Project Report (MINT 709)

Enterprise Cord

Submitted to University of Alberta in fulfillment of Degree in Master of Science in Internetworking

Submitted By:

Harsh Bassi

Under the Guidance of

Mr. Juned Noonari

Table of Content

Acknowledgment

I would like to acknowledge my gratitude and sincere thanks to my project mentor **Mr. Juned Noonari,** who guided me to successful completion of this project. His guidance and expert advice have been invaluable throughout all the stages of my project. Thank you for all the ideas and patience in guiding me to complete this project.

I take this opportunity to thanks Dr. **Mike MacGregor** and Mr. **Shahnawaz Mir** who have been supportive throughout the course and always been supportive of my career goals.

I would also like to thank my parents and friends, for their encouragement and support. Further, I also want to special thanks to all the seniors who been supportive throughout the course and always been helpful.

Enterprise CORD (E-CORD)

Abstract:

Enterprise CORD is one of the CORD use cases which provide enterprise connectivity services in wide area networks and over metro services, with the help of open source software and inbuilt commodity hardware. It is an extension of the CORD which is the combination of NFV, SDN and the cloud and it helps in bringing datacenter economies and cloud agility to the Telco central office. Their goal is to build network infrastructure with the use of few commodity building blocks using open source software and white boxes, in which the software platforms will easily create all kind of new services for the clients.

E-CORD (or ECORD) is based on CORD infrastructure to support enterprise customers and enables service providers to provide enterprise connectivity services (L2 and L3VPN).Furthermore, enterprise customers can opt for any kind of service on demand such as firewalls, WAN accelerators, traffic analytic tools, virtual routers, etc. that are installed and maintained in the service provider network.

The purpose of doing this project is to study and explain about ECORD, and research progress work of ECORD technology which is the trial mode. ChinaMobile has also implemented ECORD services in their labs. In this project report, I explain all the aspects of NFV, SDN, Cloud and all other services to create and implement ECORD. Project also explains both ECORD software architecture (OpenStack, docker, ONOS, XOS, and Atrium) and hardware architecture.

Chapter1

CORD (Central Office Re-architected as a Datacenter):

Cord will help reinvent Network edge, and it is believed that nearly 40% of all end customer traffic will use services provided by the cord till mid-2021. Cord re-architects is helping to make the central office as a data center. CORD is a reference implementation for network operators of a service delivery platform. It is a fully integrated system–sufficient for field testing–made of open source components. Being an integrated system is an important aspect of CORD, as our objective is to keep the project higher than a collection of related projects listed under a common banner. Another essential aspect of CORD is being open. This does not mean that all modules configured in CORD need to be open- source(example. proprietary services can run on the platform), but it is important that both the CORD platform be open- source and the CORD as a whole have sufficient open-source services to demonstrate its full scope.

Besides being open and integrated, CORD is designed to meet the next five requirements. The first three directly follow the objective of bringing cloud economies and agility into the Telco environment and the last two concerns the use of best system design practices in the cord.

A.CORD runs on servers and white box switches which leverage merchant silicon to the extent possible; high performance should not be dependent on proprietary or purpose-built hardware. The fact that CORD uses commodity hardware directly results from its objective of supporting the economy of cloud infrastructure. And the software running on this commodity hardware must implicitly deliver the same performance and reliability as the intended hardware of today.

B. In particular, CORD must support **services** from four dimensions:
(1) both access services (example. Fibre-to-the-Home) and conventional cloud services (SaaS).
(2) Both data plane (NFV) and control plane (SDN) services.
(3) Both trusted operator-provided and untrusted third-party services.
(4) Legacy and disaggregated Greenfield services bundled together.
In addition, CORD must provide an agile framework for the management of a range of services, regardless of how individual services achieve isolation, scale, performance and high access.

C. Extensible and Controllable: CORD is a platform that can be configured; it's not a point solution. It must provide the operator with a means to specify the desired service collection and the full OAM capacity for these services. This includes CORD configuration for various markets and access technologies: residential, enterprise and mobile. It must also provide a means to provide and parameterize these services in order to achieve the operator's business and operational objectives.

D. Multi-Domain Security: CORD has to account for multiple trust domains; it is not enough to distinguish between CORD operators and CORD users only. This includes a wide range of intermediary roles, including global operators, site-specific operators, service providers, service developers, service providers and service subscribers (other internal services).
E. Focus on Operational Robustness: CORD must account for partial and intermediate failures and incremental upgrades; it is not acceptable to ignore the operational realities of system building by integrating multiple software components that have been independently developed and deployed. CORD must be set to zero- touch: to account for the possibility that the system's operational behavior is not always synchronized with the system's target state and to automatically lead the system to the correct operating state.

1.1 Project Landscape

Following is the hierarchy of CORD:

• **CORD Vision** – The project's shared goal is to promote the construction of network edge facilities from cloud hardware and software technologies.
• **CORD Implementation** -An integrated collection of hardware and software components that implement CORD Architecture.
• **CORD Solution** - A specific CORD Implementation configuration / customization targeted at a specific use case. In today's Reference Implementation, several examples of solutions are available, each centered on a different access technology (example. R-CORD, M-CORD, E-CORD), but this is a short - term situation. A much richer collection of solutions is expected in the near future, including those supporting multiple access technologies (A-CORD).
•**CORD Platform -** CORD Implementation subset common to all CORD Solutions. An architectural definition of a CORD Platform can also be given in terms of CORD Architecture's models and interfaces.
•**Cord distributions:** There is one distribution today it's exactly the same as the Reference Implementation and includes both the Reference Platform and a collection of Reference Solutions but multiple distributions are possible, and they don't need to be excluded from each other. For example, various organizations could prepare distributions that include the same Reference Platform, but different solutions combinations.
Alignment with a common architecture enhances the ability to create interoperable components and alignment with the Reference Implementation enhances the ability to build on the open source components of each other.

1.2 CORD Vision

Here now the focus on the CORD Architecture and Reference Implementation, A large community is working towards the common goal of bringing cloud technologies to the edge of operator networks (example. Telco Central Offices). Efforts in this larger ecosystem align with the CORD Vision as long as they comply with the Architectural Requirements of CORD. Some efforts may be more focused and do not address all the requirements, but five principles are the basic requirement as follows:

1. Is built around commodity servers and software controlled switches, and controlled software switches and leverages merchant silicon to the extent possible.
2. Enables disaggregation, and is not restricted to virtual machines running bundled legacy VNFs.
3. Leverages SDN and can run fully programmable control applications, connecting virtual and physical elements as well as providing services.
4. Is extensible by design, with a common core (platform) that can be configured to include different combinations of access technologies and services.
   It adopts best practices in the construction, composition, and operation of multi-tenant cloud services that are scalable.

1.3 Reference Architecture of the CORD –

The CORD Reference Architecture is based on a set of requirements and meets all the principles underlying the CORD Vision in particular. The architecture is now specified by the public API for the implementation of the CORD reference. The architecture is now specified by the public API for the of the CORD reference implementation. Prescribe structure on implementation because they codify invariants about CORD's internal behavior (i.e., they define abstract building block components, impose interfaces on those components, and establish relationships between those components). For example, a **Service Graph** defines Solutions; a Service Graph consists of a set of Services and Dependencies between Services; a Service consists of a Controller and one or more Slices; a Slice is a logical resource container that includes a set of Instances and a set of Networks, etc.

1.4 Reference Implementation

The CORD Reference Implementation is a CORD Reference Architecture realization packaged as a combination of hardware and software approved by the **CORD Technical Steering Team (TST**) for release. This includes a material for validated hardware, instructions for how to install this hardware into a **CORD POD**, and software installation processes that build a working configuration and deploys it on the target hardware.

In the cord reference implementation following points are as follows:

- *Package of services Profiles* – A software specification for the service package to be loaded to the platform.
- *Hardware POD* –It consists of hardware configurations and wiring diagrams for the target hardware that is in use.

CORD Reference Distributions – A packaging of the CORD Reference Platform and a collection of CORD Reference Solutions approved by the CORD TST.

- *CORD Reference Solutions* – Combining the Official Service Profile with the target Hardware POD tested and included in the CORD Reference Implementation release.
- *CORD Reference Services* – These Services are included in the Service Profile for one or more levels of the CORD Reference Solutions.
- *CORD Reference Hardware* – Any hardware device included in a Hardware POD for one or more of CORD Reference Solutions.

1.4.1 Challenges

Network operators face major challenges that support ever-increasing demands for bandwidth and ever-increasing expectations for service. For example, in the last eight years, AT&T has seen data traffic increase by 100,000 percent and plans to roll out ultra-fast fiber, and access 100 cities across the US[1]are now underway. With the facing of these challenges, network operators are looking for ways to benefit from both the economies of scale (infrastructure built from a few commodity building blocks) and the agility that commodity cloud providers enjoy today (the ability to deploy and elastically scale services quickly).
Cloud economies and agility are particularly needed at the edge of the operator network — in the Telco Central Office (CO)—which includes a diverse collection of purposes — built devices, assembled over 50 years, with little coherent or unifying architecture.
Here I'm trying to explain the CORD, a Telco Central Office architecture that combines Network Functions Virtualization (NFV), Software - Defined Networking (SDN), and cloud services — all running on commodity hardware — to build cost-effective, agile networks with significantly lower CAPEX / OPEX and enable quick service creation and monetization.

1.5 Introduction to CORD:

Following three technologies in the approach for the cord technology:

SDN: SDN is about separating the network control and data planes and in this control plane can be programmable, and that help us to do more innovation. It also allows for the simplification of the forwarding devices that can be built using merchant silicon, resulting in less expensive white box switches.

NFV: This helps it to reduce the cost of CAPEX and OPEX by moving the data plane from the hardware devices to the virtual machines. It also improves operator agility and provides us the opportunity for innovation.

CLOUD: It helps to build scalable services- leveraging software-based solutions, microservice architecture, virtualized commodity platforms, elastic scaling, and service composition, to enable network operators to innovate rapidly.

These three technologies help us to reduce cost, and it also provides us sources of innovative services that telcos can offer the subscribers. These include all the control plane services (example virtual networks, cloud network, data plane services (example, Parental Control, NAT, WAN Acceleration), and global cloud services (examples, CDN, Storage, Analytics, Internet of Things).

The purpose is to not only built hardware devices but also make the central office an integral part of every Telco larger cloud strategy, enabling them to offer for valuable services. CORD architecture will also support a wide range of services. This includes both cloud services (SaaS) and access services (e.g., Fibre to the home) , services implemented in the data plane (NFV) and implemented in the control plane (SDN) , trusted operator provided services and untrusted third party services, bundled legacy services and disaggregated Greenfield services.

1.5.1 Commodity Hardware:

The CORD target hardware consists of a collection of commodity servers connected to a fabric made of white box switches. The illustrative example shown in the figure shows two important hardware configuration features:
-The fabric switches are placed on the leaf-spine topology to optimize for traffic flowing east-to-west between the access networks that connect the central office to the operator backbone. There is no conventional north-south traffic.
-The rack of GPON OLT MACS commoditized connectivity to the access network. They replace proprietary and closed hardware with an open, software-defined solution that connects millions of subscribers to the Internet.

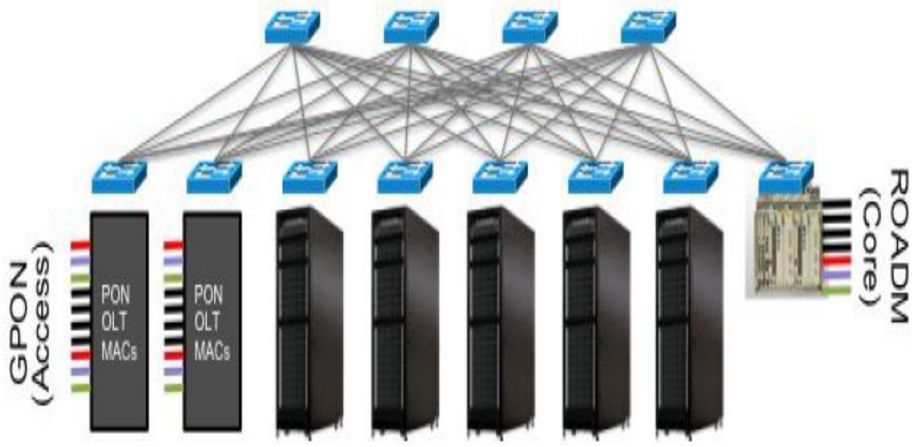


Figure 1: Hardware Built From Commodity Server, I/O blades and switches

A reference system for the cord that includes specific choices for all hardware elements organized into a rackable unit called a POD. The CORD POD reference consists of:

1. The servers are configured with 128GB of RAM, 2x300GB HDDs, and a 40GE dual port NICs. (STRATOS-S210-X12RS-IU Servers)

2. These switches are configured with 32x40GE ports(Accton 6712 switches). These switches function in the CORD fabric as both leaf and spine switches.
3. I/O blades that include the PMC Sierra OLT MAC chip. This is a predefined 1RU-blade that contains 48x1 Gbps GPON interfaces and 2x40GE uplinks.
These servers run on Ubuntu LTS 14.04 and include the Open vSwitch (OVS). The switches run the Atrium [2] software stack, which includes Open Network Linux, the Indigo OpenFlow Agent (OF 1.3), and the OpenFlow Data Plane Abstraction (OFDPA), layered on top of Broadcom merchant silicon.

1.5.2CORD Software Building Blocks

Open source software that includes Docker, OpenStack, XOS, ONOS, and. The following are the explanation of these software:

1. OpenStack [3]**:** It is a cluster management suite that provides IaaS core capability and is responsible for virtual machines (VMs) and virtual networks (VNs) creation and supply.
2. Docker [4]**:** Provides container-based means for interconnecting and deploying providers. It has a very crucial part in the deployment of CORD itself (for example, the other elements- XOS, OpenStack and ONOS) - are installed on the PODhead nodes in Docker containers.
3. ONOS [5]: The underlying whitebox switching fabric is the network operating system. It hosts a collection of control applications that implement services on behalf of Telecom subscribers and is responsible for integrating virtual networks into the underlying fabric accessed through OpenStack's Neutron API.
4. XOS [6]**:** It is used for composing and assembling services. It unifies infrastructure services, control plane services, and any data plane or cloud services (running in OpenStack provided virtual machines and Docker provided containers)

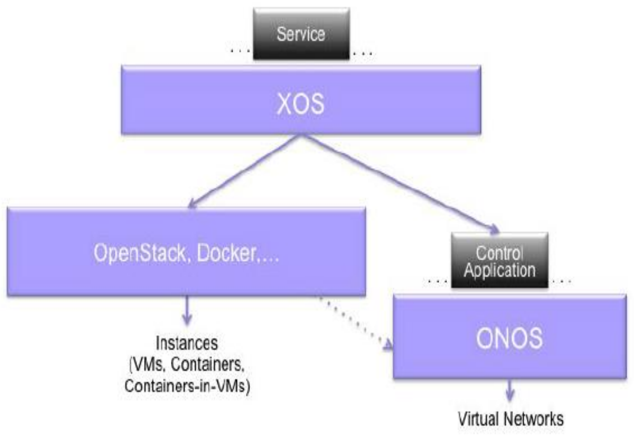


Figure 2: Open source software component to build CORD

ONOS play two roles in CORD it help in interconnect VMs,and it helps provide a platform for hosting control programs that implement cord services.

1.5.3 Transformation Process for the CORD

With this hardware / software basis, the transformation of the central office today into CORD can be seen as a two-step process.

1. The step is to virtualize the hardware device built for each purpose becomes its software partner running on commodity hardware. The key to this step is to break down and refractor the functionality of the legacy devices, to decide what is implemented in the control plane and what is implemented in the data plane.
2. It is to provide a framework in which these virtualized software elements can be plugged in; along with any cloud services the operator wants to run in the Central Office, to create a coherent end-to-end system. This framework also includes the unifying abstractions that make this collection of hardware and software elements an economical, scalable and agile system. These two steps to rearrange the Central Office in more detail are described in the following.

1.6 Virtualizing Legacy Devices

The first step in the re-architecture of the central office as a data center involves the virtualization of existing hardware devices and the transformation of each legacy device into its commodity hardware plus software. During the process, functionality is probably broken up and repackaged in a new way.
The process for the red-highlighted devices in Figure 3, includes optical line termination (OLT), customer premises equipment (CPE) and broadband network gateways (BNG).

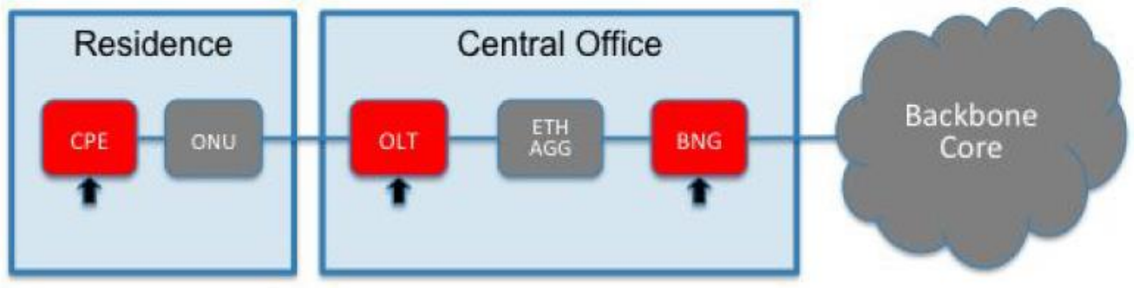


Figure 3: Legacy central office Overview, with three physical devices to be virtualized

Here we focus on GPON technology and virtualization of relevant components like OLT. However, the underlying principles apply equally to other access technologies, such as 10GPON, G.Fast copper networks, DOCSIS cable networks, and mobile BBU networks.

1.6.1 Benefits and Challenges

OLT is a large capital investment in involving closed and proprietary hardware racks that terminate access to tens of thousands of subscribers per CO. Virtualization of OLT is particularly challenging because OLT is mainly implemented in hardware, unlike many network appliances implemented by software running on vendor-branded commodity servers. At present, CPEs are distributed to tens of thousands of customer sites per CO, making them an important operating burden. This is particularly true if a service upgrade requires an upgrade of the hardware.

BNGs are costly and complex routers that have historically added much of the functionality provided by a Central Office to make it difficult to develop in an agile and cost-effective manner. The challenge is how to transform such a diverse collection of devices into software with commodity hardware systematically. The main idea is that there is a simple template for the virtualization of every physical device. It includes a combination of three elements:
1. A data plane function, which is the NFV element.
2. A control plane functions, as SDN element.
3. Merchant silicon, including both commodity servers and whitebox switches.

While SDN and NFV are overloaded terms, both are implemented for our purposes by software running on commodity servers, where it is considered an NFV element if packet processing is entirely in software and an SDN element if the software also controls commodity switches and me/ O blades via an open interface such as Open Flow.

Now I try to explain how this pattern is applied to **OLT, BNG, CPE**, and, resulting in virtual incarnations of each physical device.

1.6.2 Overview for Virtualizing the CPE

A CPE is installed at the customer's premises, sometimes referred to as a "**Home router "or "residential gateway.** "They are a major source of CAPEX and OPEX costs and a barrier to the introduction of new services because of their numbers. They often perform a collection of key functions like DHCP, NAT and optional services like Firewall, Parental Control, and VoIP for residential subscribers. There are also more sophisticated enterprise functions (for example. WAN Acceleration, IDS).By adding vCPE, new value-added services and customer care capabilities can be provided where they could not before due to hardware limitations. **Virtual Subscriber Gateway (vSG) (vCPE),** also runs a lot of subscriber functions, but it works on commodity hardware located in the Central Office and not on the premises of the customer. The device in the home (which we still call the CPE), but it can be reduced to a bare metal switch, with all the functionality that existed on the original CPE moved to CO and runs on commodity servers in a VM. Customer LAN includes a remote VM that resides in the central office and effectively provides direct access to the Telco cloud for every subscriber.

CORD allows a wide range of options for implementing subscriber bundles including a full VM, a lightweight container or a lightweight container chain. The approach is to treat the bundle as a whole (which is roughly equivalent to a VM image or a container configuration) as a standard representation and leave the means by which subscribers choose the set of functions to be included in their bundle as a choice. **The reference implementation**, for example, gives subscribers the ability to select from a small collection of functions (example. DHCP, NAT, firewall, parental filtering) but implements each function by properly configuring a **container** (as defined by a corresponding docker file). Experiments show that **1000 subscribers per server** are consistently supported by this approach.

1.6.3 Overview for virtualizing the OLT

OLT terminates the optical connection in the central office by adding a set of subscriber connections to each physical termination point. Number of OLTs and cost in a CO, the virtualization of the OLT can result in significant savings for CAPEX and OPEX.

The first challenge is to create I/O Blade with the PON OLT MAC, and AT&T has been working with the Open Compute Project to develop an open specification for a GPON MAC 1RU box.

These I/O blades are then placed under the same control paradigm based on SDN as the white box switching fabric. The resulting control program, known as virtual OLT (vOLT), runs on top of ONOS and implements all other functions normally contained in a legacy OLT chassis.

vOLT needs to implement authentication on subscriber basis establishes and manages VLANs connecting the subscriber's devices to the central office switching fabric and manages other OLT control plane functions.

1.6.4 Overview for Virtualizing the BNG

A BNG is one of the most complex and costly devices in a Central Office that provides subscribers with the means to connect to the public Internet. It minimally manages a routable IP address for each subscriber and provides some kind of network connectivity for the subscriber. However, in practice, the BNG also bundles a large collection of added value features and functions, including VPNs, GRE tunneling, MPLS tunneling, and 802.1ad termination and so on.

CORD’s virtualized BNG, known as a virtual router **(vRouter),** is implemented on behalf of subscribers as an ONOS hosted control program that manages flows through the switching fabric. More functions are bundled into a BNG device are not attempted to be reproduced, although in some cases (example. authenticating subscribers) another service provides this functionality (example., in our case, vOLT).In general, it is more accurate to believe that vRouter provides a private virtual router for each subscriber, in which the underlying fabric can be considered a distributed router with "line cards "and "backplane "installed by bare metal switches. IP network is controlled by vRouter between the attached subnetworks by the subscriber. VRouter also peers with legacy routers, example. Advertising routes for the BGP. Approach to vRouter illustrates an example of reconstruction. Historically, BNG also authenticates the user, but this functionality has been unbundled and moved to vOLT. This is because subscribers must be authenticated before they access vSG, which resides in the home but has now been transferred to the Central Office.

1.6.5 End-to-End Packet Flow

In these cases we conclude by sketching the packet flow of a subscriber through the CORD, assuming that the subscriber already has a Telco account. If the subscriber authorizes the home router, an 802.1x authentication packet will be sent to the CO via GPON. Upon arrival at I/O blade port, the packet is transferred to the vOLT control program (through ONOS), which authenticates the subscriber using an account registry such as RADIUS. Once authentication is done then vOLT assigns VLAN tags to the subscriber and installs the corresponding flow rules in I/O blade and switching fabric (via ONOS), asks vSG to spin a container for that subscriber and bind it to the VLAN. In turn, vSG requests a routable IP address from vRouter that causes vRouter (via ONOS) to install the flow rules in the fabric and software switches needed to route packets to / from the container of that subscriber. Once set up, the packets flow from the home router via a VLAN to the subscriber's container, the packets are processed according to any bundle associated with the subscriber's account (and configured in the container) and then sent to the Internet using the assigned source IP address.

This description is high-level implementation, over both many low-level details of each component (example how to assign VLAN tags, exactly what flow rules are installed in services, the exact composition of container functions) and the mechanisms by which the different agents (ONOS, Open Stack, Dockers, vOLT, vSG, vRouter) are actually plumbed.

1.7 Service Framework:

This section focuses on the second step in re-architecting the Central Office as a datacenter: orchestrating the software elements resulting from the first step (plus any additional cloud services that the operator wants to run there) into a functioning and controllable end-to-end system

1.7.1 Benefits and Challenges

Replacing hardware devices with virtual machine software is a necessary first step, but not sufficient in itself. Just as all devices in a hardware-based central office must be connected in a meaningful manner, their software counterparts must also be collectively managed. This process is often referred to as service orchestration, but if network operators have the same

agility as cloud providers, the abstractions underlying the framework of orchestration must be fully integrated
1. The elastic scaling of the resulting virtualized functionality,
2. The composition of the resulting unbundled functionality.
XaaS (everything as a service) is a way to follow the unifying principle .This brings the different functions introduced by virtualizing hardware devices in a single coherent model. The functions run as services (these functions run over ONOS, a scalable network operating system), the data plane functions run as scalable services (these functions scale over a set of VMs), Commodity infrastructure itself is managed as a service (this service is commonly referred to as the IaaS generic name) and a number of other global cloud services running in the Central Office are also managed as scalable services (as described below, CORD includes a virtualized CDN as an example).

1.7.2Scalable Services, Not Virtual Devices

Although the terms vOLT, vSG, and vRouter refer to the virtualized counterpart of the three physical devices, all three are packaged as services. While these services could be named according to their legacy counterparts, the new architecture no longer requires functionality to be bundled along with the same boundaries. It is, therefore, more intuitive to consider the above-mentioned virtualization process, which results in three generic multi-tenant services.
1. VOLT is a program of control running ONOS. It implements access as a service, where every tenant is a VLAN subscriber.
2. VSG is a scaled data plane function across a set of containers. It implements subscriber as a service, in which each tenant is a subscriber bundle.
3. vRouter is a program of control running ONOS. It uses the Internet as a service, where each tenant matches a routable subnet.

If a Content Distribution Network (CDN)—a scalable cloud service that is deployed across the network of the operator, including central office caches—Offices— to these three new services added, we have an example of the three types of services outlined in the introduction: a cloud service (CDN), a data plane service (vSG) and two plane control services (volt and vRouter).This results in the previous figure of the legacy Central Office being re-architecture in the service graph shown in Figure 4. In order to illustrate the generality of CORD as a configurable platform, this figure also includes a vG=

Each CORD service is multi-tenant and offers some service abstraction in the same sense that a conventional cloud storage service provides an abstraction of "volume "and a No SQL DB service provides an abstraction of "key store." As such, we read the service graph: A subscriber acquires a VLAN subscriber from vOLT, which acquires a subscriber bundle from vSG and finally acquires a routable subnet from vRouter.

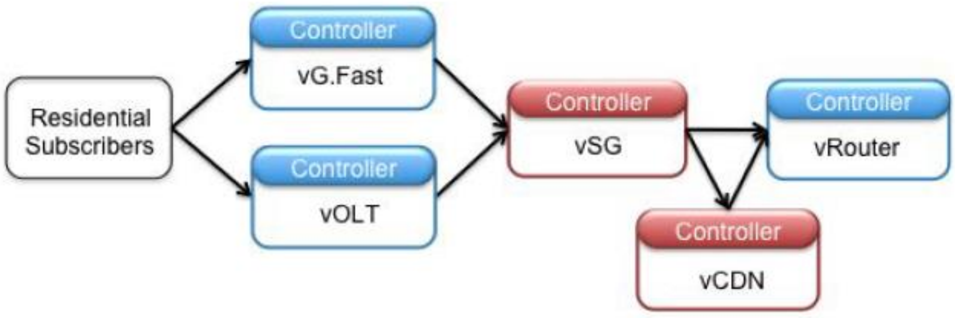


Figure 4: CORD service graph, including volt and vG.Fast

In this cases, the Subscriber is a service graph tenant in its entirety. This means, pragmatically, that once the service graph shown in Figure is configured in CORD, The subscriber can control his or her subscription (example. setting the parental control function to disallow Facebook access) by invoking operations on the subscriber object, without knowing which service implements which feature. The structure imposed on the right set of components by the XOS abstractions maps such a request.

1.7.3Layers of Abstraction

The service graph shown in the figure represents the high-level specification that a network operator would provide, but this specification must be mapped to the servers, switches and I / O blades underlying the service. These abstractions impose a structure on the services in such a way that help operators to express and enforce policies on them. Working top-down, CORD defines (and the corresponding mechanisms to implement) the following abstractions.

1. Service Graph: It represents a number of relationships of dependence between a set of services. CORD models service composition as a relationship between the service provider and the service provider (i.e., the customer).Service tenancy is anchored in a principal tenant (example. subscriber) who has one or more user accounts.

2. Service: It represents an elastically scalable, multi-tenant program, including the means to instantiate, control, and scale functionality. CORD act as a service controller that exports a multi-tenant interface and an elastically scalable set of service instances. XOS includes mechanisms for assembling a Greenfield and legacy CORD-ready service.

3. Slice: It represents a system-wide resource container in which services are carried out, including the means to determine how these resources are incorporated into the underlying infrastructure. CORD models a Virtual Machine (VM) slice and a Virtual Network (VN) set. VMs are implemented with the underlying IaaS (OpenStack and Docker) components, while ONOS implements VNs.

4. Virtual Network: This represents an interconnection between a set of instances. CORD supports several types of VN, including Private (connects instances within a Slice), Access

Direct (used by a tenant service to access a service provider by directly addressing each instance in the service provider), and Access Indirect (used by a tenant service to access a provider service by addressing the service as a whole). The latter two types of support service composition.

The support mechanism for virtual networks by CORD is implemented by a pair of control applications running on top of ONOS. The first one, called **VTN**, installs flow rules for direct or indirect addressing on each server in the OvS. 2nd is known as **Segment Routing**; implements aggregate flows across the switching fabric between servers.

Looking at Figure through the lens of NFV, each tenant abstraction in CORD corresponds to a Virtualized Network Function (VNF) in the NFV architecture. How much of a sequence of VNFs (a service chain) on a VM sequence depends on three things:
1. Whether the service is carried out in the network control plane or the network data plane.
2. How each service maps its tenants in one or more instances of service?
3. What kind of network virtual interconnects service instances?

1.8 Future Plans

CORD is a revolutionary effort to transform the legacy of the Telco network's central offices. The closed and proprietary hardware is replaced in the new Central Office as a data center by software running on commodity servers and switches. This software, on the other hand, is managed and organized as a scalable service collection. The aim of the CORD is to show the feasibility of a central office which benefits both CAPEX and OPEX from commodity infrastructure and the agility of modern cloud providers.

The implementation of the reference has been informed by the specific requirements of the field trial and is designed to be a general platform that can be set up for a wide range of deployment scenarios. For example, **mobile user configuration (MCORD)** and an **Enterprise user configuration (ECORD)** are already in operation.

1.9 Impact of CORD on Central Office:

For telecommunications service providers, the demand for bandwidth continues to increase exponentially, while subscribers and revenues increase at a much smaller rate. The main aim of the provider is to reduce the cost for both OpEx and CapEx and how to create new revenue streams efficiently. Market size will range from $54 billion to $168 billion by 2022.[7][9]

By virtualizing the network, function is the key to reduce the costs and development of the new services. The main focus of the cord is to provide the framework for the implementation of the virtualized environment.

Virtualization makes the network more agile and cost-effective. As historically all the network equipment is built with custom silicon and purpose-built hardware. All the network operations such as routing and switching, firewalls,etc. are performed by the specialized devices that are only dedicated to that function. But their all a lot of drawbacks and limitation to that approach: all the devices are vendor lock-in and proper training and knowledge is needed to operate these devices and this result in very high costs. To solve these kinds of problems cost efficiently, we have a CORD which is cloud-based virtualization approach. In this virtualized environment the virtualized software applications and tools will run of general hardware called white boxes which will help in running all the operations.

SDN AND NFV play a very important role in enabling virtualization in the data center and central office environment.

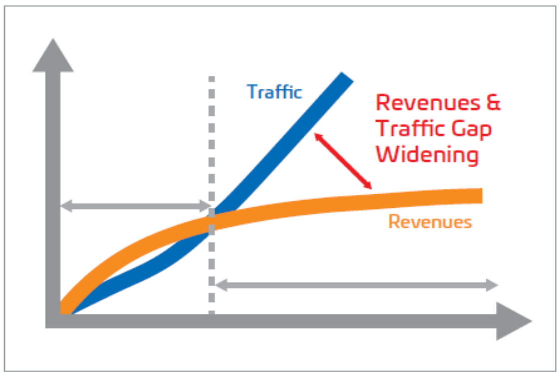


Figure 5: Overview of the Growing Gap between traffic and revenues

These two technologies help in reducing the cost. SDN/NFV is the key role in the impressive growth of the cord platform. The cord is an integrated platform that combines all three **SDN, NFV,and cloud** to provide the central office with data center economy and cloud agility.

