network services overall performance.
Impact	Medium. Network resources and services re-planning usually impacts network services performance while dealing with the changes. DC networks resources are commonly over-provisioned, so that re-planning strategies re not very popular among DC network managers, especially to avoid services disruption.
Novelty	Medium. Network services re-planning is already there and usually involves Load balancing techniques or overprovisioning of network resources.
Associated Functionalities	• Automated service re-planning tool. • Automated re-optimization engine based on different (strategic) criteria.
SDN controller Layer	SDN controller: D-CPI and core functions. Infrastructure level: Management

Table 16 - R-CP-16 Dynamic DCN reconfiguration for optimization strategies

R-CP-17 Description	**CP architecture in support of scalable DCNs and network traffic** The COSIGN CP must be designed to efficiently operate DCNs of different sizes, scaling well with an increasing number of servers and network devices and an increasing amount of intra-DC and inter-DC cloud application and management traffic with flows of different dimension. Scalability is a major concern in the data plane considerations. However, this requirement is here more connected to applications and management traffic. It is more about management than about ports and links. Scalability can depend on the amount of tenants, deployed applications, users, etc.
KPIs	• Capacity plan available • Total number of SLAs violations indicator. • Resources utilization: Percentage of network resources in use.
Impact	High. Properly scaling network service in DCs is fundamental to ensure basic DC services operation in both intra and inter DC environments. It is as simple as if the network do not scale, DC services cannot be provided within the SLA agreed levels, and consequently, "you are out of business"
Novelty	High. Traditionally, the scalability of network services relies on the HW layer. A proper monitoring of network resources together with the intelligence applied at SDN control and management layers may also enable to enhance the scalability of network services.
Associated Functionalities	• Scalability and performance tool at management or application layer.
SDN controller Layer	SDN controller: D-CPI. Infrastructure level: Management

Table 17 - R-CP-17 CP architecture in support of scalable DCNs and network traffic

R-CP-18 Description	**Scalability** The scalability of the CP is a key requirement for the proper operation of the data centre. Therefore, the size (in terms of servers and optical devices to be managed) as well as the expected huge number of traffic flows among servers should not affect the properly working of the CP operation.
KPIs	• Capacity plan available • Total number of SLAs violations indicator. • Resources utilization: Percentage of network resources in use.
Impact	High. Properly scaling network service in DCs is fundamental to ensure basic DC services operation in both intra and inter DC environments. It is as simple as if the network do not scale, DC services cannot be provided within the SLA agreed levels, and consequently, "you are out of business"
Novelty	Medium. Network service scalability is a MUST from the very beginning of a DC design. Nevertheless, novel virtualization techniques and the improvement of CP mechanisms enable with new approaches and reduce data center complexity through fewer physical connections. Through virtualization, you get better resource utilization which at the end turns into a better scalability solution.
Associated Functionalities	• Per network resource optimizer functionality (leaning on virtualization). • Per Network optimizer functionality (leaning on virtualization)

SDN controller Layer	SDN controller: D-CPI and core functions. Infrastructure level: Management Infrastructure level: HW layer

Table 18 - R-CP-18 Scalability

3 Software Defined Networking

3.1 Architectures

During the last years, many initiatives have been started with the aim to design, promote and standardise software –based programmable architectural solutions for efficient network management and control, in order to support the dynamic requirements of the traffic to be supported and the deployed services. To this purpose, Software Defined Networking (SDN) and Network Functions Virtualisation (NFV) have been identified as two complementary technologies that are being developed by both the IT and the telecom industries.

In particular, on one hand, Open Networking Foundation (ONF) [ONF] is dedicated to the promotion and adoption of SDN through open standards development. SDN decouples the network control and forwarding functions, enabling network control programmability and abstraction of underlying infrastructure. OpenFlow (OF) protocol is a foundational element for building SDN solution.

On the other, the industry-led standards development organization European Telecommunications Standards Institute (ETSI) is dedicated to define the requirements and architecture for the virtualisation of network functions [ETSI]. NFV offers the potential for enhancing service delivery and reducing overall infrastructure costs.

A strategic partnership between ONF and ETSI has been agreed, with the aim to further the development of Network Functions Virtualization (NFV) specifications. In particular, SDN can accelerate NFV deployment by offering a scalable and programmable architecture well suited to the dynamic NFV communications requirements.

In the following sections, both the SDN and NFV architectures are deeply discussed, highlighting the scope of the COSIGN project within the overall architectures.

3.1.1 ONF SDN architecture

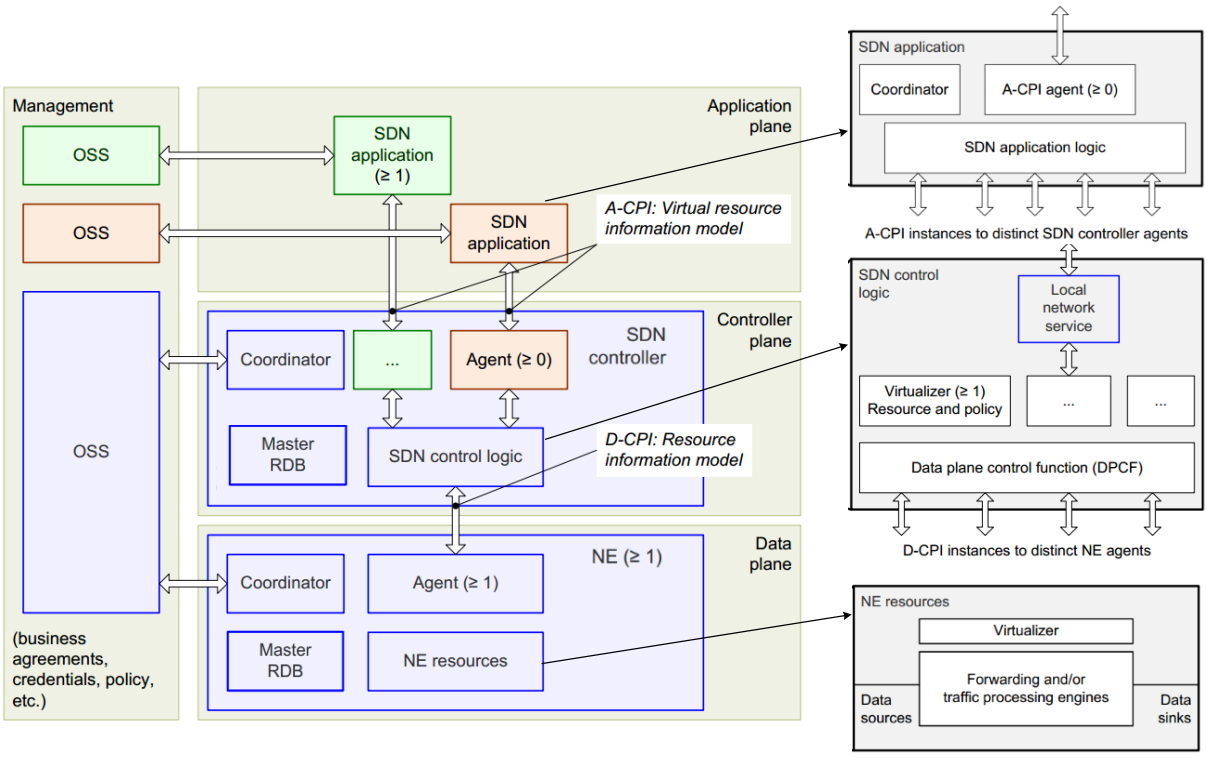


Figure 3-1 : SDN Architecture overview [SDN-ONF].

Figure 3.1 introduces the SDN components. The initial view comprises data, controller, and application planes, with transversal management function. The data plane Network Elements (NEs)

expose their capabilities toward the controller plane via southbound (from the controller perspective) interfaces, called Data Plane to Controller Plane Interfaces (D-CPI). The SDN applications are part of the application plane. They forward their network requirements toward the controller plane via northbound interfaces, called Application to Controller Plane Interfaces (A-CPI). In the middle, the SDN controller matches the applications' requirements to exert low-level control over the network elements, while providing relevant information up to the SDN applications. All the data, controller and application planes need communicate with the management plane. Specifically, in the data plane, management is required for the initial set up of the network elements, assigning the data plane resources to their associated SDN controller. In the control plane, management is required to configure the policies which define the scope of the control given to the SDN applications and monitor the performance of the system. In the application plane, management typically configures the contracts and Service Level Agreements (SLAs). Security is transversal to all planes, so that management configures the security associations that allow distributed functions to safely intercommunicate. Figure 3-1 shows distinct application, controller and data planes, with controller plane interfaces (CPIs) designated as reference points between the SDN controller and the application plane (A-CPI) and between the SDN controller and the data plane (D-CPI). The information exchanged across these interfaces should be modelled as an instance of a protocol-neutral information model.

Data Plane

The data plane incorporates the resources that deal directly with customer traffic, along with the necessary supporting resources to ensure proper virtualization, connectivity, security, availability, and quality features. Figure 3-1 expands the NE resources view. The NE resources block comprises data sources, data sinks and forwarding and/or traffic processing engines, as well as a virtualizer whose function is to abstract the resources to the SDN controller and enforce policy. The picture shows also a master Resource Data Base (RDB), the conceptual repository of all resource information associated to the network element. The data plane agent executes the SDN controller's instructions in the data plane. The data plane coordinator is used by the management to allocate data plane resources to various client agents and establish policies to govern their use. Agents and coordinators serve the same purpose in every plane of the architecture.

The interface between data and controller planes (D-CPI) includes functions such as:

- Programmatic control of all functions exposed by the RDB
- Capabilities advertisement
- Event notification

Control Plane

The SDN architecture does not specify the internal design or implementation of an SDN controller, and considers it as a black box. It is nevertheless useful to conceptualize a minimum set of functional components within the SDN controller, namely Data Plane Control Function (DPCF), coordinator, virtualizer, and agent. Subject to the logical centralization requirement, an SDN controller may include additional functions. A Resource Data Base (RDB) models the current information model instance and the necessary supporting capabilities. Functions and services that are part of a controller's externally-observable behaviour include full visibility of the information model instance under its control. Additional functions that may be required include:

- Topology awareness and Path Computation Element (PCE) (alternatively, the controller may also invoke an external service for these functions).
- The creation and maintenance of a further abstracted resource model for SDN applications, with resources bounded by enforced policy. Resource virtualization and control are potentially recursive.

The DPCF component effectively owns the subordinate resources and manipulates them under the instructions of the OSS/coordinator or virtualizer(s) that controls them. These resources take the form of an information model instance accessed through the agent of the underlying level. Since the scope of an SDN controller is expected to span multiple (virtual) NEs or even multiple virtual networks (with a distinct D-CPI instance to each), the DPCF must include a function that operates on the

aggregate. This function is commonly called orchestration. This architecture does not specify orchestration as a distinct functional component.

Also, the controller plane may comprise a set of SDN controllers, each of which has exclusive control over a set of resources exposed by one or more network elements in the data plane (its span of control). These controllers exit as peers to each other, thus multiple controller coordination strategy is also required.

A SDN controller offers services to applications by means of an information model instance. This is derived from the underlying resources, according to the policies installed through management. These policies define the level of details that is exposed to the local or external support functions. The functional entity that supports the information model instance and policy at an A-CPI is the so-called virtualizer. It presents the local trust domain boundary to the corresponding agent, which represents the client's view of the information model instance.

Application Plane

The application plane comprises one or more applications, each of which has exclusive control on a set of resources exposed by one or more SDN controllers. Additional interfaces to applications are not precluded. An application may invoke or collaborate with other applications. An application running on an SDN controller may also act as another, upper layer, SDN controller in its own right.

SDN principles permit applications to specify the resources and behaviour they require from the network, within the context of a business and policy agreement. The north-bound interface from the SDN controller to the application plane is called Application-Controller Plane Interface, A-CPI. As shown in Figure 3-1, an SDN application may itself support an A-CPI agent. This allows for recursive application hierarchies, where different levels of an application hierarchy have different degrees of abstraction.

An SDN application may invoke other external services, and may orchestrate any number of SDN controllers to achieve its objectives. The OSS link and the coordinator function recognize that, like the other major blocks of the architecture, SDN applications require at least a certain amount of a priori knowledge of their environments and roles.

Activity across the A-CPI typically includes queries or notifications about the state of the virtual network, and commands to modify and rule its state; for example to create or modify network connectivity or traffic processing functions between network client layer (data plane) handoff points, with some specified bandwidth and QoS. The A-CPI may also be used for additional functions such as to be an access point, to configure a service chain through one or more layer 4-7 services, or as an input to control virtualized network functions.

Management

Applications, SDN controller and Network Element have their own functional interface to a manager entity in the management plane. The minimum functionality of the manager is to allocate resources from a resource pool in the lower plane to a particular client entity of the higher-level plane, and to establish reachability information that permits the lower and higher plane entities to mutually communicate. Additional management functionality is not precluded, subject to the constraint that the application, SDN controller, or NE have exclusive control over any given resource. Management covers infrastructure support tasks that are not to be done by the application, controller and data planes themselves. Management may also perform operations that the application, controller and data planes are restricted from doing by policy or for other reasons. The SDN architecture recognizes classical management functions such as equipment inventory, fault isolation, and software upgrade.

One of the perceived benefits of SDN consists of allowing clients (in foreign trusted domains) to perform many of the actions that are today performed by management systems. The traditional OSS interface is expected to play a smaller role over the course of time, as customer applications take on more responsibility via SDN controllers.

Within the scope of SDN are the SDN-specific management functions, namely recording and expressing business relationships (policies) between provider and client, and configuring SDN entity environment and initialization parameters. This includes coordinating data plane handoff points,

identification conventions, reachability and credentials among logical and physical entities. The SDN architecture requires this information to be configured into the relevant SDN NEs, controllers, and applications, but does not specify the nature or structure of the OSSs.

3.1.2 Policy-based SDN architecture

Cisco Application Centric Infrastructure (ACI) [C-ACI] is a DC architecture with centralized automation and policy-driven application profiles. ACI introduces SDN into the environment with integrated physical and virtual infrastructure. ACI architecture consists of the following elements: physical switches (Cisco Nexus 9000 series) and virtual switches (Cisco Application Virtual Switch (AVS)) for the virtual network edge, a centralized policy management, SDN controller (Cisco Application Policy Infrastructure Controller (APIC)).

One of the main concepts of ACI architecture is an application-driven policy model that simplifies automated provisioning and managing of resources. Network policies in this architecture define connectivity, security, and networking requirements for agile and scalable application deployments and allow managing both physical and virtual applications. ACI application policies are created and stored at the SDN controller. It is also controller's responsibility to enforce application policies that were set based on application-specific network requirements. APIC also provides policy authority and resolution mechanisms, serving as a single point of automation and fabric element management in both physical and virtual environments.

The framework of APIC provides centralized access to other ACI components through an object-oriented RESTful APIs. Northbound APIs allow integration with existing management and orchestration frameworks. They also allow integration with OpenStack interfaces to provide network policy consistency across physical and virtual environments. Southbound APIs let you extend Cisco ACI policies to existing virtualization and Layer 4 through 7 service and networking components. The ACI southbound protocol is called OpFlex [OPFLEX]. Unlike other commonly used SDN protocols, it supplies application policy, not low-level configuration, to network devices. This approach introduces complexity into devices on one hand, but on the other hand it allows devices to configure themselves.

By centralizing policy but distributing control, networks can become much more scalable, resilient, and interoperable. OpFlex is submitted to the IETF for standardization to OpenDaylight for open source SDN implementations.

The OpFlex system architecture consists of a number of logical components: Policy Repository (PR), Endpoint Registry (EPR), Observer, and the Policy Elements (PE), where Policy Element is associated with entities comprising the policy administrative domain that is responsible for local rendering of policy. The PR serves as the single source of all policies and handles policy resolution requests from the Policy Elements. The EPR is the component that stores the current operational state of the EP within the system. PEs register the EPs with the EPR upon EP attachment to the local device where the PE is resident. The Observer serves as the monitoring subsystem that provides a detailed view of the system operational state and performance.

In OpenDaylight SDN controller the policy-based architecture in represented by the Group Policy incubation project that introduces the notion of groups of endpoints and policy abstractions that govern communication between these groups. The goal of this project is to support a northbound API that accepts abstract policy based on application requirements while offering numerous concrete southbound implementations. Similarly to the Cisco ACI architecture, the OpenDaylight group-based policy model introduces several important new primitives including:

- **Groups:** Groups include sets of network endpoints (potentially but not limited to virtual machines) with the same policies and requirements.
- **Policies:** Policies consist of sets of rules that govern how groups interact.
- **Policy rules:** Rules capture specific requirements about how groups interact. For example, a policy rule may allow HTTP traffic to a group of web servers for example. While rules capture a specific requirement, they must remain abstract and not tied to a specific hardware or software implementation.

Similarly to the ACI architecture, the group-policy architecture introduces several new components: an Endpoint Registry, a Contract Composer, and a pairwise Affinity Decomposer, and a "Native Driver". The idea is that group-based policies may be decomposed into pairwise relationships so affinity relationships can be applied prior to rendering by a specific southbound plugin.

The architecture uses several components provided in Opendaylight: the Flow Resource Manager, the OpenFlow plugin, and the Netconf south-bound plugin. The architecture may also use several proprietary CLI modules that communicate with legacy network elements.