Control plane is separate from the forwarding plane using SDN which helps in providing centralized control. SDN also allow customized network design to improve throughput, linking all the virtualized network function.

NFV help to convert the specific legacy hardware into software programs that run on common off the shelf (COTS) hardware infrastructure .These hardware infrastructures form a pool of resources that increase demand-based capacity and accelerate deployments for new applications and services.

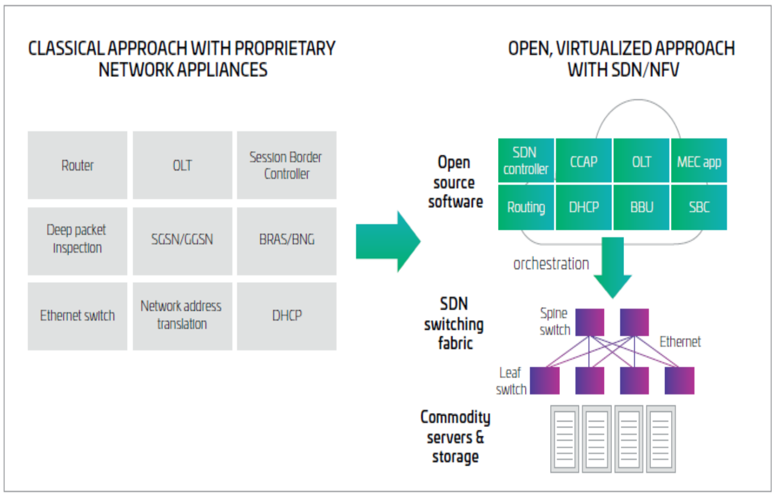


Figure 6: Journey to Proprietary appliance to an open, Virtualized Environment

SDN help in separates the network control plane from the forwarding plane, and it provides centralized control. With SDN we can customize the network design to improve throughput, linked combined virtualized all the network operation into the service chain that is independent of any type of specialized hardware.

1.9.1CORD: Data center best practices to the central office

Through CORD, we are using decades of knowledge with the data centers and CORD is an integrated open-source solution platform that combines NFV, SDN and highly flexible, scalable commodity clouds to bring the data center economics.

Bringing the edge closer

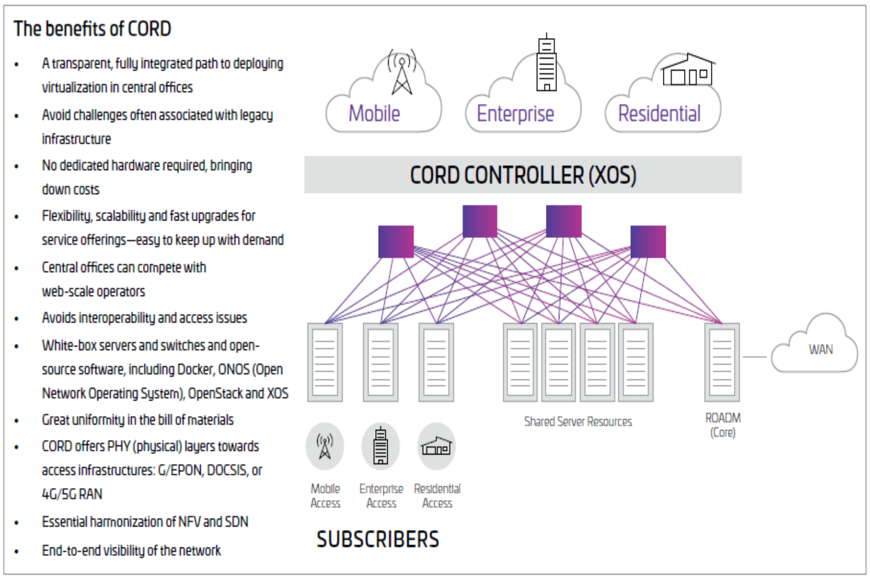
A cloud data center could be located in a traditional scenario far from the users of systems, possibly on a different continent. This distance could give an unacceptable response time or latency in processing requests for information and content delivery. Now we require less response time and lower latency for new technology like augmented reality and an autonomous vehicle that can be delivered by resources in close proximity to the end user.

Use cases of Edge data center

Content and frequently accessed applications cached on servers near end users. This improves the quality of high bandwidth applications since MEC is an essential enabler for more intelligent applications delivered with lower latency. If we have a data center close to users improve the quality of services.

CORD Infrastructure

Key rolesisSDN and NFV architecture for an increasing number of service provider’s technologies that deliver higher levels of automation, faster and more agile service deployment, new revenue streams and higher management, operations and cost efficiencies. A number of key elements must be considered when implementing a CORD infrastructure. CORD requires not only hardware and software different from traditional solutions but also an architectural change to allow for a new way of managing and delivering services.



The benefits of CORD

- A transparent, fully integrated path to deploying virtualization in central offices
- Avoid challenges often associated with legacy infrastructure
- No dedicated hardware required, bringing down costs
- Flexibility, scalability and fast upgrades for service offerings—easy to keep up with demand
- Central offices can compete with web-scale operators
- Avoids interoperability and access issues
- White-box servers and switches and open-source software, including Docker, ONOS (Open Network Operating System), OpenStack and XOS
- Great uniformity in the bill of materials
- CORD offers PHY (physical) layers towards access infrastructures: G/EPON, DOCSIS, or 4G/5G RAN
- Essential harmonization of NFV and SDN
- End-to-end visibility of the network

Figure 7: Benefits of Cord and CORD Controller (XOS)

1.9.2 Processing in the pod

The physical adoption of CORD in a central office requires a private cloud networking approach with a data center architecture (often deployed in a data center "pod "— a separate area in the processing facility). This pod architecture can benefit from lower cost multimode fiber optics savings. Migration at higher speeds is common and in some cases frequent in a fast-moving environment. Careful cabling infrastructure planning within the pod is essential to support multiple migrations.

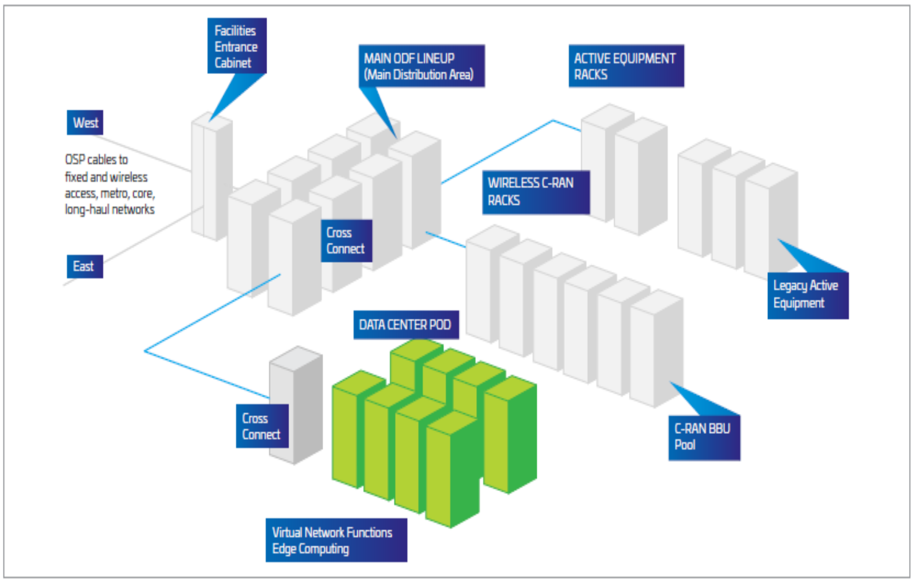

Facilities Entrance Cabinet
MAIN ODF LINEUP (Main Distribution Area)
ACTIVE EQUIPMENT RACKS
West
OSP cables to fixed and wireless access, metro, core, long-haul networks
East
WIRELESS C-RAN RACKS
Cross Connect
Legacy Active Equipment
DATA CENTER POD
Cross Connect
C-RAN BBU Pool
Virtual Network Functions Edge Computing

Figure 8: Central office with data center pod

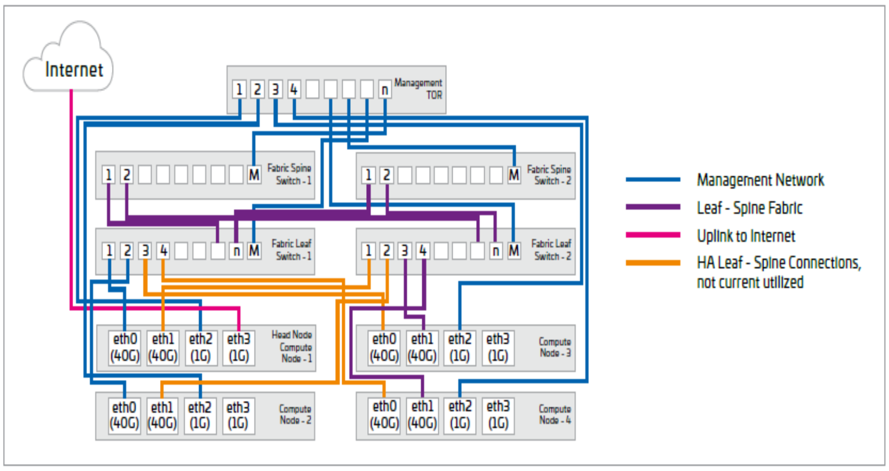


Figure 9: Compute/storage Overview

1.9.3Overview for cabling infrastructures in CORD pods

For the data center pod, engineering teams need to consider a high- speed migration path. The capacity of the lane is expected to continue to grow, reaching 400 G by 2020 and allowing the next generation of high-speed connections for fabric switches. Current requirements may require links of 10 G, 40 G, and 100 G, while the next upgrade may require links of 25 G, 50 G, 200 G and 400 G.

The increase in data center throughput speeds is driven by various factors. For example, the server density increases by about 20 percent per year and the processor capabilities increase at a similar rate.

1.9.4High-speed migration

Data center pods should need to use cable infrastructure which deals with exponentially increasing demand for higher capacity, new architectures, and ever-changing network requirements. Building blocks are the solution that enables easy and fast migration to higher speeds and different fiber optic technologies. The flexibility to deploy higher speeds or low-cost networks offers agility and minimizes data center growth costs.

1.9.5 High port density is essential to CORD success

The mesh of fiber links in a topology of leaves and spines creates a high-capacity network resource or "tissue, "which is shared with all attached devices. All connections to fabric run at the same speed. The higher the speed, the greater the tissue capacity. Tissue networks require a wide range of fiber connections, especially in the switch layer.

1.9.6 New business strategies on CORD architecture

CORD’s aim is to replace designed switching hardware with flexible and cost-effective off-the-shelf components and open source software. This new central office platform offers efficient, value-added (cloud) services based on demand. The implementation of CORD affects many areas of operation of a service provider–from hardware standards, deployment and maintenance to procurement practices.

1.10 CORD Monitoring Service:

1.10.1Introduction

The Monitoring service provides a generic platform to support real-time network observability in a Telco Central Office for fixed and mobile SDN networks so that any third - party analytical solution can be included in this framework [10]

As part of the CORD (Central Office Re - architected as Datacenter), the platform is being developed and focuses on the initial phases on the fixed / residential network environment. This platform's main goals are:

- Be scalable and multi-tenancy support.
- To instrument services in addition to computing and network devices.
- Can be helpful in adjusting the level of probing in the underlying devices.
- Aggregate probing information.
- Redirect data streams via "**probe VM**" which help in a deeper level of instrumentation that is not otherwise available from underlying devices.

As shown in **Figure 10**, the Monitoring Service collects data from various CORD network elements, including compute servers; white - box switches, I / O devices, and software-based services running on compute servers, and makes them available to one or more analytics applications running on top of CORD.XOS is firstly integrated with CORD residential services like **vSG (Virtualized Subscriber Gateway)[26], vOLT (Virtualized OLT)[25],**

vRouter (Virtualized Router)[24] but it can be used to other services like **mobility and Enterprise services** on an architectural basis.

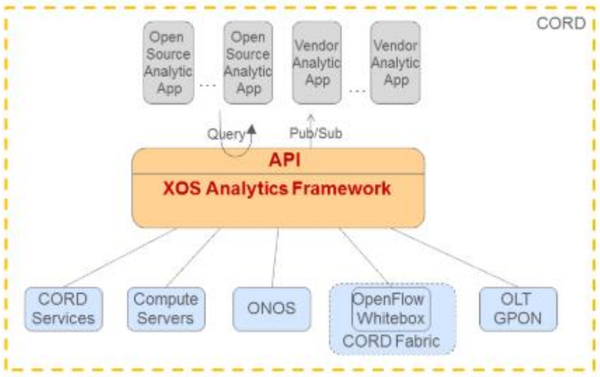


Figure 10: High-level organization of the XOS Monitoring Service.

1.10.2 Architecture

The Monitoring service leverages the OpenStack Ceilometer open source framework and improves it with a few key features as shown in Figure 11

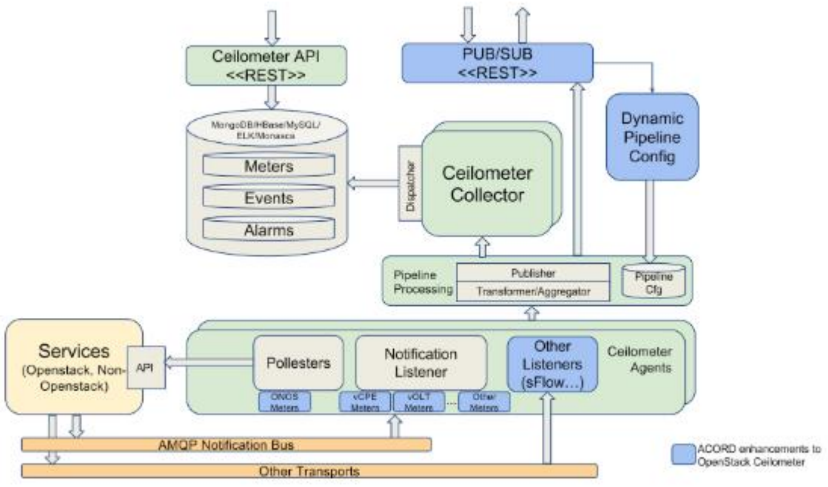


Figure 11: OpenStack Ceilometer with A-CORD enhancements.

In Figure 11, this service gives both Polling and Notification -based measurement collection methods on the southbound side and on the northbound side provides Query and Subscribe-based interfaces towards analytic applications. The analytical application correlates these events from different network sources into more meaningful events to perform any closed feedback loop operations in real time, but can also feed these events back into the monitoring framework so that other applications can use them in their analysis.

The Monitoring service also provides a platform for dynamic control of the probe information available in the underlying services and devices. It also provides flexibility for analytics applications in order to install "application - specific" probe functions in the network and potentially reflects selective traffic to these functions. This would be done to execute deeper instrumentation levels that otherwise would not be available from the underlying devices. The probe functions can feed the probe data directly into applications or feed them back into the monitoring framework so that other applications can use these events.

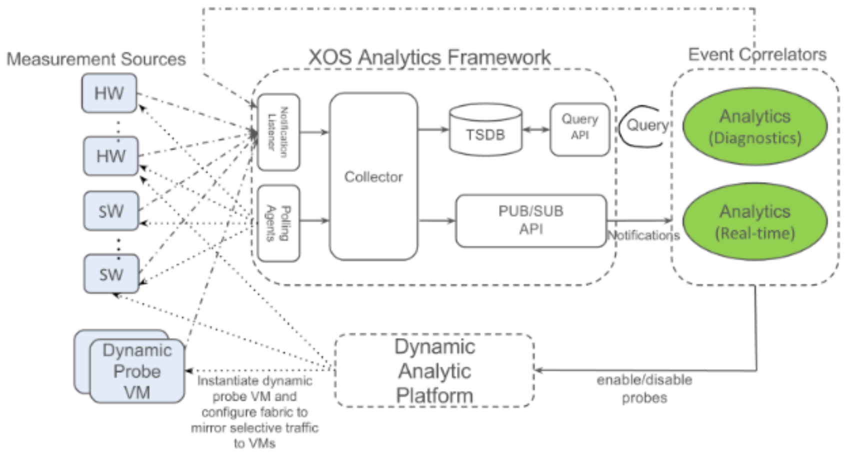


Figure 12: Internal implementation based on OpenStack's Ceilometer.

XOS monitoring framework is important as an elastically scalable, multi-tenant service in line with the core XOS principle of everything - as - a - Service (XaaS). Using the XOS service building toolkit (example, Data Model, Synchronizer), it does this. Based on the workload, each of the components can be scaled up and down. Multi-tenancy, mention in Figure 13, is done by developing a lightweight proxy containers for every tenant of the Monitoring service, such that every tenant is able to access only to the instrumentation data of the network resources belonging to that tenant. The tenants would typically be the applications for analytics that run on top of this service.

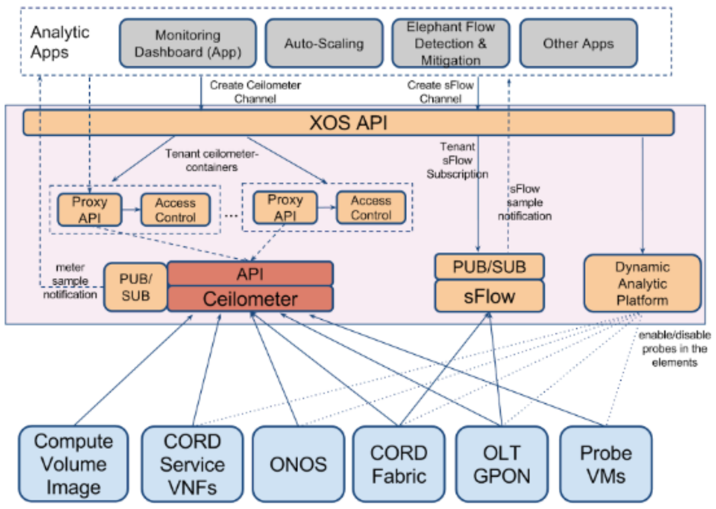

Analytic Apps
Monitoring Dashboard (App)
Auto-Scaling
Elephant Flow Detection & Mitigation
Other Apps
Create Ceilometer Channel
Create sFlow Channel
XOS API
Tenant ceilometer-containers
Tenant sFlow Subscription
Proxy API
Access Control
Proxy API
Access Control
sFlow sample notification
meter sample notification
PUB/SUB
API
Ceilometer
PUB/SUB
sFlow
Dynamic Analytic Platform
enable/disable probes in the elements
Compute Volume Image
CORD Service VNFs
ONOS
CORD Fabric
OLT GPON
Probe VMs

Figure 13: Monitoring service integrated into XOS.

1.10.3 Software Components of CORD

Following are the component of XOS:
1. Authoritative state for the service definition to Django data models.
2. For defined users to do interaction with the service and update the model's state we are using Front-end API.
3. Back end system is a synchronizer that implements the service in sync with the model state.

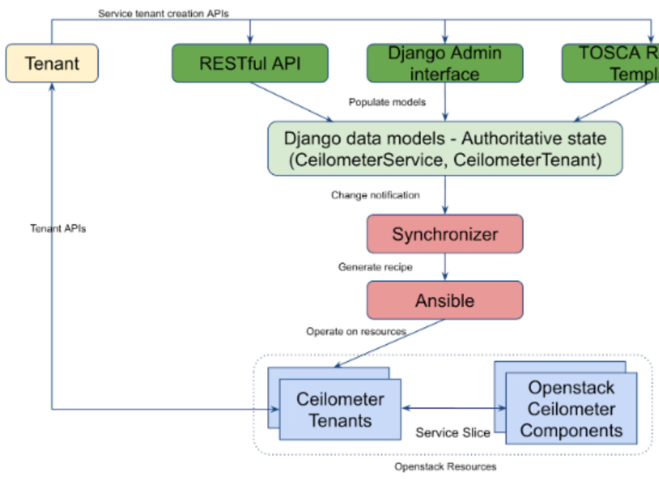


Figure 14: Software components of XOS monitoring services

1.10.4 Models

For the Ceilometer service, there are two Django models defined:

1. Ceilometer Service
2. Monitoring Channel, also called CeilometerTenant.

The relationship of these models with XOS core models is described as follow:

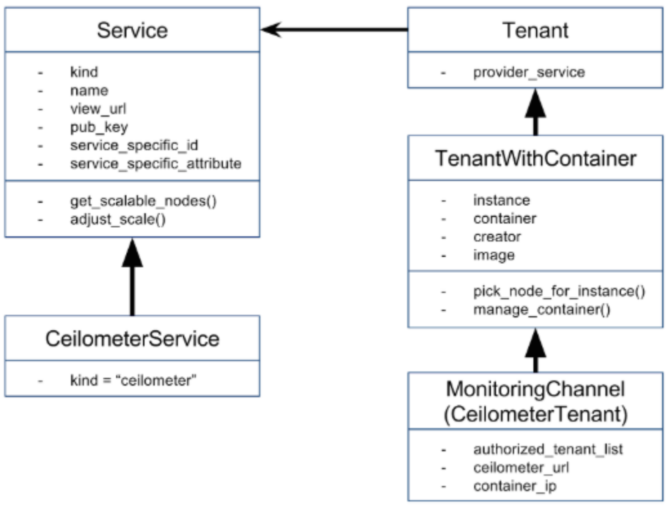


Figure 15: Overview to Ceilometer service data model.

1.10.5 Synchronizer

The Ceilometer synchronizer's role is to monitor the associated data model for any changes and to ensure that the backend system status is up - to - date with the latest changes to the data model. Ansible playbook is used by Ceilometer synchronizer to reflect any changes in the backend of the data model. When creating or updating a new Ceilometer tenant data model, the synchronizer will carry out the following operations:

- Synchronizer finds a VM to host docker containers for tenant ceilometers. If no VMs or existing VMs are fully used, the synchronizer installs a new VM and installs on that VM any necessary software (python, docker, pipework, etc.).
- Synchronizer creates a new Docker container for a tenant in VM and creates port-forwarding from VM port to Docker container port 8000. It also writes in the tenant docker container the information related to the authorization (such as the authorized tenant list ...etc.)**.**
  - web server uses the preloaded docker image that runs on port 8000.
  - The web server is a ceilometer proxy server waiting for any Ceilometer query from the applications and forwarding it to the OpenStack ceilometer service after the configured access controls have been applied.

1.10.6 Front-end APIs

three front-end mechanisms provide by XOS to interact with these services:

- RESTful interface
- Admin GUI interface
- TOSCA Resource Templates

1.10.7 Summary

The monitoring service operates in CORD infrastructure, providing data from different elements in real time. This allows network operators to run one or more Analytics applications to monitor and respond to various aspects of the network, including detection and mitigation of anomalies, fabric congestion avoidance, troubleshooting / diagnosis of customer care, etc.

More detailed about XOS can be found here:
https://wiki.opencord.org/display/CORD/CORD+Monitoring+Service

1.11 ONOS (Open Network Operating System)

ONOS stands for **O**pen **N**etwork **O**perating **S**ystem)[11][12]. ONOS provides software-defined network (SDN) control plane, manages network components such as switches and links, and runs software programs or modules to deliver end hosts and neighboring networks communication services. It is the leading open source SDN controller for building next-generation SDN/NFV solutions. ONOS is meeting the need of clients who want to build carrier-grade solutions. Using white box merchant silicon hardware to leverage economics while offering flexibility to create and deploy new dynamic network services with simplified programmatic interfaces. Two key features which are Configuration and real-time control are supported by ONOS, eliminating the need to run routing and switching control protocols inside the network fabric. Innovation is enabled by moving intelligence into the ONOS cloud controller, and end users can easily create new network applications without changing the data plane systems. The ONOS platform includes:

1. A platform and set of applications that perform extensible, modular, distributed SDN controller.
2.It is scale-out architecture to provide the resiliency and scalability required to meet the rigors of production carrier environments.

- ONOS provides some analog features, including APIs and abstractions, resource allocation and permissions, as well as user-facing software such as CLI, GUI, and system applications.
- Rather than a single device ONOS able to manage all the network that can greatly simplify the administration, configuration and deployment of new software, hardware, and services.
- ONOS platform and applications behave as a modular, extensible, distributed SDN controller.

An operating system's most important benefit is that it provides a useful and usable platform for software programs designed for a specific application or application case.

Figure 16 shows the architecture of ONOS and emphasizes its use of a distributed core that enables ONOS to boast of high availability and resiliency to match the carrier-class standards. This distributed core layer is sandwiched between the northbound core APIs and southbound core APIs. Both of these cores serve the purpose of offering protocol-agnostic API to the distributed core from their respective directions. The distributed core takes care of the coordination across the cluster, as well as state and data management from the northbound and southbound cores. It also makes sure that the various controllers across the network work coherently and keep a common view of the entire network view and not just segregated view based on their visibility. The application interacting with ONOS should have this one view of the network. This is where ONOS' strength and benefit of its distributed nature come in.

On the southbound core APIs, ONOS uses pluggable adapters, which allows it to support many popular SDN southbound protocols, as well as to be flexible. The northbound APIs allow for applications to interact with and use ONOS without requiring knowledge of the distributed ONOS deployment.

ONOS releases are named after bird species; even the ONOS logo is that of a flying bird. Its latest release, called Hummingbird is available at the time of this writing. One of the most popular uses of ONOS is its inclusion in a new open architecture called Central Office Re-architected as a Datacenter (CORD) that is designed to facilitate adaption of SDN and NFV by vendors.

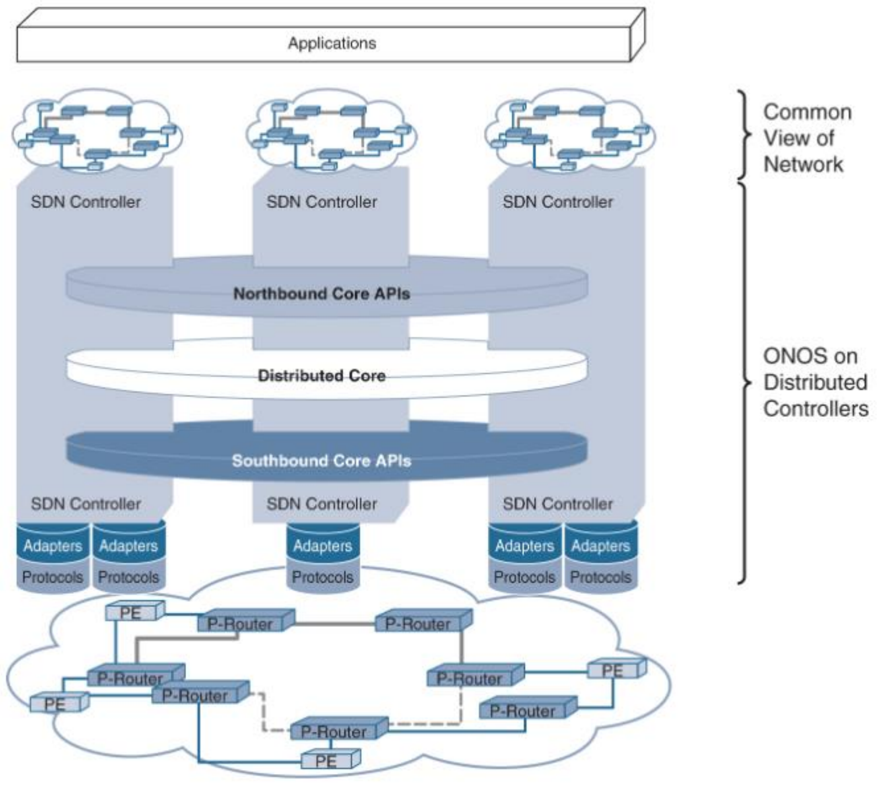


Figure 16: ONOS Architecture

On the multiple servers,ONOS run as distributed system, enabling multiple server CPU and memory resources to be used while providing fault tolerance in the event of server failure and potentially supporting hardware and software upgrades live / rolling without interrupting network traffic.

More information about ONOS here:
https://wiki.onosproject.org/display/ONOS/Wiki+Home

1.12 CORD Reference Implementation

1.12.1 Introduction

CORD is a reference for network operators to implement a cloud-based service delivery platform. CORD was designed to deliver cloud economies by leveraging infrastructure built from a few commodity building blocks and cloud agility by enabling fast deployment and elastic scale of services. [13]
Reference implementation as follows:
(1) it is built from open source software components;
(2) It is a fully integrated system that is comprehensive enough to support field trials
(3) it exercises the full breadth of capabilities it is designed to support.

1.12.2 Design and Technology

CORD goes behind design decisions and technology choices as a reference implementation. These decisions are subject to change, and alternatives can be used to replace individual components.

Displaying all possible services and access technologies in a single configuration is not as practical as an extensible platform. Other services implementation will also be added over time, including M-CORD (a configuration of CORD suitable for a Mobile use case) and E-CORD (a configuration of CORD suitable for an Enterprise use case).

Our technology choices are influenced by two factors. The Ist is to decide which component bakes in policy decisions that ought to be left on top of CORD to orchestration software. The objective is to separate policies (specified by the operator) from mechanisms (implemented by CORD) cleanly. The 2ndis to look forward to looking design over the edge of backward compatibility. In particular, CORD makes no effort to accommodate legacy hardware, which frees the design from constraints that could otherwise compromise the ability of CORD to achieve its goals.

1.12.3Hardware – CORD POD

Collection of commodity servers and white-box switches in combination with a "disaggregated packaging" of media access technologies such as GPON. Rackable unit contains this hardware—called a POD—that is suitable for deployment in a Telco Central Office.

1.12.4Commodity Elements for the CORD

The reference architecture of CORD POD are as follows::

- Servers – Open Compute Project (OCP) qualified QUANTA STRATOS-S210-X12RS-IU server. Each server has a 128 GB RAM, 2x300 GB HDD, and 40GE dual port NIC configuration.
- Switches – The OCP-qualified and OpenFlow-enabled Accton 6712 switch. 32x40GE ports are used to configured switches and can be extended to double as both leaf and spine switches in the CORD fabric.
- **The function of Input/output blades** – Celestica is the ODM for the PMC Sierra OLT MAC chip "OLT pizza box." This is a 1u - blade with GPON interfaces of 48x2.5Gbps and uplinks of 6x40GE.Ubuntu LTS 14.04.3 runs the servers, including Open vSwitch (OVS 2.3.0).

1.12.5Scenarios for the Configuration and Sizing

The reference CORD POD is sized to focus on CORD's functionality and performance, as well as to serve in a field trial. Basic overview of the pod is as follow:

- Two racks which are virtualized with two leaf/ToR switches and a set of servers/blades are being used here.
- It consists of six Accton switches arranged in a leaf-spine configuration, with four leaf switches (two per virtual rack) and two spine switches. 24 ports are used a leaf-spine as downlinks to the rack's servers and 8 ports as uplinks to the spine, while the spine switches use all 32 ports as downlinks to the leaf switches.
- It includes two Celestica OLT pizza boxes. All I / O blades are connected to the corresponding pair of leaf switches in the first "virtual rack."
- It has six Quanta servers, with three serving as "head" nodes and as "compute" nodes. the Two virtual racks are used as compute servers and connected to the corresponding pair of leaf switches.

Selected leaf switches have the capacity to operate up to 24 dual-port servers,and the spine switches have sufficient capacity to operate up to 16 racks.

1.12.6 Software – CORD Distribution

Other Integrating other open source projects are also being used to build CORD, each of which has been configured with its own set of parameters, model definitions, and software plug-in. These software components collectively define the extensible platform of CORD, in addition to which a specific service profile is included in the reference implementation.