The architecture is shown in the following figure:

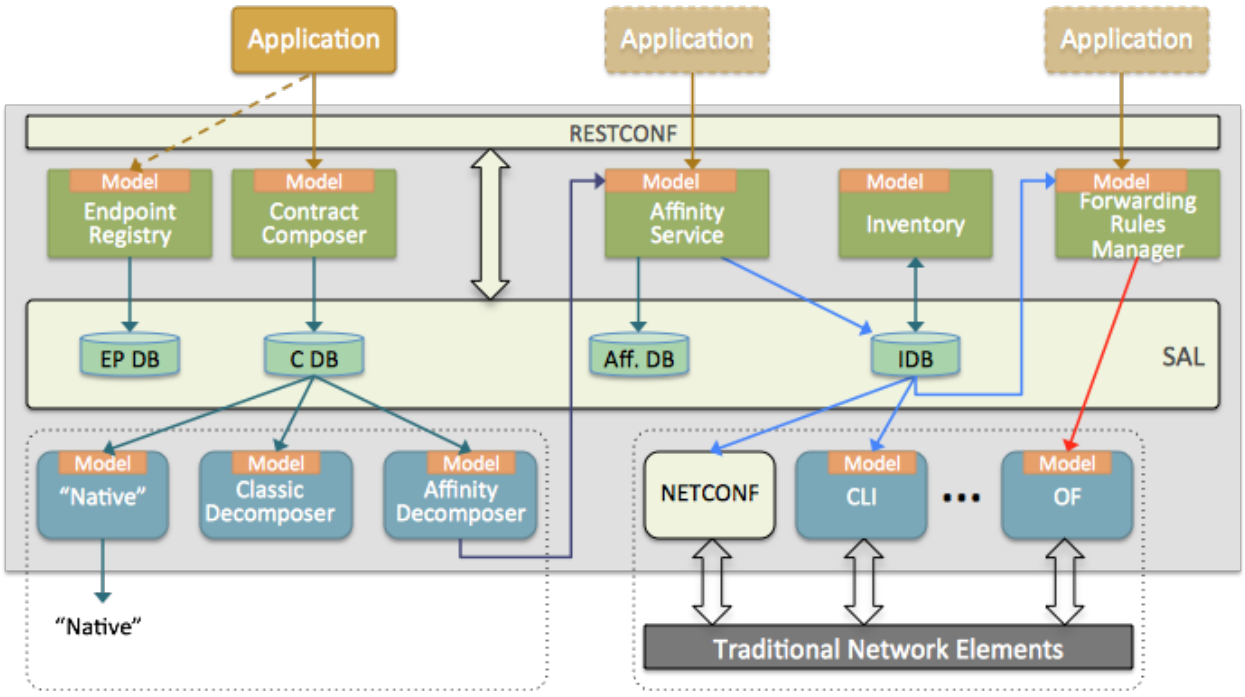


Figure 3-2 : Group-policy architecture.

The Endpoint Registry keeps track of all configured and discovered endpoints, as well as assignments of Endpoints into EPGs (Endpoint Groups). Contract Composer is used as the entry point through which applications specify contract policies. While "Native" Driver talks to systems that can understand contract policies directly, the Affinity Decomposer transforms Contract policies into pairwise affinity specifications and uses Affinity Service to provide affinity services. FRM (Flow Resource Manager) provides flow operations over one or more domains. The presented architecture uses existing southbound protocol plugins (OF, CLI, Netconf). Finally, Inventory keeps track of all endpoints in the system.

3.2 Protocols

The SDN architecture identifies control and management protocols for the interaction with a programmable data plane. The first type of protocol, which includes OpenFlow [OF] (see section 3.2.1), is used for the operation of the programmable devices at runtime through the dynamic configuration of the forwarding plane. This type of protocol is adopted at the D-CPI interface between an SDN controller and the underlying controlled virtual or physical devices and it can be used to query the hardware capabilities (e.g. the device ports), retrieve packets from the data plane, configure the flow tables or collect monitoring information.

The other type of protocol operates at the management plane; it is used to configure the programmable devices during the deployment phase and, broadly speaking, handles actions that take place at a longer timescale if compared to OpenFlow commands. These protocols include the OpenFlow Management and Configuration protocol (OFConfig [OFC]) and the OVSDB protocol [OVSDB] (see sections 3.2.3 and 3.2.4) and do not necessarily require the presence of an SDN controller. Typical actions enabled

by these protocol are the activation and the configuration of some ports, e.g. with specific queues. This configuration can be usually retrieved in read-only mode by the SDN controllers through operation protocols like OpenFlow.

3.2.1 OpenFlow

The Open Networking Foundation (ONF) [ONF] is the organization dedicated to the promotion and adoption of software-defined networking (SDN), funded in 2009 by companies interested in the development of SDN technologies (such as Google, Facebook, Yahoo, Microsoft, Verizon and Deutsche Telekom). ONF emphasizes an open, collaborative development process that is driven from the end-user perspective. To date, ONF's foremost achievement is introducing the OpenFlow™ standard, which enables communication between the SDN controller and the forwarding plane. Although SDN and OpenFlow™ started as academic experiments initially by Stanford University in 2008, they gained significant interest from the industry over the past few years. Many vendors of commercial switches now include support of the OpenFlow™ API in their equipment.

OpenFlow™ is almost certainly the most popular open standard that allows direct access and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based). It was the absence of an open interface to the forwarding plane that has led to the characterization of today's networking devices as monolithic, closed, and mainframe-like and hence, OpenFlow™ has been developed to overcome that pitfall, allowing the network control out of proprietary network switches and promoting an open source, locally managed control software [SDN-OF]

OpenFlow™ can be compared to the instruction set of a CPU. Figure 3-3 shows the protocol specific basic primitives that can be used by an external software application to program the forwarding plane of network devices, similarly to the way an instruction set of a CPU would program a computer system [SDN-OF].

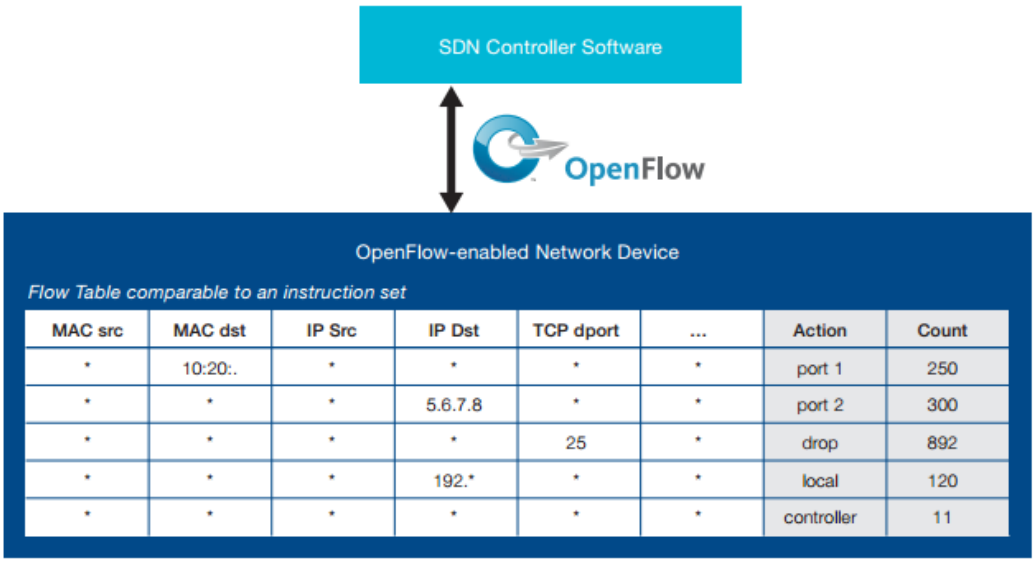


Figure 3-3 : Example of OpenFlow Instruction set. Source [SDN-OF].

Table 19 lists the most important versions of OpenFlow since ONF acquisition together with the most relevant contributions to the protocol.

Version	Release Date	Improvements	Link to Spec. Doc.
0.8.9	February 2009	IP Netmasks New Physical Port Stats IN PORT Virtual Port Port and Link Status and Configuration Echo Request/Reply Messages Vendor Extensions Explicit Handling of IP Fragments	OpenFlow_0.8.9_release_notes

		802.1D Spanning Tree Modify Actions in Existing Flow Entries More Flexible Description of Tables Lookup Count in Tables Modifying Flags in Port-Mod More Explicit New Packet-Out Message Format Hard Timeout for Flow Entries Reworked initial handshake to support backwards compatibility Description of Switch Stat Variable Length and Vendor Actions VLAN Action Changes Max Supported Ports Set to 65280 Send Error Message When Flow Not Added Due To Full Tables Behaviour Defined When Controller Connection Lost ICMP Type and Code Fields Now Matchable Output Port Filtering for Delete Flow Stats and Aggregate Stats	
0.9	July 2009	Failover Emergency Flow Cache Barrier Command Match on VLAN Priority Bits Selective Flow Expirations Flow Mod Behaviour Flow Expiration Duration Notification for Flow Deletes Rewrite DSCP in IP ToS header Port Enumeration now starts at 1 Other changes to the Specification	OpenFlow_0.9_release_notes
1.0	December 2009	Slicing Flow cookies User-specifiable datapath description Match on IP fields in ARP packets Match on IP ToS/DSCP bits Querying port stats for individual ports Improved flow duration resolution in stats/expiry messages Other changes to the Specification	openflow-spec-v1.0.0.pdf
1.1	February 2011	Multiple Tables Groups Tags : MPLS & VLAN Virtual ports Controller connection failure Other changes	openflow-spec-v1.1.0.pdf
1.2	December 2011	Extensible match support Extensible ’set field’ packet rewriting support Extensible context expression in ’packet-in’ Extensible Error messages via experimenter error type IPv6 support added Simplified behaviour of flow-mod request Removed packet parsing specification Controller role change mechanism Other changes	openflow-spec-v1.2.pdf

1.3	June 2012	Refactor capabilities negotiation More flexible table miss support IPv6 Extension Header handling support Per flow meters Per connection event filtering Auxiliary connections MPLS BoS matching Provider Backbone Bridging tagging Rework tag order Tunnel-ID metadata Cookies in packet-in Duration for stats On demand flow counters Improved version negotiation Other changes	openflow-spec-v1.3.0.pdf
1.4	October 2013	More extensible wire protocol More descriptive reasons for packet-in Optical port properties Flow-removed reason for meter delete Flow monitoring Role status events Eviction Vacancy events Bundles Synchronised tables Group and Meter change notifications Error code for bad priority Error code for Set-async-config PBB UCA header field Error code for duplicate instruction Error code for multipart timeout Change default TCP port to 6653	openflow-spec-v1.4.0.pdf

Table 19: OpenFlow™ versions with corresponding improvements

Also worth noting that OpenFlow™ version 2.0 is currently being defined and will be released soon.

It is a fact that OpenFlow™ switching has already been incorporated into a number of infrastructure designs, both physical and virtual, as well as SDN controller software. There are commercially available network services and business applications that use OpenFlow™ and interact with SDN controllers, proving an enhanced integration and coordination between them.

3.2.2 Network Configuration protocol (NETCONF)

NETCONF is a protocol defined by the IETF to "install, manipulate, and delete the configuration of network devices" [RFC6241]. NETCONF operations are carried out based on a Remote Procedure Call (RPC) layer (using XML encoding) and allows for operations to query and edit configurations of a network device. A NETCONF session is the logical connection between a network administrator or network configuration application and a network device.

The NETCONF protocol can be conceptually partitioned into four layers, as shown in Figure 3-4. Their detailed description can be found in [RFC6241].

1. The Content layer consists of configuration data and notification data.
2. The Operations layer defines a set of base protocol operations to retrieve and edit the configuration data.
3. The Messages layer provides a mechanism for encoding remote procedure calls (RPCs) and notifications.
4. The Secure Transport layer provides a secure and reliable transport of messages between a client and a server.

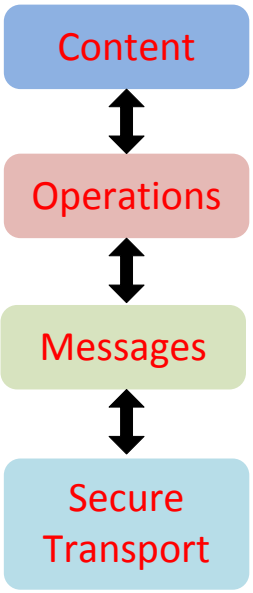


Figure 3-4 : NETCONF layers.

The NETCONF protocol has been implemented in network devices (e.g. routers, switches, etc.) major equipment vendors. A key aspect of NETCONF is that it allows the functionality of the management protocol to closely mirror the native functionality of the device. This reduces implementation costs and allows timely access to new features. In addition, applications can access both the syntactic and semantic content of the device's native user interface [RFC6241]. NETCONF's support for robust configuration change transactions involving a number of devices is an additional strength.

NETCONF is also used, together with YANG, in OF-Config [OFC], which is the complementing protocol to OpenFlow (OF) in the SDN southbound interface.

3.2.3 OpenFlow Management and Configuration protocol (OFConfig)

The OpenFlow Management and Configuration protocol (OFConfig, [OFC]) has been standardized by the ONF and it is aimed at enabling the remote administration of OpenFlow capable switches. Different from the OpenFlow protocol, which operates on a per-flow basis, OFConfig works at the network operator level. This means that the time scales of each protocol are notably different. While OpenFlow takes active part during the data plane operation (i.e., when the flows are installed, uninstalled and transmitting data), OFConfig uses a slower time scale since it is devoted to configuration and maintenance tasks such as enabling and disabling ports that will be further used or removed. Hence, both protocols work in parallel but have different responsibilities and timings.

From an architectural perspective (Figure 3-2), OFConfig introduces the concept of the OpenFlow Capable Switch context as a set of OpenFlow Logical Switches that are managed (via OFConfig) by one or more configuration entities (i.e., OpenFlow Configuration Points [OFC]). An OpenFlow Logical Switch is defined as an OpenFlow switching unit that is operated by an OpenFlow controller as a single entity. More specifically, an OpenFlow Logical Switch can be seen as a single data plane. Hence, an OpenFlow Capable Switch is able to host several OpenFlow Logical Switches or data planes, which, in turn, are a set of OpenFlow resources (e.g., ports, queues, etc.) of a physical OpenFlow switch that operate together and separated from the rest.

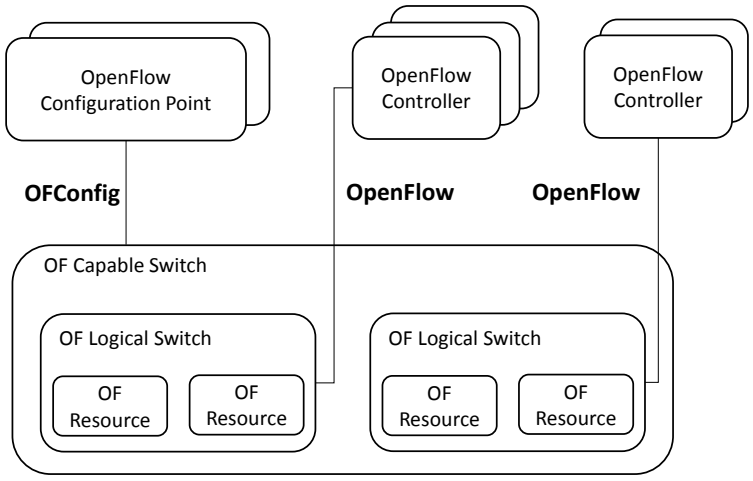


Figure 3-5 : OFConfig components [OFC].

From an operational perspective, the OFConfig protocol functional requirements fundamentally aim at configuring the logical switches of an OpenFlow Capable Switch. In this context, the first function of the protocol is to instantiate one or more data planes (i.e., OpenFlow Logical Switches) in the OpenFlow Capable Switch. Once instantiated, OFConfig has to implement a procedure to assign one or more OpenFlow controllers to the logical switches. Being the logical switch a set of OpenFlow resources (ports, queues, etc.) of the physical switch, the protocol offers mechanisms to configure such resources as well. Besides, given that the communication between the logical switches and the controllers uses secure transport protocols [OF], OFConfig has as a requirement the configuration of the certificates that enable such secure communication. Finally, the protocol also implements a mechanism to discover the capabilities of the logical switches. [OFC] gives further detail not only on the functional requirements but also on the operational ones.

In light of the above, since no data planes are instantiated at a starting point, the OpenFlow Capable Switch owns all the resources. Then, the OpenFlow Configuration Point uses the OFConfig protocol to instantiate one or more data planes (i.e., OpenFlow Logical Switches). These logical switches are assigned an OpenFlow controller that will take care of the flow-based operation as defined by the OpenFlow protocol standard [OF].

The ONF is working on the support of the emerging Negotiable Datapath Model (NDM, [OF-NDM]) inside the OFConfig protocol. An NDM is an abstract definition that specifies forwarding behaviours of the logical switches that can be controlled by means of the OpenFlow protocol. The goal of this feature is to allow implementers to develop optimized and more complex forwarding behaviours. The support of NDM in OFConfig is currently set as optional.

For the messaging, OFConfig uses the NETCONF protocol [RFC6241]. NETCONF is an XML-based protocol designed to configure network devices. NETCONF provides an extensible structure that allows OFConfig to extend the protocol for its purpose, and uses secure transport (SSH, TLS, SOAP and BEEP), which is a requirement for OFConfig as well. The OFConfig protocol message specification is depicted in [OFC].

3.2.4 OVSDB management protocol

OpenFlow devices or Open vSwitch (OVS) instances can be programmed and controlled via OpenFlow protocol [OF], while their configuration and management can be performed through the Open vSwitch Database management protocol (OVSDB), based on JSON-RPC version 1.0 [JSON-RPC], as specified in the RFC 7047 [OVSDB]. In particular, the OVSDB protocol supports operations like CRUD (Create, Read, Update, Delete) actions on OpenFlow datapaths (i.e. bridges), ports, tunnel interfaces and queues, configuration of the SDN controller(s) and manager(s) to which the device should connect, configuration of QoS policies and collection of statistics.

The configuration of an Open vSwitch instance is structured and maintained in a database organized following the schema specified in [OVSDB-S]. RFC 7047 specifies the protocol used to interact with this database, in order to retrieve the schema and the stored information (i.e. retrieve the current

configuration or statistics) as well as to add new entries or modify the existing ones (i.e. modify the current configuration).

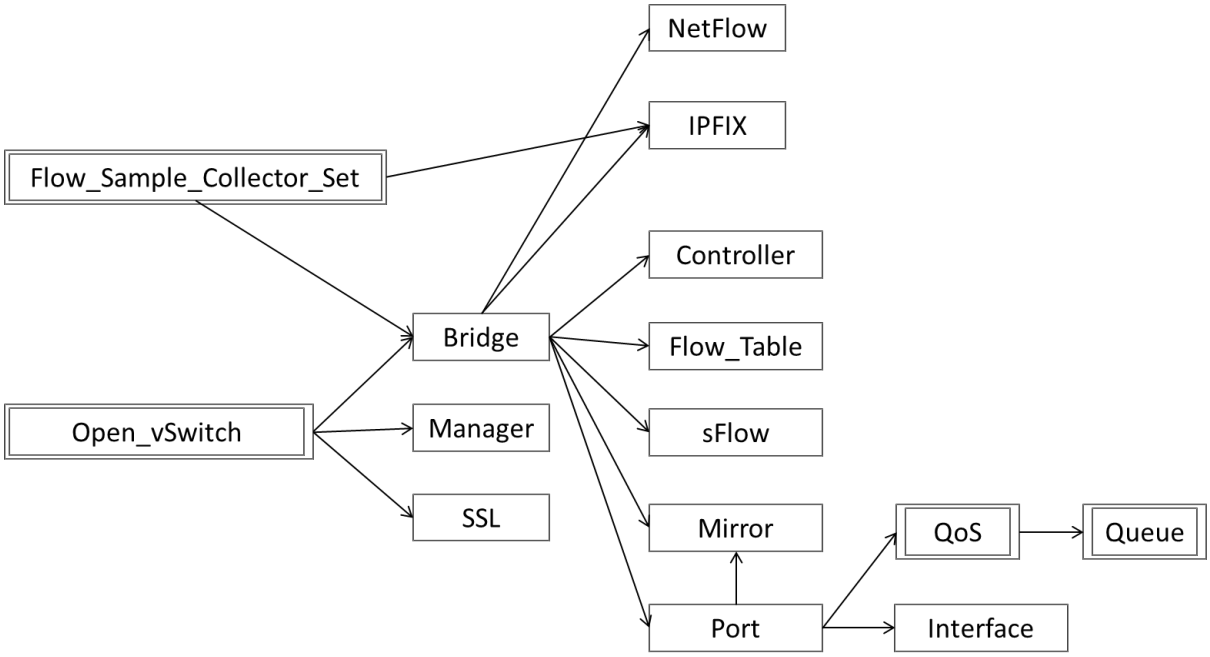


Figure 3-6 : OVSDB schema: tables.

The OVSDB schema is specified in a separated document ([OVSDB-S]); the latest version available today is 7.10.1. The different tables specified in the schema are shown in Figure 3-6 and the most relevant ones are summarized in Table 20, with the description of the main configuration parameters.

Table	Description
Open_vSwitch	The global configuration of the (virtual) switch, including bridge, default behaviour at start up, maximum numbers of flows, statistics, etc.
Bridge	Bridge configuration. A bridge record is the logical representation of an Ethernet switch with one or more ports (modelled by the "Port" table). The configuration options include OpenFlow parameters (e.g. datapath_id, flow tables, protocol version), spanning tree (STP) and rapid spanning tree protocol (RSTP) configuration.
Port	Port configuration. A port record is the representation of a port within a bridge. Each port has usually exactly one interface, while ports with more than one interface are defined as "bonded ports". Typical parameters associated to a port are related to VLAN configuration, bonding configuration (e.g. for link failure detection, Link Aggregation Configuration Protocol (LACP) and rebalancing configuration), STP and RSTP configuration, and port status and statistics for monitoring purposes.
Interface	Configuration of the logical representation of a physical network device in a port. Configuration parameters include MAC address, OpenFlow port number, tunnel options, ingress policing, administrative status, management parameters for connectivity faults and VM identifiers. Counters for statistics are available in terms of packets or bytes transmitted and received, errors and dropped packet.
Flow_Table	Configuration of an OpenFlow table, including flow limit and policies to be applied in case of overflow.
QoS	Configuration of Quality of Service. QoS can be enforced through Linux hierarchy token bucket (HTB) or Linux hierarchical fair service curve classifier (HFSC). The QoS table identifies the Queues (see next field) configured for each QoS entry and the maximum rate shared by all the queued traffic.
Queue	Configuration of a QoS output queue. The configuration parameters include the dscp code for marking the egress packets and the parameters specific for the QoS type applied in the queue (i.e. min/max rate, burst and priority for HTB and min/max rate for HFSC).
Mirror	Configuration of port mirroring, to send selected frames to special mirrored port. The configuration parameters include the criteria to select packets for mirroring and the mirroring destination. Statistics about transmitted packets and bits are available as mirror counters.
Controller	Configuration of the SDN OpenFlow controllers, both primary and

	service controllers.
Manager	Configuration of the OVSDB management connection to an OVSDB client.
NetFlow	NetFlow configuration.
SSL	SSL configuration, including private key and certificate.
sFlow	sFlow configuration, for the remote monitoring of the switches.
IPFIX	IPFIX configuration.
Flow_Sample_Collector_Set	Flow_Sample_Collector_Set configuration.

Table 20 – OVSDB schema: main tables

3.3 Network Function Virtualization

The Network Function Virtualisation (NFV) paradigm, defined by ETSI [NFV], is based on the software virtualization of Network Functions (NFs) which can be provided using general purpose hardware (servers and storage devices) avoiding the high CAPEX investments on a variety of dedicated hardware devices. The decoupling between NFs and physical devices allows to share the hardware among different network functions and to easily deploy their virtual instances in the most convenient locations in a flexible and dynamic manner.

The NFV approach allows to introduce new services and features very quickly through the deployment of software based NFs, without the need of complex installations and configurations. Moreover, the virtual entities which constitute the NF blocks can be dynamically activated, suspended, migrated and re-allocated to optimize the workloads, reduce energy consumption, improve the infrastructure utilization through resource sharing and enhance service resiliency in case of failures and infrastructure disruptions. The automation of NFVs management and operation is fundamental to fully exploit these potential benefits and simplify the procedures to efficiently deploy and provision virtual network functions in DC environments. Thus, the COSIGN approach for DCN control and management should consider NFV as a possible applicability case and be compliant with the corresponding components and interfaces defined in the NFV architectural framework [NFV-af].

Figure 3-7 shows the high level NFV framework defined by ETSI, highlighting the positioning of the COSIGN DC network control plane and net/IT orchestration platform. In the picture, the Virtual Network Function (VNF) boxes represent the software instances of network functions which run over the NFV Infrastructure (NFVI) and can be considered as VMs running in COSIGN data centres. The NFVI includes all the physical resources supporting the execution of the VNFs, together with the corresponding virtual resources. This entity can be mapped to the entire DC infrastructure, where COSIGN will introduce the innovative optical devices and the related enhancements for optical network virtualization (addressed in Work Package 3). Finally, the NFV Management and Orchestration is dedicated to handle the whole set of physical and virtual resources of the NFVI, in combination with the lifecycle of the VNFs instances. Its functions cover the dynamic construction and management of VNF instances, including their dependencies and inter-connectivity in case of explicit VNF Forwarding Graphs (VNF-FG) for service chains or general VFN Sets where connectivity requirements are not specified.

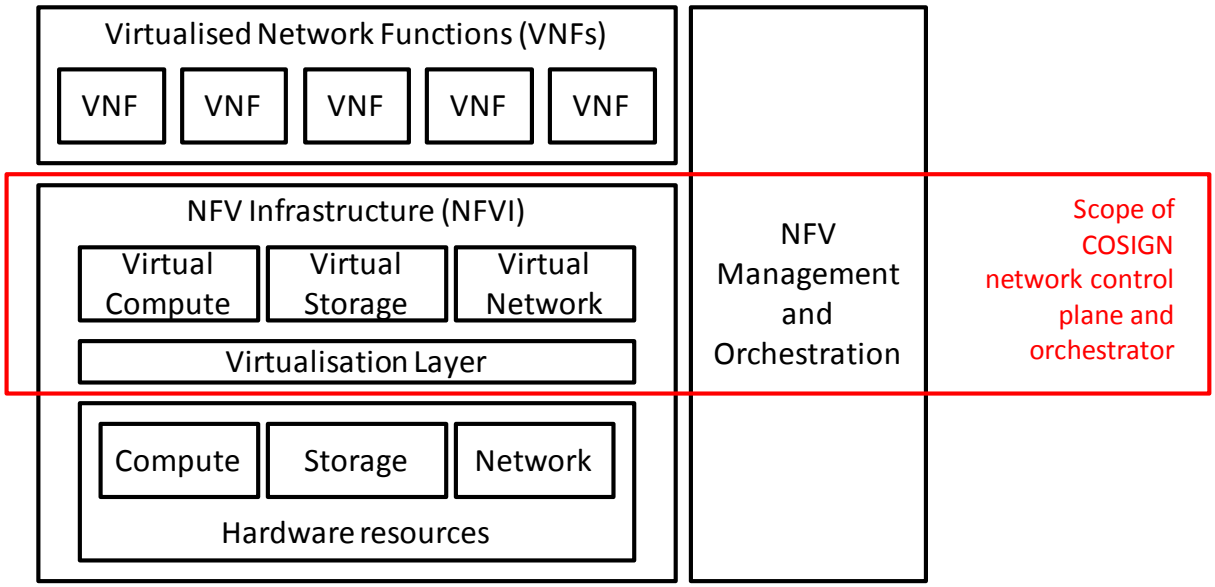


Figure 3-7 : COSIGN scope in the high-level NFV framework.

In COSIGN, where the focus is mostly on the networking aspects but taking into account the joint allocation of computing, storage and network resources for DC optimization purposes, these orchestration and management functions are provided by the combined actions of the SDN-based network control plane (Work Package 3) and cloud orchestration platform (Work Package 4). In particular, the COSIGN network control plane will implement the functions of the Virtualized Infrastructure Manager in scope for the networking part: resource management (network NFV infrastructure inventory and allocation of network connectivity) and operation actions (management and visibility of the network NFV infrastructure, monitoring and fault detection, capacity optimization). On the other hand, the COSIGN orchestration platform will cover the functions of the NFV Orchestrator, which realizes the network services over the NFVI using the software resources, and the VNF Manager, which is responsible for the VNF lifecycle management from instantiation to termination.

4 Survey of existing control plane technologies for intra-Data centres network

This section provides a survey of existing control plane technologies, discussing their possible adoption in DC networks. Section 3.1 focuses on legacy control plane solutions currently available in DC management systems proposed by different vendors. On the other hand, section 3.2 analyses open source solutions based on the SDN paradigm, with a survey of the main SDN controllers. Beyond the evaluation of the overall platform and the supported functions or applications, this analysis will mainly focus on two aspects:

- The integration between the SDN controllers and the main Cloud Management Systems (e.g. OpenStack [OpenStack], CloudStack [CloudStack]), in order to evaluate the SDN controller applicability to DC environments;
- The flexibility of the south-bound interface, e.g. in terms of support of multiple D-CPI protocols or drivers and extensibility of the information model used to describe network resources. This aspect is fundamental to evaluate the possibility to enhance the SDN controller beyond the traditional control of L2-L3 devices and integrate the support of optical devices with new drivers and extended protocols for optical technologies.

4.1 Legacy DCN control plane solution

Traditionally, DC networks were built using Ethernet technology to interconnect terminals using switching fabrics. Different switched topologies were developed in order to optimize connectivity in the DC networks [fat-tree]. Most of these topologies are characterized by multi-tier layers that provide network access, traffic aggregation and core networks. The data flow is usually forwarded up from access network, via aggregation layer to the core network, and then back down to the access layer of the destination. In order to assure loop-free paths between source and destination at the L2, switches run smart distributed algorithms for path computation and building Spanning Trees. At the L3, routers run sophisticated protocols, such as OSPF, IS-IS, RIP, EIGRP for collecting the topology information, calculating optimal paths and building forwarding tables.

As a result, each network element, router or switch, participates in complex distributed protocols. Centralizing network management in the smart network elements brought a wide range of complex, vendor proprietary features and functions and introduced significant complexity to the DC network control plane and management. Together with the evolution of the network requirements, more and more incremental changes and sophisticated features were added to the switching and routing protocols, resulting in network complexity and fragility.

In recent years, virtualization of the modern DC compute and network resources allowed heavy server utilization and introducing of heterogeneous networks with very different requirements for storage, compute and appliances networks [ODIN2012]. The traditional DC network control approach became un-scalable for managing these complex networks and fulfil all their requirements. These issues led to re-considering of the control plane solution and to introducing of the SDN paradigm [Green4D] both in industrial and in open source solutions. In the SDN, the network control plane is moved from the switches and routers toward the network controller. This way the network elements and their management are significantly simplified, while the centralized controller bears all the complexity of the control plane. An extensive overview of the industrial SDN controllers is presented in Section 5.1.

4.2 Analysis of existing SDN control plane solution

4.2.1 OpenDaylight

OpenDaylight [odl] is an open source platform for network programmability hosted by the Linux Foundation that enables SDN through a combination of components including a fully pluggable controller, interfaces, protocol plug-ins and applications. The core part of the project, the modular,

pluggable and flexible controller, is implemented in Java and can be contained within its own Java Virtual Machine (JVM), so that it can be run on any platform that supports Java. Besides, the northbound and southbound interfaces have clearly defined and documented APIs. The combination of these elements allows vendors, service providers, customers and application developers to utilize a standards-based and widely supported SDN platform.

The OpenDayLight SDN platform architecture is composed of three main layers (Figure 1):

- *The Network Application Orchestration and Services:* This layer is composed of business and network logic applications for network control, provisioning and monitoring. In addition, more complex and advanced orchestration applications needed for cloud, data centre and virtualization services also reside in this layer. These heterogeneous applications use the open northbound API framework exposed by the SDN controller platform which is implemented by a set of bidirectional REST interfaces. It is important to highlight that the OpenDayLight northbound framework is open and flexible to either extend already existing APIs, or create new REST APIs to expose new user-defined internal network services capabilities.
- *The Controller Platform:* This is the layer where the SDN abstraction happens. It is composed by a collection of dynamically pluggable modules to perform a variety of network tasks and services. The SDN controller platform offers a set of base network services such as the Topology Manager, the Statistics Manager, the Switch Manager, the Shortest Path Forwarding and the Host Tracker. Each of these modules exposes its own northbound APIs to let network applications and orchestration services program the network and the controller behaviour. In addition, platform oriented services and other enhanced network services can be implemented as part of the controller platform for enhanced SDN functions.
- *Southbound Interfaces and Protocol Plugins:* OpenDayLight supports multiple southbound interfaces through a heterogeneous set of dynamically loadable plugins. In this way, OpenDayLight implements a wide set of southbound protocols, such as OpenFlow 1.0, OpenFlow 1.3, BGP Link State (BGP-LS), Open vSwitch Database (OVSDB), PCEP, etc. A newly created OpenDayLight project is also providing the implementation of a dedicated OpenContrail plugin to enable OpenDayLight to utilize OpenContrail's networking capabilities [odl-openc-pl]. These modules are dynamically linked into a Service Abstraction Layer (SAL) which actually provides the SDN controller platform abstraction capabilities. Below the SDN controller platform, a wide range of physical and virtual devices, switches, and routers that provide the actual connective fabric between the network endpoints can be controlled.