1. Core Platform: Combining ONOS, XOS, OpenStack and Docker, CORD's core software platform is built.

- OpenStack provides core IaaS capabilities and is responsible for virtual machine (VM) and virtual network (VN) creation and delivery. CORD uses Nova, Neutron, Keystone, and Glance subsystems from OpenStack**.**
- Docker provides a means of deploying and interconnecting services based on containers. Docker help in the configuration and deployment of CORD itself (for example, the other elements — XOS, OpenStack, and ONOS — are installed on the PODhead nodes in Docker containers).

- ONOS manages underlying white-box switching fabric, as well as the software switches (OvS) running in all server. It hosts a set of control applications implementing services and is responsible for the integration of virtual networks into the underlying network — two of these control applications: VTN and Fabric Control -re part of CORD's core platform.
- XOS is a service assembly and composition framework, and it helps to combine data plane services supported by OpenStack and Docker, and the control plane services running on ONOS. XOS is a system that defines a data model (built using Django) and a backend component controller framework (written in Python using Ansible extensively).

In these implementing choices, there are three notable design decisions that are significant First, in support of the collection of services (a) the reference implementation supports services running in virtual machines called(KVM), (b)in containers running directly on bare metal called (Docker), and (c)in containers inside virtual machines called(Docker in KVM).

Secondly, ONOS have two important roles in the reference implementation.

1. Managing the switching fabric and implementing the virtual networks that the rest of the framework requires. This functionality is implemented by the combination of ONOS applications (called Segment Routing and VTN, respectively), and it is accessed indirectly through OpenStack's Neutron API.
2. To help in hosting program to implement CORD services(vOLT and vRouter), both of which access ONOS using its Flow Objectives NBI.

Third, XOS can also act as a controller for CORD, and as such, it defines the CORD's NBI. These interfaces have both a RESTful version and a model - based TOSCA (YAML)-based interface. CORD also includes a lot of GUI-based views (examples. one for third-party service developer, another for network operators, and one for residential subscribers).

2. Service Profile

CORD is designed to be expandable with network operators specifying the service graph that they wish to deploy in a particular environment. The implementation of the reference includes the following set of services, the first three of which are targeted at residential subscribers (vOLT, vSG, vRouter).

- **vOLT** – It Provides a Subscriber VLAN that connects residential customers to CORD. A control application running on ONOS is implemented, and RADIUS is used for authentication.
- **vSG** – Provides a Subscriber Bundle with a set of consumer characteristics (e.g., parental control, firewall). Implemented as a bare metal Docker container with features implemented using different Ubuntu mechanisms.
- **vRouter** – Provides a routable subnet that connects the container of a subscriber to the public Internet. A control application running on ONOS is implemented, and Quagga is used to peer with other routers.
- **Ceilomete**r – It is a monitoring channel that queried by different hardware and software resources for meters reported. A combination of OpenStack subsystems in VMs and Docker containers is implemented.

1.12.7POD Assembly

1. Hardware

Canonical POD has servers, switches,I/O blades in a are assembled into two virtual racks (single physical rack) as illustrated in Figure 17. Figure 17 is incomplete in that it does not show that Leaf-2 is connected to an upstream (legacy) router and the OLT blades are connected via GPON to residential gateways. Figure 16 is also limited in that is shows only the 3x40Gbps uplink from each OLT blade to Leaf-1, but there is a parallel connection to Leaf-3. This second path is only active if the first one fails; the two paths cannot be used at the same time.

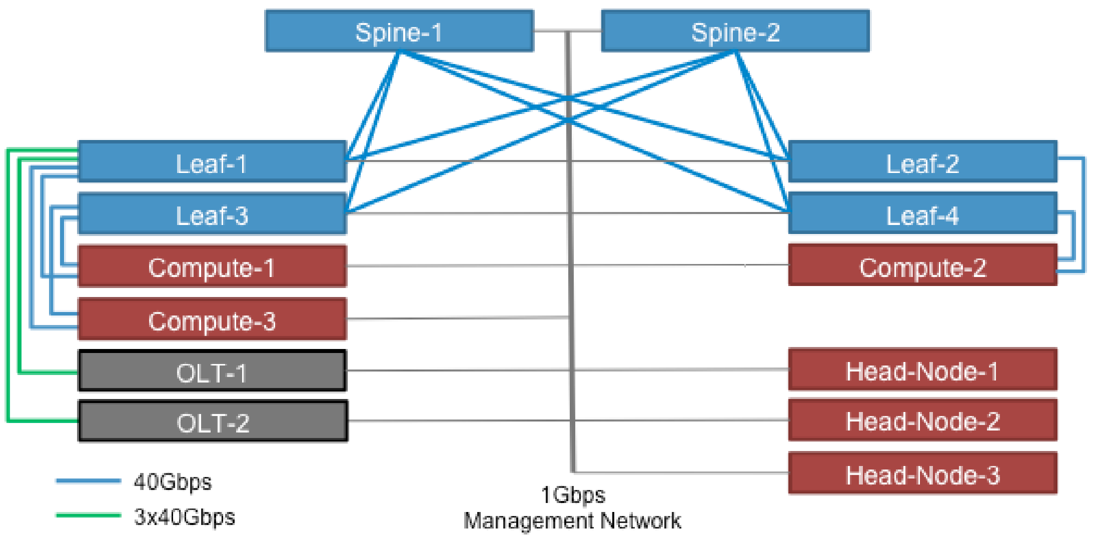


Figure 17: Overview to POD Hardware Configuration.

As depicted in Figure16, the POD includes a global management network that connects all the servers and fabric switches to the control processes running on the CORD head nodes (The physical switch is implemented separately from the white - box switches implementing the CORD data plane). There is a local management network (per - server) that logically connects the instances running on that server, but local management networks are not interconnected with each other or with the global physical management network. Instead, some of the management functions are implemented by proxies or application bridges that are connected to both the global management and each local (per-server) management network. For example, for management purposes, ssh proxy is used to log in an instance.

2.Software:

The process of build cord supports Continuous Integration / Continuous Delivery (CI/CD), with two initial deployment targets:

(1) a full hardware POD

(2) a development environment running on a single server (also known as CORD-in-a-Box)., Installing and configuring CORD is a four-step process:

1. Boot Bare Metals – Leverage Metal-as-a-Service (MaaS) to boot the collection of servers and switches in a CORD POD.

2. Install Container Support services – Configure servers with container support, including docker-engine and docker-compose.
3. Start in built Platform Components – Initialize core platform components (OpenStack, ONOS, XOS) in containers.
4. On-Board active Services – CORD service graph is initiated XOS.

1.13User Interfaces

The user interface collection available for CORD configuration and control. The collection includes an extensible graphical interface, a programmatic RESTful interface, and a configuration interface based on TOSCA. [22]

1. Graphical User Interface

The CORD GUI consists of the following elements:
(1)an auto-generated global view for the CORD data model.
(2) A set of custom views for particular users or workflows.
These views because each provides a different view of the underlying data model – the global view is a straightforward representation of the data model (typically, each page maps directly to a Django model), while each custom view typically combines and cuts across models for the sake of some specific user community.

2. Global View

The global view is a collection of Django Admin package generated pages. The Django Suit is also used to extend Django Admin and to stylize the content.All the templates can be found in /XOS/templates/admin. The global view includes pages for all of the core models (example. Services, Slices Users, Sites, Deployments) and can be used to read the update and delete objects in the CORD data model.

Besides pages corresponding to the core models, the global view also includes a set of pages for each service that has been embedded in CORD.Part of the onboarding process is to define the service-specific extensions to the Django Admin framework. These extensions can be found in the XOS/admin.py file of each service repository (Example,Service repository). A service will typically define one page that can be used to manage the entire service and a second page that can be used to manage individual tenants (instances) of the service.

3. Custom Views

There is an important need to present combined information that crosses multiple models; one can define (and load in CORD) a custom Javascript-driven view. Each view is a standalone AngularJs application that communicates through REST APIs with the CORD data model.4. REST APIs.

A programmatic northbound interface to CORD is defined by a REST API. The API defines a set of REST endpoints, prefixed with /API and divided into four main categories:

- /api/core – Core models; this is an endpoint generated automatically.
- /api/utility – Utility end-points; Not necessarily linked to a model but useful for system management (e.g., Login / Logout, Onboard Service)
- /api/service – Service-related endpoints; Defined per - service (see On-Boarding Service)
- /api/tenant – Tenant/Instance-related endpoints; Defined per service (see On-Boarding Service)

5. TOSCA

TOSCA (Cloud Application Topology and Orchestration Specification) is a YAML - based language used primarily by CORD to configure the set of services running in a CORD POD. TOSCA recipe examples are defined in the Service Profile repository. Each service also defines a service-specific on-boarding recipe as part of its service definition (see Service On-Boarding). TOSCA templates are typically pushed by a specific REST API to CORD. The GUI's global view service page also provides a button that can be used to export a specified service's TOSCA recipe.

1.14 Trellis: CORD Network Infrastructure:

Trellis is the leading open - source, multi-purpose L2/L3 spine - leaf switching fabric for data center (DC) networking. Trellis uses the ONOS Controller to create a non - blocking fabric for data centers using white box switching hardware and open source software. Unlike traditional networking approaches, a control protocol (such as BGP, OSPF, or RSTP) is not run by the fabric itself.Instead, all the intelligence is moved to applications running on the clustered ONOS controller. The entire fabric can be optimized by using a holistic view of all activity and new features and functionality can be deployed without upgrading the switches. [27]

Optionally, Trellis can provide both an underlying network and multiple tenant overlay networks. The underlay connects all physical servers and switches, while the overlays create private networks on top of the underlay for separate users, applications or services.Trellis also provides industry-leading vRouter solution for external connectivity, where the entire fabric acts as a distributed virtual router data - plane.
Traditional ways of building spine - leaf fabrics rely on control protocols running in each of the switch fabric nodes to create the underlying network in the network. Several control protocols layer 2 or layer 3 (such as BGP) is used, requiring each fabric node to implement and run one of these complex protocols. It results in a scenario where switches need more powerful CPUs, memory and all the complexity of this software. This adds costs and vulnerability to failure.

In addition, overlays are then constructed on top of the underlay to create private tenant domains. These overlays are completely independent of the underlay, eliminating the ability to coordinate and optimize overlays to change the underlay. This construction requires hair - pinning tenant traffic through virtual (software) routers in VMs that act as gateways between virtual and physical networks.

Trellis is an important part of the CORD platform, providing the network infrastructure for all connectivity in the CORD POD. Trellis is used independently of CORD for general purpose data center networking needs.

Trellis: Enabling Network Infrastructure for CORD

1. Help in eliminating the complex control protocols in the fabric networks.
2. Reduces costs of the network
3.Reduces the risk of software failure in nodes in the network operations.
4. It also combines both underlay and overlay configuration and control
5.It helps in the operation of the entire fabric as an ‘SDN island.’

6.It helps in proving solution in the management and minimizes the number of BGP instances in the overall network in the topology, thus help in speeding up route recalculation times and minimizing BGP processing load over the network.

Trellis can handle these challenges by:

1. It simplifies the switching nodes between the networks.
2. It helps in creating a single routing instance for the entire fabric network and it also simplifying how the fabric connects to the external network.
3. It helps in the Coordination of the underlay and overlay to optimize resource usage and connectivity.
4. For multi-pathing and segment routing principles to offer fine-grained path control for select traffic, it uses ECMP routing.
5.It Fully integrates the software switches running on each server, to provide a complete solution for interconnecting servers, applications, VMs and containers with dynamically created tenant domains and service-chains

Trellis is a combination of virtual networking of the underlying leaf - spine fabric overlay and unified SDN control over the underlying and overlay. Figures 17 shows where Trellis fits in the CORD architecture. [23]

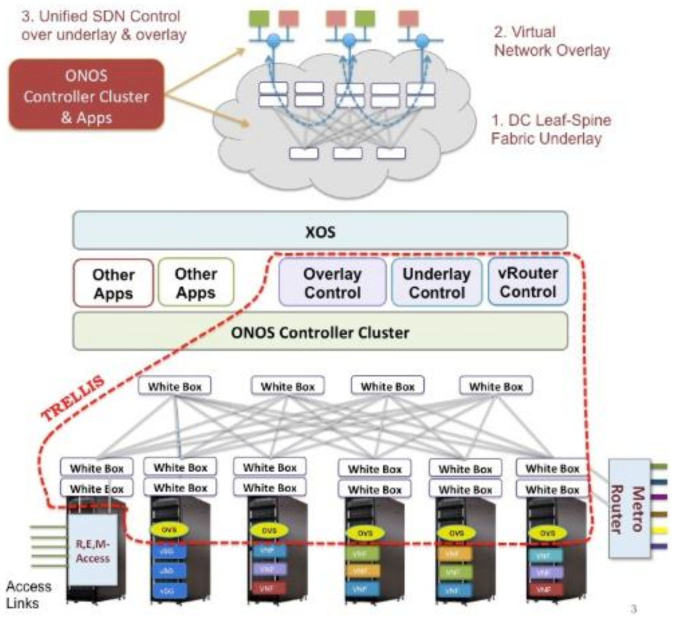


Figure 18: Trellis Architecture

1.14.1 Introduction

An application (called segment routing) in ONOS controls the underlay hardware fabric. It interacts with a number of other applications such as vRouter, vOLT,and multicast to deliver CORD services. Briefly, the architecture of the CORD network includes:

SDN - based Leaf - Spine fabric built with hardware and open - source switch software based on bare metal (OCP certified).This is a puree OF/SDN solution—while use ARP, MAC, IPv4,VLAN, MPLS, VXLAN, and other data plane headers.

1. The fabric has the following characteristics:

a. Rack handled at leaf-switches (ToRs) with the help of l2 switches.
b. L3 transmission across racks using ECMP hashing and routing of the MPLS segment.
c. vRouter combinations for interfacing with upstream metro-router, providing reachability to publicly routable IP addresses.
d. VLAN inter-connect feature to switch QinQ packets between OLT I/O blades and vSG containers (R-CORD feature).
e. IPv4 multicasting forwarding and pruning for IPTV streams (with vRouter integration) from upstream router to residential subscribers (R-CORD feature).
f. XOS integration with REST API calls to configure end-hosts and VLAN cross-connects at runtime dynamically.
g. Ability to utilize rack as fabric in single-rack or multi-rack configurations.

2. The fabric forms the underlying network in overlay/underlay architecture. The overlay is also based on SDN, with the following features:

a. Use of software switches (e.g., OVS with DPDK) with a service chain custom - designed pipeline.
b. Load - balancing distribution per service in each OvS.
c. VLAN tunneling in OvS for virtual networks based on overlays.

The advantage of common SDN control over both the overlay infrastructure as well as the underlying fabric is that they can be orchestrated together and optimized to deliver the features and services **that Central Offices require, with the agility and economies of data center operations.**

More Information about Trellis is as follows:
https://wiki.opencord.org/display/CORD/Trellis+Underlay+Fabric

1.15 CORD: A Big Market with a Diverse Ecosystem:

As CORD has released its version 4.1 so it is getting better day by day, and the market value of cord is increasing at great extent. CORD evolves into a powerful platform with a range of capabilities to enable new edge services while reducing costs and meeting latency requirements. CORD is supported by a broad ecosystem of global carriers and vendors ranging from software developers to white box and networking suppliers.
The cord is expected to $300 B markets in the next coming years. This is why every enterprise wants to deploy cord in the next coming years.

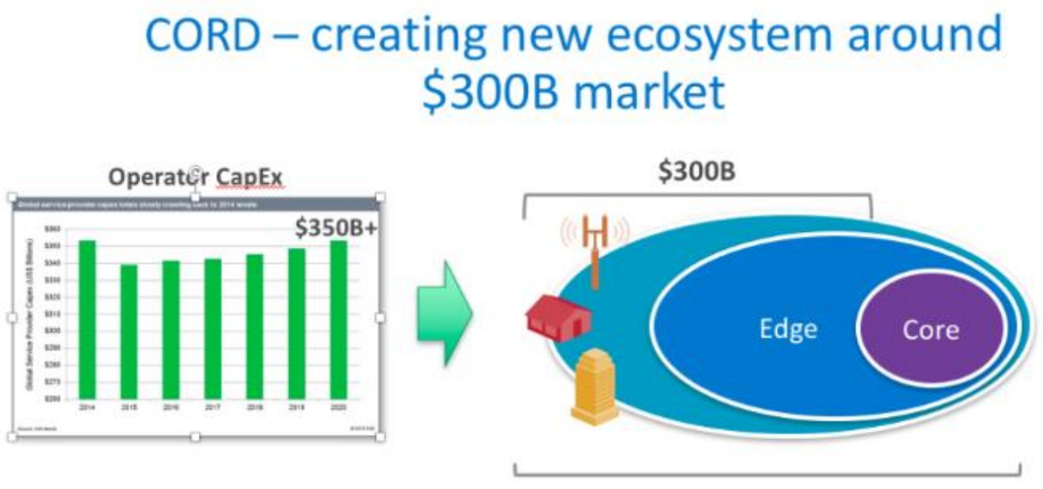


Figure 19: CORD Market Value

With CORD rapidly become the open source platform for the operator edge, the list of companies participating in CORD is becoming a "who's who" of operator networking -

including service providers, vendors, and system integrators. The companies like Cisco, Ericsson, Fujitsu, Huawei, NEC, Nokia, and Samsung are Incumbent Vendors of cord. Argela/Netsia, Ciena, Flex, Radisys are playing integrator part and ARM, Broadcom, Cavium, Intel are the silicon Manufacturers. CORD has made truly tremendous progress in last year with 60 participating companies and global operator interest. CORD is getting better, and the ecosystem keeps getting stronger.

Chapter 2

Software-defined Networking (SDN)

Software-defined networking (SDN) technology is a cloud computing approach that facilitates network management and allows network configuration that is programmatically efficient to improve network performance and monitoring. SDN is intended to address the fact that the static architecture of traditional networks is decentralized and complex, while current networks require greater flexibility and an easy problem-solving process. SDN attempts to centralize network intelligence in a single network component by separating the network packet (data plane) forwarding process from the routing process (control plane). The software that implements the networking function in a conventional network device has several roles. These roles can be classified as functional planes that relate independently using proprietary or open application program interfaces (APIs), following are the high- level classifications of these roles:

• **Control plane**: One of the roles is to determine the path the data must take as it flows through the device, the decision to allow or disable transiting data, the queuing behavior of these data, and any manipulation required for the data, etc. This is called the control plane.

• **Forwarding plane**: It out the transmission, queuing and processing of data throughout the device processing of data throughout the device on of data throughout the device on the basis of the control plane instructions. This role is referred to as a data plane or forwarding plane. Therefore, the control plane facilitates and decides the processing of data entering the device while the data plane takes action on the basis of these decisions.

• **Management plane**: While control and forwarding planes handle data traffic, the management plane is responsible for configuring, monitoring faults and managing the network device's resources.

• **Operational plane**: The device's operating state is monitored by the operating plane, which has a direct view of all device entities. The management plane works directly with the operational plane and uses it to collect device health information and to update the configuration to manipulate the device's operating state.

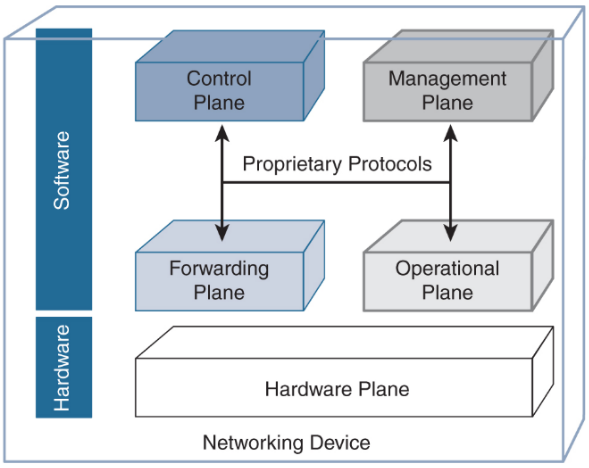


Figure 1: Overview of Planes in Traditional Networking Device

Let us use a router as an example to explain these concepts in the context. The management plane responsible for router configuration provides a mechanism to define parameters such as hostname, IP address of the interfaces to be used, routing protocol configuration, Quality of service (QoS) thresholds, and classifications, etc. The operating plane monitors the interface states, CPU usage, memory usage, etc. and communicates to the management plane the status of these resources for fault monitoring purposes**.** The routing protocol, defined by the management plane, runs on the router, forms a control plane and predetermines the data flow to populate the route lookup table (referred to as a routing information base or RIB) that maps these data against the router's exit interface. The forwarding plane uses this route search table and programs the data path through this router.

Since the control plane is integrated into the device software, the resulting network architecture has a distributed control plane where each node computes its own control plane. These control planes can exchange information between them, for example. Although the management plane is localized, network management systems (NMS) have played a role in centralizing it by adding another layer to the bundled management plane. Protocols such as Syslog, Simple Network Management Protocol (SNMP) and Net Flow have traditionally been used for monitoring and configuration using proprietary CLIs, APIs, SNMP or scripts. Figure 2 shows a picture of the deployment architecture.

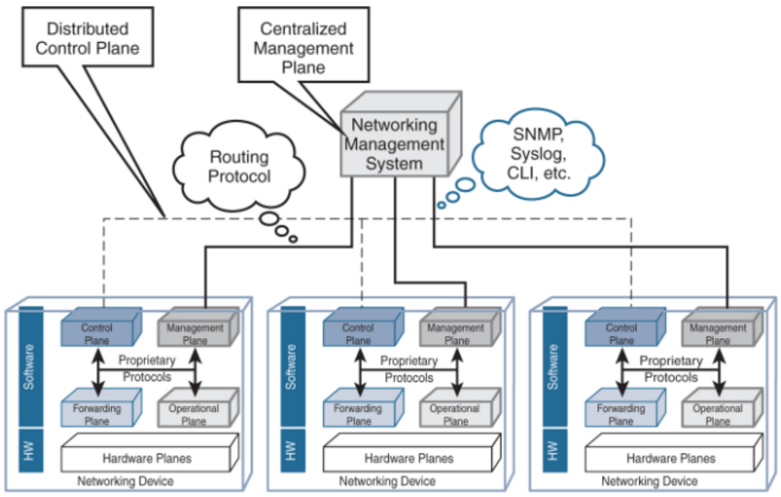


Figure 2 - Traditional Network Architecture Diagram

2.1 What is SDN?

In the traditional deployment of network devices presented earlier, the responsibility for decision making on the basis of the entire network state lies independently with each network device. This can potentially become a bottleneck whenever the ability to perform the control plane functions meets the limitations of the device, even if the bandwidth of the data plane is still underutilized. Furthermore, when a decision on a control plane involves information from more than one node, for example in the case of the resource reservation protocol (RSVP), further communication between the nodes is necessary to collect this information. This creates unnecessary overheads on the device.

SDN defines a methodology for centralizing the control plane from the network devices to a central device or cluster. This decouples the forwarding and control planes and discharges the control plane functions from the network devices, enabling the device to perform pure forwarding plane functions. SDN also aims to replace the proprietary interfaces between these planes with open industry-accepted communication protocols when the control and forwarding planes are separated. SDN, therefore, makes it possible for a vendor-neutral heterogeneous network where control plane interacts with multiple data plane implemented by different vendors.

The goal of SDN is to separate the control plane from the forwarding plane, but it does not require the centralized control plane to be confined to a single node. The control plane can be extended horizontally to form a control plane cluster for scalability and high availability purposes. The blocks in this cluster can communicate via protocols such as the Border Gateway Protocol (BGP) or the Path Computational Element Communication Protocol (PCEP)[1] and act as a single centralized control plane**. Figure 3** shows SDN concepts and changes in relation to traditional network architecture. Note that the figure does not emphasize the interaction between planes and hardware or the operational and management plane, as SDN focuses on control and forwarding planes.

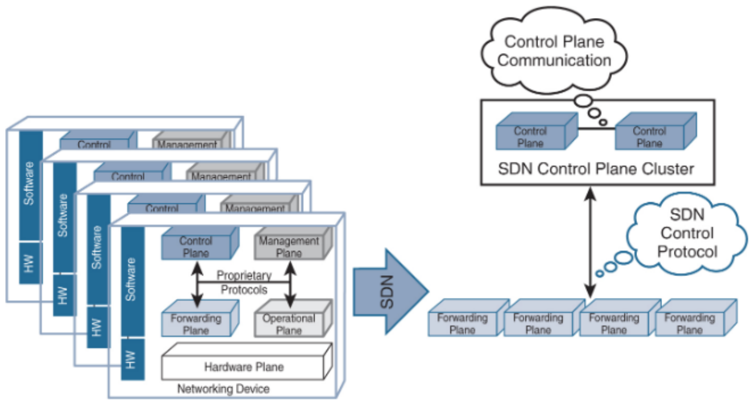


Figure 3: Transition from Traditional to SDN Architecture

2.2 Application Plane

The control plane can be managed by an application in an SDN-based implementation that can interact with both control and management planes. This application can abstract management plane device information and configuration, as well as network topology and traffic path information. Therefore, the application can have a comprehensive consolidated view of the network and use this information to make decisions that it can pass on to the control or management plans, as shown in **Figure 4**.

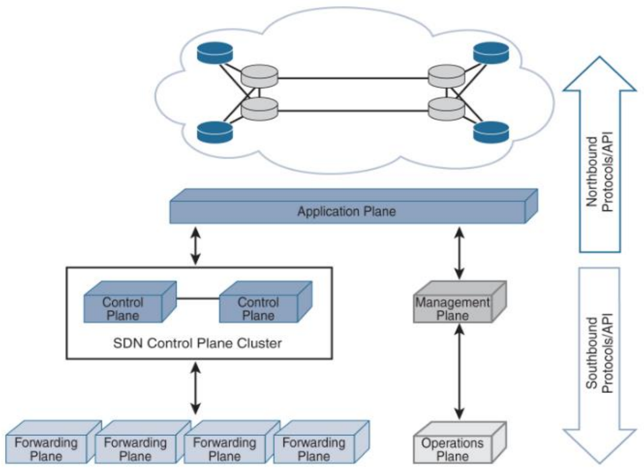


Figure 4: Application Plane and Northbound/Southbound Protocols

Bandwidth on demand could be an example of such an application. This application could monitor network traffic and provide additional traffic routes at certain times of the day or when a predefined threshold is crossed. The management plane would have to provide information on the status and use of the network interfaces in this application, while the control plane would offer the current real-time forwarding topology. The application then combines the information to determine whether additional routes for traffic of interest are to be provided. You can use user-defined policies to set the thresholds for this application. The application can communicate its action by instructing the management plane to take the new path and telling the control plane to begin its use.

Figure 4 also introduces the concept of northbound and southbound protocols and APIs. These terms relate to the contexts in which they are referred to, the SDN control plane and the management plane in this case. The southbound protocols, therefore, refer to communication between the control or management planes and the lower planes. The interface provided to the upper-level planes by management and control planes, such as application layers, is referred to as northbound APIs or protocols.

2.3 Advantages of SDN

When the SDN idea was initially introduced, its benefits were not sufficiently compelling to seriously pursue this direction for vendors or service providers. Scaled deployments still used partially automated supply and management mechanisms, and the tight integration of control and data planes was not seen as a major growth bottleneck. Therefore, SDN was initially regarded as an academic subject and not a practical, beneficial technological change. As the networking industry expanded and faced the factory increase in demand, the standards of the networking world began to look like constraints.NFV is an example of how the industry has become innovative and has adopted new techniques to overcome the barriers of vendor lock-in. SDN is another example of this. The adoption of SDN was driven by its potential to implement networks that are flexible, scalable, open and programmable.

1. Programmability and Automation

The ability to control the network through applications may be the most important advantages of SDN. Today's networks require greater network restoration agility, massive scalability, faster deployment and optimization of operating costs. The maximum use of automated tools and applications has become a requirement for meeting the demands of the network. Automation and programmability are necessary to support the provision of networks, the monitoring,and interpretation of device data and the implementation of changes in run-time based on traffic loads, disruptions, known and unknown occurrences. The methods available through vendors were traditionally specific to their devices or operating systems (OS) and offered limited (if any) support to enable an external device to make decisions based on logic and network constraints.

SDN offers a solution by connecting applications to the network and bridging the void with manual management and control processes. Since SDN places the intelligence in a central control device (i.e., the SDN controller), programs and scripts that automatically react to anticipated and unexpected events can be built into the controller directly. Alternatively, applications can run on the controller via the northbound APIs to pass the logic to the controller and the forwarding devices. The application can handle failures and increasing demand, promptly remediate and restore. This approach can reduce operating costs by significantly reducing service downtime, improving delivery time and increasing the ratio of devices to network operating personnel.

2. Support for Centralized Control

The centralization of the control plane facilitates the implementation of control logic because all the important information is easily available.SDN makes this consolidated network view possible. It simplifies the network control logic and reduces operational complexity and costs.

3. Multivendor and Open Architecture Support

SDN breaks vendor-specific control dependence by being a proponent of standardized protocols. The traditional proprietary methods offered by vendors for accessing and configuring devices are not easy to program and present a hurdle in the development of applications and scripts to automate some configuration and management processes**.** The applications must take into account changes and differences in the device interfaces, especially in a mixed-seller (or even mixed-OS) environment. Furthermore, if there are differences in how the vendors implemented a standardized control plane protocol (perhaps because of differences in interpretation), interoperability problems could arise. These challenges have existed in classical networks, but since SDN removes control from the devices and leaves only the data plane behind, it implicitly solves the problems of interoperability of control planes in the deployment of mixed vendors.

2.4 SDN IMPLEMENTATION AND PROTOCOLS

As I already explained the concept behind SDN separates the control plane from the forwarding plane and separates it. This can be implemented with a control plane in an external device (that is, called anSDN controller) and leave a forwarding plane on the data path devices. However, if you look deeper into the SDN philosophy, the basic goals of centralized control and offloading of the data plane can be achieved in different ways. A few possible implementations are introduced in this section. Let’s first formally introduce the SDN controller.

2.5 SDN Controller

The SDN controller is an independent device that acts as the SDN controller and communicates to the networking devices its control plane decisions. The controller also collects information from these devices in order to make informed decisions in the control plane. It uses protocols referred to as SDN control protocols to communicate with the devices.

The SDN controller (or controllers if more than one is used) must not be collocated geographically with the network devices, but they should be able to communicate with the network devices they control. Different SDN controller open source and commercial flavors are available.

2.6 SDN Implementation Models

It is not always technically feasible for vendors to separate the control plane completely from the networking devices and allow them to carry out pure forwarding functions. Therefore, vendors have taken different approaches to implementing SDN, which do not exactly correspond to the SDN implementations discussed to date. Service providers may have operational challenges that make it difficult to transform their networks completely into SDN and therefore have good reasons to accept one of these alternative SDN deployment methods. Until the SDN benefits are compromised and the implementation of the idea of separating the planes remains true, these approaches are valid ways of implementing SDN. Three ways to implement SDN are used.

1. Open (Classic) SDN

In this approach, the classic way of control and forwarding plane separation is implemented. Since the network devices built by vendors have not been able to do this, support for SDN is added by replacing the local control plane with a support layer for SDN. The new code is intended to work with the SDN controller and the device forwarding plane. It makes the device capable of communicating with an SDN controller using one of the SDN protocols. It also has the ability to manipulate the forwarding plane directly.

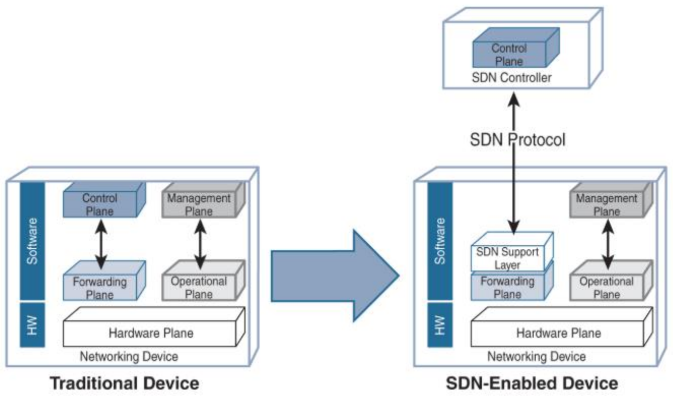


Figure 5: Open SDN

2. Hybrid SDN

Many vendors have approached the modification of the control plane of the device with anSDN support layer and claim that their devices are ready for SDN. This does not necessarily mean that the device's local control plane is no longer in existence. With the control plane implemented via the external controller, local intelligence can still be used**.** Since the device runs its own (distributed) control plane in this implementation and the external SDN controller complements the device's intelligence by modifying the routing parameters used by these protocols or directly modifying the forwarding plane, the approach is called the hybrid SDN. Figure 6 shows the hybrid SDN method. Note that the biggest difference here is the use of a local control plane on the devices compared to the classic SDN approach.