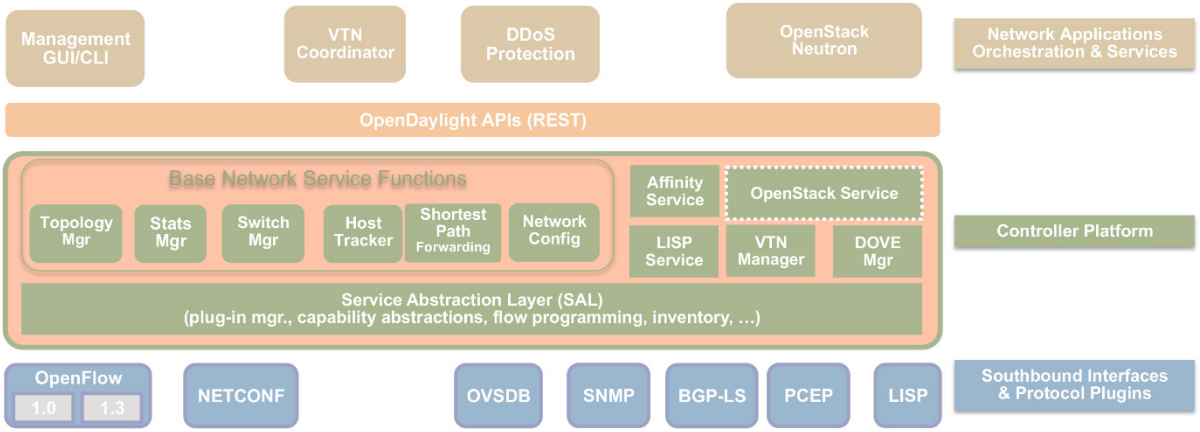


Figure 4-1 OpenDayLight control plane framework

The first OpenDayLight official release, Hydrogen, was launched in February 2014. This release provides three different editions: *Base*, *Virtualization* and *Service Provider*. The first one is conceived for those users and software developers who are exploring SDN and OpenFlow for proof-of-concepts or academic initiatives in physical and virtual environments. The second one is meant for data centre environments and it includes all the components of the Base edition plus functionalities for creating and managing Virtual Tenant Networks and virtual overlays. It also includes applications for security

and network management. Finally, the Service Provider edition is the version for providers and carriers who want to manage their existing networks by means of SDN. It includes the Base edition modules and provides support for protocols that can be commonly found in service provider networks (i.e. BGP, PCEP), as well as security and network management applications.

The second OpenDaylight official release, Helium, was launched in October 2014 and includes several additional plugins, with new applications, south-bound interfaces and features. One of the new components, the Group Based Policy, defines an application-centric policy model which separates information about application connectivity requirements from the information about the underlying network infrastructure. This concept may be of interest for COSIGN scenarios, where service requests at the orchestration level should be expressed in terms of application needs, independently on the details of the DCN technologies and topology.

4.2.2 Floodlight

Floodlight is a Java based, modular SDN controller platform. Some of its characteristics are:

- The modules can be configured (enabled/disabled) at compile time (it's not based on OSGi).
- It does not come with a database although it contains an internal API for integration with databases and it currently provides an in-memory key-value store through this API.
- It is well integrated with Eclipse IDE.
- It uses Maven for dependency management and as a build tool.
- It does not provide a High Availability solution.
- Last official release was in December 2014, and the community is quite active.

Existing functions (they also provide a REST API):

- Firewall: reactively allows/denies flows based on rules (packet header matching) defined at the controller;
- Load Balancer: limited implementation of load balancing for ICMP, UDP and TCP services;
- Virtual Network Filter (VNF): simple network virtualization module which provides a REST API for integration with Openstack. It allows traffic between MAC addresses (VMs) based on their association with a virtual network: the VNF create flows between 2 VMs only if they belong to the same virtual network.
- Static flow pusher: proactive OF rule insertion.
- Forwarding (based on Dijkstra's algorithm): can work with Openflow (OF) islands connected through non-OF devices.

Floodlight southbound interface supports Openflow 1.0 and Openflow 1.3 in the official release.

Floodlight offers a REST API with JSON data as basic northbound interface, which allows access to the basic OpenFlow commands and metrics like:

- Retrieving information for the controller, connected switches or single switchports
- Inventory of all devices tracked by the controller
- Basic topology information

In addition there is an API for a "Static Flow Pusher", which allows a single flow entry to be added to a specific switch in the environment. This API is utilized by the sample "Circuit Pusher" REST application which allows bidirectional flows between two IP addresses across several switches to be installed. Firewall, LoadBalancer and OpenStack Quantum modules also provide JSON-based RESTful APIs.

Northbound Interface (NBI): The NBI is based on RESTful web services. The NBI provides capabilities for interrogation of the network topology, setting up OF rules in devices, setting up virtual networks (VNF) and defining firewall rules.

Regarding the cloud management integration, Floodlight is integrated with Openstack through the REST API provided by VNF.

The latest official version of Floodlight is v1.0 and its release date is 30th of December 2014.

4.2.3 OpenContrail

OpenContrail is an open-source initiative launched by Juniper in September 2013, corresponding to the open-sourced version of Juniper's commercial product Contrail. OpenContrail is released under Apache 2.0 software license and its community of contributors includes partners like Nokia, IPnett, Tcp Cloud, CodiLime and Cloudwatt.

The Contrail system has been designed to create and orchestrate virtual networks over physical devices. It automates the creation of virtual network services in support of cloud networking for a variety of scenarios, from Private Cloud to Virtual Private Cloud or Hybrid Cloud. On the other hand, the Contrail solution strongly focuses on Network Function Virtualization for the orchestration and management of networking functions in virtual machines to provide value added services in service providers' edge networks.

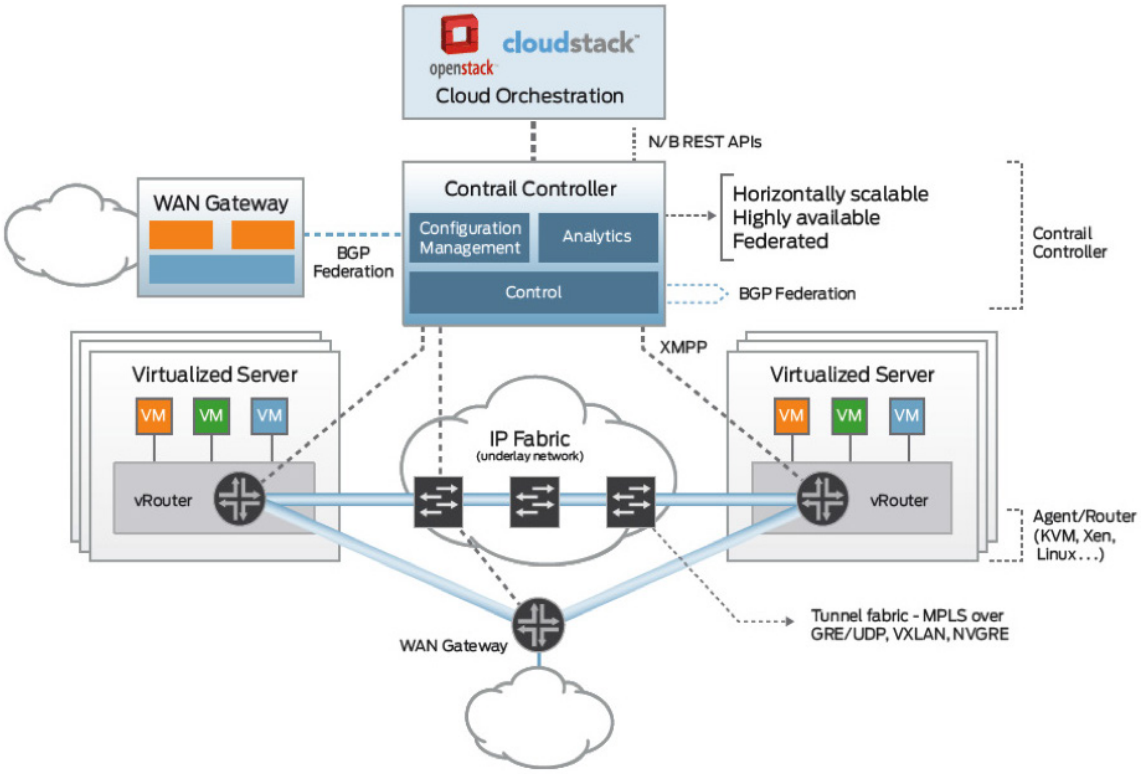


Figure 4-2 Juniper Contrail solution

The OpenContrail system (see Figure 4-2) consists of two main components: the OpenContrail Controller, a logically centralized and physically distributed SDN controller, and the OpenContrail vRouter, a distributed router that runs in the hypervisor of a virtualized server.

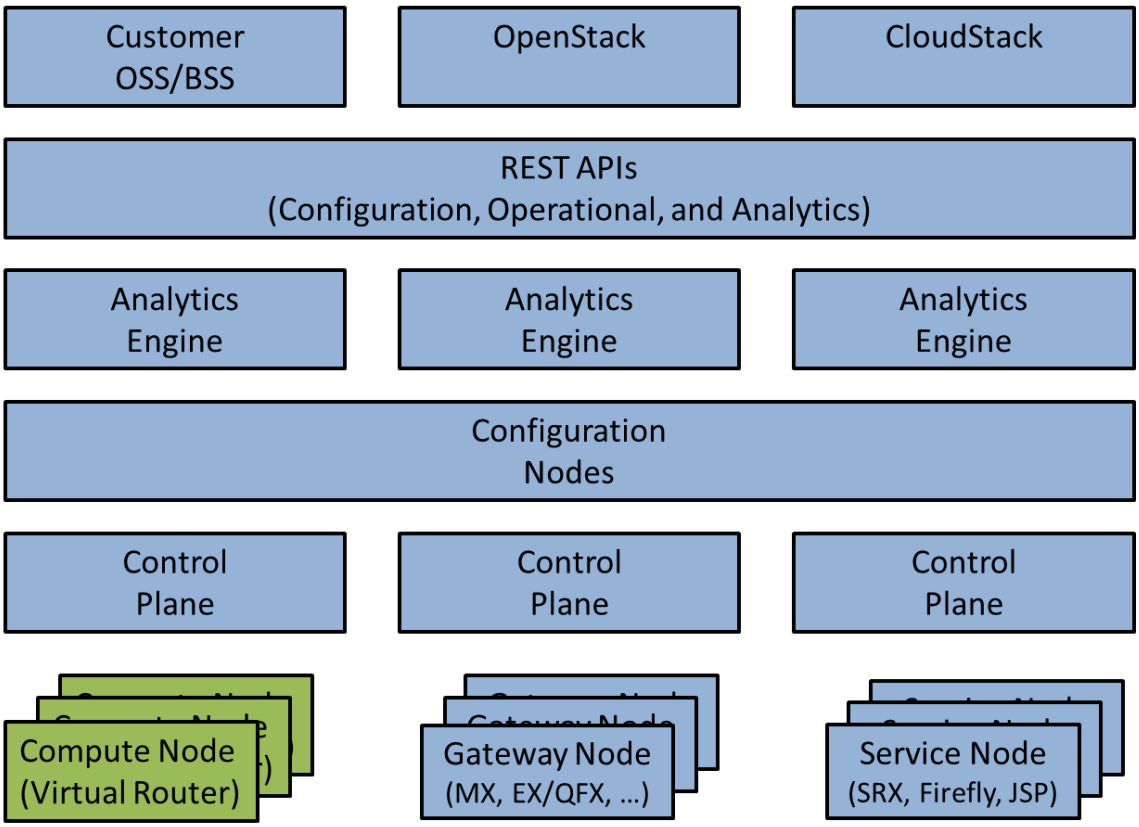


Figure 4-3 Juniper Contrail architecture

The vRouter, represented by the green Compute Node in Figure 4-3, is similar to a virtual switch like Open vSwitch but provides also routing and higher level functions. The OpenContrail Controller provides management, control and analytics functions and it is organized in three types of node, as shown in Figure 3:

- Configuration node: responsible for the management layer functions, it provides north-bound REST APIs to configure the system or collect operational data. These REST APIs are based on a high-level service data model that, following a declarative approach, describes entities with a high level of abstraction mapping the services from the end-user's perspective (e.g. a virtual network, a connectivity or a security policy). The Configuration node includes a translator, in charge of mapping the high-level service data model into the low-level technology data model, which represents objects related to specific network protocol constructs (e.g. a VXLAN network identifier).
- Control node: implements the logically centralized part of the control plane and it is in charge of enforcing the desired network state, as computed and described by the configuration node through the technology data model. The current version of OpenContrail adopts a combination of south-bound protocols including Extensible Messaging and Presence Protocol (XMPP) towards vRouters, Border Gateway Protocol (BGP) and Network Configuration (Netconf) protocols toward physical routers.
- Analytics node: responsible for collecting monitoring and event records generated from the other OpenContrail components. It aggregates and stores analytics information into an internal database, providing north-bound REST APIs for analytics queries.

The OpenContrail system is integrated with open source cloud management systems such as OpenStack and CloudStack. This integration is enabled through the north-bound REST APIs provided by the Configuration node towards orchestration systems or other applications, as shown in Figure 4-4 for OpenStack. In this case, the Contrail Neutron Plugin within OpenStack Neutron is used on the client side of the north-bound interface to trigger the creation of the virtual networks, while their

configuration is handled by the vRouter Agent following the indications of the OpenStack Nova Agent.

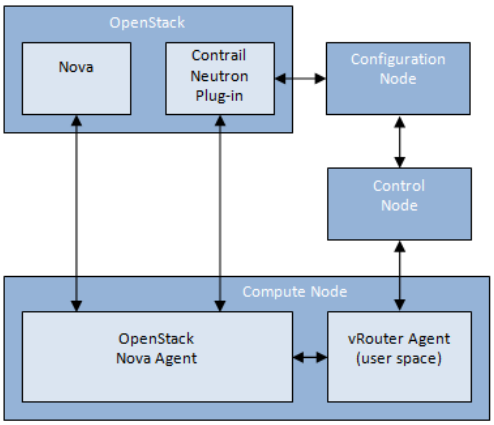


Figure 4-4 Integration between OpenContrail and OpenStack

The OpenContrail system can be used in several DC-based use cases, mainly to translate the abstract requests coming from the cloud orchestrator into concrete actions on the physical or virtual network devices, as shown in Figure 4-5. Examples of services supported by the OpenContrail system are the creation and management of isolated virtual tenant networks, tenants' connectivity to Internet or enterprise networks via VPN and DC interconnections over a Wide Area Network, network monitoring or traffic steering in support of Service Function Chains (SFCs).

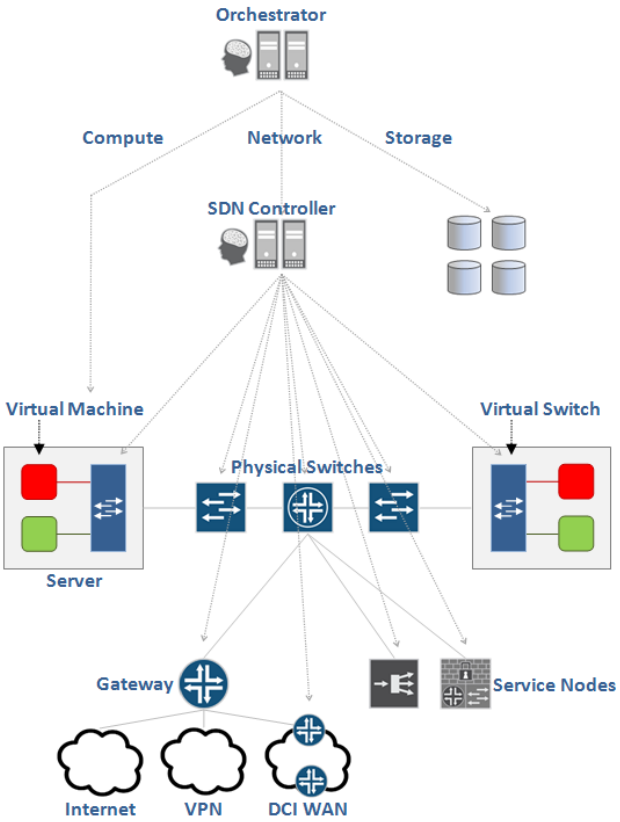


Figure 4-5 Role of orchestrator and SDN controller in data centres

4.2.4 Ryu Controller

Ryu SDN Framework is an Open Source Software (OSS) system developed and supported by NTT. As shown in Figure 4-6, Ryu framework supports development of SDN applications and supports

multiple types of network switches through multiple control (south-bound) protocols. For OpenFlow enabled switches, Ryu handles a wide range of OpenFlow versions, including 1.0, 1.2, 1.3, 1.4, and Nicira extensions. Ryu community effort includes certification program for verifying interoperability with a wide range of vendor and open OpenFlow switches. For non OpenFlow network devices, Ryu framework includes libraries for other south bound protocols, such as Netconf and SNMP. Additional south-bound protocols supported by Ryu are: OVSDB and OF-Config 1.1 for switch configuration and management and netflow and sflow for monitoring and statistics collection.