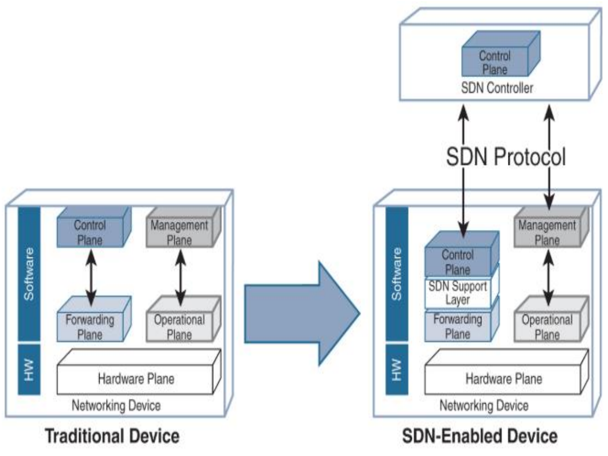


Figure 6: Hybrid SDN

3. SDN via APIs

Some vendors responded to SDN by providing API access for the supply, configuration,and management of devices. Therefore, these APIs allow applications to control the forwarding plane of the device and are equivalent to the southbound APIs used by controllers on network devices. Since APIs can be directly connected to the application, however, there may be no need for an SDN controller using standard southbound protocols in this implementation.

This is a shift to a more open and collaborative direction compared to the proprietary command-line interface (CLI) used by vendors. The implementation may not be genuinely open, however, because these APIs may not be compatible across vendors and the proprietary aspect is therefore not completely eliminated. Applications using this API-based SDN approach must be aware of the equipment of the vendors with which they communicate to use the correct APIs. The one way to approach is that the goals of SDN are still met by allowing applications to influence transmission decisions and the APIs are openly available to anyone who wants to create an application and use the APIs. This makes the network programmable but not necessarily flexible (since southbound APIs are proprietary).Some vendors have solved the problem of flexibility by offering their own controllers that use proprietary APIs southbound (toward networking devices) and APIs based on standards northbound to applications. Figure 7 shows a number of API implementation scenarios for SDN.

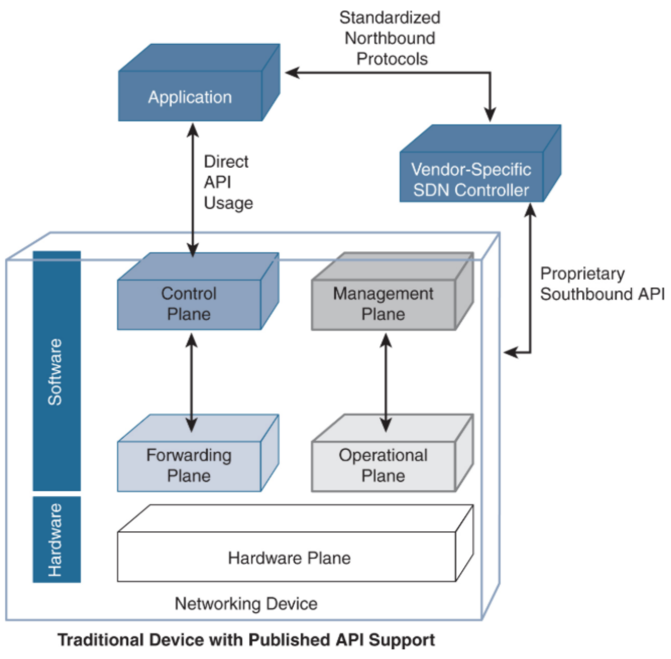


Figure 7: SDN via API

2.7 SDN Protocols

Regardless of the methodology used to implement SDN, certain types of protocols must be used to communicate and exchange information between transmission devices, applications, and controllers. From the SDN controller perspective, these protocols can be grouped into northbound and southbound protocols. As described above, southbound protocols are located between the control planes such as the SDN.

1. Southbound SDN Protocols

The southbound protocols can be further subdivided into two categories. This classification is based on the reasoning that the control plane can communicate directly with the forwarding plane or can influence the forwarding plane indirectly by using a management plane to change the parameters of the device. The protocols that interact directly with the forwarding plane called SDN control protocols.

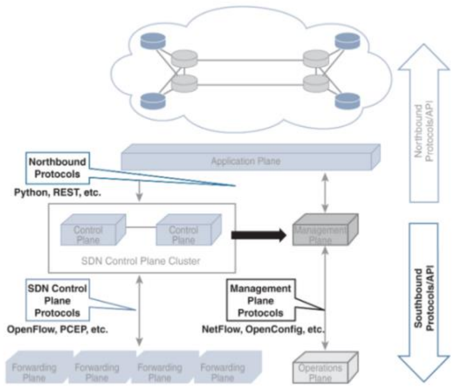


Figure 8: SDN Protocols

2. SDN Control Plane Protocols

The control plane protocols work on network devices at a low level and program the device hardware directly to manipulate the data plane. The most commonly referenced SDN control plane protocols include OpenFlow, Path Computation Element Communication Protocol (PCEP) and BGP Flow Spec [2].

2.8 OpenFlow Overview

In the same device, communication between the control plane and the forwarding plane takes place in the traditional vendor-developed network device. These devices use proprietary protocols for communication and internal procedural calls. In the SDN environment, a standard protocol with multivendor support was required to communicate between the control plane and the forwarding plane. For this purpose, OpenFlow was developed. OpenFlow was the first open source control protocol to communicate the forwarding plane between the SDN controllers to the network devices

OpenFlow has evolved from the initial lab version and developed into production-grade software with versions 1.3 and higher. OpenFlow maintains what it calls a flow table on the device containing information on how the data is to be transmitted. The SDN controller can then use OpenFlow to program an OpenFlow-enabled switch's forwarding plane by changing this flow table.

The OpenFlow architecture supports two modes of operation, reactive and proactive, to program the forwarding information and set the path across the network. The reactive mode is the default implementation method for SDN with OpenFlow and assumes that there is no intelligence or fragment of a control plane running on the network devices. In this mode, the first data traffic packet received on any of the forwarding nodes is sent to the SDN controller, and the SDN controller uses this information to program the flow through the entire network. This creates the flow table and switches the data traffic accordingly on all subsequent devices in the path. The SDN controller is pre-configured in a proactive mode with some default flow values, and the traffic flow is pre-programmed as soon as the switch is raised.

As the SDN controller and switches exchange information flow over the network, it is recommended that a secure channel such as Secure Socket Layer (SSL) or Transport Layer Security (TLS) be used to connect OpenFlow.

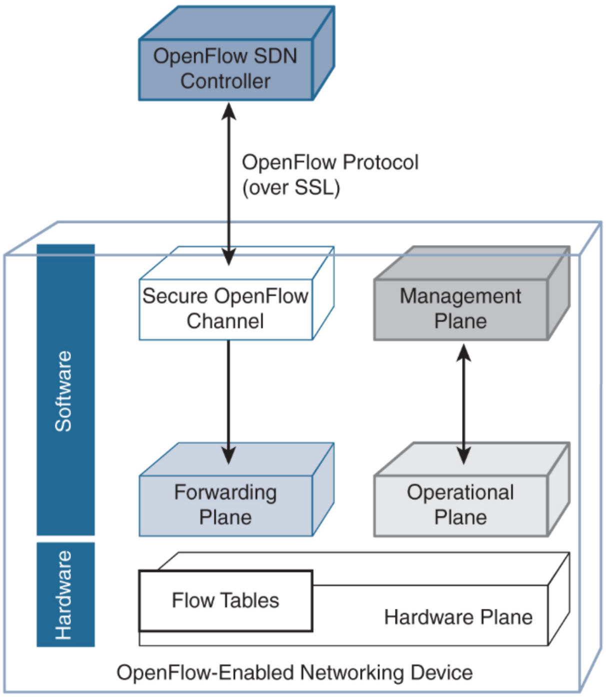


Figure 9: OpenFlow Architecture

2.9 OpenConfig Framework

OpenConfig is a framework that supports a vendor-neutral network device programmatic interface. It was launched by a forum of network operators like Google, AT&T, BT and so on. Drive this industry effort to create a case-based use model for programmatically configuring and monitoring a network device.

OpenConfig uses YANG modeling [3] as a data transport standard. It does not specify any underlying operations protocol, but NETCONF has been adapted to support the OpenConfig framework by a few vendors. OpenConfig also supports network monitoring by supporting streaming device telemetry data.

2.10 Northbound Protocols

These protocols are used as the interface between the SDN controller and the application above, as shown in Figure 10 the usual purpose of the applications is to perform service orchestration or to make and implement decisions based on the logic or policy defined in the application. The communication between the SDN controller and the application is no different from that between two software entities, so no special new protocol is required. The existing commonly used protocols, such as RESTful APIs or libraries in programming languages such as Python, Ruby, Go, Java or C++, are used for northbound communication.

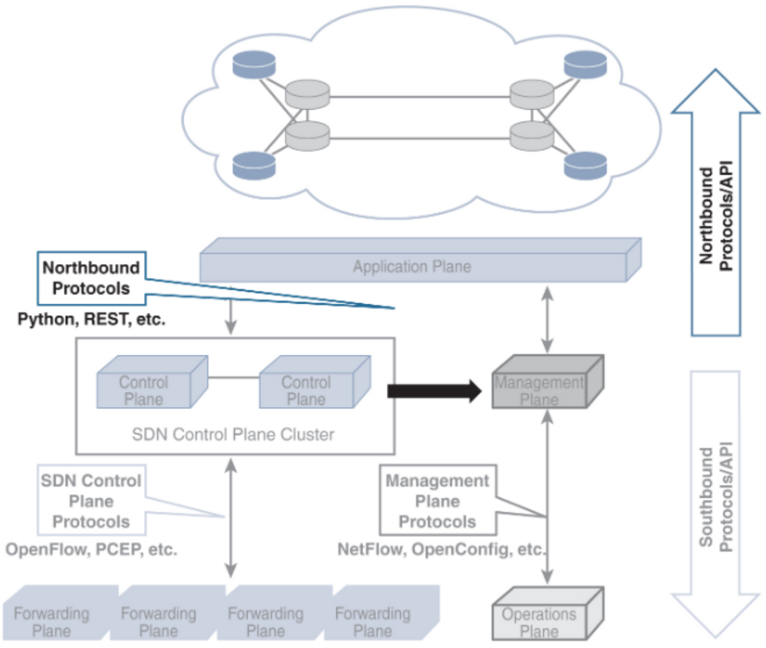


Figure 10 - Northbound Protocols/APIs

2.11 SDN in Wide-Area Networks (SD-WAN)

Enterprise and business customer networks are spread across geographical regions, with multiple branches connecting to headquarters. These sites use dedicated broad area networks (WAN) circuits, such as T1 or T3, or dedicated lines from a VPN service provider to connect between different offices. These links or services have high price tags and increase network operating expenditure. In order to reduce overhead costs, companies have moved their connectivity to secure Internet connections through the use of new technologies such as Dynamic Multipoint VPN (DMVPN) or MPLS VPN. The overlay network can be dynamically set up with additional data protection encryption, allowing the use of a shared Internet connection for the private traffic of the enterprise. However, Internet connectivity cannot guarantee SLA. It may also not be considered suitable for the exchange of highly sensitive company data, despite the encryption. This approach, therefore, reduces the requirements for bandwidth on the dedicated link but does not completely eliminate the need for private WAN links. The traffic can be controlled to either the Internet link or the dedicated WAN links based on the SLA required or the sensitivity of the data. In addition to the primary dedicated link, the customer has a cost-effective way to connect different branch offices and headquarters as shown in figure 11.

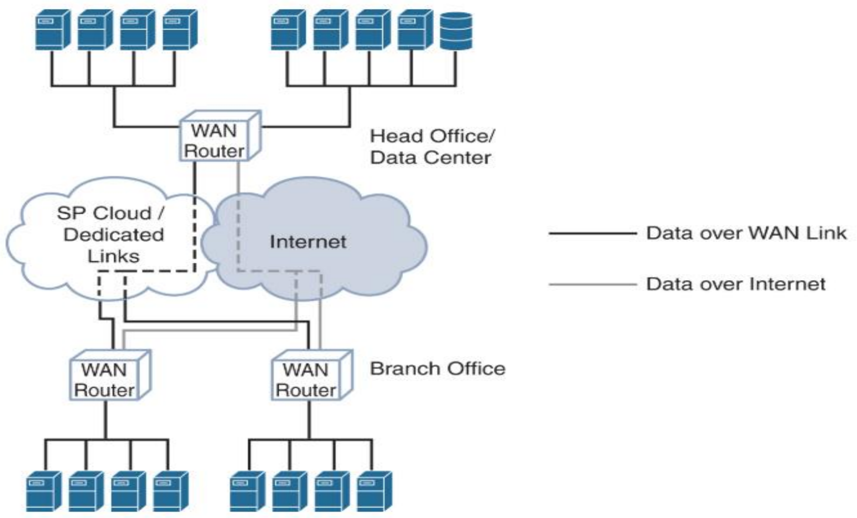


Figure 11: Enterprise WAN Connectivity

2.12 Problems and Challenges for SD-WAN: For large- scale WAN deployments with perhaps hundreds of sites, maintaining interconnectivity between branches while still using divided connectivity between dedicated and linked links becomes a complex task. The policy that determines which types of traffic can be diverted to the Internet connection must also be managed at each location, and this alone can become an overhead management. It is not easy either to optimize these policies on the basis of real-time measurements or to make those policies dynamic with frequent changes. The cost and performance benefits were clearly visible, but without a management system that could centrally manage the flows and configure the WAN routers, it was not a feasible objective to achieve this result.

2.13 SDN Solution

Using the SDN model of a centralized network topology for the deployment, the WAN network with multiple links of connectivity can be abstracted to a central management system, which is termed a controller. The controller can monitor the SLA on the Internet links and instruct the branch or head offices to utilize the right links for the data. It also makes it possible to manage the traffic flows based on traffic type and nature of sensitivity and to split the traffic to utilize the dedicated circuit and the Internet link efficiently. An additional advantage of link utilization on the remote routers can be taken into consideration to make the decision of traffic flow egressing the source router. Therefore, the operational expense savings that were otherwise very hard to realize are now achieved using the SDN-based solution. This solution is referenced as SDN for WAN, or simply SD WAN. Vendor's offerings such as Viptela's vSmart, Cisco's IWAN, or Riverbed's Steel-Connect are some examples of commercial offerings of SD WAN. Figure 12 offers a pictorial view of this solution. As shown, a central SDN WAN controller manages the WAN routers to monitor performance for uplinks and alter their traffic flows using centrally managed policies based on link performance, nature of traffic, time of day, etc.[7][11]

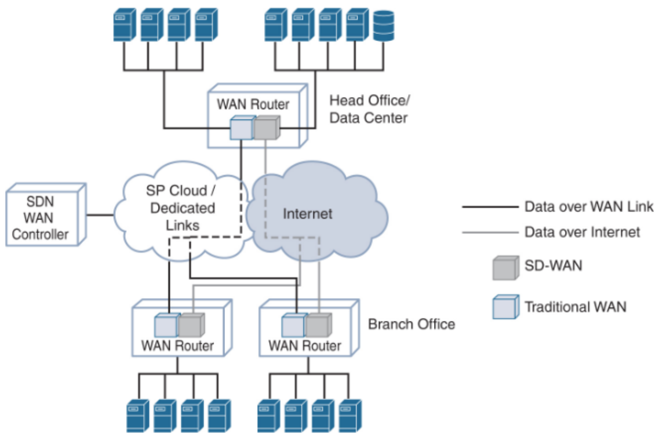


Figure 12: SD WAN

2.14 Enterprise SDN:

An enterprise network consists of network devices spanning local LAN and WAN connectivity. Depending on the scale of the enterprise's business the local-area network (LAN) could consist of wireless and wireline for connectivity to the computers, printers, voice/video endpoints, and another network device. The WAN connectivity can have a dedicated WAN link or an internet link to connect to the remote branch office or data centers. Several services are deployed on the network, such as an internal office and external world communication voice network, a data network connecting local users and data storage in a private data center or public cloud. Large companies separate their networks according to departments such as engineering, finance, marketing, sales,and partners. Figure 13 shows a picture of the network of this company.

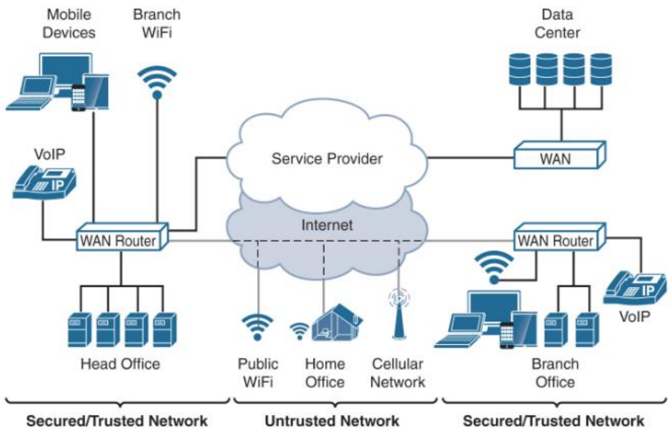


Figure 13: Typical sketch for Enterprise Network

2.15 Problems and Challenges for Enterprise SDN

The requirements for enterprise networks may require different policies depending on whether they are accessed via LAN or WAN. The deployment and management of this type of network architecture require a high degree of security and flexibility. When private cloud architectures are adopted by companies, network access, firewall, and security practices must evolve to benefit from this deployment of private clouds. With so many new types of business agility models such as BYOD (bring your own device) and the proliferation of computer devices (laptops, mobile devices, tablets, etc.) and the ability to access the network from any device anywhere, the IT department needs to support this whole architecture without affecting the network. The security policy for accessing the network is extremely dynamic and complex. For example, user access devices, such as laptops running multiple Oss, require different security procedures, the network transport layer requires VPN and encryption and data privacy on data center servers. The user policy must be provided at all user access points, and the QoS policy must be implemented throughout the entire network.

Since the user implements wireless and mobile device network transport has a fluid nature of network access, all these policies must follow the access ports on the edge of the network to which the user connects.

2.16 Enterprise SDN Solution: The enterprise challenges are being addressed using SDN’s centralized model of the network view and its ability to program the network from a single source. For BYOD, the SDN controller can detect the presence of the device's entry into the network and, based on the user or device; the corresponding profile can implement the device's access policy and also program the edge and transport of the network with the appropriate policy implementation. With this approach, which creates a dynamic policy from a central source, the company customer can now provide its users / employees or partners with the ability to access the network from any location without having to be present physically in a single office / desk. Without the SDN model, it would have been a difficult task for IT staff to dynamically configure QoS / security on the network, on the basis of each device or user. If the organization has thousands of employees and is spread over several locations, such a dynamic configuration with traditional methods is an almost impossible task. In addition, SDN plays an important role in protecting the enterprise network from any DDoS or other security attacks. In this role, when an attack signature is detected or learned from any source, the SDN controller can mitigate this security risk throughout the entire company network and protect business data. This use of the BGP flow specs to handle DDoS attacks. Once the attack traffic is identified, this can be notified to the BGP server running on the SDN controller, and the server can then use BGP-FS to instruct the peering edge devices to drop or deflect (for scrubbing) all the traffic that matches the attack traffic profile. Without this method, this is done today by injecting a static route (which can only be used to match the destination of traffic) for a remote triggered black hole (RTBH) to adjust the routing of each edge device to divert traffic towards scrubbing centers.[6]

Enterprise networks are embracing the new network architecture in combination with SDN. It is bringing business benefits such as the self-service IT, lower operational expenses, security, and compliance with scale and agility.

2.17 CORD—An Example of NFV and SDN Working Together:

CORD is a joint effort between AT&T labs and ON.Lab to build a new architecture to transform the way a central telecommunications office is built. It is considered to be a platform for the scalable and agile deployment of next-generation network services.CORD makes use of both SDN and NFV as core components of the architecture, as well as open orchestration tools and application programmability combined with data center deployment architecture concepts. The combination of SND and NFV makes it the perfect example of how these technologies are used collectively to revolutionize the design, implementation,and deployment of new networking services. Both NFV and SDN are focusing on open source software and breaking vendor boundaries. This is also true for CORD because the CORD architecture is essentially knitted together on the merchant's silicon-based generic hardware from open source software.[16]

The CORD architecture consists of an entire stack from hardware, software, and service orchestration. ONOS is used as the SDN controller. Network services are implemented by the VNFs running on COTS hardware, OpenStack performs NFVI orchestration, and an open cloud operating system, XOS, stitches these pictures together to make it possible to create, manage, run, and offer services using all these elements. Figure 14 presents a generic diagram for the CORD architecture. Commodity servers and the underlying network are built into spine - leaf architecture in the data center style. The SDN controller and the applications on top perform control plane functionality, while XOS and OpenStack perform service and NFVI orchestration accordingly.

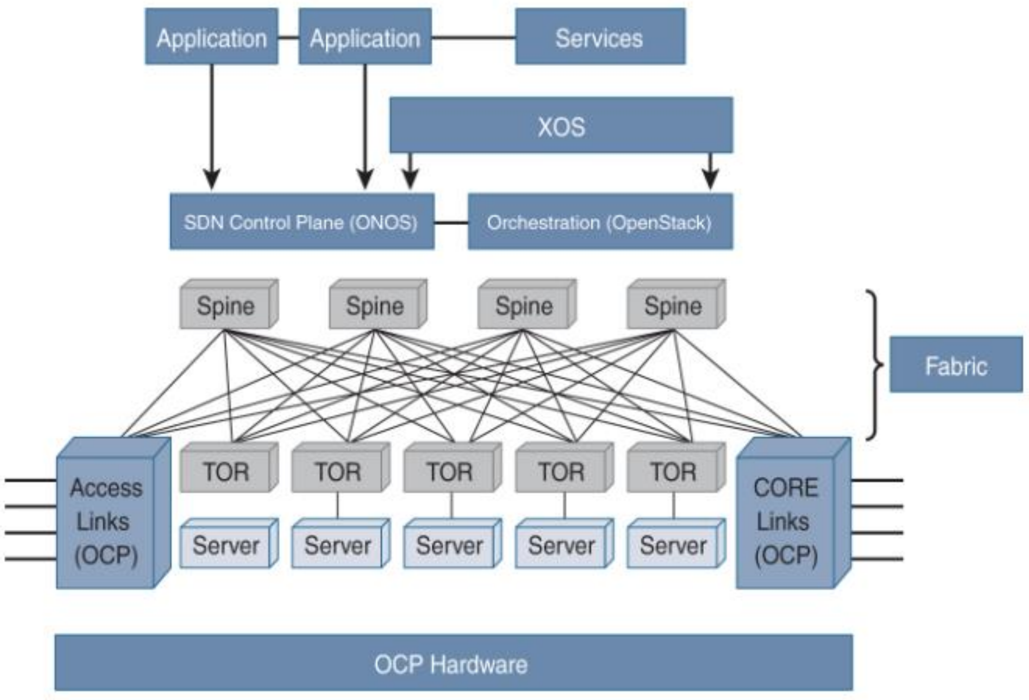


Figure 14: CORD Architecture

Different networking segments have started to utilize CORD infrastructure to build use cases for their specific domains. For example, mobile vendors are looking to leverage CORD for mobile 5G-based services, called M-CORD. In M-CORD, the concepts of NFV are applied by virtualizing the Mobility Management Entity (MME), Serving Gateway (SGW), and other elements into their virtual equivalents, while the use of a methodology like SD-WAN optimizes the network usage by deflecting traffic to caching servers or the Internet as needed. Other domains, such as Enterprise CORD (E-CORD), are being tapped to offer customized on-demand networks by using vFW, vLB, as well as SDN concepts to make it programmable and adaptable.

Chapter 3

NFV (Network functions virtualization)

The server virtualization approach is already proven in data centers, where stacks of independent server hardware systems have mostly been replaced by virtualized servers running on shared hardware.

NFV builds on this server virtualization concept. The concept extends beyond servers and extends the scope to include network devices. The ecosystem can also manage, supply, monitor and deploy these virtualized network entities.

NFV to refer to the entire ecosystem comprising virtual network devices, management tools and the infrastructure that integrates these computer hardware components. However, NFV is more precisely defined as the method and technology that allows you to replace physical network devices with one or more software programs that perform the same network functions while running on generic computer hardware. An example is the replacement of a physical firewall device with a virtual machine based on software. This machine provides the firewall functions, runs the same operating system and looks and feels the same - but on unused, shared and generic hardware.

NFV can implement network functions on any generic hardware that provides the basic resources for processing, storage, and data transmission. Virtualization has matured to the extent that the physical device can be masked, enabling commercial off-shelf (COTS) hardware to provide NFV with the infrastructure.

3.1 COTS (Commercial Off The Shelf)

Commercial off-shelf (COTS) means any product or service which is commercially developed and marketed. COTS hardware refers to general purpose computing, storage and networking equipment that is constructed and sold for any use that requires such resources. The use of proprietary hardware or software is not required.

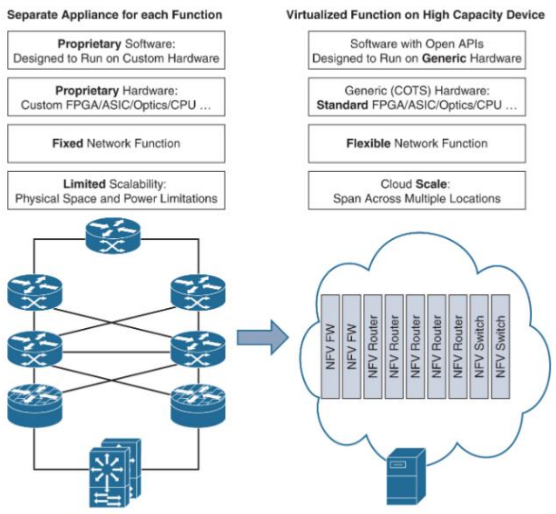

Separate Appliance for each Function
Proprietary Software: Designed to Run on Custom Hardware
Proprietary Hardware: Custom FPGA/ASIC/Optics/CPU ...
Fixed Network Function
Limited Scalability: Physical Space and Power Limitations
Virtualized Function on High Capacity Device
Software with Open APIs Designed to Run on Generic Hardware
Generic (COTS) Hardware: Standard FPGA/ASIC/Optics/CPU ...
Flexible Network Function
Cloud Scale: Span Across Multiple Locations
NFV FW
NFV FW
NFV Router
NFV Router
NFV Router
NFV Router
NFV Router
NFV Switch
NFV Switch

Figure 1: Transition to NFV

Most of the network vendor not concerned about the hardware on which their code is run, as the hardware is developed, customized and used for the specific network function as dedicated equipment. They have full control of both the hardware and the software on your device. This enables vendors to design hardware and its performance factors in accordance with the roles these devices play in the network.

In the case of virtualized network functions, assumptions about the capabilities that hardware has to offer are not realistic, and it is not possible to integrate very closely with the bare hardware.

Network virtualization opens up new opportunities for the deployment and management of networks. The flexibility, agility, savings in capital and operating costs and scalability that NFV offers opens up new innovations, design paradigms, and new network architectures.

3.2 NFV Architecture

Traditional network architecture that defines devices is quite basic, as both hardware and software are customized and integrated closely. In contrast, NFV enables vendor-developed software to run on generic shared hardware to create multiple management touch points.

The NFV framework is developed to ensure that these contact points are standardized and compatible with each vendor's implementation. Here I explain the framework and rationale behind its blocks. Understanding the framework allows readers to see NFV’s flexibility and freedom of choice.

3.2.1 Need for a Framework

The architecture is fairly basic since both the hardware and software are customized and tightly integrated. NFV alsoallows software developed by the vendors to run on generic shared hardware creating multiple touch points for management.Network functions virtual implementation is referred to as virtualized network function (VNF). A VNF function is an example. Router, switch, firewall, load-balancer, etc.

The architecture is quite basic, as both hardware and software are customized and integrated closely. In contrast, NFV allows vendor-developed software to run on generic shared hardware to create multiple management touch points. The virtual implementation of network functions is called a **virtualized network function (VNF**) in the NFV jargon. A VNF is intended to perform a specific network function, for example. Router, switch, load balancing, firewall, etc.

3.3 VNF (Virtualized network function)

These VNFs can be offered by different vendors,and service providers can select a combination of vendors and functions that best suit their needs. They provided need to creates the need for a standardized method of communication and management in the virtual environment between VNFs. The management of NFV shall take the following considerations into account:

• Implementations of VNFsfor the use of multivendor.

• life cycles and interactions of these functions are being managed.

• managing the hardware resource allocations

• monitoring the utilization

• The configuration of the VNFs

In order to implement these management roles and maintain the system open and non-proprietary, a framework for standardization needs to be defined. This standard framework should ensure that the deployed VNF is not connected to specific hardware and does not need to be adapted to any environment. It should provide vendors with a reference architecture that can be consistent and consistent with the deployment methods of any VNF they implement. In addition, it must ensure that the management of these VNFs and the hardware on which they are running does not depend on a vendor. In order to implement the network functions in this heterogeneous ecosystem, no special modification should be required. In essence, this framework must provide the architectural basis for the seamless operation of VNFs, hardware and management systems within the well-defined limits.

3.4 ETSI Framework for NFV

NFV was first introduced by a consortium of key service providers at the SDN OpenFlow World Congress in 2012.They referred to the main challenges facing network operators, in particular, their dependence on introducing new hardware to enable their customers to offer innovative services. The group focused on the challenges of the following concepts

• design changes around the new equipment

• need for expertise to manage and operate the new proprietary hardware and software

• new proprietary equipment need to deal with hardware complexity

• restarting the cycle before returns from the capital expenses and investments are fully realized

The group proposed NFV to tackle these challenges and improve efficiency by using standard IT virtualization technology to consolidate many types of network equipment into the industry- standard high- volume servers, switches and storage located in data centers, network nodes and end- user locations.[1]

ISG group was formally launched in early 2013 to define requirements and an architectural framework that can support the virtualization of network functions carried out by custom hardware devices from vendors. Following are some points used by these groups:

• **Decoupling:** complete separation of hardware and software

• **Flexibility:**Function in networks need to implemented properly and in a scalable manner.

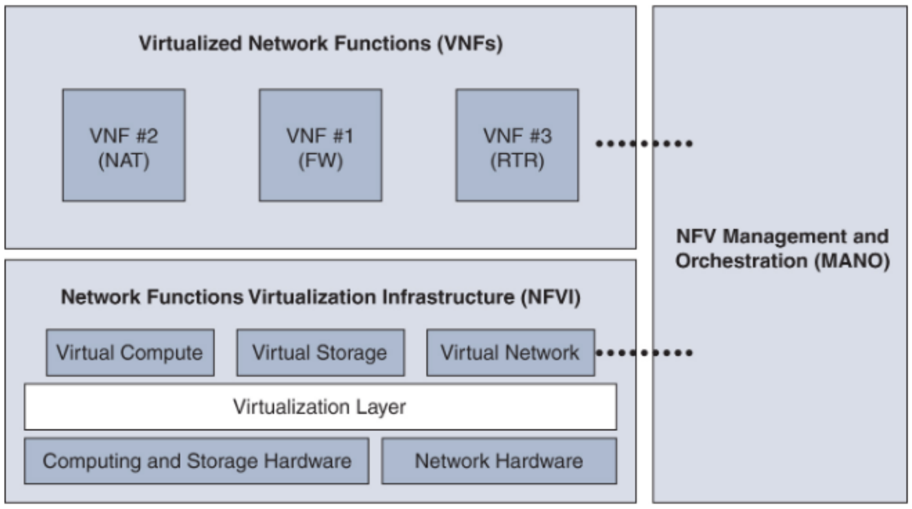

Virtualized Network Functions (VNFs)
VNF #2 (NAT)
VNF #1 (FW)
VNF #3 (RTR)
NFV Management and Orchestration (MANO)
Network Functions Virtualization Infrastructure (NFVI)
Virtual Compute
Virtual Storage
Virtual Network
Virtualization Layer
Computing and Storage Hardware
Network Hardware

Figure 2: High-Level ETSI NFV Framework

Its architectural framework forms the basis of the work of standardization and development and is often referred to as the ETSI NFV framework. The framework at a high level covers the management of VNFs, relationships,and interdependencies; data flows between VNFs and allocation of resources.ETSI ISG classified these roles into three high-level blocks, namely a block of infrastructure, a block of virtualized functions and a management block. [2]

3.5 ETSI Framework

The ETSI framework and the process of thinking behind its high- level blocks can be better understood by examining the construction process that led to this framework. Let's start with the basic concept of NFV, like virtualizing the network device function. This is done by VNFs.