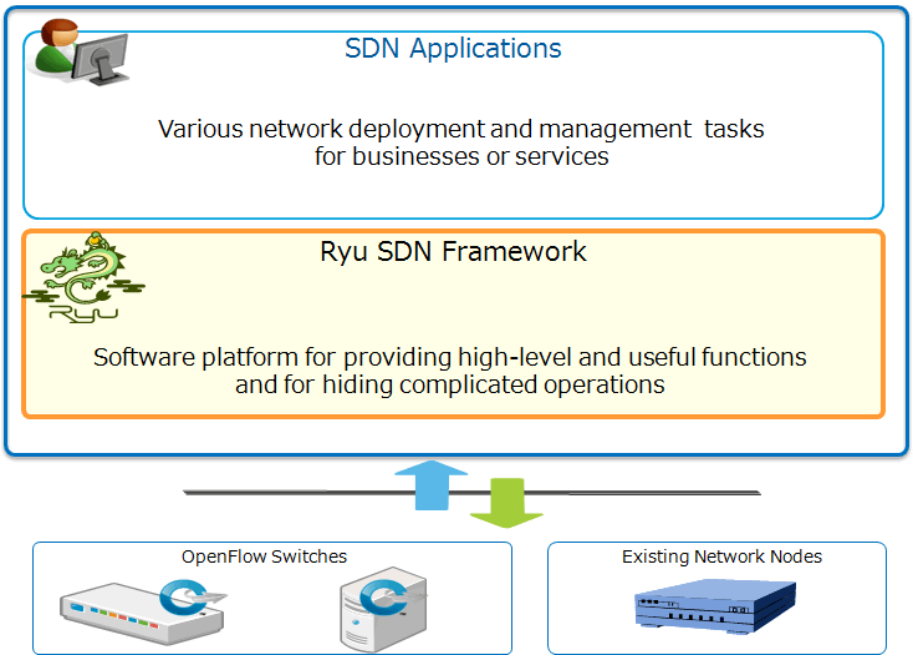


Figure 4-6 Ryu SDN Framework Overview

Written entirely in Python programming language, Ryu framework is component based and its messaging service does support components developed in other languages. Existing components, shown in Figure 4-7, include south bound protocols support, event management, messaging, in-memory state management, application management, infrastructure services and a series of reusable libraries (e.g., NETCONF library, sFlow/Netflow library). Additionally, applications like layer 2 switch, firewall, IDS (Snort), GRE tunnel abstractions, VRRP, as well as services, like topology discovery and statistics, are available.

At the northbound, Ryu offers a RESTful API with JSON data format for access to information and configuration. In general, every module can offer its own REST API. In particular, Ryu has an OpenStack Neutron plug-in that supports both GRE based overlay and VLAN configurations. Moreover, Ryu supports REST APIs for:

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

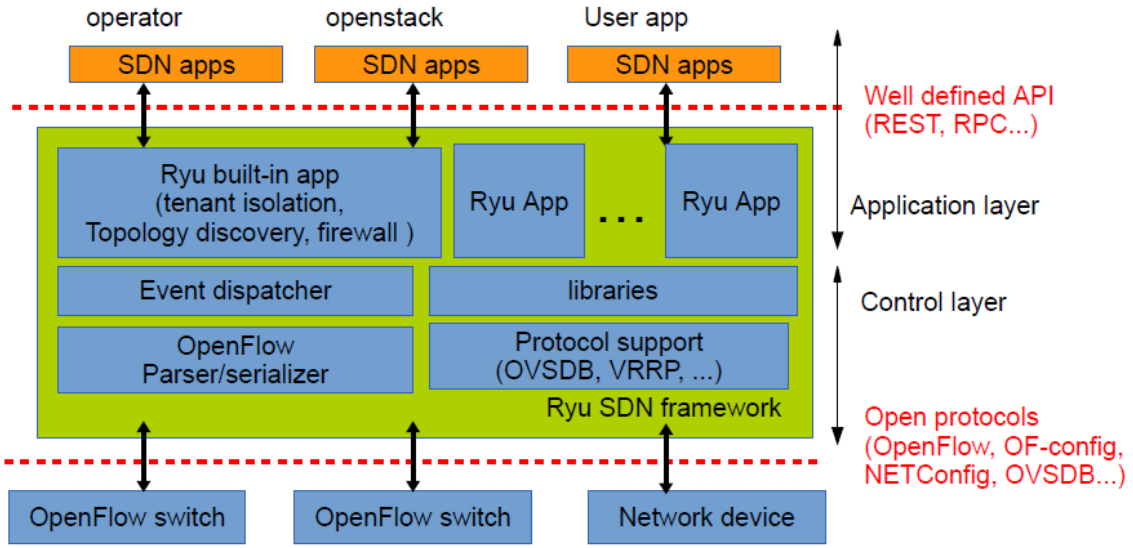


Figure 4-7 Ryu SDN Framework – core components and interfaces

While Ryu supports high availability via a Zookeeper component, as shown in Figure 4-8, it does not support a cooperative cluster of controllers. In addition, all known reported performance evaluations ([contr-perf], [sdn-course], [contr-comp]) report that Ryu is slow and not scalable as compared to other controllers. This can be mainly attributed to the programming language choice (interpreted Python) and to the fact that the whole framework is essentially single-threaded. Ryu's usage of greenlets for cooperative concurrency simplifies the programming on the one hand (by ensuring exactly one piece of code is executed at any given time), but on the other hand it renders the controller unable to utilize more than a single CPU core.

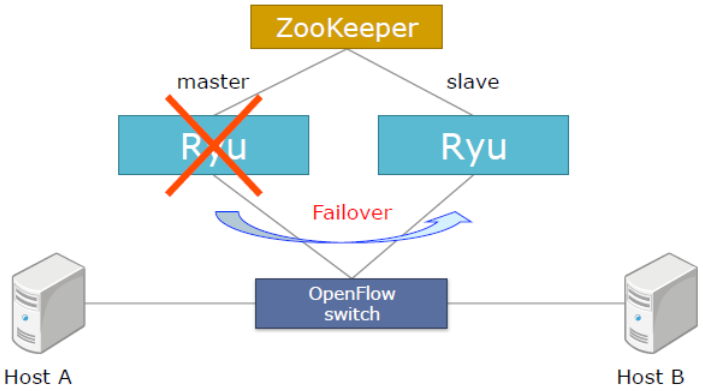


Figure 4-8 Ryu High Availability with Zookeeper

Ryu framework is developed under Apache 2 license, enjoys broad community support, and has good and reliable documentation. In summary, Ryu is a good candidate controller platform for research, education, and quick prototyping.

4.2.5 POX

POX [pox] is a simple open source SDN controller platform written in Python that can run on Linux, Windows or Mac OS machines. It is widely used in research and education environments, since it allows a fast and easy prototyping of network applications. The latest stable version, called *carp*, has been released in fall 2013 under Apache 2.0 license, while the current version, *dart*, is still being actively developed.

At the southbound interface, POX provides only support for OpenFlow 1.0 (including some OpenFlow extensions for Nicira and Open vSwitch devices) and OVSDB. Network applications can be developed as python modules making use of POX's APIs and can be launched through the POX command line. POX's APIs provides a powerful set of functions to receive and process or generate and send OpenFlow messages, including objects to handle the most common OpenFlow structures like flow tables and related entries, OpenFlow matches, statistics and actions. Moreover, a set of stock components is already included in the POX platform (e.g. hub, L2 learning switch, spanning tree, packet dump, nat, TCP load balancer, host tracker, etc.) and can be easily extended and re-used as needed to compose additional applications.

The main limitations of POX for its possible integration in COSIGN DCNs are related to the low flexibility in terms of external interfaces, modularity and extendibility. For example, the POX platform does not natively support multiple plugins on the south-bound interface, which is a fundamental feature to support the heterogeneity of technologies available in COSIGN data-plane. On the other hand, the lack of REST APIs for the interaction with external applications makes the integration with cloud orchestration and management systems more complex.

4.2.6 Comparison between SDN controllers

Table 21 provides a summarized comparison of the main SDN controllers analysed in our survey. From the architectural point of view, OpenDaylight provides most of the features that are required for the SDN controller of COSIGN DCNs. This is especially true for what concerns the modularity that allows to support a variety of south-bound plugins dedicated to the different technology of the COSIGN data-plane. Moreover, the internal architecture and applications of OpenDaylight provides a solid basis for network abstraction and virtualization functions which constitute one of the main pillars of COSIGN concepts. However, a final decision about the reference SDN controller platform for COSIGN developments will be taken in WP3, considering also software aspects.

Feature	OpenDaylight	Floodlight	OpenContrail	Ryu	POX
Support of south-bound plugins to interact with heterogeneous data-plane	OpenFlow and SNMP. Configuration (e.g., VLANs, STP, flooding protection), interface statistics, and topology information (e.g., via LLDP).	Supports OpenFlow 1.0 and 1.3.	XMPP, BGP and NETCONF supported at the moment. Plans for OpenFlow support.	Supports fully OpenFlow 1.0, 1.2, 1.3, 1.4 and Nicira Extensions; OF-CONFIG; NETCONF and OVSDB.	OpenFlow 1.0, with extensions for Nicira and Open vSwitch devices, and OVSDB. No support for multiple plugins.
Resource abstraction	Comprehensive list of REST APIs. Model Driven Service Abstraction Layer.	Supports abstractions for various network resources such as: node, port, link, network, isolated tenant networks.	Declarative high-level service data model to describe services from the end-user's perspective (e.g. a virtual network, a connectivity or a security policy).	Supported.	Only functions to manage OF structures.

Internal functionalities and extendibility	Many functions already available (e.g. topology manager, switch manager, forwarding rules manager, …). Easy to extend with new service plugins.	Several functions are already available (e.g. topology, device, host managers). Can be easily extended with new modules.	Setup of tenants connectivity to Internet or enterprise networks via VPN. DC inter-connections over a WAN. Network monitoring. Traffic steering for SFCs.	Supported.	Limited modularity and extendibility. Availability of functions to manipulate the most common OF structures.
Available apps for network virtualization	Virtual Tenant Network. OpenDOVE.	MAC-based network isolation.	Creation of isolated virtual tenant networks. Implementation of Neutron services.	Supported.	--
Support for external net apps	Via northbound REST APIs exposed by all the services.	External applications are supported through REST APIs.	Northbound REST APIs to configure the system or collect operational data.	Supported.	Network applications must be developed as python modules using POX's APIs. They are launched through POX CLI.
Integration with cloud management platforms	OpenStack integration (through Neutron service).	Integration with Openstack (Quantum).	OpenStack integration.		REST APIs not available.

Table 21 – Comparison of SDN controllers

5 Network Virtualization technologies

Network virtualization constitutes one of the main functionalities of a network control plane and orchestration platform for Data Centre environments, with the objective of deploying multi-tenant virtual networks sharing a common underlying infrastructure and able to support the insertion and composition of L4-L7 service chains. Network virtualization techniques are mostly based on SDN principles, as analyzed below. In particular, section 5.1 is dedicated to industrial products while section 5.2 provides a survey of open-source tools and applications for network virtualization, most of them implemented as SDN application on top of SDN controller platforms.

5.1 Industry network virtualization solutions

Several solutions for network virtualization are available today on the market, from well-known networking vendors but also from some new startups that are leveraging on the novel opportunities offered by the SDN paradigm. An interesting market report comparing the main network virtualization products available today has been recently released by SDN Central [sdn-2013].

As shown in the report, the global trend is clearly oriented to SDN-based network virtualization, following two main approaches:

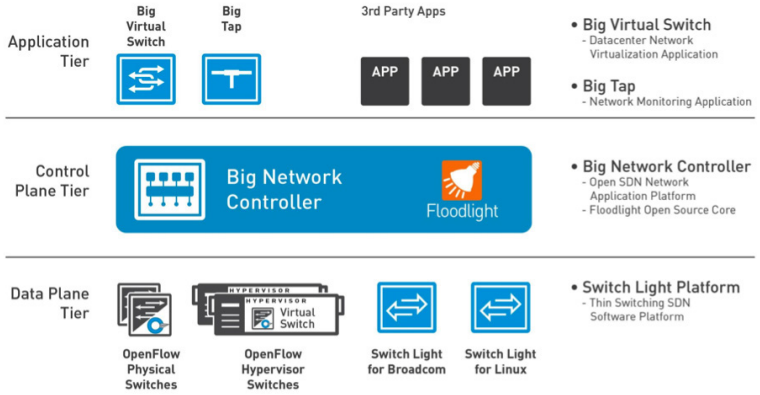
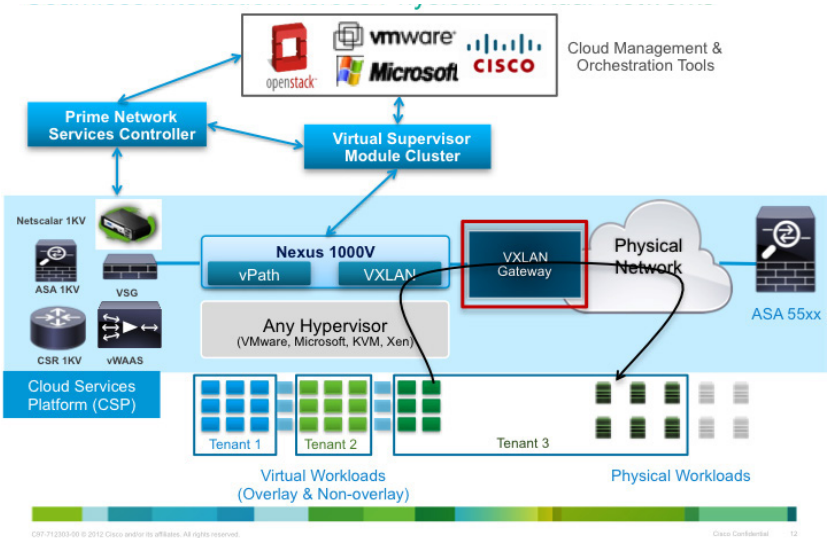
- Overlay-based network virtualization, where the traffic coming from the end-points is encapsulated and tunnelled through the physical networks. Main encapsulation formats include VXLAN, NVGRE and STT.
- Network virtualization based on the direct fabric programming, to create direct paths through the virtual or physical network fabric to compose isolated private networks.

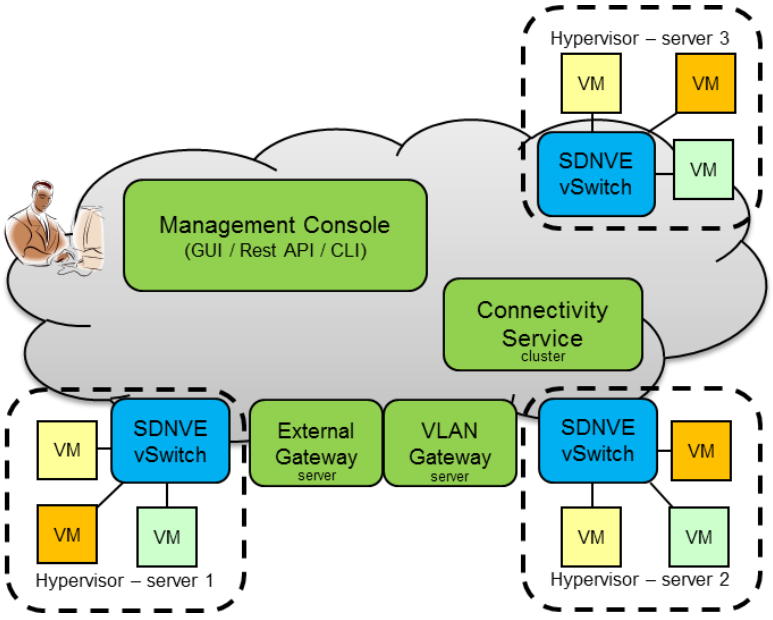
The overlay-based network virtualization is the most popular form of virtualization and, while not strictly requiring an SDN approach, it is often integrated with SDN mechanisms to support the composition of multi-tiered networks and L4-7 service chains. Its main advantage is the possibility to keep the intelligence in the edges of the network. This allows for introducing new services and features more rapidly, without imposing any requirement on the upgrade of existing equipment in the physical network. Most of the vendors adopt the VXLAN data plane technology, together with proprietary solutions to deal with multicast or broadcast services and overcome scalability issues.

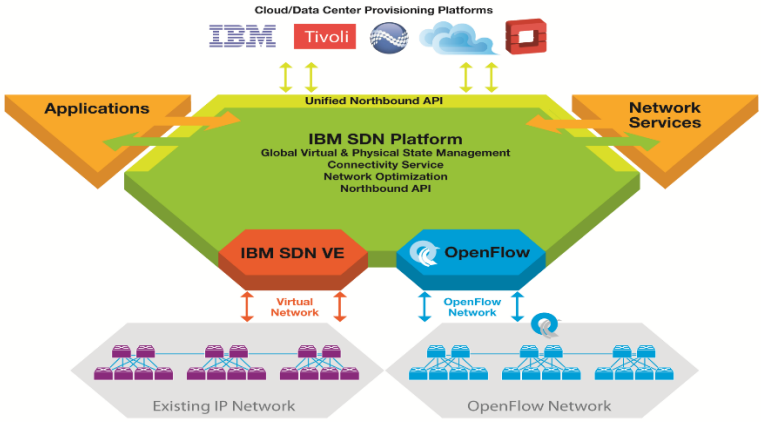
However, the overlay-based virtualization has several great limitations. First, as overlay solutions fully decoupled from the underlying network, they do not allow for advanced features like bandwidth tuning for QoS, fast recovery from link failures, optimization of performance and network utilization. Second, monitoring and troubleshooting are extremely complicated due to the lack of visibility on the underlying network. Third, requirement to encapsulate each and every packet sent over overlay virtual network, increases the latency, decreases the bandwidth and, especially if implemented in SW, overloads the TEP nodes. For these reasons, the research is investigating joint overlay/underlay solutions, debating on how much cooperation, abstraction, and level of awareness is required between the control of the underlying infrastructure and the management of the overlay networks.