VNFs can be deployed as standalone entities or as a combination of multiple VNFs for the implementation of the network service. The protocols for the function virtualized in a VNF need not be aware of the virtualized implementation. As shown in Figure 3, the VNF implementing the firewall service (FW), NAT device (NAT) and routing (RTR) communicate without knowing that they are not physically connected or run on specific physical devices.

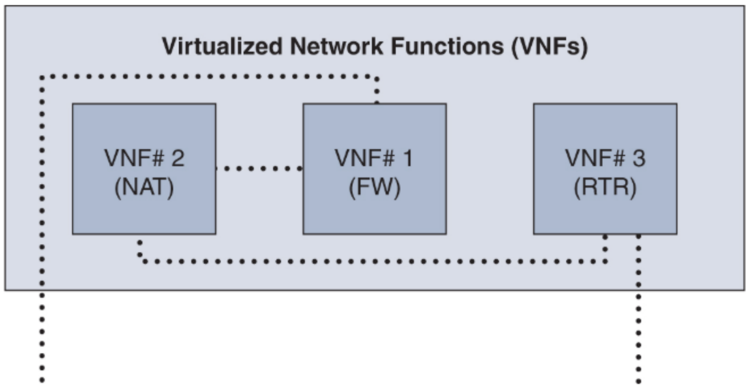


Figure 3: Network Functions Working as VNFs

Since no dedicated or custom hardware is designed to run these VNFs, a generic hardware device with generic hardware resources such as a processor (CPU), storage, memory,and network interfaces can be used to run these VNFs.This can be done using COTS hardware. It must not be a single COTS device; it can be an integrated hardware solution that combines the hardware resources required to run the VNFs. [3]

Hardware virtualization offers the VNF an infrastructure to run on. This NFV (NFVI) infrastructure can use COTS hardware as a common resource pool and develop subsets of these resources to create "virtualized "computing, storage and network pools that can be allocated as required by the VNFs.

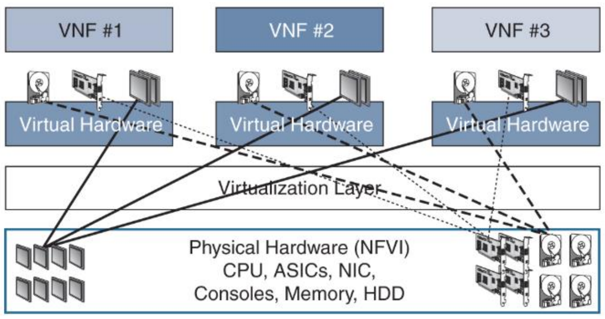


Figure 4: Virtual Computing, Storage, and Networking Resources for VNF

The VNF vendor recommends a minimum requirement for the resources that should be made available for its implementation, but the vendor cannot control or optimize this hardware parameter. For example, the vendor can make a recommendation on the CPU cores necessary to execute the code or storage space and the memory that the VNF needs but the vendors are no longer free to design the hardware according to their specific needs. The physical hardware virtualization layer can accommodate the VNF resource request. The VNF has no visibility in this process, nor is the VNF aware of the existence of other VNFs that can share their physical hardware.

In this virtualized network architecture, multiple resources are now available for different levels of management and operation. In comparison, the management of today's network architecture is vendor-specific and has limited vendor knobs and data points. Any new requirements or improvements in management capacity can only be achieved with the support of the vendor. With NFV, the entities can be managed on a more individual and granular level. Therefore, the NFV architecture would not be complete without defining the methods for managing, automating, coordinating and interconnecting these layers and function blocks in an agile, scalable and automated manner.

This leads us to use one more functional block to the framework that communicates with the VNF and NFVI blocks and manages them, as shown in Figure 5.This block manages the deployment and interconnection of VNFs on the COTS hardware and assigns the hardware resources to these VNFs.

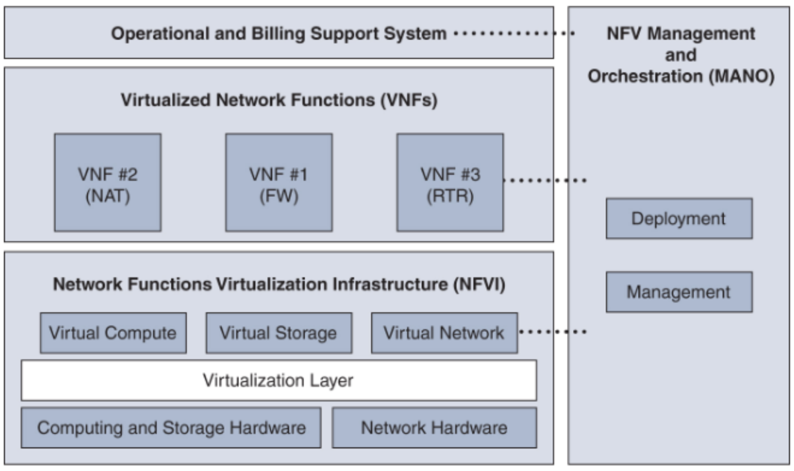

Operational and Billing Support System
NFV Management and Orchestration (MANO)
Virtualized Network Functions (VNFs)
VNF #2 (NAT)
VNF #1 (FW)
VNF #3 (RTR)
Deployment
Management
Network Functions Virtualization Infrastructure (NFVI)
Virtual Compute
Virtual Storage
Virtual Network
Virtualization Layer
Computing and Storage Hardware
Network Hardware

Figure 5: Management and Orchestration Block for NFV

Since the MANO block is intended to give the entities full visibility and is responsible for managing them, they are fully aware of their utilization, operational status and usage statistics. This makes MANO the most appropriate interface for the operating and billing systems to collect the data for use.

3.6 NFV Framework

The objective of defining the framework and, more specifically, the individual function blocks and reference points is to eliminate (or more realistically reduce) interoperability challenges and standardizes implementation. In the framework, the purpose and scope of each of these blocks are well defined. In the same way, interdependencies and paths of communication are defined by reference.

Vendors can develop these functions independently and use them to function smoothly with other function blocks developed by other vendors. Theis help in scope and role defined by the framework communicate with the other blocks using open methods at the points of reference; the NFV can be deployed heterogeneously. This means that service providers can choose between vendors for different function blocks in complete flexibility. This contrasts with the traditional deployment of networks, in which service providers were linked to the hardware (and its limitations) and software of the vendor (and the challenges to adapt to all operational needs) and had to deal with the interoperability concerns of mixed-vendor networks. NFV provides service providers with the ability to overcome this limitation and deploy a scalable and agile network using hardware and NFV function blocks from any vendor combination.

This magically does not eliminate the higher-level protocol interoperability problems that may arise between VNFs implemented by various vendors. For instance, a BGP implementation by a vendor of a VNF may have some problem when it looks at another VNF developed by another vendor. There is already a standardization process, and it will continue to play a role in these types of interoperability problems. In addition, NFV does not require vendors to provide an open and standard way to manage the VNF configuration and monitoring. That's compensated by EM in the NFV framework. However, the operational support system should be able to work with VNFs using standard methods in an implementation closer to the ideal. This takes place through a parallel shift in technology to software-defined networking (SDN).Although NFV and SDN are not interdependent; they complement each other's benefits and advantages.

3.7 NFV Benefits

Limits related to the use of traditional network equipment have been identified. The virtualization of network functions addresses most of these restrictions directly and offers many additional benefits. It provides a framework for completely transforming the architecture, deployment, management, and operation of networks while offering many levels of improvement and efficiency. Figure 6 lists a few of the benefits that NFV offers that are discussed in the sections that follow.

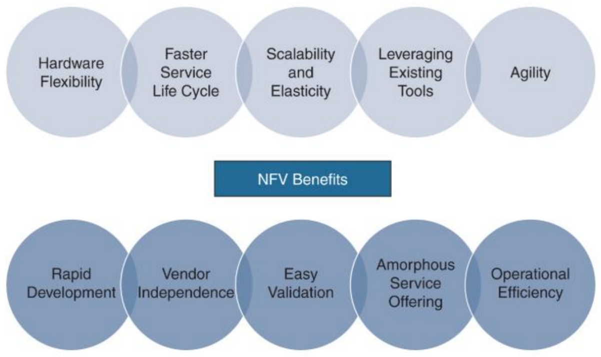


Figure 6: Network Functions Virtualization Benefits

1. Hardware Flexibility

Since NFV uses regular COTS hardware, network operators can choose and build the hardware as efficiently as possible to meet their needs and requirements.

Hardware offered used by network vendors traditionally has very limited options for its computing, memory, storage and networking capabilities, and any changes lead to a hardware upgrade costing operators time and money. With NFV, suppliers can now choose between many different vendors and have the flexibility to select the optimal hardware capabilities for their network architecture and planning. For example, if the Internet gateway used runs out of the capacity to store the entire Internet table and needs a memory upgrade, it can only be achieved in most current implementations by upgrading the controller or upgrading the full device. More memory to the VM can be given by a provider that hosts this VNF in NFV.

2.Faster Service Life Cycle

Unlike physical hardware, the VNFs can be created and deleted by flying. The VNFs can be much shorter lifespine and dynamic compared to physical devices, as these functions can be added when needed, easily supplied by automated software tools that do not require on-site activity and then released as soon as the need is over. It contrasts with the deployment effort required to add a new function to an existing network that would have required an on-site physical installation that could be time consuming and expensive. One of the most important advantages of NFV is its ability to add new network functions (deployment agility) quickly.Services can now be commissioned or shut down by pressing a button without the need for a delivery truck, reducing deployment times dramatically from weeks to minutes.

3. Scalability and Elasticity

New services and capability-hungry applications in today's networks are keeping network operators, especially cloud providers, up to speedy consumer demands. The service providers have met these requirements, and it takes time, planning and money to increase the traditional network equipment capacity. This is solved by NFV, which allows changes in capacity by providing a way to increase and reduce the resources used by VNF.For example, if any of the VNFs requires additional CPU, storage or bandwidth, it can be requested from the VIM and from the hardware pool allocated to the VNF. In a traditional network device, a complete device replacement or hardware upgrade would be required to change any of these parameters. However, since VNFs are not restricted by custom physical hardware, they can offer this elasticity. Therefore, networks need not be substantially over provided to meet capacity requirement changes.

4. Leveraging Existing Tools

Since NFV uses the same infrastructure like data centers, it can reuse and leverage existing deployment and management tools in data centers. The use of a single centralized glass panel for virtual network management and virtual servers offers the advantage of faster adaptation for new deployments without the need to develop new tools and thus eliminates the costs of deploying, familiarizing and using new tools.

5. Rapid Development and Vendor Independence

Since NFV offers the means to easily deploy a different vendor solution without the heavy costs associated with replacing the deployment of an existing vendor, network operators are not locked in a specific vendor. Operators cans see and compare vendors and functions and choose between them based on the availability of features, software licensing costs, model of support for post-deployment.

New solutions and features can be quickly produced without waiting for the currently deployed supplier to develop and support them. Furthermore, NFV's inherent support for the use of open source tools and software facilitates this rapid deployment.

6. Validation of New Solutions

Service providers are looking for new solutions, services,and functions before introducing them into their production networks by deploying them in test setups. They have traditionally had to replicate a subset of their production environment for in-house tests, increasing their operating budget. The construction and management of such test installations have become much more cost-effective with NFV.

7. Amorphous Service Offering

A deployment based on the NFV is not limited to a single design and deployment. It can adapt to specific market needs and offer a targeted range of services to meet changing requirements. By combining elasticity and deployment agility, the location and capacity of network functions can be quickly changed, and workload mobility can be achieved. For example, suppliers can implement a "follow the sun network "using constantly moving, day-to-day virtual machines and spin up or expand new VNFs to meet the service and capacity requirements of the network as they change during peak and off-peak use or when major events take place in any geographical region.

8. Operational Efficiency and Agility

Common hardware hosting different VNFs can centralize business tasks such as inventory management, procurement process. This reduces the operational overhead compared to the separate deployment of multiple hardware devices for different network services.

NFV is inherently automation friendly and can enhance the benefits of using Machine to Machine tools (M2M).For example, it can be used to monitor a device with an automation tool to determine the need for more memory in a network function. With NFV, this tool can go ahead and ask for this memory to be allocated without any human intervention.

Network activates related to maintenance can also benefit greatly from NFV by reducing possible downtime.NFV enables the spin-up of a new VNF, transfers the workload temporarily to that VNF and releases existing VNF for maintenance. This makes it possible to achieve 24/7 self-healing networks in-service software upgrades (ISSU) and minimize the operational loss of revenue due to network failures.

9. Movement to Cloud

With the advent of new smart devices, bandwidth-hungry applications, the new generation of connected devices and IoT technologies, the demand for and use of networks has exponentially increased. These recent changes have created market demand for services on any device, anywhere and anytime. In order to meet this market shift, suppliers aim to develop and offer cloud-based services that meet the new needs.

Research publications forecast that between 2015 and 2020, the NFV market will grow to beyond $9B with a compound annual growth rate (CAGR) of 83.1%. This is a huge market for the traditional providers to miss, and many new providers are now jumping into this market, such as cloud providers, service providers, enterprise, start-ups, etc. [4]

3.8NFV MARKET DRIVERS

NFV is more than a technology for transformation. Like any new technology which has brought major changes and new advantages, NFV has had to accept and adopt the market. The drivers of the NFV market are very important, clear and promising. These have helped NFV to move beyond its infancy in research laboratories and to put it into mainstream deployments in a very short period.

Internet access and the worldwide trend towards digital services create a large market for network service providers. The scale and bandwidth requirements are already tightening up the existing infrastructure. The improvement of this traditional network infrastructure requires a high level of time, money and suppliers ' resources. This has forced suppliers to rethink the network architecture and use innovations that can keep pace with the new world of cloud and digitization .One of the main drivers is the cloud technology movement coupled with the use of mature technology such as Virtualization and COTS hardware. Network providers now use the same cloud infrastructure as computers (servers) and storage devices and add the network function to these elements to meet new market requirements. By adopting this approach, they gain significant cost savings, the ability to market new services more rapidly and the ability to adapt rapidly to any changes in the market environment.

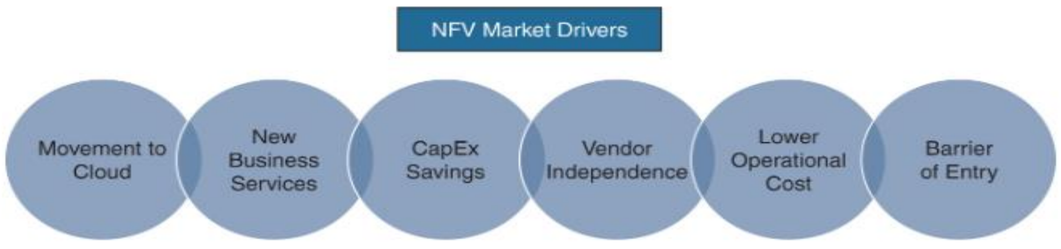


Figure 7: NFV Market Drivers

The NFV market drivers that bring new business opportunities have made network operators eager to transition to NFV. Figure 7 lists some of this market driven

3.9 Enterprise NFV

Here I will try to explain the use of NFV in Enterprise with the help of Cisco NFV papers. Many business organizations have a problem: the proliferation at branch sites of networking hardware. A typical branch site may have a mix of routing, telephony, security, wireless LAN, WAN, and Layer 2 switching devices. Traditionally, each function requires its own preconfigured box resulting in high cost and long deployment and upgrade times.[5][6]

The upgrade and configuration of numerous devices in each branch takes a lot of network staff time and may require new hardware to be shipped to each branch, which is a very high-cost mechanism. Also problematic is the proper connection of new or upgraded hardware to existing hardware and the transition from old to new hardware. Software upgrades are typically an all-or-nothing switch because the platforms often can only run one instance.

3.9.1 Enterprise NFV solution:

●Will reduce the number of hardware elements to be managed at the branch, thus minimizing the need for costly site visits for hardware installations or upgrades.
● Increases the speed of network service deployment by providing a software-based approach to service delivery.
● It Automates branch function deployment, management, and operation, thereby reducing operating expenses.
● Facilitates the deployment of best - in - class functions through an open networking approach.
● improves the flexibility of network operations by using virtualization techniques such as moving virtual machines, snapshots or upgrades.

3.9.2 Benefits of Enterprise NFV?

Although the virtualization of the branch has some advantages, Enterprise NFV also simplifies daily operations by providing a central orchestration portal that is easy to use. It allows customers to reduce the number of devices they need to deploy and manage at each location and to deploy new services quickly. Enterprise NFV also gives you the freedom to select the appropriate hardware deployment platform from Cisco UCS E - Series running on Cisco 4000 Series Integrated Services Routers (ISRs) to Cisco 5400 Enterprise Network Compute System.

3.9.3 Architectural components of Enterprise NFV?

The CISCO NFV solution for network and related application services (see Figure 8). The main features are
● An orchestration environment in which virtualized network services consisting of multiple VNFs can be easily automated.

● Virtualized network functions (VNFs) that provide the required network functionality or even non - networking software applications at a deployment site.
● The NFV Infrastructure Software (NFVIS) is a platform to facilitate the operation and deployment of VNFs and hardware components.
● X86-based computing resources to provide the CPU, memory, and storage needed to run and deploy VNFs.

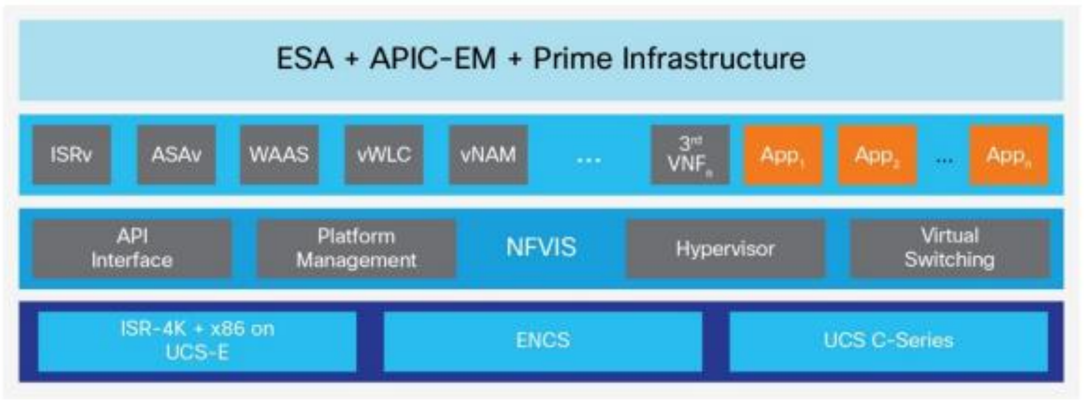


Figure 8: Component of Cisco Enterprise NFV

3.9.4 Use Cases of Cisco NFV:

1. Healthcare

Straumann Group is the global leader in implant, restorative and regenerative dentistry, headquartered in Basel, Switzerland. Managing more than 50 locations with centralized automation and monitoring was also critical for them. They evaluated the Cisco Enterprise NFV solution and deployed a solution using ENFV with the ENCS5408 platform of Cisco after rigorous design and testing. Straumann Group has been able to run virtualized services on ENCS including Virtualized Network Functions (VNFs) from Cisco for advanced encryption routing, WAN Optimization, and 3rd party firewall services. The deployment uses two ENCS 5408s for high availability and redundancy in most locations.

2. Banking

Many banks are rolling out Mobile Branch Banking Services to extend access to remote and mobile customers. Cisco NFV company has a North American customer who needed the IT team to provide a lean solution to increase agility and provide reliable uptime and predictable connectivity to achieve business initiatives. Cisco NFV company has a North American customer who needed the IT team to provide a lean solution to increase agility and provide reliable uptime and predictable connectivity to achieve business initiatives.

They rolled out the Cisco ENCS 5412 running virtualized services on NFVIS after testing and eliminating multiple vendors, including advanced routing features, firewall services and secure 4 G connectivity. They also used the integrated switch to power IP phones and access points with PoE capabilities and provided dual 4 G WAN access.

Chapter 4

Integrated NFV/SDN Architectures:

Note: In this section, I am using the MICHEL S. BONFIM, KELVIN L. DIAS, and STENIO F. L. FERNANDES, paper to explain the main features of SDN/NFV. I find this paper suitable to explain some sections of my project.

Network Functions Virtualization (NFV) and SDN are new paradigms for moving towards open software and network hardware. While NFV aims to virtualized and deploy network functions into general hardware, SDN enables networks to be programmed by separating control and data planes. NFV and SDN are complementary technologies that can supply a network solution.
SDN can provide automated connectivity between Virtual Network Functions (VNFs), while NFV can use SDN in the service chain.NFV / SDN architectures are designed in many studies in different environments. Researchers have tried to deal with the problems of reliability, performance, and scalability using various architectural designs.

4.1 INTRODUCTION to SDN and NFV

Proprietary hardware equipment is available in companies, homes, and data center networks everywhere. Each vendor examines and takes advantage of its platforms ' maximum capacity to meet the performance, reliability and availability requirements of various types of users. This approach has, however, led to incompatibility between the manufacturer's various technologies. Limited licensing agreements and proprietary source code have also contributed to this restriction.

Due to such platform incompatibility and the need for network engineers to add new features to their networks (example. firewalls, load balancing), they often have to buy new equipment from different vendors. Each equipment has a share of traffic processing that requires specific management strategies. Difficulties in the management and configuration of such heterogeneous environments are the norm. .The requirement for this flexibility in configuration and the deployment of new network functions must be met in new business and engineering models. The complexity of the current networking environment leads to high operating (OPEX) and capital (CAPEX) costs. **[1]**

Network Functions Virtualization (NFV) aims to solve these problems by transferring networking functions from vendor-specific and proprietary hardware to Common-Off-The-Shelf (COTS) software (i.e., standard processing, memory, and storage components).
These systems usually provide network services with different operations on virtual machines (for example. Virtual Network Functions-VNFs) **[2].**NFV has the potential to reduce costs

and speed up network expansion. Furthermore, NFV has the potential to increase network flexibility in the rapid delivery of services, an option that is difficult to achieve with traditional methods [3]. Its main feature is that the network control plane is separated from the data plane compared to current networks in which the IP layer integrates both planes vertically into the network devices[4].The SDN control plane, represented by a software called the SDN controller, which decides how to manage the network traffic underlying network policies and rules. The SDN controller can run separately on COTS systems from the forwarding devices. The data plane, which is used as network devices, is responsible for data transmission under a set of rules. The SDN controller allows these rules to be created and managed via an application programming interface (API) in the northbound interface. It has direct control over the data plane elements through the southbound interface protocols. This separation offers certain definite advantages, such as simplification and flexibility in network policy implementation, network configuration facilitation, development and innovation[5]. It also poses challenges for research and development that attracted researchers from industry and academia.

4.2 Combination of Computer and Network Resources

This concept aims to create an abstraction layer (AL) for both computer and network resources to offer a unique and centralized view of the entire environment. Initially, NFV / SDN solutions focused on creating this Abstraction Layer (AL) for the delivery of intelligent network services-performance and reliability for different customer profiles such as end, retail and enterprise users, as well as Over the Top (OTT) providers and developers **[15].** The AL is responsible for two orchestration functions (see Figure 1), namely Resource Orchestration (RO) and Network Service Orchestration (NSO) [29]. RO utilizes NFV VIM component and SDN to perform global resource management with resource virtualization provisioning and managing. On the other hand, the NFV uses the NSO functions to implement the life-cycle management of Network Services (VNFFGs), coordinating groups of VNF instances as network services. These functions use the services exposed by the VNFM and by the RO allowing joint instantiation and configuration along with connectivity and dynamic changes management.

In this context, different architectures try to achieve this unification, such as **UNIFY [30, 31**] and **T-NOVA [32],** both EU-funded 7th Framework Programme (FP7) projects.

T- NOVA's goal is to implement anSDN- based MANO framework for federated network and cloud resources management. Its primary objective is to automate and optimize the delivery of third-party network functions (NFs) to operators ' customers by introducing a "Network Function Store." For NF management, an Orchestrator platform was developed on top of common open source components such as OpenStack **[33]** and Open Daylight **[34**]. Besides, the WICM (WAN Infrastructure and Connectivity Manager) provides network connectivity between NFVI-PoPs (Points of Presence) and manages traffic steering in virtual networks.

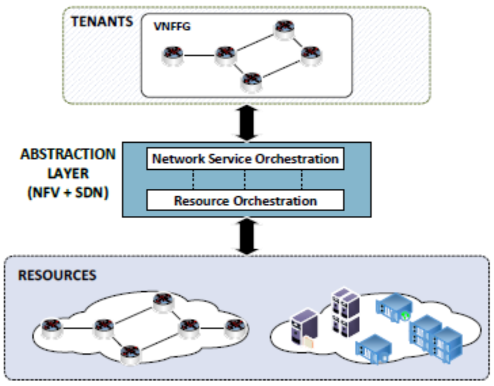


Figure 1: Abstraction layer sketch for unifying computer and network resources.

On the other hand, UNIFY aims to combine computer and network resources in a common framework for management. The aim of the UNIFY NFV / SDN architecture is to create and manage dynamic end-to-end network services from the home and **Enterprise networks** to the data center of the operator. It provides a MANO framework comprising both Cloud and WAN domains and three layers, namely the Service Layer (SL), the Orchestration Layer (OL) and the Infrastructure Layer (IL). **Figure 2** shows our simplified view of the UNIFY architecture, highlighting the main functional components of ETSI's NFV (left box) and ONF's SDN (right box) reference models.

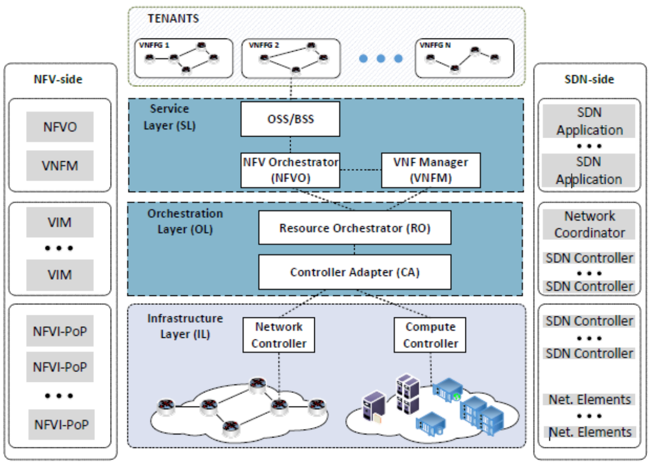


Figure 2: Overview of UNIFY architecture

The SL includes management related to the service life cycle, providing operational support system (OSS) and business support system (BSS) functions related to services provided by different tenants (example. business users, service providers, etc.).The SL also performs NFVO and VNFM functions in the management of the NS and VNF life cycle. It is important to note that SL management functions are infrastructure-agnostic and only deal with the services offered**[35]**

TOL acts as a VIM component-providing resource orchestration (RO module) to provide SL with virtual resource views and policy implementation. The Controller Adapter (CA) and a multi-domain, multi-technology and multi-vendor controller are also included. The CA offers computing and networking abstraction by collecting virtualized resources from domain-specific controllers on a lower level and organizing them into a global virtualized view of resources. The CA is an independent technology control to the RO module. Finally, in the OL, SDN is used for the creation and integration of virtual networks in both domains.

The IL manages all IT and network (physical and virtual) resources required to run the VNF.For this purpose, the Compute Controller uses two types of domain-specific controllers to manage computer resources and the Network Controller to manage the network resources. The IL considers various types of resources such as network nodes enabled by SDN (example. Open Flow switches) and data centers enabled by the cloud (example. OpenStack, Cloud Stack etc.).

The UNIFY architecture was used to implement several NFV / SDN solutions to various problems, such as Middlebox Virtualization and Virtualized Customer Premises (vCPE).

4.3On-demand and Application-specific Traffic Steering

Traffic management is the ability to direct the requests of users to the relevant service/content sources. Traffic management could be based on many factors, such as the available customer and server networking resources and capabilities, user permissions and location, and the like. A certain user may, for example, request a video streaming service with strict application performance requirements. In this case, on-demand and application-specific traffic management could guarantee the efficient user use of network resources and better quality of experience.

In the context of the NFV framework, SDN can improve traffic management between VNFs by providing dynamic service chaining. With the separation of control and data planes, SDN allows the exchange of information between the application and network layers, enabling user services to gain an overview of the general state of the network and to make intelligent decisions (to meet service requirements) on how to better control traffic via VNFs.

Carella et al.(2015)[36] proposed an NFV / SDN architecture to provide a cross-layer interface between the application and network layers to enable on-demand and application-specific traffic management network services to be deployed. Figure 3 shows our insight into this architecture. It consists of three levels: application, management, and infrastructure. To communicate your network requirements (example. bandwidth, maximum latency), these services must interface with the Control Plane API.The control plane has a global view of computer and network resources and offers the Application Layer traffic control capabilities. Its main component is the Cross-Layer Orchestra (CLO), which acts as an NFVO and VNFM to manage the cloud (OpenStack) and WAN (OpenFlow) service life cycles. The CLO was implemented over the OpenSDNCore Orchestrator **[37]**, using Java programming language.

4.4 Middleboxes Virtualization

According to [38], middleboxes (45% of network devices) were increasingly used in enterprise networks. They are used to improve performance (example. traffic shaping, load balancing and optimization of TCP) and to provide security features (example. firewalls, intrusion detection, and prevention systems-IDPS and Deep Packet Inspection-DPI) for both incoming and outgoing traffic.

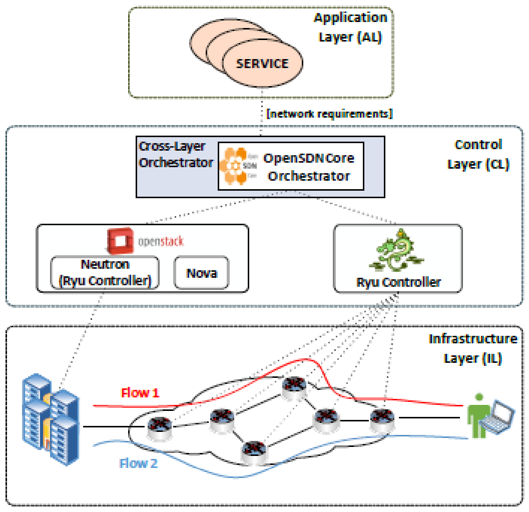
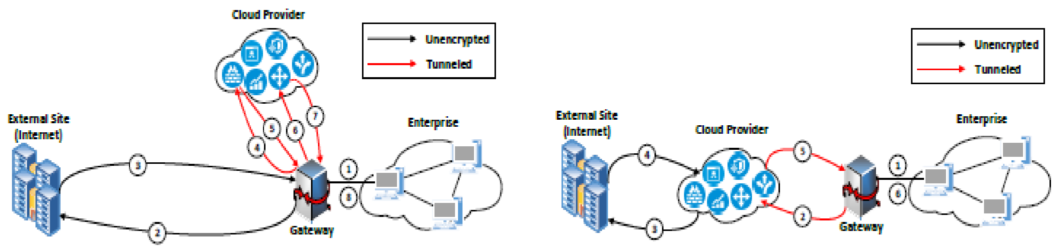


Figure 3: NFV/SDN architecture for application-specific traffic steering.

However, hardware-based middleboxes have the following disadvantages **[39**].First, due to the complexity of the management, they incur high operating costs (OPEX).These middleboxes are from various manufacturers and must be individually deployed, configured and managed. Secondly, middleboxes based on hardware incur capital costs (CAPEX).When new network functions are needed, companies must purchase one or more middleboxes due to the inflexibility of proprietary hardware that limits innovation and lock-in.
These challenges may be addressed using the NFV / SDN architectures. The main goals are to reduce both CAPEX and OPEX and deliver network function, elasticity, and dynamic service chaining quickly.NFV manages virtual middleboxes in these works, while SDN provides interconnection between VNFs for the provision of network services.
In this context,[38] consider two approaches(see Figure 4), namely Bounce and IP redirects, to redirect traffic to virtualized middleboxes for further processing. For Bounce redirections, a certain company gateway uses tunneling techniques to redirect input and output traffic to the virtual middleboxes (grouped into service chains), as shown in Figure 4a. It requires minimal configuration (i.e., few static rules) at the gateway because it only redirects traffic to a middlebox cloud provider. This approach could, however, increase the end- to- end delay

for each packet because of these redirections. To avoid the additional round trips of the redirection of the bounce (see figure 4a), The IP redirection allows traffic to and from the cloud provider directly, as shown in Figure 4b.The cloud will be located amid communication between companies and external sites in this approach. The provider must, however, announce the name and address on behalf of the company (example. DNS redirect).



(a) Bounce Redirection. (b) IP Redirection.

Figure 4: Middleboxes virtualization with NFV and SDN

4.5 Introduction to Virtualized Customers Premises Equipment (vCPE)

CPE (customer premises equipment) means any equipment (router, modem, etc.) that receives a communication service within the customer domain. Due to the high costs of maintenance, management difficulties and the impossibility of remote upgrades, CPEs were a barrier to the current objectives of both telecommunications companies and service providers. As an alternative, a solution is the CPE virtualization using NFV architecture, also known as Virtualized CPE (vCPE) **[50].**

According to a 2016 IHS Markit Survey, 100% of the service providers consulted said they intended to deploy NFV at some point. This deployment is expected to roll out by 81 percent by 2017. The majority of service providers (over 80%) prefer to deploy vCPE.

vCPE is a service that virtualizes some or all CPE-related functions[50].One of the main problems associated with virtualizing CPEs is how network services can be installed in distributed infrastructures (using multiple NFVI -PoPs)[51].In this type of scenario(see Figure 5), also called Distributed NFV[52], the VNFs are placed either on the Cloud(Cloud CPE) service provider or on-site CPE, depending on where they are most efficient in terms of latency, resources available, etc.

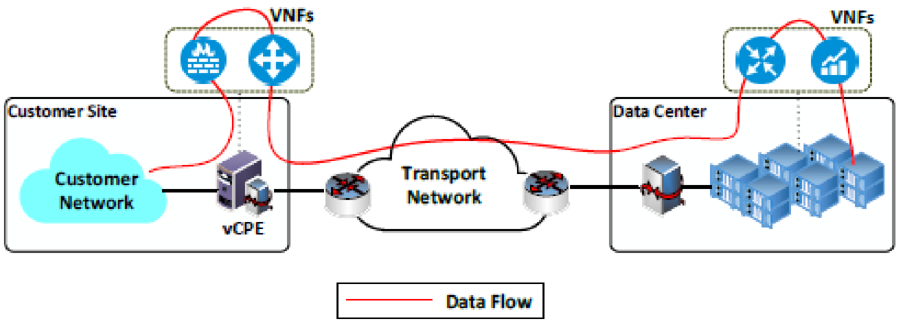


Figure 5: NFV/SDN architecture for Virtualized Customers Premises Equipment (vCPE).

Cerrato et al. (2015) [51] proposed a service-oriented NFV/SDN architecture for Telco networks that delivers generic network services selected by **telecom operators (DHCP and NAT)** or end users (Bit Torrent client). The deployment of these network services can occur in a distributed manner either in the telecom data center or the CPE.

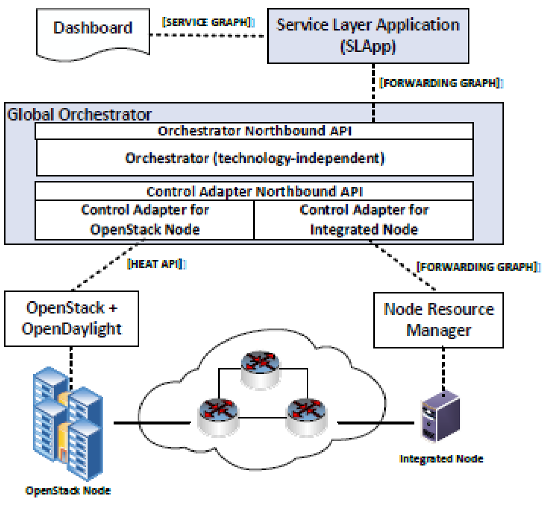

Dashboard
[SERVICE GRAPH]
Service Layer Application (SLApp)
[FORWARDING GRAPH]
Global Orchestrator
Orchestrator Northbound API
Orchestrator (technology-independent)
Control Adapter Northbound API
Control Adapter for OpenStack Node
Control Adapter for Integrated Node
[HEAT API]
[FORWARDING GRAPH]
OpenStack + OpenDaylight
Node Resource Manager
Integrated Node
OpenStack Node

Figure 6: An NFV/SDN architecture design for vCPE.

This solution is based on the UNIFY architecture including its three layers. The Service Layer Application (SLApp) represents the UNIFY SL enabling different players (operators and end users) to select their network services. For this, the SLApp includes an authentication mechanism and provides a high-level data model for denying network services (including traffic steering primitives), called Service Graph (SG). Also, the Global Orchestrator (GO) represents the UNIFY Orchestration Layer (OL). The GO manipulates the Forwarding Graph (FG) received from SLApp to enable the network service deployment according to the VNF requirements and infrastructure capabilities. To allow distributed NFV, the GO implements multiple Control Adaptors to coordinate different infrastructures and an Orchestrator component responsible for the centralized coordination of multiple Control Adaptors.

Then, the GO selects one of the infrastructures to implement all the network service requested. The authors proposed two different infrastructures (UNIFY Infrastructure Layer - IL) to host network services: the integrated node and the OpenStack node.

The integrated node represents the CPE (home gateway). It receives an FG from the GO through the Node Resource Manager (NRM) via REST API. The NRM will instantiate all VNFs using Docker containers, DPDK process or any hypervisor supported by libvirt. For traffic steering, the NRM uses an extensible Data-Path daemon (xDPd) to create an OpenFlow switch (and its correspondent Controller) for each FG. Separately, the OpenStack node represents the Telco data center and uses the OpenStack Cloud Platform for network service deployment. In this case, the KVM hypervisor creates the VNFs, and the OpenDaylight [13] and Open vSwitch control the traffic steering. The works of Soares [53, 54] proposed the Cloud4NFV platform, an NFV/SDN framework for Telco network virtualization. This platform considers multiple NFVI-PoPs and WAN domains when deploying new Service Function Chaining. Cloud4NFV considers a topology with multiples customer sites (NFVI-PoPs). All NFVI-PoPs include an OpenStack distribution working as a Cloud VIM and an OpenDaylight controller to provide VNF connectivity. The VNFs are CPE functions, and they are deployed as VMs in the NFVI-PoP closest to the customer. Further, Cloud4NFV includes a WAN VIM to provide a view of a unified WAN domain connecting the NFVI-PoPs and Telco Data Center.

4.6 TAXONOMY for NFV/SDN ARCHITECTURES DESIGN

This section provides support to answer the third research question by describing taxonomy to organize the various decision-making levels for the design of NFV / SDN architectures. Figure 7 shows our proposed taxonomy for the use of SDN in an NFV framework. It was derived from the architectures and implementations found in the selected studies,and the NFV / SDN reference architectures published [15, 102]. This taxonomy is a useful guide to simplify the work of researchers in the study of NFV / SDN architectures or the development of new solutions. In this taxonomy, the design of the NFV / SDN architectures was divided into two parts: the NFV and the SDN. On the NFV side, we have to decide whether to use two features of the architectural design or not. These features are described as follows.

Distributed NFV (D-NFV): In D-NFV, the MANO framework places Virtual Network Functions (VNFs) where they could be most efficiently and economically be deployed, such as in data centers, forwarding devices, or the CPEs [52].

Multiple VIMs: A designer could place Multiple VIMs in different NFVI-PoPs to support the multi-domain administration or in the same NFVI-PoP to provide scalability and performance [15].

As far as we are concerned with NFV, it is important to identify the NFV Management and Orchestration (MANO) tools. Tools such as OpenMANO [103] and OpenBaton [104] can provide complete solutions for MANO. On the other hand, the OpenStack enables VIM implementation to provide support for existing or new VNF Managers and NFV Orchestrators. On the SDN-side, an rst step is the placement of SDN elements in the NFV Framework [15]. These elements are described below:

SDN Resources: Comprise of both physical and virtual switches and routers;
SDN Controllers: Responsible for controlling the SDN resources, determining the behavior of network traffic;
SDN Applications: Interfaces with one or multiple SDN controllers to enforce high-level network policy, such as firewall, network address translation, QoS, and network management.

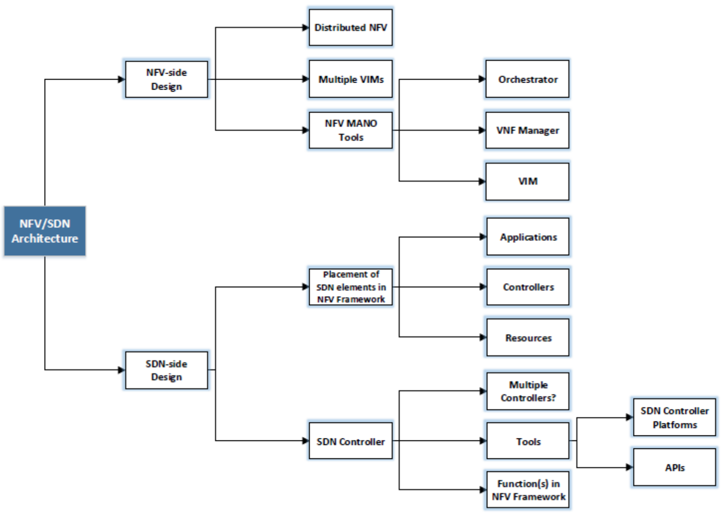


Figure 7: A taxonomy to NFV/SDN architectures design

We list some possible locations for the placement of SDN Resources in the NFV Framework, as follows [15]:
- Physical switch or router.
- Virtual switch or router.
- E-switch, software-based SDN-enabled switch in a server NIC.
- Switch or router as a VNF.

There are also some possible locations for the placement of SDN Controllers in NFV Framework **[15]**. They are the following:
- Merged with the Virtualized Infrastructure Manager (VIM);
- Virtualized as a VNF;
- As part of the NFVI and not as a VNF;
- As part of the OSS/BSS;
- As a Physical Network Function (PNF).

Finally, some locations for the placement of SDN Applications in NFV Framework are listed below **[15]:**
-As part of a PNF.
-As part of the VIM;
-Virtualized as a VNF;
-As part of an EMS;
-As part of the OSS/BSS.

There are also some architectural decisions to be made when one considers the SDN Controller, as follows:

- Does the solution implement multiple SDN Controllers?
If so, what is the main objective? Multiple SDN Controllers are hierarchically distributed to provide performance, scalability, reliability, administrative domains interaction, or Network as a Service (NaaS) management in an NFV Framework.

-What is the function of SDN Controller in the NFV Framework?
Network Connectivity in the NFVI, Control of Virtual Networks, Interconnecting VNFCs, and Interconnecting VNFs are some of these functions (extracted from the studies selected in this SLR). Finally, we must identify the SDN Controller tools, including its underlying software and the used bound interfaces (at South, North, West, and East). As SDN Controllers we can cite: OpenDaylight [34], Floodlight [105], ONOS [106], Ryu [107], and POX [49].

4.7 Auxiliary Tools

This section presents the tools used in the studies carried out to implement NFV/SDN architectures. The primary focus is on free or open-source tools, which are thus freely available for future implementations.

4.7.1VNF Instantiation

The following open source tools have been used in NFVI to enable virtualization of network functions.

Kernel Virtual Machine (KVM) [127]
Xen Hypervisor [114]:
ClickOS [128]:
Docker Engine [48]:
Data Plane Development Kit (DPDK) [130]:

4.7.2 OpenStack Cloud Platform

The OpenStack Cloud Platform provides an Infrastructure as a Service (IaaS) solution through a set of related services, written in Python. Below we describe the main services.

Keystone: Represents the Identity Service. It provides functionalities such as authentication, authorization, and service catalog.

Nova: Represents the Computing Service. It is responsible for managing the lifecycle of Virtual machine instances. Currently, it provides support for different hypervisors: Xen, XenServer/XCP, QEMU, KVM, UML, VMware, vSphere, and Hyper-V.

Neutron: Represents the Networking Service. This service provides connectivity to the interfaces of the VMs. It has an exible API that allows the construction of complex networks: at and shared networks, VLANs, VXLAN, GRE, DHCP, IPv6, and SDN.

Glance: Represents the Image Service. It stores and retrieves disk images of virtual machines. It also stores metadata of these images.

Swift: Represents the Object Storage Service. It stores and retrieves unstructured data objects. It works with data replication to provide a highly tolerant and scalable architecture.

Cinder: Represents the Block Storage Service. It provides a persistent storage block for running instances.

4.7.3 Virtual Switches

The Open vSwitch (OVS) [131] was the most adopted virtual switch. Currently maintained by the Linux Foundation, OVS aims to automate network tasks. Therefore, it supports many protocols, such as OpenFlow (versions 1.0 [132] and 1.3 [14]), OVSDB [133], NetFlow, sFlow, IPFIX, and the like. It also provides many additional features, such as VLAN isolation, traffic filtering, traffic queuing, and traffic shaping. Furthermore, the OVS has also been integrated into popular cloud platforms including oVirt, OpenNebula, and OpenStack.

4.7.4 SDN Controllers

The table lists all SDN controllers used in implementations of NFV/SDN architectures. Such controllers have been used as NFVI, VIM or VNF components.

4.7.5 SDN Controller Overview:

OpenDaylight [26, 35, 39, 44, 47, 51, 53, 54, 57, 75, 79, 81–86, 89–94, 108, 109, 115, 125]
ONOS [55, 120]

Floodlight [21, 40, 41, 81–84, 124]
Ryu [17, 20, 23, 36, 42, 45, 74, 117]
POX [35, 45, 46, 111]

4.7.6 SDN Southbound Interfaces

1. OpenFlow - The ONF is an organization committed to the development and dissemination of SDN. For this, ONF is responsible for the creation of open standards for SDN, such as the OpenFlow specifications. OpenFlow is frequently updated, thus adding new features from version 1.0 [132] to 1.3 [14]. Multiple row tables, group and meter tables, and MPLS support are some of the advances since version 1.1. The OpenFlow comprises three main elements, namely switches, controllers, and protocols (Southbound API). The OpenFlow Switches are responsible for the data packet forwarding (i.e., data plane) according to the rules created and maintained by the Controller [14]. Multiple Flow Tables stores these rules. The controller is the main component of OpenFlow. In architectural terms, the controller supports the network applications, determining the rules to be stored and applied by switches. There are several OpenFlow controllers available, such as OpenDaylight, Floodlight, ONOS, Ryu, POX, and NOX.

2. Forwarding and Control Element Separation (ForCES) - ForCES is an SDN standard Defined by the Internet Engineering Task Force (IETF) [122]. In ForCES, the plane separation occurs by dividing the Network Elements (NE) into two entities: Forwarding Elements (FE) and the Control Elements (CE). FEs represents the Data Plane and comprises both physical and virtual switches. For CES models FEs by denying one or many Logical Functional Blocks (LFBs) classes, realized by an XML-based modeling language. An LFB comprises input and output ports and acts as a packet processing resource performing different functions, such as classification, and measurement. Multiple LFB instances in the same FE can be connected in a directed graph to create a network service. Each LFB provides operational parameters, capabilities, and events to a CE that acts as an SDN Controller. CE uses the ForCES protocol as a Southbound API to perform the per-LFB controlling.

4.7.7 Some examples of SDN Northbound Interfaces

Regarding Northbound Interface (NBI), all studies that used this type of interface are as follows- chose to work with REST API [137]. According to [138], the REST API has become a prevalent choice for the NBI in SDN, because it is highly extensible and maintainable for managing services from both data and control planes. HTTP [139] is usually adopted as a protocol for the communication between REST services (called web services). Procera [140] and Frenetic [141] would be alternatives for NBIs, but they run on top of a single OpenFlow Controller, and they are not so extensible as RESTful interfaces [138].RESTful interfaces are available in the main SDN Controllers, such as OpenDaylight, Floodlight, ONOS, and Ryu.

SDN Northbound API

REST API [16–18, 20, 23, 26, 36, 39, 40, 44, 47, 51, 53, 54, 73, 75, 79, 81–86, 89–94, 100, 101, 108, 109, 115, 120, 125]

4.7.8 Examples for Vendor-specific Tools

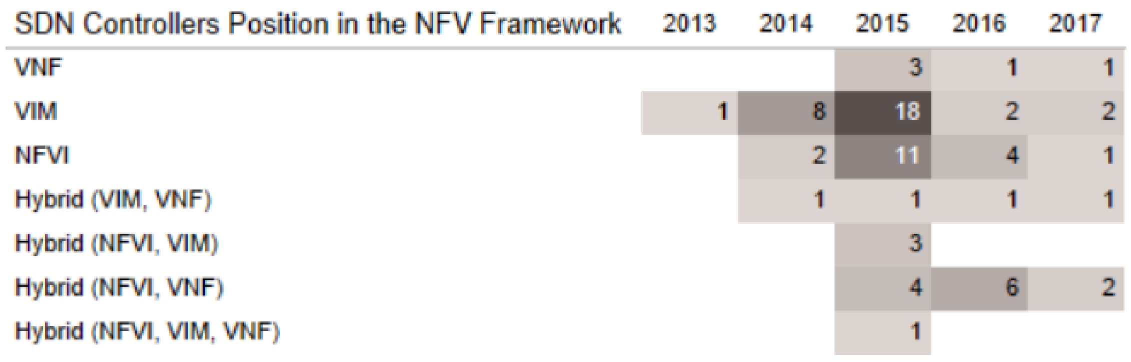
The Vendor-specific solutions are only used in some PoCs from ETSI NFV ISG, for all NFV components. Some of the features used are: HP [22, 24, 26, 27] and Huawei [25] NFV Orchestrators (NFVO); ZTE [24], Riverbed [25], Samsung [22], Telcoware [22], HP [26] and F5 [27] VNF Managers (VNFM); and Samsung [22], Telcoware [22], HP [24, 27] and ZTE [24] Virtual Infrastructure Managers (VIM).

4.8 Placement of SDN Elements in the NFV Framework

This section uses the SDN-side of the taxonomy to organize the NFV/SDN solutions regarding the SDN
The position of virtual switches in NFV Framework
NFVI [3, 16–20, 22, 24–28, 35, 36, 39–47, 51, 53–55, 57, 73–76, 79–94, 100, 101, 109–112, 115–120]
VNF [19, 59, 71, 100, 101, 121]
The table above shows how virtual switches are placed in the given NFV framework by the study. The NFVI is the most widely used SDN resource location. This scenario is a common approach to providing VNFs with network programmability and the ability to connect and steer traffic. Some works, however, include virtual switches as VNFs as well. These works aim to provide various customers with an SDN - enabled virtual network. In [71, 121], this placement is possible because they work with the Forwarding and Control Element Separation (ForCES) protocol [122] as the SDN Southbound API and consider the Logical Functional Blocks (LFBs) as VNFs On the other hand, Neves et al. [100, 101] created an abstraction for network service deployment by installing Virtual Network Elements (VNE).VNEs are VNFs running a process of virtual switching that processes packets (networking services) through network traffic. VNEs may be distributed in the NFVI - PoPs core or edge

The position of SDN Controllers in the NFV Framework.
NFVI [3, 18, 21, 25–27, 35, 36, 42, 43, 45, 46, 51, 53, 54, 75, 76, 79–86, 89– 94, 109, 113, 115]
VIM [3, 16, 17, 22, 24, 28, 35, 36, 39–41, 44, 45, 47, 55, 57, 59, 70–73, 78, 87, 88, 100, 101, 108, 110–112, 116–121, 123–125]
VNF [19, 20, 23, 24, 45, 73, 74, 77, 81–86, 89–94, 100, 101]

Above table shows how the studies positioned the SDN Controllers in the given NFV Framework. According to the [15], an SDN Controller can run in (5) places: NFVI, VIM, VNF, OSS/BSS, and can be a Physical Network Function (PNF). The table shows how the studies positioned the SDN Applications in the NFV Framework. According to the [15], SDN Applications can run in five (5) points: as part of a PNF, as part of the VIM, virtualized as a VNF, as part of an Element Manager (EM), and as part of the OSS/BSS. In this work, we did not provide and references to SDN Applications as part of a PNF or an EM. The VIM is the most used as a position for both SDN Controllers and Applications (see Figure 18). The VIM is the best place for these elements because it offers a global view of both NFVI physical and virtual infrastructures and the VNFs. This property allows the implementation of different functionalities, such as VNFs or VNFCs interconnections, network connectivity in the NFVI, and the control of virtual networks as shown in Table 8. However, the NFVI have also been widely adopted as SDN Controller placement. According to [15], this scenario is a classic case of the SDN controller providing network connectivity in the NFVI. In this work, we consider the SDN controller as an NFVI component when the VIM and its functions are distinguishable, as in the case of OpenStack controlling OpenDaylight.



SDN Controllers Position in the NFV Framework	2013	2014	2015	2016	2017
VNF			3	1	1
VIM	1	8	18	2	2
NFVI		2	11	4	1
Hybrid (VIM, VNF)		1	1	1	1
Hybrid (NFVI, VIM)			3		
Hybrid (NFVI, VNF)			4	6	2
Hybrid (NFVI, VIM, VNF)			1		

Figure 8: Position of SDN Controllers in the NFV Framework.

There are three VNF types: Ctrl App, OF Ctrl, and SDN-enabled virtual switches. The Ctrl App and OF Ctrl are deployed as VMs in OpenStack (Cloud domain), while SDN switches are deployed in a Mininet emulator (representing an OpenFlow domain). On the data plane, an elastic router comprises one or more SDN switches. On the control plane, the SDN Ctrl manages the topology creation on the top of SDN switches. Moreover, the SDN application Ctrl App monitors the SDN Flow statistics and triggers topology changes (if needed), adding more or fewer SDN switches**.** A hybrid solution for the positioning of SDN controllers was proposed. In this case, the POX Controller [49] was placed on VIM to support the creation and management of network services in OpenFlow domains. The OpenDaylight [34] was placed on NFVI and is used by the OpenStack [33] to provide connectivity in the Cloud domain. Finally, the SDN Ctrl is a Ryu Controller created as a VNF to coordinate the SDN-enabled virtual network.

The position of SDN Applications in the NFV Framework

VIM [3, 16–18, 20, 24–28, 35, 36, 39, 41–47, 51, 53–55, 57, 59, 73, 75, 80–86, 89– 94, 100, 101, 108–113, 115, 119, 120]

VNF [19, 21–24, 40, 45, 70, 71, 73, 74, 76–79, 87, 100, 101, 116–118, 123]
OSS/BSS [3, 41, 72, 81–86, 88–94, 109, 112, 119, 121, 124, 125]

Regarding SDN Applications placement, the SDN App also runs as a VNF on top of SDNCtrl (Ryu Controller). For OpenFlow domains, the SDN applications run as a VIM component, on top of POX. Finally, for Cloud domains, the Neutron service (VIM) performs the control of OpenDaylight instance. The Operations support systems (OSS) and Business Support Systems (BSS) have also been widely adopted as SDN Applications placement. Application at this level enables multiple tenants to control dedicated SDN networks to provide their services. This scenario is common in works that propose solutions for 5G Cellular Networks [88, 100, and 101]. Examples of SDN application placement as OSS/BSS management task are the works of Munoz and Vilalta [81–86, 89–94]. In those works, a tenant SDN Controller runs as a VNF to control an underlying VTN. The end-users or service provider operators (components of OSS/BSS element) have direct access to this controller and can implement customized applications for VTN control and management.

4.8.1 SDN Controller Functions in the NFV Framework

Following table shows the possible functions for SDN Controllers when applied in the NFV Framework. Below, we describe these functions:
Interconnecting VNFs/VNFCs: The SDN Controller might be used to connect and manage the traffic between VNFs and VNFCs to enable Network Services, by creating Service Function Chaining (SFC). Network Connectivity in the NFVI: The SDN Controller is used to provide L2/L3 connectivity among end-to-end devices;
Control of Virtual Networks: The SDN Controller might be responsible for creating and managing virtual networks for different customers.

Possible functions for SDN Controllers when applied in NFV Framework
Interconnecting VNFs/VNFCs [3, 16–28, 35, 36, 39, 41, 42, 44–47, 51, 53–55, 59, 70–73, 75–80, 88, 100, 101, 110–113, 115, 117–121, 124]
Network Connectivity in the NFVI [3, 16, 18, 20, 21, 23, 24, 26, 27, 35, 36, 40, 41, 43, 45–47, 51,53–55, 57, 59, 70, 72, 74–77, 79–87, 89–94, 100, 101, 108,109, 111, 113, 115, 116, 123, 125]
Control of Virtual Networks [16, 27, 41, 55, 57, 70, 73, 81–94, 100, 101, 125]

Objective	**Studies**
Distributed Performance	**[26]**
Scalability	**[46]**
Reliability	**[81–86, 89–94]**
Administrative Domains Interaction	**[3, 16, 20, 35, 36, 40, 45, 70, 80–86, 89–94, 100,101, 109, 113]**

NaaS Management	**[3, 16, 20, 39, 44, 45, 47, 51, 80–87, 89–94, 100,101]**

List of objectives for using Multiple SDN Controllers.

The table shows the main objectives for implementing Multiple SDN Controllers in the NFV Framework as well as the use of Multiple VIMs multiple SDN controllers are organized hierarchically to provide the following objectives.
Distributed Performance: When the VNFs have to be distributed to the chosen location, they are interconnected by location-specific SDN Controllers;
Scalability: Multiple controllers manage an NFVI-PoP infrastructure;
Reliability: Fault tolerance, disaster recovery, and full isolation management.
Administrative Domains Interaction: Communication management of different scenarios (e.g., Cloud and WAN) using different SDN Controllers integrated hierarchically in a unique platform;
A network as a Service (NaaS) Management: The concept of NaaS is related to the provision of virtualized network services to customers with different requirements [126]. This function includes network virtualization (Network Slicing).

4.9 Opportunities and Challenges

1. Deployment of Network Services

The Service Function Chaining (SFC) provides the ordered traffic flow service list[9].The rapid deployment and provision of NFV by the SFC and the centralized and exemplary control of SDN have opened up new opportunities in this area. Solutions that improve performance and optimize the use of resources in the deployment of functions are still required. Below, we describe some of the challenges associated with NFV / SDN solutions for network function deployment.

VNF Performance: Virtualized network functions should meet the performance requirements of the user, in particular, if the SDN controller is a VNF.For fast packet processing in standard servers, current hypervisors must be optimized to achieve high I / O speed, short transmission delays and so on. The DPDK [130], NetVM [142], and ClickOS [128] are some of the initiatives. Also, the use of container-based virtualization, such as Docker containers, is of paramount importance in environments requiring high performance with low resource consumption, such as edge devices (example. vCPEs, MECs, etc.) because they have relatively low capacity compared with traditional NFV servers. However, there are some challenges to be considered when choosing the containers technologies for deploying VNFs in NFV/SDN architectures [47, 143]:

-**Orchestration:** All containers and their services and settings share the same kernel. This feature increases the complexity of the orchestration and management platforms because the allocation process must take into account whether a VNF has unique kernel requirements, such as a specific module or configuration.

- **Security:** Container-based virtualization has a wider attack surface than other virtualization techniques because it has a more sophisticated interface than hardware emulation interfaces. It also provides weaker functional insulation in cases where containers may require different conflicting kernel configurations. It also has more vulnerable performance insulation because containers in the same host can be placed under resource pressure by external attacks or new instances (example. memory or CPU overconsumption).

VNFs Scheduling and Placement:

High-level Policies: High-level policies need to be defined to simplify the configuration of NFV Orchestrator operations, such as resource allocation and optimization mechanisms, and to meet customer requirements (OSS / BSS interfaces).In this case, the template languages OpenStack 's HOT (Heat Orchestration Template) and TOSCA (Cloud Application Topology and Orchestration Specification) could be used[33].

Traffic Steering: In NFV/SDN solutions, traffic management and network function deployment should be jointly optimized by providing a network-conscious scheduling mechanism [51, 115] that deploys VNFs taking into account both the paths expressed in the forwarding graph and the network behavior (available bandwidth, latency, jitter, etc.).
As a result, more variables are introduced, and heuristic algorithms to reduce the computational complexity should be created.

Elastic Network Function: The dynamic service scaling at runtime improves the use of resources, reduces both CAPEX and OPEX and maintains service levels[3].NFV / SDN solutions must be able to scale (in / out or up / down) networking services and monitor the resources of both servers and networks to more elastic, pay-as-used services.

Orchestration: Orchestration services are necessary for the deployment, supply, and management of elastic, adaptable and autonomous network functions. For NFV MANO (Management and Orchestration) tools such as OpenMANO [103] and OpenBaton [104] could be used as a solution.

2. Help in the improvement of the Programmability:

SDN and NFV are the critical enablers for realizing some of the expected features in 5G networks, such as network programmability (example., network abstraction, infrastructure sharing, and re-configurability), adaptability (example., self-healing, self-configuration, self-protection, and self-optimization) and capabilities (example., network slicing and MEC) [65]. However, some improvements should be provided so that the existing SDN standards such as OpenFlow can be applied in this type of scenario. OpenFlow is the most used protocol for the Southbound API in NFV/SDN solutions; however, currently; it does not support application layer packet processing.

Inspection and classification of the application layer are necessary to distribute various network services and thus provide intelligent service chaining. Finally, OpenFlow is not suitable for wireless networks (example. Wi-Fi, LTE,etc) because the tables only contain Ethernet-based switches rules. Wireless communication is more complex, as wireless links are time-varying and vulnerable to interference. For this, extensions must be implemented to allow Wi-Fi programming rules, enabling the matching and monitoring of wireless frames [41]. Besides, SDN must provide support to Radio Access Network (RAN) virtualization infrastructures [147]. In this case, SDN approaches must support both legacies (example., 3G and 4G) and new radio access technologies (example., 5G and the narrowband Internet of Things, NB-IoT), ensuring radio resource isolation.

3. **Multiple SDN Controllers:**

NFV / SDN solutions could be used for controlling and managing heterogeneous network resources (optical, MPLS, IP, etc.), distributed in various geographical areas. Hierarchical and federated SDN controllers should, therefore, be used to fulfill the requirements of scalability, availability, reliability and end-to-end (multi-domain) supply. For such solutions, tools such as FlowVisor and North / East / Westbound interfaces from popular SDN controllers (OpenDayLight [34] and ONOS [106]) can be used.

4. Security Features:

In addition to current security problems that are unique to each technology [5, 6], the NFV/SDN solutions also have security challenges related to the integration process, such as the lack of authentication and authorization mechanisms in the communication interfaces between SDN and NFV modules. Also, security services should be developed to address malfunctioning software (example. detection and prevention of exploits) or attacks caused by malicious adversaries (example. intrusion detection and prevention systems) such as Distributed Denial of Service (DDoS)[97]. Such services should take into account attack surfaces at all levels of the infrastructure, including network, edge and core data centers, virtualization, and user devices.