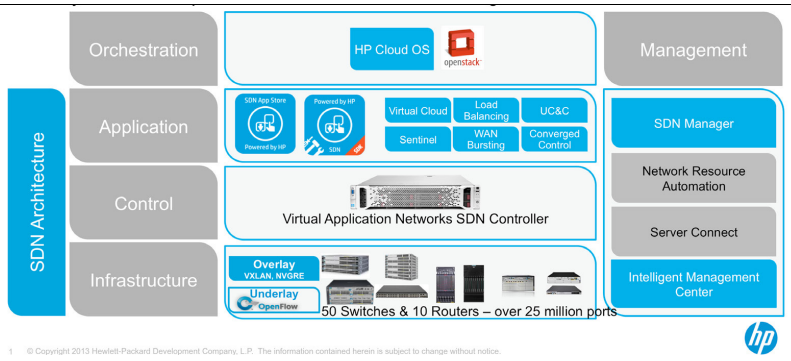
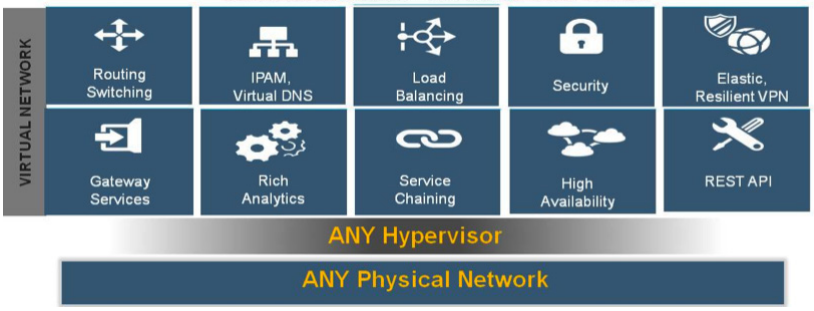
The following table provides an overview of the network virtualization products available in the market.

Vendor	Product	Description
Big Switch	Switch Light Big Network Controller Big Virtual Switch Big Tap	Big Virtual Switch is a network virtualization application based on direct fabric programming through the Big Network Controller. Big Switch solution is suitable for environments deploying multiple OpenFlow-enabled physical and virtual switches and it is of particular interest for multi-tenant, single-site cloud environments. The virtualization application is integrated with OpenStack via REST APIs and supports L4-7 service insertion at the network edges.

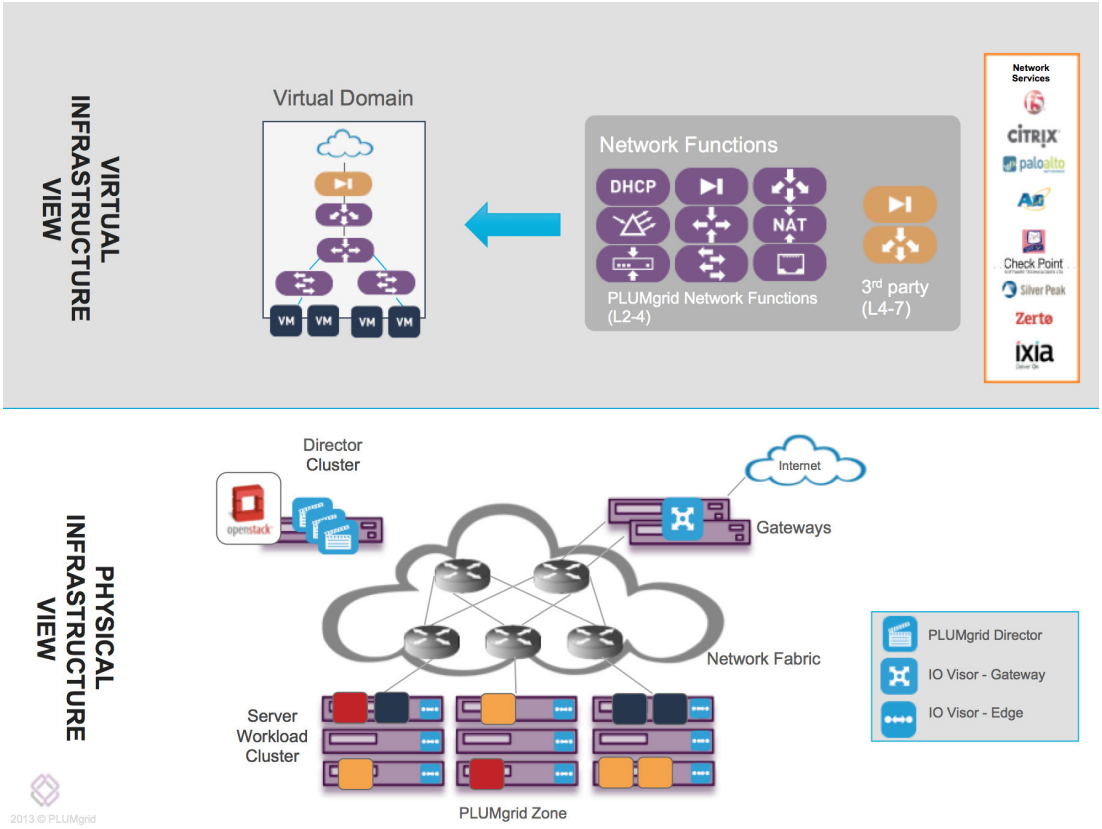
	Floodlight	Unified network monitoring is provided through the Big Tap application. One of the main benefits of Big Switch solution is the centralized provisioning and automation that allows to optimize the network operation. Big Switch has released Floodlight, an open-source (Apache-licensed) centralized controller development platform.  *Big Switch Networks Open SDN Suite*
Cisco	Open Network Environment (ONE) Cisco Nexus 1000V Cisco 1100 Series Cloud Service Platform Cisco Nexus 1000V InterCloud Cisco Application Centric Infrastructure (ACI) – including Cisco Nexus 9000 switches, Cisco Application Policy Infrastructure Controller (APIC) and Cisco Application Virtual Switch (AVS)	Nexus 1000V is a virtual switch that integrates directly with VMware vSphere hypervisors, Microsoft Hyper-V and KVM. The overlay-based virtualization solution adopts VXLANs tunnels, with a proprietary extension to manage broadcast, unknown unicast and multicast traffic. The Cisco vPath 2.0 features allow the Nexus 1000V to optimize the deployment of chains of L4-L7 virtual network services. Moreover, Cisco portfolio includes InterCloud, a cross-data centre solution to build hybrid clouds and extend private data centres to public clouds, with support for workload mobility, end-to-end security, and centralized monitoring across heterogeneous clouds.  *Cisco Cloud Networking and Services* Cisco Application Centric Infrastructure (ACI) provides a policy and management framework for automated infrastructure provisioning following a model based on application policies. The ACI solution includes Cisco Nexus 9000 switches, Application Virtual Switches (AVS) for the virtual network edge and the Cisco Application Policy

		Infrastructure Controller (APIC). APIC is a centralized policy-engine for physical, virtual and cloud infrastructures, which provides features like QoS, high-availability, per-application and per-tenant monitoring and telemetry. It can be integrated with OpenStack through new set of APIs, based on group-based policy abstraction and supporting XML and JSON languages. On the south-bound side, APIC supports the OpFlex protocol, used to supply application policies to compliant forwarding devices.
IBM	SDN VE IBM SDN Platform	Over the last several years IBM has offered several network virtualization products for virtualized environments, e.g. the v5000 distributed virtual switch for VMware clouds and the Programmable Flow Controller coupled with the OpenFlow enabled switch hardware. Today IBM offers the IBM SDN platform and ecosystem, centred on the IBM SDN VE (SDN for Virtualized Environments) solution. SDN VE is a multi-hypervisor virtual network overlay that uses existing (traditional or OpenFlow) IP infrastructure. The solution exists for VMware and for KVM environments and offers the following capabilities: • Keep your existing physical network unchanged • Setup or tear down virtual networks as quickly as virtual machine, reducing configuration time from days to hours • Provision 16,0005 virtual networks with the first release and up to 16 million networks later • Seamlessly connect virtual and nonvirtual networks through gateways SDN VE consists of the following components: the Management Console for virtual networks Provisioning and configuration, the Connectivity Service cluster, the set of vSwitch instances for data plane connectivity and TEP interposition, and the set of gateway products for connecting overlays to the external networks or non-virtualized systems (servers and appliances). SDN VE components are presented in the figure below: 

		IBM SDN platform is the third, integrated edistion of SDN VE product that combines the overlay and the OpenFlow capabilities into the unified network management product capable both of creating and managing virtual overlays and of controlling the physical OpenFlow enabled infrastructure, as shown below:  This most recent release of SDN VE leverages industry standards for solid foundation and is based on the OpenDaylight SDN platform.
Dell	Active Fabric Manager Active Systems Manager	Dell solution supports different virtualization approaches, from the direct fabric programming based on OpenFlow to the overlay-based approach via VXLAN/NVGRE/STT. The Dell Active Fabric Manager provides integration with OpenStack and other orchestration stacks. The Dell Active Fabric architecture follows a modular approach so that it can be easily integrated into pre-existing environments. It can be adopted in hybrid and private cloud scenarios and multi-site data centre deployments, supported through hypervisor tunnelling and inter-DC gateways.
HP	Virtual Application Networks Intelligent Management Centre HP Virtualized Services Router	HP solution is based on a suite of products including an OpenFlow-based controller and many SDN applications for security, communication, and a WAN bursting app for private-public cloud bursting. HP has also announced an open SDN ecosystem, with an SDN Developer Kit and an SDN App Store. The HP Virtual Application Networks framework provides an end-to-end virtualization solution that, in combination with the HP Intelligent Management Centre, integrates policy-based network automation and orchestration. HP's overlay-based virtualization is based on VXLAN as tunnelling protocol (NVGRE planned), while OpenFlow 1.0/1.3 and OVSDB are adopted for data-plane configuration.

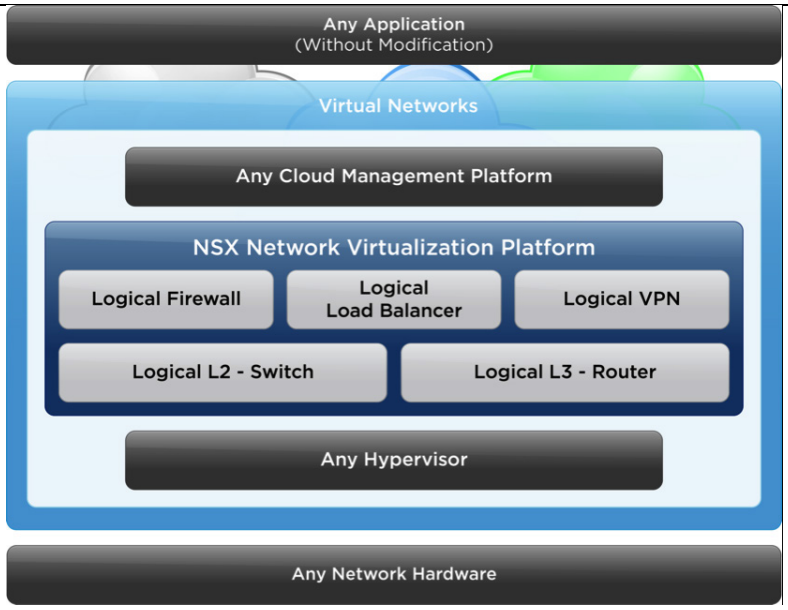
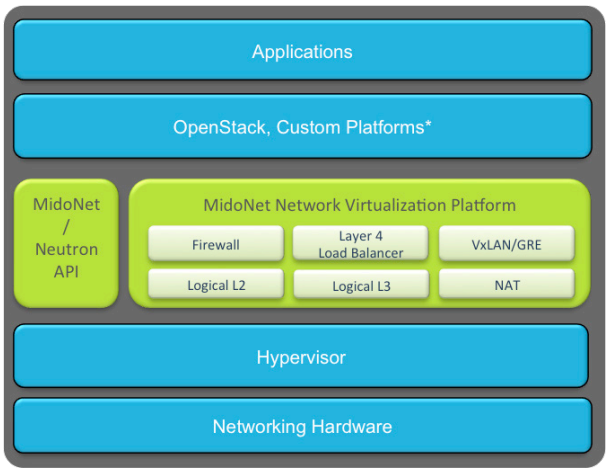
		 *HP Virtual Application Networks*
Juniper	Juniper Networks Contrail: Contrail controller Contrail vRouter	Juniper's solution adopts several encapsulation approaches, including MPLS over GRE, MPLS over UDP, and VXLAN (NVGRE planned). The Contrail platform provides mechanisms for the integration with the WAN, particularly suitable for hybrid clouds and cloud bursting, supports L4-L7 service chaining and includes analytics features for troubleshooting and diagnostics. Interoperability with existing physical networks is guaranteed through the usage of networking protocols like BGP and MPLS. While OpenFlow is available in some Juniper's physical switches and routers, the Contrail controller uses Netconf to configure the network physical devices and XMPP to control vRouters. On the orchestration side, Contrail is integrated with CloudStack and OpenStack via RESTful APIs. Juniper has also released an open-source version of Contrail, called OpenContrail, under Apache 2.0 license.  *Juniper Contrail controller*
NEC	Programmable -Flow Networking Suite	NEC solution is based on direct fabric programming where NEC's OpenFlow-enabled switches are controlled through the Programmable Flow Controller (PFC). NEC's platform supports QoS and failure recovery and it is suitable for multi-tenant environments that require high network flexibility for complex infrastructures. While originally focused on single-site data centers, NEC's solution has evolved with the Unified Network Coordinator, which allows to create and orchestrate virtual networks across multiple controllers within a single data centre and in inter-DC environments. NEC's SDN Application Centre allows to integrate L4-7 services through a RESTful interface.
PLUMgrid	PLUMgrid Platform: PLUMgrid IO Visor	PLUMgrid solution is based on a network virtualization platform that supports the major hypervisors and is based on the VXLAN protocol. The main characteristics are the modularity and scalability of the architecture, and the support for more flexible network abstractions.

	PLUMgrid Director	The PLUMgrid Platform is a distributed software suite that enables the automated and on-demand creation of programmable virtual network infrastructures in a multi-tenant scenario. These virtual networks can be dynamically cloned or migrated, but also updated loading or unloading virtual network functions on-demand.



PLUMgrid Platform

VMware	VMware NSX network virtualization platform	NSX solution is involving a wide ecosystem of actors, from hardware to L4-7 software vendors. This brings benefits for the possibility to integrate several components from physical gateways to service chaining features. The NSX vSwitch-based solution includes L2-7 virtual network services through a north-bound API controlled by the VMware NSX controller cluster. It is suitable for scenarios with strong requirements of network flexibility for automation and re-configuration, but average performance. The virtualization approach is overlay-based and supports GRE, STT and VXLAN as tunnelling protocols. Multi-site deployments are supported through NSX Edge Gateways, while the RESTful NSX API allows the integration with several orchestrators, like VMware-centric vCloud and vCAC orchestration stacks. Dedicated plugins are also available for OpenStack and CloudStack.

		 *VMWare NSX Network Virtualization Platform*
Midokura	MidoNet	MidoNet is a distributed network virtualization software platform integrated with OpenStack. It provides an overlay-based virtualization solution, using VXLAN and GRE as tunnelling protocols. It is based on a distributed architecture with a control plane entity (MidoNet Agent) on each hypervisor host. MidoNet provides L2-L4 services, including L4 load balancing and firewall, while the interaction with orchestrators like OpenStack (or CloudStack) is enabled through RESTful APIs. The MidoNet Control Panel provides monitoring information, troubleshooting tools, and mechanisms for centralized configuration. The MidoNet Gateway enables the interconnection with external networks and provides support for cross-DC communication and hybrid cloud deployments via BGP protocol.  *MidoNet framework*
Plexxi	Plexxi Control Plexxi Switches	Plexxi virtualization solution is based on the concept of affinities to describe applications workload requirements in non-networking terms. Plexxi traffic analysis provides an automatic way to discover the requirements of traffic flows between endpoints. The Plexxi Control, a centralized SDN controller, translates the traffic requirements expressed through affinities into actions that configure the network. This translation is based on internal algorithms to

		compute optimized topologies over the current network status, given the constraints described through the affinities. The Plexxi Control supports RESTful APIs and can be integrated with tools like OpenStack and Chef.
Nuage Networks	Virtualized Services Platform (VSP)	VSP eliminates the constraints that have held back the responsiveness and efficiency of the datacenter network by: making the datacenter network as dynamic & consumable as compute infrastructure through automated instantiation of network services; eliminating cumbersome configuration-driven processes for datacenter networking that are today's norm; separating and simplifying the definition of network service requirements and policies from the manner in which network services are established; seamless connectivity across ANY datacenter network infrastructure (Layer2 through Layer4) incorporating both virtualized and non-virtualized compute environments; scaling to meet the demands of thousands of tenants with unique application requirements and enterprise policies. VSP consists of the following components: • Virtualized Services Controller (VSC) serves as the robust control plane of the datacenter network, maintaining a full per-tenant view of network and service topologies. Through network APIs using interfaces such as Openflow, the VSC programs the datacenter network independent of datacenter networking hardware. • Virtualized Services Directory (VSD) serves as a policy, business logic & analytics engine for the abstract definition of network services. Through RESTful APIs to the VSD, administrators can define and refine service designs and incorporate enterprise policies. • Virtual Routing & Switching (VRS) is a module that serves as a virtual endpoint for network services. Through the VRS, changes in the compute environment are immediately detected, triggering instantaneous policy-based responses in network connectivity to ensure that the needs of applications are met. In addition, 7850 Virtualized Services Gateway (VSG) helps bridging the virtualized environment (overlays) into the rest of the datacentre.