5. Multi-Tenant, Multi-Service and Multi-Domain Support

For the provision of quality of service (QoS) and SLA enforcement in multi-tenant environments with end-to-end services, an NFV / SDN architecture supporting multiple domains or NFVI-PoPs is required. It remains a challenge, however, as the orchestral functions must support the following characteristics [65]:

-Multi-domain orchestration of diverse programmable infrastructure technologies (example. RAN, transport, and core networks, data centers, etc.), possibly belonging to different operators;

- Northbound interface for Network Slicing management, providing multi-tenancy and Multi-service support;
-End-to-end network slices that are flexible to the dynamic requirements of different services (example. IoT, smart cities, etc.) And mobile operators, providing a multi-service and context-aware adaptation of network functions;
-Advanced autonomic network management platforms to address complexity in such scenarios.

Chapter 5

Enterprise CORD

5.1 Introduction:

Enterprise CORD (E-CORD)[1] is the first initiative to offer business connectivity services across metro and large-scale networks using only open source software and commodity hardware. CORD (Central Office Re-architected as a data center) combines NFV, SDN,and commodity cloud elasticity to provide the Telco Central Office with data center economy and cloud agility. The open reference implementation of CORD uses white box switches, commodity servers with docker, OpenStack, ONOS and XOS open source software. E-CORD builds on the CORD infrastructure to support business customers and offers business connectivity services (L2 and L3VPN) to service providers. Company customers can use E - CORD to quickly establish on-demand networks between any number of endpoints or branches of the company. These networks can be dynamically configured, which means that connection attributes and SLAs can be specified and supplied on the fly. Besides, enterprise customers can use network functions such as firewalls, WAN accelerators, traffic analysis tools, virtual routers, etc. as on-demand services provided and maintained within the service provider network.[2][10][12][13]
E-CORD enables a variety of services for enterprise customers, including:

- Private VPNs
- Internet Connectivity
- Content Delivery Networks
- SD-WAN traffic optimization over multiple backbones
- Application traffic prioritization
- Firewall and security services

What is the key business or technical challenges this solution addresses?
1. Software defined control of a converged packet/optical wide area network that have innovative multi-layer and delegated control primitives.
2. Connectivity services at the carrier level have elaborate service model specifications and deploying them on white box infrastructure introduces problems with hardware support.
3. A disaggregated ROADM platform control and configuration require careful design of abstract interfaces.
Finally, maintaining high transmission performance and signal integrity in the optical white box model is an open challenge.

Following are the component which helps in the open setup of the ECORD Platform:

● OpenStack provides a basic IaaS capability and is responsible for virtual machine (VM) and virtual network (VN) creation and delivery. CORD uses Neutron, OpenStack's Nova, Ceilometers, Keystone, and Glance subsystems.

● ONOS hosts control programs implementing services and is responsible for the integration of virtual networks into the underlying network.

● XOS is a service assembly and composition framework. It combines data plane services supported by OpenStack and Docker and control plane services running on ONOS.

● The atrium is the software stack which runs on every whitebox switch. It includes Open Network Linux, Indigo OpenFlow Agent (OF 1.3) and OpenFlow Data Plane Abstraction (OFDPA), layered on top of Broadcom merchant silicone. It also demonstrates the world's first disaggregated whitebox ROADM, built using commodity hardware components and controlled using open interfaces. [14]

5.1.1 Enterprise WAN Current State Overview

1. A very important part of enterprise infrastructure which is all over the enterprises is MPLS VPN and leased lines are primary approaches.
2. Represents a significant cost.
3. Is relatively static-As we cannot create (virtual) networks on demand for apps or user groups and cannot get resources (example. bandwidth) on demand.
4. Does not offer a programmatic interface to observe, control, & adapt to the changes in the network.
5. Has increasing complexity-With BYOD, IoT, private and public cloud, develops the model, privacy regulations

5.1.2 What Enterprise Wants

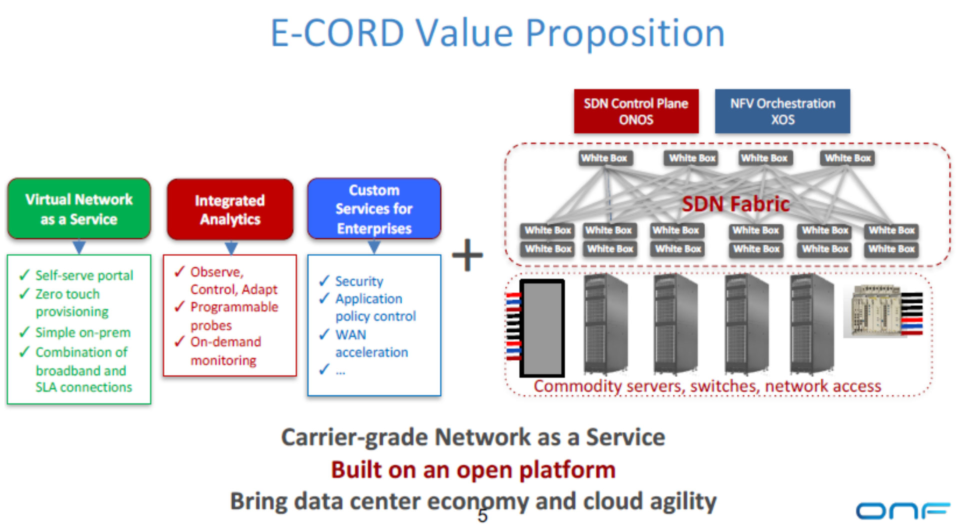
1.Customized network on-demand service for their infrastructure.
   - For different apps or user groups and the different services.
   - On-demand bandwidth
   -safe & insulated from other networks
2. Software-defined to observe, control, and adapt
   - With its own portal and programmatic interface
3. Ability to cope with increasing complexity as the size of the company will grow
4.Lower cost to match that of other infrastructure components
   - Commodity/open source building blocks

***5.1.3 Current State for Enterprise*:** The current state of the enterprise is quite old, and it comes with a very high cost, some of them are as follows:
Enterprises represent a very important revenue source so one should able to handle the cost of the enterprises.
1.SP infrastructure built with closed proprietary boxes
– High capex/opex
– Do not allow the creation of new services
Customer demands are outpacing the ability to change

2.At risk of losing enterprises to a new class of providers
 – Example. Viptela, Cloudgenix

***5.1.4 What a Service Provider Wants*:** Now the new features and services which are required by the enterprise are described as follows:
1.Ability to offer (virtual) networks on demand
– Customer spec: resources with SLAs
– Allow customers to manage their networks: observe, control, & adapt according to their need.
– Secure and isolated from other virtual networks for secure transmission.
2. Software stack that enables the creation of innovative services for customers as per requirement.
3. Simplify on-demand equipment and its management to make the operation easy and fast.
4. Infrastructure with commodity and open source building blocks.

5.1.5 Enterprise WAN: Costly, Static & Increasingly Complex
Optical WAN SDN control requires innovative multi-layer and delegated primitives in control. Moreover, Hardware support problems due to white box infrastructure carrier-grade connectivity services. To control and configure the disaggregated ROADM platform, careful design of abstract interfaces is necessary,and it Maintains high transmission and signals integrity performance levels in the optical white box model. [9]


E-CORD Value Proposition
SDN Control Plane ONOS
NFV Orchestration XOS
White Box
SDN Fabric
Virtual Network as a Service
✓ Self-serve portal
✓ Zero touch provisioning
✓ Simple on-prem
✓ Combination of broadband and SLA connections
Integrated Analytics
✓ Observe, Control, Adapt
✓ Programmable probes
✓ On-demand monitoring
Custom Services for Enterprises
✓ Security
✓ Application policy control
✓ WAN acceleration
✓ ...
+
Commodity servers, switches, network access
Carrier-grade Network as a Service
Built on an open platform
Bring data center economy and cloud agility
5
onf
Figure 1: Overview of the ECORD Value Proposition

5.1.6 Cloud-based Enterprise Services with ECORD: The cloud-based enterprise services have a lot of benefits for the customers as follows:
1.Zero-touch virtual networks.
2. Strong SLAs / QoS for enterprise traffic.

3. Built-in analytics to support complex network.
4. Commodities hardware and open source software.

5.1.7 E-CORD: SD-WAN:The ECORD SD-WAN have a lot of features;some of them are as follows:
1.Zero-touch virtual networks on demand
2. Strong QOS/SLA for enterprise traffic
3. Software stack that allows innovative services to be created and manage according to the client needs.
4. Simplified on-site maintenance and equipment.

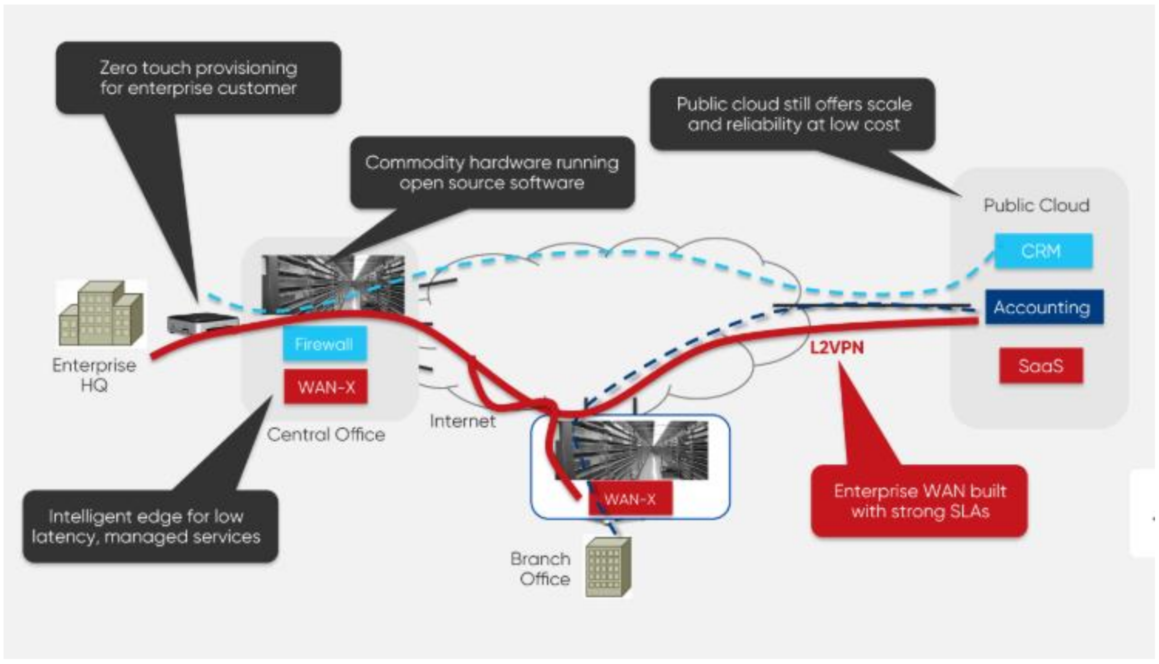

Zero touch provisioning for enterprise customer
Public cloud still offers scale and reliability at low cost
Commodity hardware running open source software
Public Cloud
CRM
Accounting
SaaS
L2VPN
Firewall
WAN-X
Enterprise HQ
Central Office
Internet
WAN-X
Enterprise WAN built with strong SLAs
Intelligent edge for low latency, managed services
Branch Office

Figure 2: ECORD Overview

5.1.8 Virtual Networks as a service
• SD-WAN Overlays: It has many advantages in the enterprise cord network as follows:
1.Combines one or more broadband connections with LxVPN
2.Simple to use, highly scalable, but best effort service on broadband
3.All service needs to support on-demand creations between all location, dynamic re-provisioning (SLA changes etc.) and zero-touch configuration.

• Value-Added Services Examples
Following are the value-added service we can add in ECORD environment:
1.Traffic Analysis
2.Active Testing: Probe network to verify connectivity and SLA
3.Application Policies & Traffic Steering: Prioritize traffic based on user, locaCon, application

4.Firewall: Access control and traffic control
5.WAN Acceleration: Minimize end-to-end traffic
6.Encryption: Configurable end-to-end security

• **L2VPN:** Following are some feature of L2VPN in an enterprise network
1. It Provides broadcast domain between enterprise locations.
2. Simple to use, scaling limitations, strong SLAs.
3. Utilizes a standards-based service model (MEF).
4. Offers a limited choice of value-added services.

• **L3VPN:** L3VPN have a lot of features in enterprise network some of them are Pseudowire / VLAN xconnect, Provides private IP network, Requires routing configuration, highly scalable, strong SLAs and Offers a broad choice of value-add services**.**

All services need to support: All the services need to support on-demand spawning, Customer interface for configuration and to observe and should support Customer interface for configuration and observing

5.1.9 Integrated Analytics:

Observe: The cord integrated analytics platform need to observe the following things:
1. Individual CORD components (CPE, VNF, fabric...)
2.Transport network
3.End-to-end connection status (OAM probes)
4.Global view across multiple CORD sites

Analyze: Need to analyze the following component of the cord:
1.Root cause analysis across the multiple-domain network
2.SLA validation
3.More sophisticated analytics can be plugged in by others

Control: Need to control the following services in cord environment.
1. Adding additional service capacity
2.Automatic load balancing across sites
3.Automatic healing, routing around faults

Analytics should allow
1. Service Providers to**:** Should able toMonitor CO health (servers, fabric, edge devices) and to Monitor transport network
2. Customers to: Should able to monitor end-to-end connection status and SLAs and Monitor VNF performance

5.1.10 Implementation of Virtual Networks as a Service

• **Metro Orchestration:** It Identifies the transport path(s) and End-to-End resource constraints are given services and virtual network type and helps in Conveys constraints and service requirements to each CORD site(s). It also helps in CORD sites configure fabric and service(s) for LxVPN.

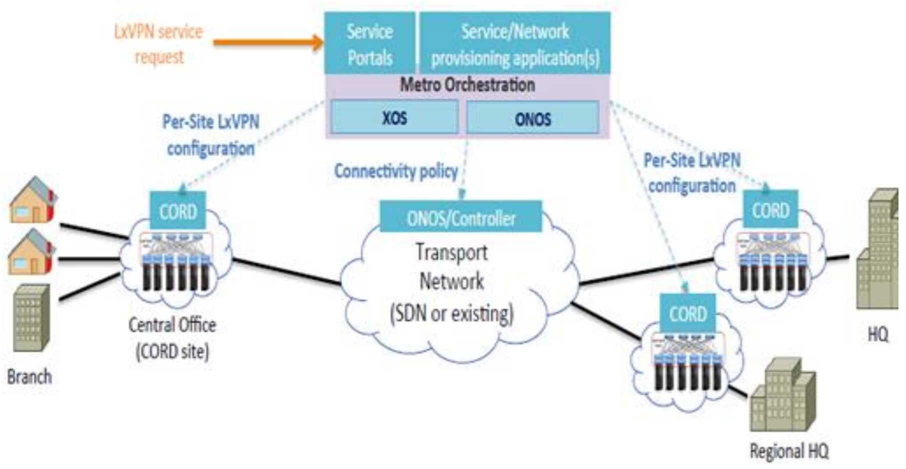

LxVPN service request
Service Portals
Service/Network provisioning application(s)
Metro Orchestration
XOS
ONOS
Per-Site LxVPN configuration
Connectivity policy
Per-Site LxVPN configuration
CORD
ONOS/Controller
Transport Network (SDN or existing)
CORD
CORD
Central Office (CORD site)
Branch
HQ
Regional HQ

Figure 3: Virtual Networks as a Service

5.1.11 Implementation to CORD site:There should be Per-Site LxVPN configuration specified as High-level service graph. We need to do Site orchestrator refines High-level service graph into site-local service graph, and XOS instantiates the service graph, LxVPN app on ONOS configures the networking for service instances.

a) Ethernet Edge (EE) & vEE: classifies traffic from the user and maps them to site-local service graph.

b) Transport Edge: Add/Remove headers to be routed through the transport network.

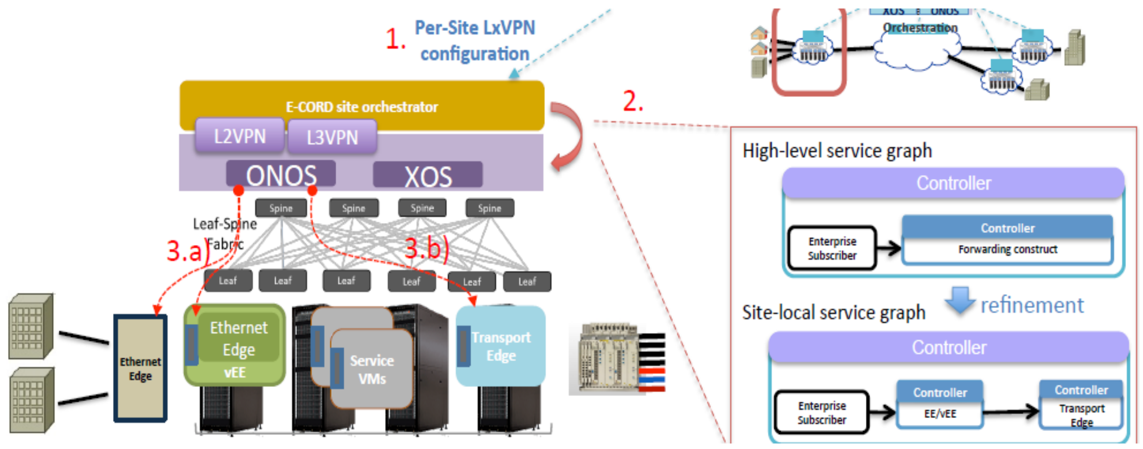


Figure 4: Implementation: CORD site

5.1.12 Implementation of Transport Network Control: ONOS controller is used for the transport network control. The detail about the ONOS will be in the next sections. It Handles request from orchestration layer to provide connectivity between CORD sites (example.MEF LSO Presto, ONF TAPI, etc.)

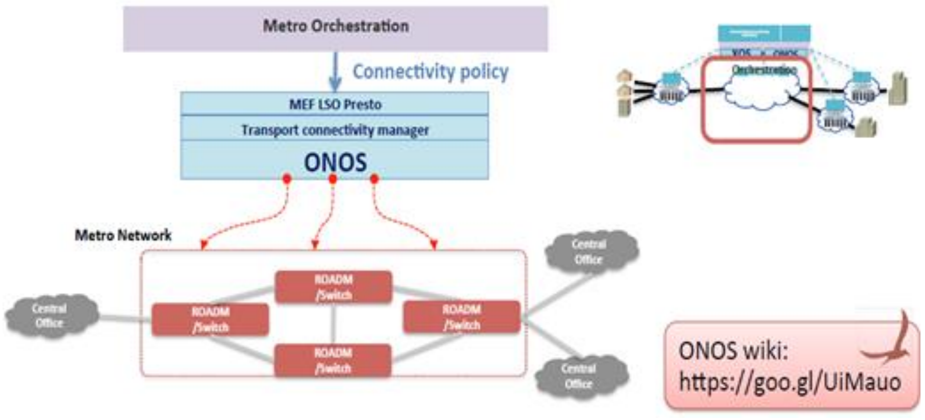
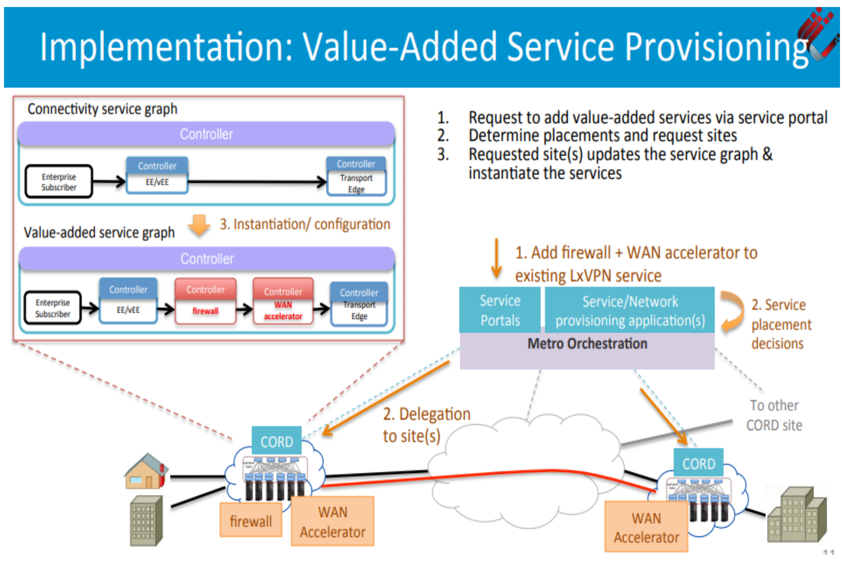


Figure 5: Implementation: Transport Network Control

5.1.13 Implementation of Value-Added Services

1. Implementation: Value-Added Service Provisioning



Implementation: Value-Added Service Provisioning

1. Request to add value-added services via service portal
2. Determine placements and request sites
3. Requested site(s) updates the service graph & instantiate the services

Connectivity service graph
Controller
Enterprise Subscriber
Controller EE/vEE
Controller Transport Edge
Value-added service graph
3. Instantiation/ configuration
Controller firewall
Controller WAN accelerator
1. Add firewall + WAN accelerator to existing LxVPN service
Service Portals
Service/Network provisioning application(s)
Metro Orchestration
2. Service placement decisions
2. Delegation to site(s)
To other CORD site
CORD
firewall
WAN Accelerator

Figure 6 -Value Added Services Provisioning

2. Overview of Value-Added Service Placement: There are alot of following a set of rules which can be followed for the value-added services placement.
1. Placement of the services can depend on ‘weight’ and complexity of the application
2. Most services should be near the CORD site
3. Lightweight but latency sensitive services should be placed at CPE
4. Need to Calculate and store heavy service on the CORD / Cloud DC "hub.”

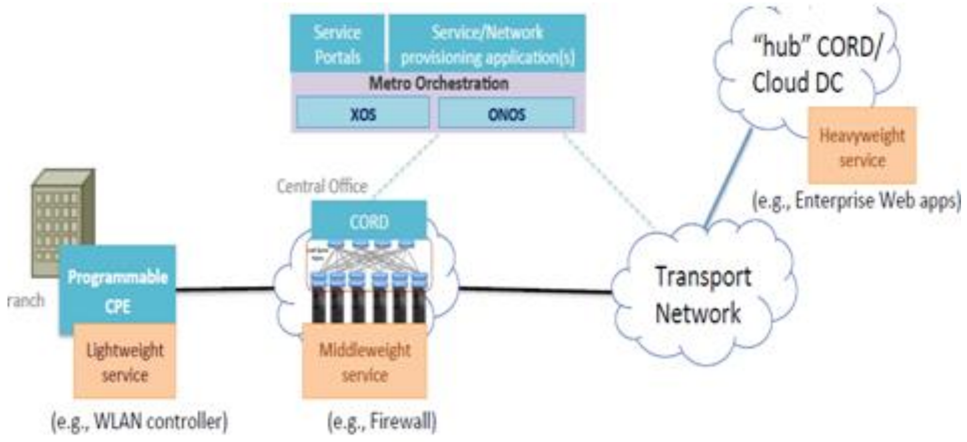


Figure 7: Placement for the Value-Added Service

5.1.14 Implementation to Integrated Analytics

1. Programmable Probes: Enterprise Service VNFs

Service Level Events

Services	**FunctionsAssigned**
vEE & vTE	Packet & byte counter , Queue length
Firewall	Throughput, Open connections
vOAM	Latency distribution, Packet-loss rate

Infrastructure Metrics Examples: Here are the some of the infrastructure metrics examples which are used in implementation of services:
- instance.cpu
- instance.vcpu

- instance.memory.usage
- instance.network.incoming.bytes.rate
- instance.network.outgoing.bytes.rate
- instance.disk.usage
- instance.create
- instance.delete

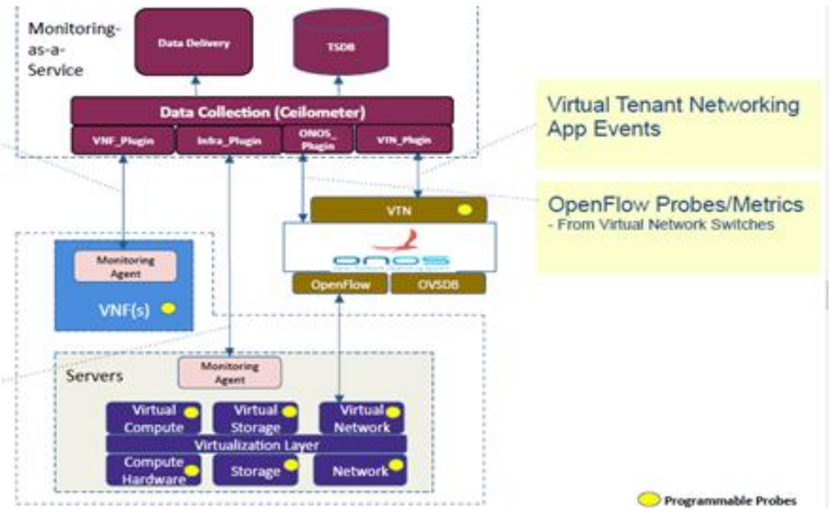


Figure 8: Enterprise Service as VNF

2. Implementation of Site-local analytics: This figure gives the overview of the site-local analytic

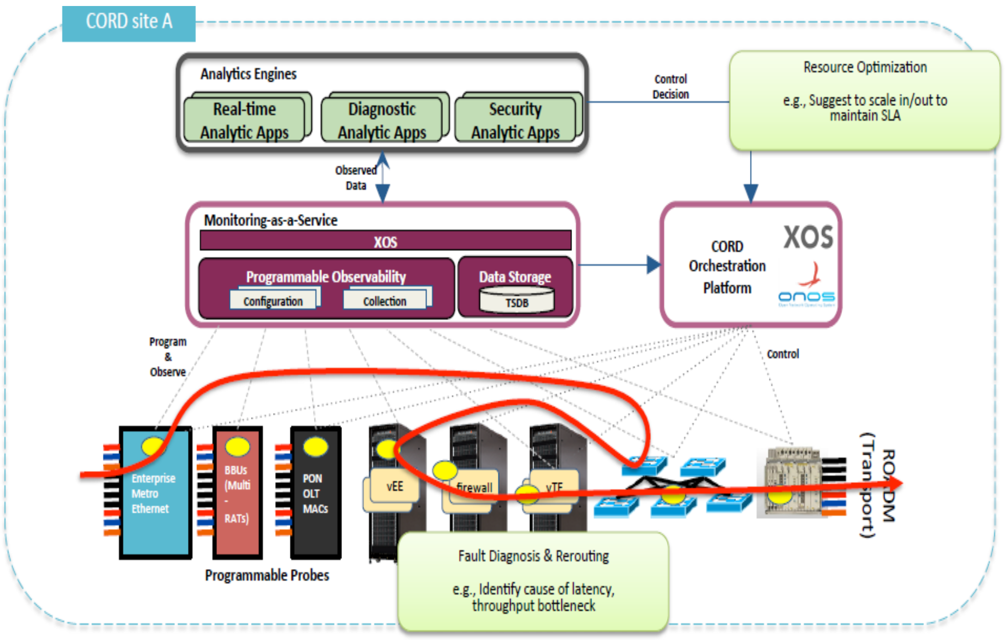


Figure 9: Implementation: Site local analytics

3. Integrated Analytics of Cross Sites Action

Example: Cloud-based Service migraCon across sites
1. Trigger alarm requiring cross sites acCon: Example. It can detect high loads at the site a natural disaster early warning and help in disaster situations
2. Analyze available options in the region and execute coordinated actions according to the given rules and regulations: Example. It can relocate the service instance to site B, with some spare capacity.

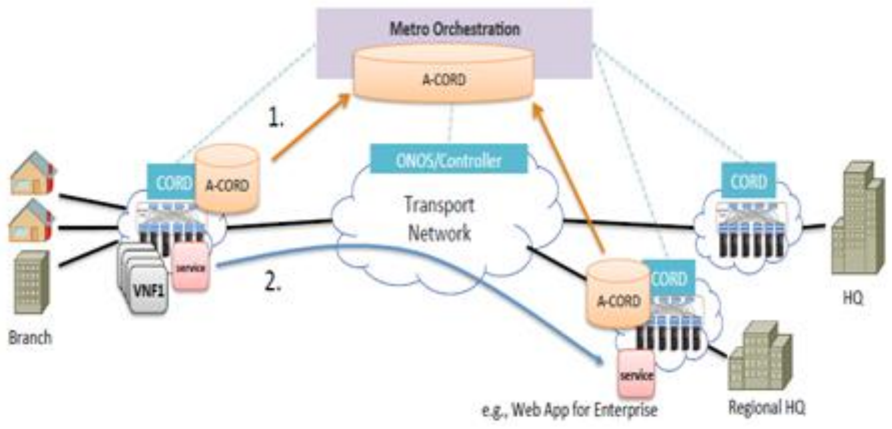


Figure 10: Integrated Analytics: Cross Sites Action

***5.1.15: Hierarchical End-to-End Network Control*:** There are two domain controllers in the end to end network control are as follows:

Global domain-agnostic controllers:
-Maintains an aggregated view of the underlying topology.
-Handle service requests from global orchestrator.

Local domain-specific controllers
Controls and manage an actual portion of the network

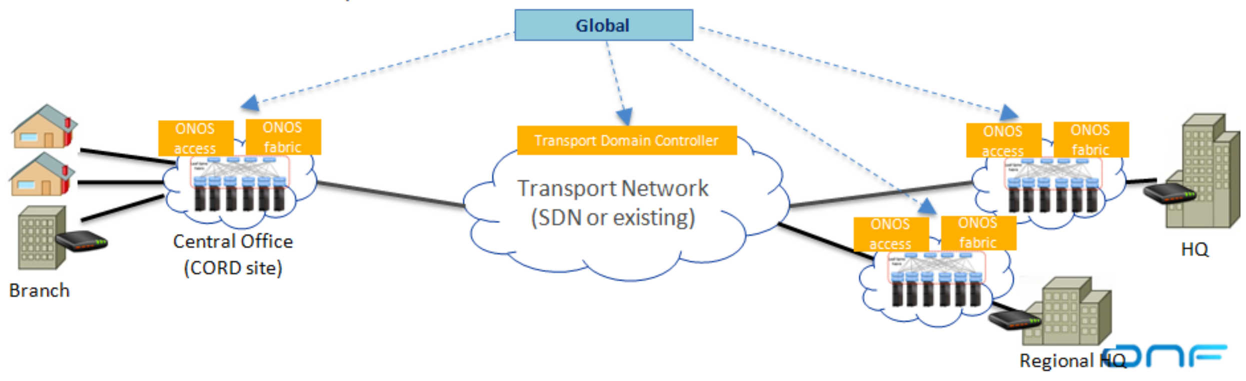


Figure 11:End-to-End Network Control

5.1.16 Overview to Local CORD POD Packet Operations

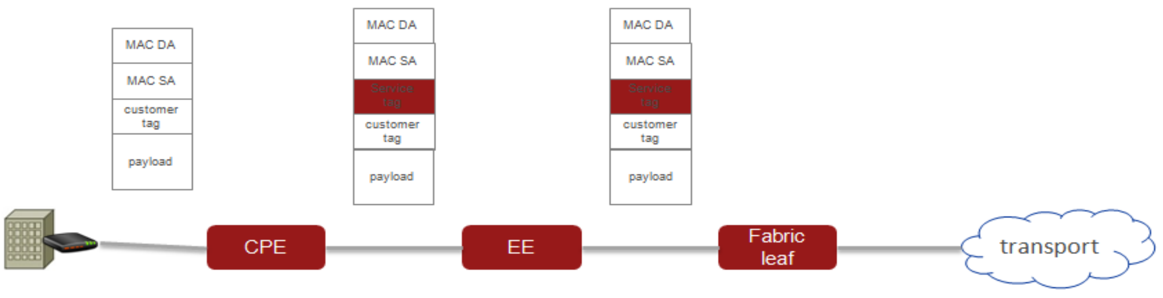


Figure 12: Local CORD POD Packet Operations

- CPE: Perform the Push/pop service tag ,QoS ,OA
  EE:It helps in managing Policing/QoS , Forward to Fabric
  Fabric leaf: The fabric leaf help in Cross-connection to transport network
  The more about CPE,EE and fabric leaf will be explained in the next sections.