Table 22 – Industry network virtualization solutions

5.2 Open network virtualization solutions

5.2.1 FlowVisor

FlowVisor is the ON.Lab network slicer that acts as a transparent proxy between OpenFlow switches and various guest network operating systems. The current FlowVisor supports network slicing and allows a tenant or an experimenter to control and manage some specific traffic from a subset of end points. This approach enables multiple experimenters to use a physical OpenFlow network without interfering with each other.

Like the virtualization layer on a computer, FlowVisor sits between the underlying physical hardware and the software that controls it (see Figure 5-1). And like an operating system uses an instruction set to

control the underlying hardware, FlowVisor uses the OpenFlow protocol to control the underlying physical network. Open-Flow exposes forwarding control of a switch's packets to a programmable entity, i.e., the OpenFlow controller. FlowVisor hosts multiple guest OpenFlow controllers, one controller per slice, making sure that a controller can observe and control its own slice, while isolating one slice from another (both the datapath traffic belonging to the slice, and the control of the slice).

Generally speaking, OpenFlow provides an abstraction of the networking forwarding path that allows FlowVisor to slice the network with the following main characteristics:

- FlowVisor defines a slice as a set of flows running on a topology of switches.
- FlowVisor sits between each OpenFlow controller and the switches, to make sure that a guest controller can only observe and control the switches it is supposed to.
- FlowVisor partitions the link bandwidth by assigning a minimum data rate to the set of flows that make up a slice.
- FlowVisor partitions the flow-table in each switch by keeping track of which flow-entries belong to each guest controller.

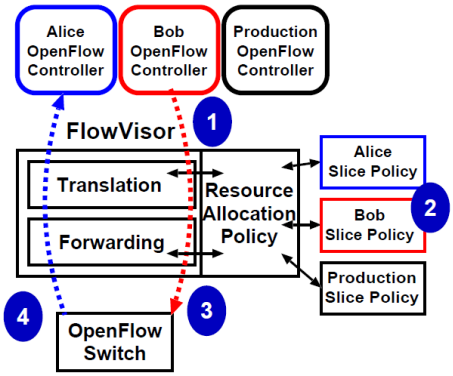


Figure 5-1 The FlowVisor intercepts OpenFlow messages from guest controllers (1) and, using the user's slicing policy (2), transparently rewrites (3) the message to control only a slice of the network. Messages from switches (4) are forwarded only to guests if it matches their slice policy. [flowvisor]

The set of flows that make up a slice can be thought of constituting a well-defined subspace of the entire geometric space of possible packet headers. For example, the current version of OpenFlow supports forwarding rules, called flow entries, that match on any subset of bits in the fields of the packet header (from the physical port the packet arrived on, the MAC addresses, through to the TCP port numbers). Also, a slice can be considered as defined by a set of (possibly non-contiguous) regions, called slice's "flowspace". In general, FlowVisor slices traffic uses flowspaces. Given a packet header (a single "point"), FlowVisor can decide which flowspace contains it, and therefore which slice (or slices) it belongs to. FlowVisor can isolate two slices by making sure their flowspaces don't overlap anywhere in the topology; or it can decide which switches can be used to communicate from one slice to another. It can also allow a packet to belong to two or more slices; for example, if one slice is used to monitor other slices.

5.2.2 OpenNaas

Virtualization techniques exert a very strong impact on network services development and constitute a key driver to for supporting DC Network Hardware Abstraction Layer (Net-HAL). Network service delivery capabilities and new virtualization techniques constitute key drivers to shape innovative service offerings for both the R&E community and the industry based on the Network as a Service (NaaS) concept. NaaS is a business model related to network infrastructure servicing for delivering network services virtually over service providers´ infrastructures and the Internet.
The NaaS model has been brought forward with the OpenNaaS framework. OpenNaaS [OpenNaaS] is the outcome of the cooperation between several stakeholders and was born with the goal of creating an open source project fostered by a lively community of contributors, benefactors and beneficiaries. OpenNaaS was designed and built in the FP7 Mantychore project [MANTY] (2010-2013). It consists

of a robust and extensible open source framework based on the recent evolution of virtualization techniques in networking and infrastructure services that makes network requirements processing easy and provides with scheduled, dynamic and flexible network connectivity services. OpenNaaS framework is currently being used in several European projects to build the prototypes for diverse cutting edge network applications and Cloud services, ranging from SDN QoS [OFERTIE], SDN network service awareness [FI-PPP-XIFI], Open Access Networks control [SODALES], containers for Virtualized Network Functions in NFV, Management as a Service tools and others [Géant3+].

In terms of virtualization, OpenNaaS follows a resource-based modelling approach in which network devices are no longer seen as bare metal with configuration interfaces. Such approach leans on an information modelling framework. OpenNaaS abstracts the underlying network complexity and offers network functions and capabilities as a service enabling the possibility to easily deploy and operate customized advanced network services to all-optical future DC network solutions. Figure 5-2 shows a scheme on OpenNaaS a virtualized based approach based on resources and capabilities, placed on top of physical equipment. In this resource-capability model, a **Resource** models a device, set of devices or system. A Resource can be (among other things) a switch, a router, a link, a logical router or a network, and each resource is composed of one or more **Capabilities**. A capability is an interface to a given resource functionality (e.g. OSPF configuration in router resource). The main benefit of virtualizing and modelling underlying the network in terms of resources and capabilities is the possibility for OpenNaaS to abstract such network to get information from it and trigger configuration actions, on top of it. From OpenNaaS perspective, virtualization is the basic act of decoupling an infrastructure service from the physical assets on which that service operates. Therefore, the proposed model pretends to be flexible enough to procure different designs and orientations, but fixed enough so common tools can be built and reused across plugins.

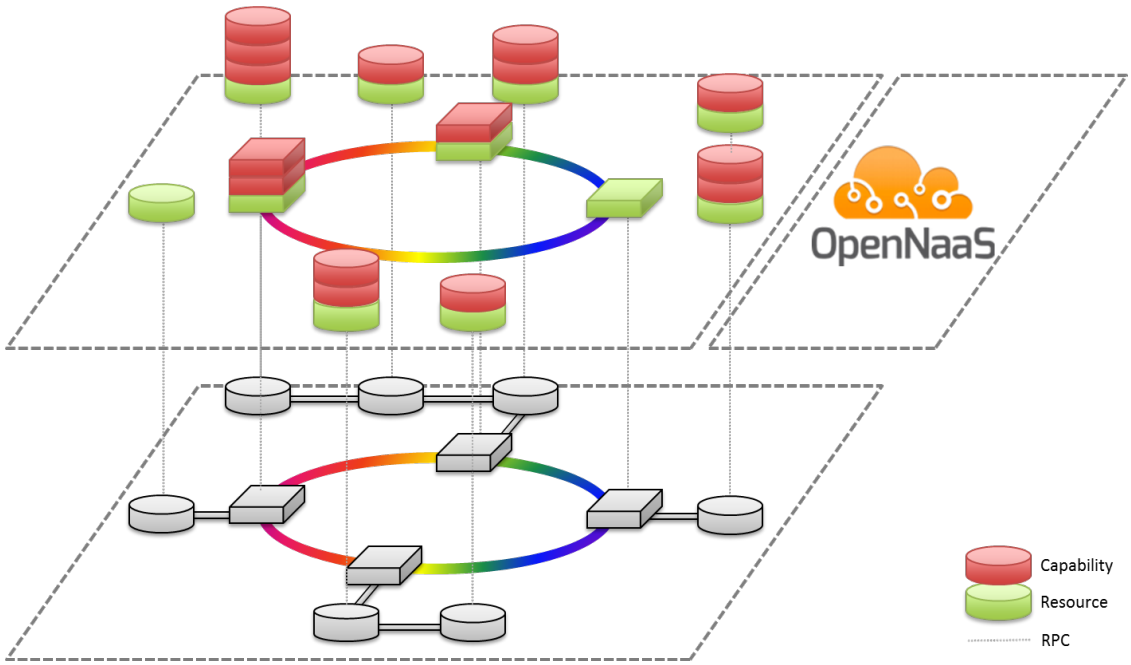


Figure 5-2: OpenNaaS physical device virtualization scheme.

More specifically, OpenNaaS virtualization approach would enable to address the following requirements across different layers of COSIGN architecture:

- At business level:
  - OpenNaaS virtualization approach would enable Network control and functions to be virtualized (CAPEX reduction)
- At management level:

  - OpenNaaS presents a Lightweight Abstracted operational model (HAL) that enables to decouple from actual vendor-specific details and consequently, reduce network management complexity, by decoupling network resource management/control from actual services delivered.
  - In enables the abstraction of management/control functions for logical manipulation in the service stratum.
- At application and service levels:
  - It allows the possibility to provide with abstracted resources, potentially shared for efficient multi-tenancy and in an isolated way.
  - Tailored networks with flexible size for flexible network service composition

Among the suite of OpenNaaS services, a set of them are directly related to OpenNaaS virtualization model which may be of relevance for the design and implementation of future-all optical DC networks and it´s management:

- Virtual network on Demand provisioning.
- Virtualized network functions (NAT, FW, Routing, etc.).
- Console for unified network management.
- Console for Data Centre administration (based on network and Cloud management tools) [IDCMngmnt]
- Virtual CPE service [vCPE]

All in all, OpenNaaS virtualization approach aims to accommodate a lightweight abstracted operational model providing to upper application layers and tenants a significant step towards the vision of "the network is a resource composed of capabilities that can be controlled and managed" balancing solution expressiveness and complexity. The abstraction, resource sharing and isolation principles of the OpenNaaS platform, constitute the base of virtualization for network service delivery, fact which also applies for future all-optical DC network deployments.

5.2.3 DOVE

The open source network virtualization solution, contributed by IBM is called OpenDOVE - Distributed Overlay Virtual Ethernet. Just like in SDN for Virtual Environments (SDN VE) – an industrial solution by IBM presented in previous chapter, the open DOVE architecture is based on a centralized control plane, and is aligned with the Software Defined Networking (SDN) architectural concepts. It is based on overlay networks that achieve an advanced network abstraction, allowing to enrich provider-tenant contracts with network services [OpenDOVE].

DOVE solution allows the consolidation of multiple abstractly defined networks on large scale commodity physical infrastructures, completely delegating network administration and control to their tenants. The major improvement provided by overlay networks is their separation from the underlying infrastructure, and from each other. This separation facilitates independent address spaces, ensures isolation, and allows different virtual networks to be managed by different administrators.

OpenDOVE is a network virtualization platform with a full control plane implementation for Open Daylight and data plane based on Open vSwitch. The building blocks of DOVE virtualization solution are Open DOVE Management Console (ODMC), Open DOVE Gateway, Open DOVE Connectivity Server (ODCS) and Open DOVE virtual switches, as presented in the Figure 5-3.

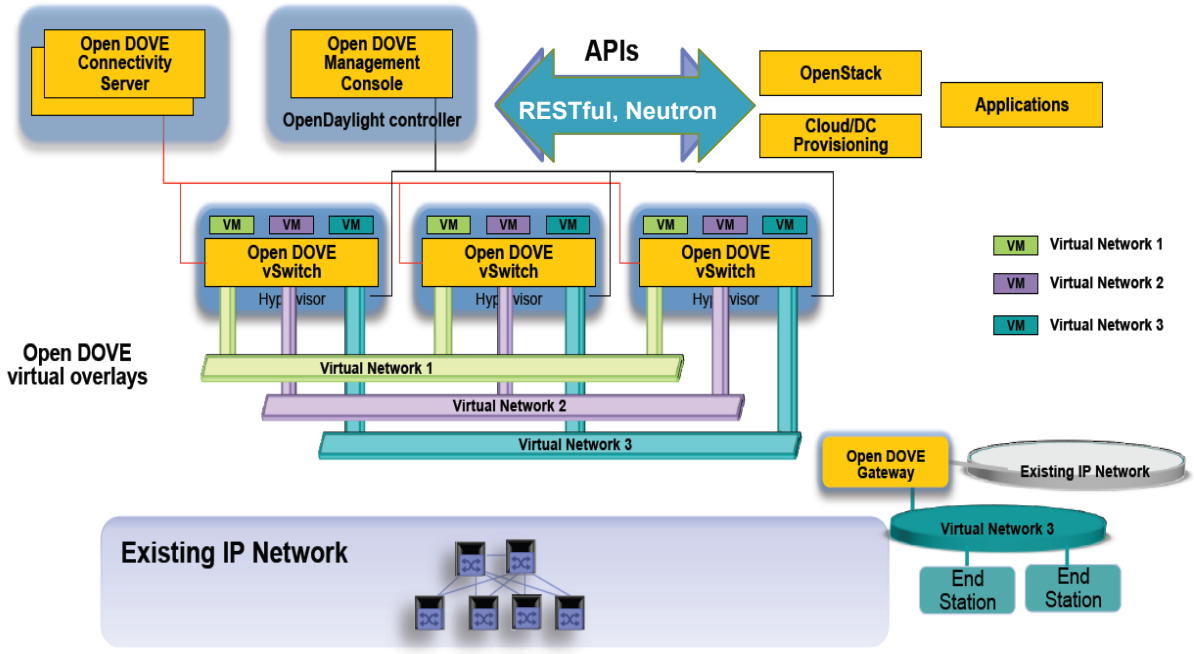


Figure 5-3 : OpenDOVE architecture.

Clients of DOVE networks use REST API to the ODMC to manage programmable virtual networks. The management console is connected to Neutron, which provides network-as-a-service to the OpenStack [OpenDOVES]. ODMC provides user–facing interface for DOVE and central configuration repository. In addition, it allows for the creation of the virtual networks and specification of the network policies. Another important role of ODMC, is configuration of OpenDOVE Gateway, which provides connectivity between virtualized DOVE network and external, non-DOVE networks, as shown in the Figure 11.

Control plane implementation of the OpenDOVE includes, but not limited to addressing and mobility management, as well as network policies management.

DOVE Connectivity Service (ODCS) is a directory-based service that provides address dissemination and network policy information to OpenDOVE vSwitches. ODCS has built in support for clustering, scale-out and also high-availability.

At the data plane, OpenDOVE vSwitches turn to the ODCS in order to retrieve network policy information and address resolution to the communicating endpoints hosted by the switches. DOVE switches leverage OVS native encapsulation/tunneling support and use the data retrieved from ODCS in order to encapsulate client's traffic and build overlay networks that span virtualized hosts in the data center. These overlay networks actually implement client's virtual networks presented in the Figure 11. Also vSwitches are responsible for network policies enforcement inside and between virtual networks.

5.2.4 Virtual Tenant Network (VTN)

The Virtual Tenant Network (VTN) is an OpenDaylight application to create and operate multi-tenant virtual networks over a physical network controlled by OpenDaylight. The VTN application is part of the OpenDaylight Hydrogen release and it is included in its Virtualisation Edition.

The VTN application is based on a logical abstraction plane that allows the users to request the desired network following an abstracted model to describe generic L2/L3 networks, without any need to consider the complexity of the underlying physical infrastructure. The specified VTN is then automatically mapped into the physical network, which is configured through the south-bound protocols. The VTN information models is based on elements representing virtual nodes, interfaces or links, as specified in Table 2.

Element		Description
Virtual node	vBridge	Logical representation of a L2 switch function
	vRouter	Logical representation of a router function, supporting routing, ARP learning and DHCP relay agent functions
	vTep	Logical representation of a Tunnel End Point (TEP)
	vTunnel	Logical representation of a Tunnel
	vBypass	Logical representation of connectivity between controlled networks, through an external domain not controlled by SDN controllers.
Virtual interface	interface	Representation of end point on the virtual node
Virtual link	vLink	Logical representation of L1 connectivity between virtual interfaces

Table 23 – VTN elements

The application provides REST APIs for the provisioning of virtual networks, allowing Create, Read, Update, Delete (CRUD) actions on the elements composing a VTN, in the planning and operation phases. VTNs are programmable and manageable entities, where the tenant can maintain a certain control on the traffic flows running over them. In particular, the VTN application APIs provide mechanisms to apply flow filters to any virtual node interface for forwarding, dropping, re-directing or re-marking packets and to enforce some policing for QoS control. Moreover, monitoring information for traffic statistics and failure events can be collected.

The VTN application is implemented in two main components, as shown in Figure 5-4:

- VTN Manager: a plugin of the OpenDaylight controller which implements the elements specified in the VTN model (e.g. vBridge, vRouter, interface, …). It provides REST APIs on its north-bound interface to create and configure these elements in the controller.
- VTN Coordinator: an external application which exposes REST APIs towards the users or other applications (e.g. an orchestrator, or OpenStack) to invoke VTN services. It handles the VTN requests and interacts with the VTN Manager instance(s) to implement the VTN configuration on the underlying physical network.

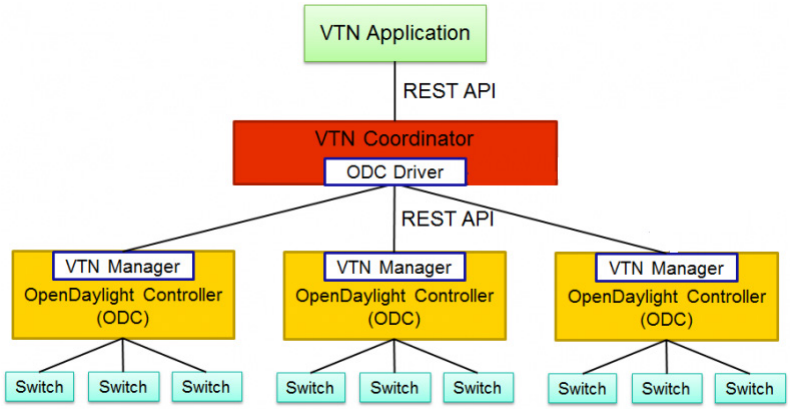


Figure 5-4 VTN architecture

It should be noted that the VTN Coordinator is able to orchestrate multiple SDN controllers, so that the VTN abstraction can be actually mapped across network domains handled by different controllers, guaranteeing a certain degree of scalability. This solution is particularly suitable for multi data centre environments where cloud services span among different DCs, each of them deploying separated SDN controller instances.

5.2.5 OpenVirteX

OVX is a network hypervisor that can create multiple virtual and programmable networks on top of a single physical infrastructure. Each tenant can use the full addressing space, specify their own topology, and deploy the network OS of their choice. Networks are reconfigurable at run-time, and OVX can automatically recover from physical network failures.

OVX implements network virtualization as a proxy that sits between the network and the tenants' network OSes. For each tenant, OVX rewrites OpenFlow messages to translate between what a tenant's NOS sends and receives from its virtual network, and what the infrastructure should receive in order to produce the behaviour consistent with the network seen by the tenant. This approach has two consequences:

- it allows OVX to present programmable virtual networks that support OpenFlow, so a tenant can control the virtual network with its own NOS, and
- it makes OVX transparent – from the network, it appears to be a controller, and from the tenants, as a collection of OpenFlow-capable switches in their network

As part of this function, OVX acts as a (de)multiplexer for OpenFlow messages that flow between the tenant control planes and the infrastructure. Notably, this process relies on the assumption that each network host belongs to just one virtual network. Host hardware addresses play a key role in associating tenants with their network traffic.

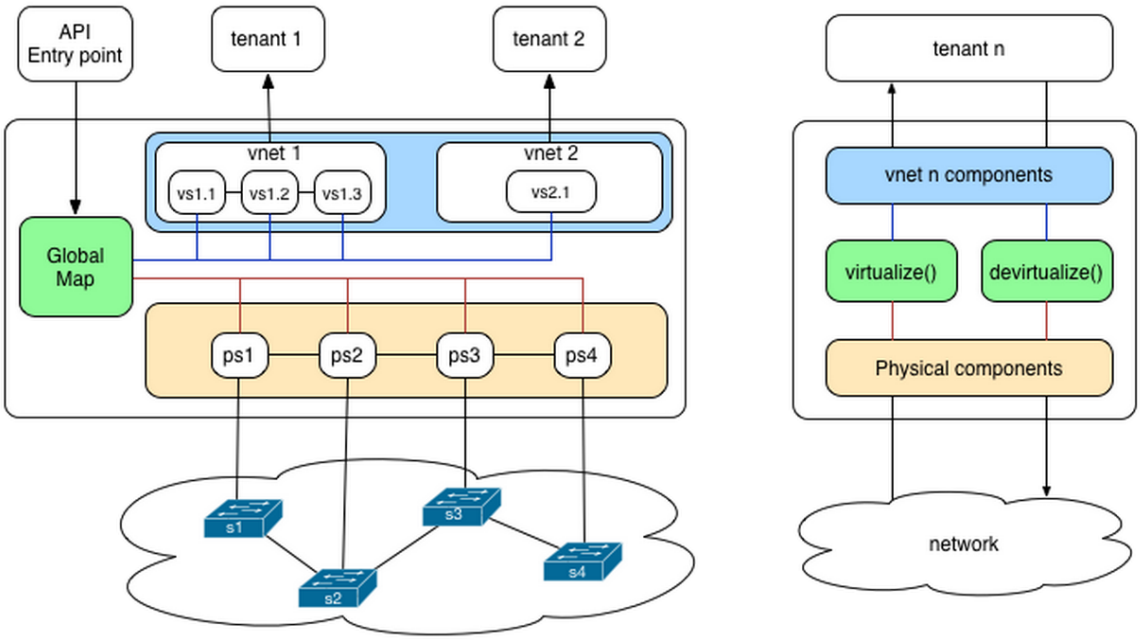


Figure 5-5 OpenVirteX : A High-level View [ovx]

Within OVX (see Figure 13 left), the representation of the physical network (yellow) and the tenant networks (blue) are logically separate. Tenants 1 and 2 can only 'see' the virtualized topology presented to them by OVX, labelled vnet 1 and vnet 2, respectively. The topologies of vnet 1 and 2, and the mapping between them and the physical network, are configured via API calls and stored in a global map (green). The contents of the map can be accessed from both physical and virtual networks (orange and blue lines). Right) The physical-virtual split is crossed by the OpenFlow message virtualization/de-virtualization process, which references the global map.

5.2.6 Neutron

Neutron is a networking project in the OpenStack cloud management software. Neutron was created in October 2011 as an incubation project in Diablo release of OpenStack. The main goal was to allow for programmatic management of the virtual networks in the multi-tenant environments. Before Neutron, networking in OpenStack was statically configured and there was no way to create, delete, or modify networks as tenants requirements evolve. Instead, network configuration was created per OpenStack

installation, by the infrastructure owner, providing the same network service for all the tenants and all the workloads irrespectively of their diverse needs. In addition, only the simplest networking infrastructure was supported by OpenStack networking then – flat L2 domain or VLANs in the physical network and LinuxBridge in the hypervisor. OpenStack community has specified the objectives for the new networking project as follows: REST API for programmatic management of virtual networks by the cloud administrator and by the cloud tenants, pluggable architecture to allow for diverse implementations, extensible design to allow for feature addition and evolution. The first implementation was created by merging network virtualization proposals by several partners – Rackspace, Citrix, Nicira, Cisco, NTT, Midocura. It has exposed the first version of network virtualization API and included two plugins – LinuxBridge and OpenvSwitch (OVS). Since then, rapid evolution has occurred along several dimensions – the initial API was replaced by a more elaborated one, advanced APIs for extended features were added, new plugins have been released to support diverse back-end solutions.

Today, Neutron is a de-facto standard for network virtualization in cloud environments. Neutron virtual network management API represents the common denominator around which the industry converges. Most network virtualization solutions in existence today have been integrated with Neutron through one or more plugins, most of which are part of the OpenStack codebase while some are available from their owners. New solutions continuously diversify the Neutron codebase and APIs, calling for changes in its software architecture and in functionality it exposes. In addition, community is working hard on stabilizing the code and on adding non-functional features like high availability, fault tolerance, scalability, etc.

At the moment, most large scale OpenStack cloud installations are still using rudimentary networking available through the pre-Neutron OpenStack networking service, nova-network. This happens due to the perceived lack of stability, difficulty in choosing the right set of plugins and configuring them correctly, and a very initial high availability and scalability support. As Neutron evolves, experience is gained by the community and best practices are created and distilled by users and integrators. It can be foreseen, therefore, that more and more deployments will switch to Neutron, to enjoy a more advanced network virtualization features it includes.

6 CONCLUSIONS

This deliverable has been focused on the analysis of COSIGN control plane requirements, standardization achievements in the area of SDN architectures and protocols, as well as existing open source solutions for SDN controller platforms and industrial or open products for network virtualization. This study has allowed to define the major functions for the COSIGN SDN controller and to identify the layers that will compose the overall COSIGN network control plane, in compliance with the SDN architecture definition from the ONF standards.

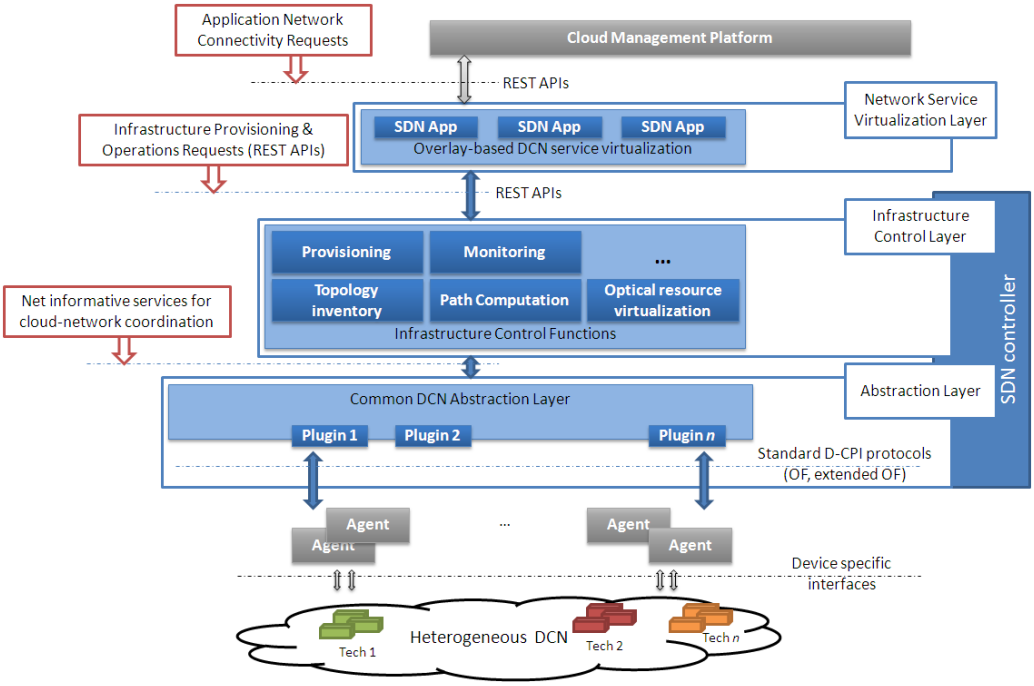


Figure 6-1 COSIGN Network Control Plane: architecture and functions

The architectural decomposition of the COSIGN Network Control Plane results in a three-layered approach (see *Figure 6-1*). Following a bottom-up direction, the Abstraction Layer provides a protocol independent view of the DCN exposing APIs to the internal services of the SDN controller that operates at the Infrastructure Control Layer. These APIs are used for programming the multi-technology COSIGN data plane following a unified approach.

On top of this, SDN applications specialized for overlay network virtualization are placed in the Network Service Virtualization Layer. These applications implement connectivity service virtualization offering advanced application-based Network-as-a-Service functions via REST APIs to a Cloud Management Platform in charge of controlling the entire DC. The abstraction layer is a concept implemented to some degrees by most of the SDN controller platforms analysed in this deliverable. The most promising approach seems the Service Abstraction Layer (SAL) of the OpenDaylight platform, which can load several plugins to support different south-bound protocols and data plane technologies while exposing generalized but customizable models towards the controller's internal functions. This option can be adopted in COSIGN control plane through the definition of an abstract model for optical DCN devices and its binding with dedicated OpenFlow plugins which extend the OpenFlow protocol to support the optical capabilities of the COSIGN data plane.

At the upper layers, the analysis of challenges and limitations in current network virtualization solutions highlights that network virtualization based on the overlay approach (implemented at the Network Service Virtualization Layer) and virtualization techniques based on direct fabric programming (as implemented in the Infrastructure Control Layer) must cooperate tightly. For this reason, the COSIGN architecture will define a powerful interface between these two layers, allowing

to exploit a mutual knowledge of applications and overlay traffic dynamics on one side and infrastructure capabilities and performance on the other side. Moreover, cross-layer awareness between network and cloud domains is a fundamental aspect to achieve more efficient performance in the utilization of the overall DC infrastructure and support the composition and provision of QoS enabled multi-tenant networks combined with chains of virtual L4-L7 services. For this reason, the network control plane, through the abstraction layer, will expose a suitable subset of information about infrastructure topology and performance through informative services that can be consumed by external applications, including the cloud management system and its orchestrator in particular. Finally, the application-driven policy model may be adopted to automate the deployment of virtual applications through network policies that express general requirements for connectivity and networking services at the north-bound API and are enforced at the SDN controller level.

These considerations about the COSIGN network control plane will be further developed in WP3, which will release a first design of its architecture in deliverable D3.1, defining its main services and functions, layer components and interfaces. The survey of SDN controller platform constitutes an additional input for WP3, which will further analyse these controllers from a software point of view to take a final decision about the reference platform for COSIGN developments.

7 REFERENCES

[C-ACI] Cisco Application Centric Infrastructure (ACI), web-site: http://www.cisco.com/c/en/us/solutions/data-center-virtualization/application-centric-infrastructure/index.html

[CloudStack] CloudStack web-page: http://cloudstack.apache.org/

[contr-comp] Rahamatullah Khondoker, Adel Zaalouk, Ronald Marx, Kpatcha Bayarou, Feature-based Comparison of Software Defined Networking (SDN) Controllers, ICCSA' 2014, Sousse, Tunisia, 17 - 19 January 2014

[contr-perf] Alexander Shalimov, Dmitry Zuikov, Daria Zimarina, Vasily Pashkov, and Ruslan Smeliansky. 2013. Advanced study of SDN/OpenFlow controllers. In Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR '13)

[COSIGN-D11] COSIGN, Deliverable D1.1, "Requirements for Next Generation Intra-Data Centre Networks Design", June 2014

[ETSI] http://www.etsi.org/technologies-clusters/technologies/nfv

[FI-PPP-XIFI] XIFI project, web-page: https://www.fi-xifi.eu/about-xifi/what-is-xifi.html

[flowvisor] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, "FlowVisor: A Network Virtualization Layer", October 2009, http://openflowswitch.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf

[IDCMngmnt] J.I. Aznar, Jara, M.; Rosello, A.; Wilson, D.; Figuerola, S." OpenNaaS Based Management Solution for Inter-data Centers Connectivity, "IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), December, 2013.

[Géant3+] Géant3+ web-page: http://www.geant.net/Pages/default.aspx

[JSON-RPC] "JSON-RPC Specification, Version 1.0", http://json-rpc.org/wiki/specification

[MANTY] Mantychore project, web-page: http://www.mantychore.eu/

[NFV] NFV white paper, "Network Functions Virtualisation. An introduction, benefits, enablers, challenges and call for actions. Issue 1", October 2012, available at http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[NFV-af] ETSI GS NFV 002 v1.1.1, "Network Functions Virtualisation (NFV); Architectural Framework", October 2013

[OF] Open Networking Foundation, "OpenFlow Switch Specification", version 1.3.4, March 2014

[OFC] Open Networking Foundation, "OF-Config 1.2 – OpenFlow Management and Configuration protocol", 2014

[OFERTIE] FP7 OFERTIE project, web-page: http://www.ofertie.org

[OF-NDM] Open Networking Foundation, "OpenFlow Controller/Switch NDM Synchronization – v1.0", August 2014

[ONF] Open Networking Foundation web-site: https://www.opennetworking.org

[OpenDOVE] L. Lewin-Eytan, K. Barabash, R. Cohen, V. Jain, "A Levin Designing modular overlay solutions for network virtualization", IBM Technical Paper, 2012

[OpenDOVES] R. Cohen, K. Barabash, L. Schour, "Distributed Overlay Virtual Ethernet (DOVE) integration with Openstack", Integrated Network Management (IM 2013)

[OpenNaaS] OpenNaaS web-page: http://opennaas.org/

[OpenStack] OpenStack web-page: http://www.openstack.org/

[OPFLEX] M. Smith, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, N. Weidenbacher, "OpFlex Control Protocol", IETF draft, April 2014, https://tools.ietf.org/html/draft-smith-opflex-00

[OVS] OpenVSwitch web-site: http://openvswitch.org

[OVSDB] B. Pfaff, B. Davie, "The Open vSwitch Database Management Protocol", RFC 7047, December 2013

[OVSDB-S] Open_vSwitch database schema, available at http://openvswitch.org/ovs-vswitchd.conf.db.5.pdf

[ovx] OpenVirtex web-page: http://ovx.onlab.us/

[RFC6241] R. Enns et al. RFC6241, Network Configuration Protocol, June 2011

[sdn-2013] SDN Central, "Market Report. Network Virtualization Solutions. An analysis of solutions, use cases and vendor and product profiles", October 2013.

[sdn-course] Nick Feamster's Coursera, "Course on SDN", web site: https://www.coursera.org/course/sdn/

[SDN-OF] "Software-Defined Networking (SDN): The New Norm for Networks". Open Networking Foundation

[SODALES] FP7 SODALES project, web-page: http://www.fp7-sodales.eu/

[vCPE] Minoves, P.,; Frendved, O. ; Bo Peng ; Mackarel, A. ; Wilson, D., "Virtual CPE: Enhancing CPE's deployment and operations through virtualization, " in CLOUDCOM 2013 - IEEE fourth International Conference on Cloud Computing Technology and Science, pp.687-692, DOI 10.1109/CloudCom.2012.6427560