5.1.17 Overview of Logical Local Service Graph

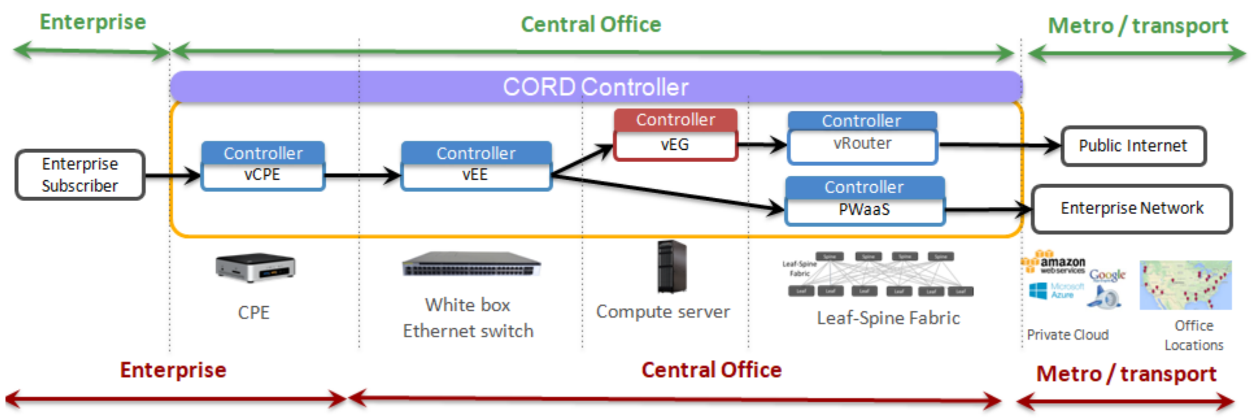


Figure 13: Logical Local Service Graph

VCPE:vCPE have a lot of features but some of the highlighted features are as follows:
1.Service classification
2.Programmable and on-demand OAM
3.Off-loaded to hardware

VEG: Following are some action performs by the vEG
1.DHCP for all.
2. NAT for Internet traffic
3.firewall
4.Extensible encryption, etc.

VEE: The following are features of vEE:
1.QoS: metering &queuing
2.Differentiate between public (go to vEG) and private traffic (go to PW)
3.Register to the global level

Pseudowire / VLAN xconnect: Following are a lot of features of Pseudowire / VLAN xconnect:
1.The fast path through the fabric
2.Connect EE-NNI, or EE-vEG and vEG-NNI
3.Applies NNI VLAN tag (at ingress)

5.2 What is E-CORD?

Enterprise CORD (E-CORD) [3] is a CORD use case that offers enterprise connectivity services over metro and wide area networks, using open source software and commodity hardware.

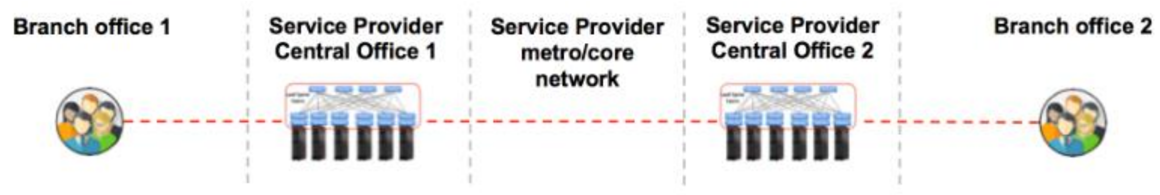

Branch office 1
Service Provider Central Office 1
Service Provider metro/core network
Service Provider Central Office 2
Branch office 2

Figure 14: ECORD Overview Diagram

E-CORD builds on the CORD platform to support enterprise customers, to provide services to the enterprises**.** It can also include Virtual Network Functions (VNFs) and Composition of service capabilities to support disruptive cloud-based enterprise services.
Enterprise customers can use E - CORD to quickly create on-demand networks between any number of endpoints or branches of the company. These networks are dynamically configurable; connection attributes and SLAs can be specified and provided on the fly. Also, Enterprise customers can choose to run network functions such as firewalls, WAN accelerators, traffic analytics tools, virtual routers, etc**.** as on-demand services provided and maintained within the service provider network.

Following are the terms we used in ECORD:

- **Local POD**: Each local CORD POD is a standard CORD POD equipped with specific access equipment. It is usually located in the Service Provider's Central Offices which help in:
1) Service Provider network to be connected by local pod.
2) Run value-added user services (example. firewalls, traffic analytic tools, WAN accelerators) at the edge of the network, close to the enterprise sites. Upstream, the POD usually connects to the Metro / Transport service provider or the public internet.

- **Global Node**: A global node is a machine running either in any other part of the Service Provider network or in the cloud, or, used to coordinate connectivity and services among all the local PODs of the E-CORD deployment. It has an instance of XOS (global orchestrator) and ONOS (transport network interconnecting edge sites).

5.2.1 ECORD System Overview

A typical E - CORD deployment consists of one global node and multiple (minimum two) CORD sites (PODs) connected to the same transport network.

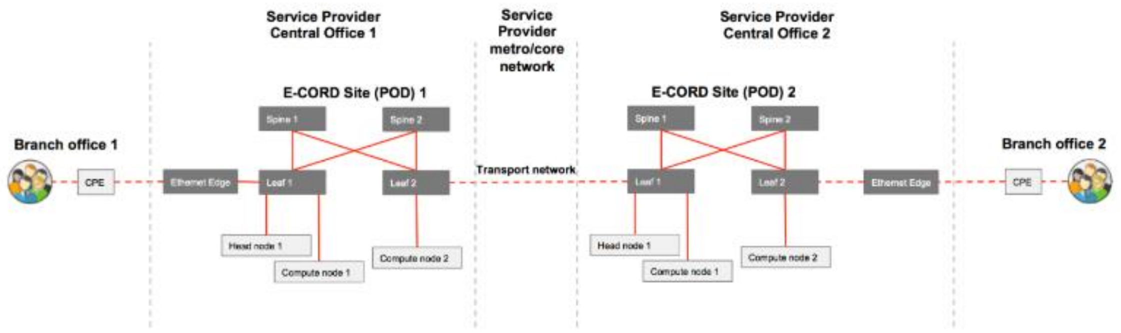


Figure 15: E-CORD Deployment

Each site is a standard CORD POD consisting of one or more compute nodes and one or more fabric switches. The transportation network provides connectivity between CORD sites. It can be converged packet-optical network or to a single packet switch for demo purposes. The transport network can be composed of legacy equipment, white-box switches, or a mix of both. The minimum requirement for the deployment of E - CORD is to provide Layer 2 connectivity between the PODs, specifically between the leaf fabric switches facing the upstream/metro COs network.

5.2.2 Global Node

The global node is responsible for organizing user requests and managing connectivity and service delivery between the POD. It runs only an instance of XOS and ONOS. Modules of global nodes are as follows:
1.**Global HTTP Channel**: Help in communication with the local sites.
2.Virtual Provider: Abstract view of E-CORD sites topologies is managed.

5.2.3 Local Sites

Local sites are responsible for collecting user's traffic and forwarding it either to other local sites or to the Internet. The Carrier Ethernet application uses both controllers to manage the physical network:

- **ONOS_CORD**: Help in running the application controlling edge access which also including CPE devices and Ethernet Edge (EE) devices.
- **ONOS_Fabric**: Help in running the application configuring the cross connections within the fabric of CORD, to bridge the CPEs to the transport network and the remote sites.

The following components for ONOS_CORD:

- CE-API
- Bigswitch service

- HTTP channel
- CE-VEE

The following components for ONOS_Fabric:

- CE-API
- Bigswitch service
- HTTP channel
- CE-fabric

The CE-API, the bigswitch service,and the HTTP channel are common to both ONOS_Fabric and ONOS_CORD. It manages in communication the ONOS running on the global node and the ones running on the local sites.
The local sites will abstract their network topology and expose it to the global node.

5.2.4 Service Graph

This responsible for managing end - to - end connectivity between PODs.
Each local site implements multiple services that manage the underlying hardware and software, and are thus able to connect the Enterprise Subscriber from the edge of the Central Office to the upstream network.

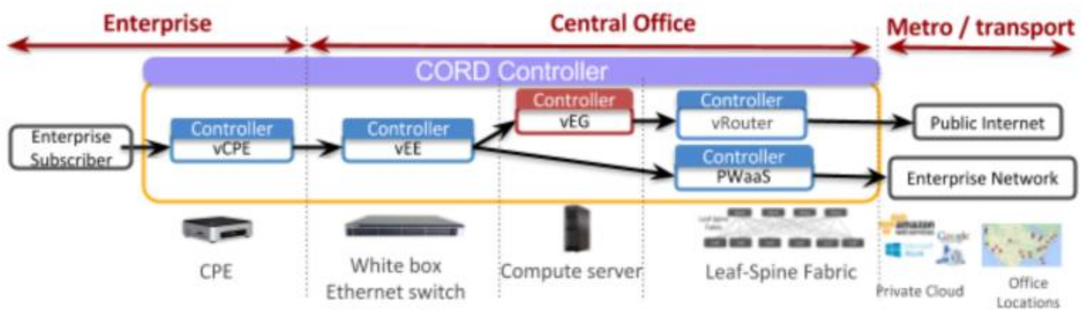


Figure 16: E-CORD local site service graph

5.2.5 Topology Abstraction

The global node help in maintaining an abstract view of the underlying topology for the sake of scalability and separating domain - specific and domain-agnostic concerns. In single CORD POD, there are two independent ONOS controllers: ONOS_CORD and ONOS_Fabric.
For each local site, the E-CORD application has two abstract devices to the global node: one is for ONOS_CORD and the other for the ONOS_Fabric. They represent a combination of

the real network elements that compose the topology of a local site.
Figure below shows how the two topologies got abstracted and exposed to the ONOS instance running on the global node.

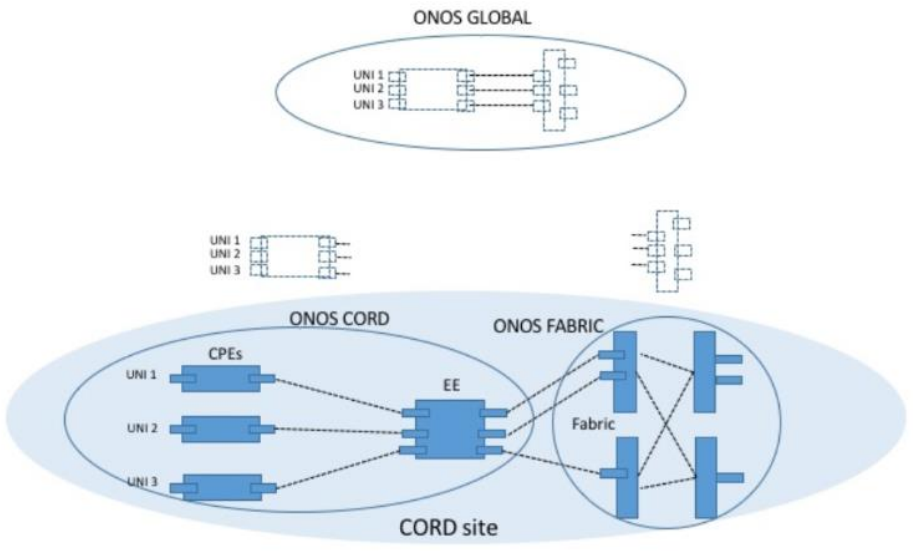


Figure 17: Topology running on global node

With the help of this, the global ONOS has very fewer devices and link data structures to manage the Path computation involves only these aggregated items, while the actual network provisioning is achieved by the local site controllers.
Topology information for the global node includes:

- User-to-Network Interfaces (UNIs): Can be found in MEF specs.[17]
- Network-to-Network Interfaces (NNIs): Can be found in MEF specs.[17]
- Associated bandwidth capacities: For admission control.

At the local level, the inter-domain link connects to devices controlled by different ONOS controllers.

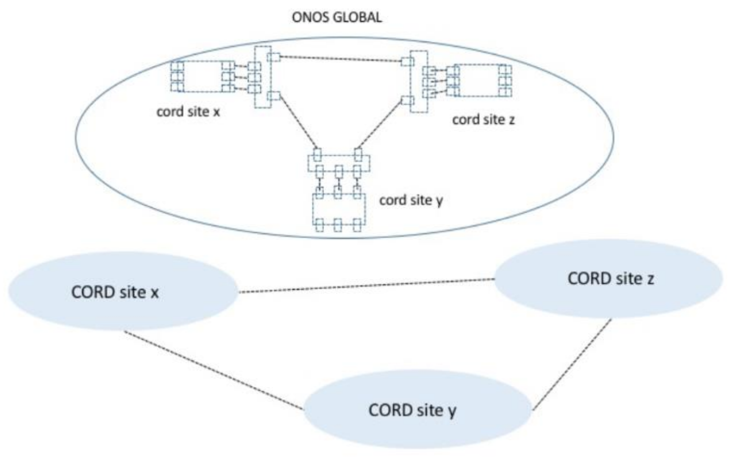


Figure 18: Global node topology

Installation Guide for E-CORD

https://guide.opencord.org/cord-5.0/profiles/ECORD/installation_guide.html
https://wiki.opencord.org/display/CORD/Enterprise+CORD

Development Environment for ECORD:
https://wiki.opencord.org/display/CORD/Development+Environment

E-CORD Developer Guide
https://guide.opencord.org/cord-5.0/profiles/ECORD/dev_guide.html

Building and Installing CORD
https://guide.opencord.org/cord-5.0/install.html

All the codes for the CORD platform:
https://github.com/opencord?tab=repositories

All the code for ONOS platform:
https://github.com/opennetworkinglab/onos

Internals of the CORD Build Process
https://wiki.opencord.org/display/CORD/Internals+of+the+CORD+Build+Process

5.3 Layer 2 Monitoring in E-CORD

5.3.1 Connectivity Fault Management (CFM): This is IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks. They run on VLANs (enterprise services/carrier services).CFM defines maintenance domains and maintenance points as there constituent, and the managed objects required creating and administering them. It defines the relationship between maintenance domains and the services offered by VLAN-aware bridges and provider bridges. It helps in explaining the procedures and rules used by maintenance points to maintain and diagnose connectivity faults within a maintenance domain; and Provides means for future expansion of the capabilities of maintenance points and their protocols.[15][4] There are types of test we can perform with this:

• Loopback
• Link trace
• Continuity Check Messages
– Describe Maintenance Association Endpoint (MEP) as a test node
– Logically partitioned into
• Maintenance Domains (MD)
• Maintenance Associations (MA)

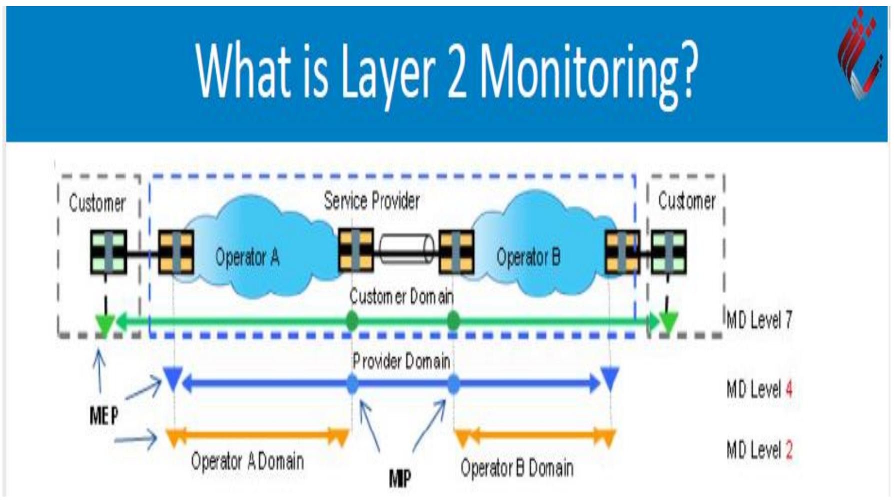


Figure 19: Layer 2 monitoring Overview

The following are the standard used in E-CORD:
1. Maintenance Domain (MD) can cover companies, suppliers or operators– Levels 0-7 discern the scope
• MEP provides test scope at either end of the domain (and MIP between)
   – Multiple MEPs can be involved - not just point to point
•Maintenance Association (MA) relates to a VLAN (service) within an MD and can contain 1 or more MEPs.
2. SOAM (Service Operations Administration and Management)
   – Metro Ethernet Forum (MEF) has defined MEF-17
3. Adds to MEP with extra tests
   – Delay Measurement (delay and jitter)
   – Loss Measurements
   – Signal Test
4.For Carrier Ethernet services
   – At each endpoint, the MEP supports E-Line, E-Tree,and E-Access.
   – Generally, a maintenance association is created by EVPL

5. E-CORD uses ONOS's Carrier Ethernet services to provide enterprise connectivity
6.CFM and SOAM have been added to ONOS which act as an API, a REST NBI and driver behaviors and all are in a loosely coupled way (does not strictly depend on carrier Ethernet)
7. E-CORD acts as an orchestrator
– Managing CE services
– Monitoring
– Incorporating A-CORD for "programmable observability

5.3.2 How is it applied to E-CORD and why?

- E-CORD Global orchestrates CE services across sites
- E-CORD Local implements CE services and CFM through ONOS CORD

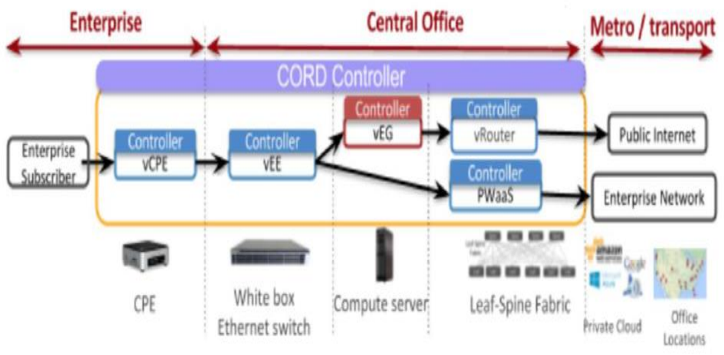


Figure 20: Basic ECORD Functions

• Within ONOS_CORD, all the Service runs and supports **REST NBI**, at any point of time MEPs can be created with devices that support behaviors and the TODO perform the following two functions:
– Persisting MEPs
– CLI for MEPs

5.3.3 Relationship to A-CORD: Closed Loop Control: This is a feature of ACORD where we can do implement features like Derive control decisions, Dynamic change of observability, Execution of control decision. And these features have been implemented in ECORD as follows:

- is key to the creation of autonomous (self-managed, self-protected) and SLA networks
- Means Continuous monitoring and analysis of the data of the sample for control decisions
- Optimizing the system status and then implementing these control decisions
- May change the observability level dynamically (Broader vs. Deeper) during the decision-making

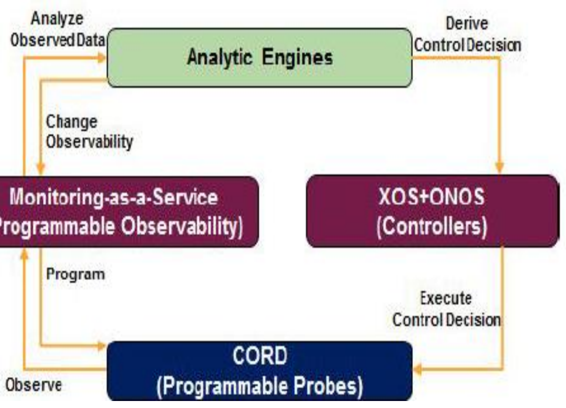


Figure 21: Relationship with ACORD

5.3.4 Summary: Layer 2 monitoring is a very important part of the ECORD environment. Layer 2 Monitoring in E-CORD is a feature that implements CFM and SOAM in VPNs, since CORD 4.0 and now Micro semi EA1000 device supports it today. With the help of layer 2 monitoring feature, the ECORD is the revolutionary telecom game changer. For device vendors, open source platforms supported by this industry (example. CORD, ONOS, etc.) Need to reconsider existing software systems.

5.4 Carrier Ethernet Application

Enterprise CORD offers L2VPN connectivity services using a Carrier Ethernet application in ONOS. [5]

5.4.1 Overview

The CE app is used to install and manage MEF-defined Carrier Ethernet services, along with relevant components like UNIs and Bandwidth Profiles (BW profiles).

5.4.2 Prerequisites

The application is added in the ONOS applications to perform CORD repository and can be activated from the ONOS CLI as follows:

Onos> app activate org.onosproject.ECORD.carrierethernet

5.4.3 Introduction of Classes

CarrierEthernetManager.java: Help in Performs CE service validation, installation,and removal. The validation part checks whether there do enough resources to accommodate the

UNIs comprise the CE service request. The UNIs that cannot be accommodated are omitted to receive request services.

CarrierEthernetVirtualConnection.java: Represents a CE Ethernet Virtual Connection (EVC) including the associated UNI and the service type. Three types of CE EVCs:
1:POINT_TO_POINT
2:MULTIPOINT_TO_MULTIPOINT
3:ROOT_MULTIPOINT.
Now, VLAN-based connectivity across the Carrier Ethernet Network (CEN) is supported; each EVC has a globally unique VLAN ID which is automatically generated by the CE app. A unique **EVC id** also generated by the application upon service creation with the help of format which indicates the EVC type, whether it is virtual, as well as the VLAN ID used by it.

CarrierEthernetBandwidthProfile.java: It Represents a UNI BW profile and includes the entire relevant attributes example: Committed Information Rate (CIR), Committed Burst Size (CBS), Excess Information Rate (EIR) and Excess Burst Size (EBS). Three possible types of BW profiles are **INTERFACE, EVC, COS.**

CarrierEthernetUni.java: Represents a UNI and includes relevant attributes like the associated CE-VLAN IDs and BW profiles. Currently, the class can be used to denote either a UNI from EVC(in which case only a single BW profile may be present) or a so-called "global" UNI. For the first type of UNIs ,it is a type attribute (ROOT or LEAF) is used to indicate the role of the UNI within the EVC. On the other hand, global UNIs exist independently of service UNIs.

CarrierEthernetProvisioner: Connectivity between Packet network domain is managed by this class, which includes the selection of link paths and appropriate method calls from CarrierEthernetPacketNodeManager.java to establish forwarding and apply bandwidth profiles on network nodes.
CarrierEthernetPacketNodeManager.java: Abstract class used to control packet nodes across the CENforconnectivity in CEN

More Information about Carrier Ethernet Application is in the link below:

https://wiki.onosproject.org/display/ONOS/Carrier+Ethernet+Application#CarrierEthernetApplication-TableofContents

5.5 E-CORD Developer Environment

Now here I try to explain how to set up a Mininet environment for emulating metro (multi-domain) networks. It also contains all the details steps involved in testing various ONOS features related to the Packet/Optical and E-CORD use cases. This example I am taking from opencord wiki, and I am trying to explain the E-Cord development environment. [6]

5.5.1 Overview

Here an example of the model of a metro network with multiple "central office fabrics" (anything ranging from a single OVS to a set of CPqD soft switches in a clos topology) joined together with a "metro core" of Linc-OE soft ROADMs. Figure 22 shows an example with three central offices (blue squares) connected to an optical core of three ROADMs (pink square).

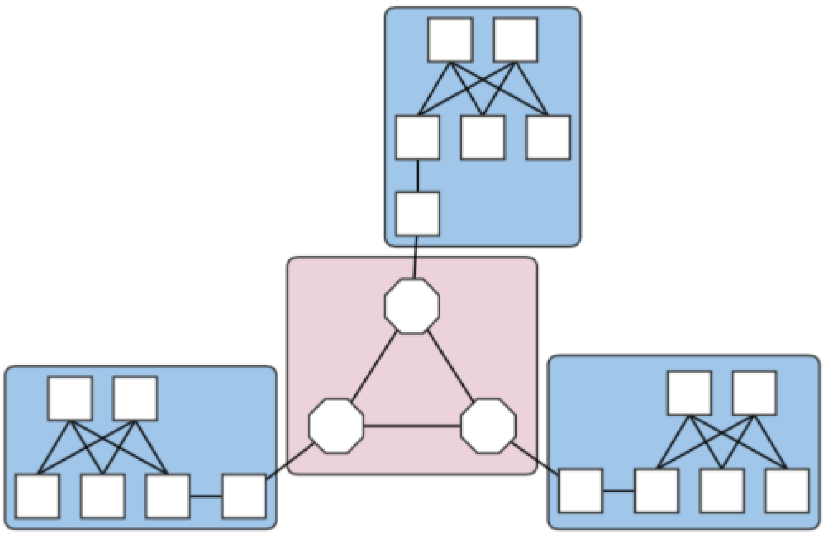


Figure 22: A four-domain metro network topology

All central office and the core are the different domains, which mean each region controlled by different (sets of) controllers.

5.5.2 Environment Setup

In the setup of the Mininet ECORD Environment we need to configure control planes, deployment host, ONOS hosts, we need to enable the path discovery,and all this is explained in the link below. So we need to follow all these steps for E-CORD Developer Environment

ECORD Development environment details:
https://wiki.opencord.org/display/CORD/Development+Environment

5.6 The E-CORD Pod

5.6.1 Overview

The E-CORD **"pod"** is multiple CORD sites (pods) connected by some transport network**.** The former can be a typical CORD site consisting of a head node, one or more computer nodes and one or more fabrics. The latter, which give connectivity between the CORD sites, can be anything, from an optical network requiring a converged view of the logical layers, to a single packet switch. The transport network can be composed of white boxes, legacy equipment, or both of it together. [7]

5.6.2 CORD Site Deployment

CORD sites are deployed by following quickstart_physical. The high-level procedure for deploying a CORD site is as follows:

1. Set up a CORD build Vagrant VM. This VM, called corddev, is where all required bits are downloaded and built; using a series of Ansible scripts invoked using Gradle. Now we need to build the Docker images and registry onto the build VM. The host of this VM is the build machine, typically a laptop, or perhaps an additional machine not used as part of the CORD site in the rack.
2. Publish the images to the designated CORD head node on the build VM. This sets a registry on the head node.
3. Need to edit the pod configuration file then need to specify the target head node IP, and address blocks to use for the CORD site internally.
4. Set MAAS and XOS on the head node. Like the other steps, the build process is initialized from the VM build. This step will arrange the network configurations on the head node to fit the CORD's site organization. This step will set up the head node as a PXE / DHCP / DNS server to manage the computer node(s) and switch within one CORD site.
5. Then PXE boots the compute node(s) and fabric switches against the head node. This step will do a fresh reinstall of the node(s)/switches and configures the network on the devices so that they logically reside within the CORD site management network.
6. Set up fabric configurations and connect the switches to the ONOS instance for managing the fabric. Likewise set up the XOS/CORD ONOS as appropriate (in the quick start, this would be the VTN app).

5.6.3 The ON.Lab Deployment

Overview

The E-CORD pod will contain three CORD sites connected by one transport node (packet switch). Each site is essentially a central office. In the E-CORD pod, each site is a *half-pod*, which contains:

- 1 fabric switch
- 1 head node
- 1 compute node

The pod also has an additional node which will help in running the global orchestrator components and which is indirectly connected to the CORD sites and transport switch through the management network. There are two different management network types in ECORD-POD:

1. The rack management network, accessible from the test bed network of the office
2. The **CORD management networks**, one per site and used by the head node to manage its corresponding compute and fabric nodes. From the rack management network, these are configured as VLANs at the top of the rack switch so that traffic does not logically leak into the test bed network.

5.6.4 Logical Topology

The deployment scripts for CORD assume that the compute and head nodes have four interfaces, two 1GBe, and two 40GBe. It also assumes that each has a particular role, and will rewrite the interfaces file (in Ubuntu), to configure them to fit that role. How the interfaces are configured itself guided by the pod configuration YAML file.

The following is a diagram 23 that attempts to conclude the logical and physical network configurations for the one-half pod, and how it connects to the ToR switch and global node. Only the configurable pieces that can be gleaned from the *interfaces* file and brctl are shown.

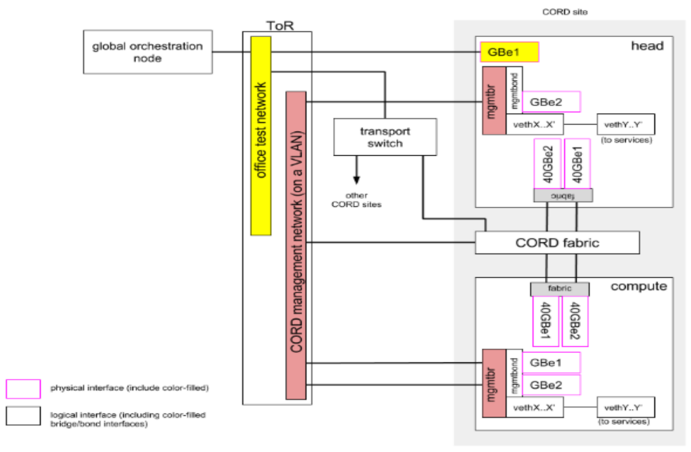


Figure 23: Logical and Physical network configuration

***5.6.5** The Head Node*

Once a CORD site is set up according to quickstart_physical [16], the head node will run a series of Docker containers and VMs that are part of the various CORD and XOS services. It will have three bridges for Linux:

1. Docker0–Have the docker registry and maven repo containers
2.br-<6-byte string> - CORD (MAAS, provisioner, dhcp-harvester, etc.) containers
3.mgmtbr - XOS component VMs (OpenStack, XOS, ONOS instances, ceilometer, etc)

Of the three, mgmtbr incorporates a bond, mgmtbond, that in turn one of the 1GBe interfaces (GBe2 above) are part of. Services like MAAS will use this interface to bootstrap computing nodes and switches and is the head node's CORD management plane - facing interface. Since the services include things like DHCP and expect their address block, mgmtbris only connected to a VLAN for the particular site. The virtual Ethernet links that expose the services hosted on the headnode to the management network are also connected to mgmtbr.

The other 1GBe interface is the external (WAN)-facing interface not associated with any bridge devices and is the means of access from the outside world to the head node (and the rest of the devices on the site). It, therefore, connects to the network of the office.

The interfaces of 40GBe are bonded together and connected to the fabric switch. Note that these fabric interfaces are not used in the head node**.**

The Compute Node

The one compute node of a half-pod is booted off of the MAAS service running on the head node. Since the compute node is only working on the CORD management network, both of its GBe interfaces are bonded together and attached to **mgmtbr**. A CORD site's head node is the entry point to reach the rest of a site's nodes, including the fabric switch's compute node and management shell.

The Global node

The global node interacts with the head node of each site and the transport network, so it connects to the office network along with the transport switch.

Global Head Node

Note: We also need to install the Docker Engine as well as Docker Compose on the machine that will function as a global head node. Refer to the Docker documentation for further instructions.

This node is a simplified XOS installation because it does not require any OpenStack components or OpenStack synchronizer. As already explained in the XOS install guide) [18], all we needed are the **PostgreSQL and XOS Docker containers**. It can easily build by cloning the XOS repo and generating the Docker containers as below.

```
$ git clone https://gerrit.opencord.org/xos
$ cd xos/containers
$ cd postgresql; make build; cd -
$ cd xos; make build; cd –
```

We use Docker Compose with a customized YAML file to run the images.Which is basically the same as docker-compose.yml with all comments except xos - db (db refers to the PostgreSQL database) and xos containers. Save this new file using any name you like; we use ECORD-global.yml.

```
$ Docker-compose -f ECORD-global.yml up –d
```

More detail about ECORD Pod is as follows:
https://wiki.opencord.org/display/CORD/The+E-CORD+Pod

5.7 XOS metro-net Service in ECORD

5.7.1 Overview

XOS brings the organizing principle of everything - as - a - Service (XaaS) to the CORD architecture. It also solves several of CORD's design requirements, to integrate control plane (SDN) and data plane (NFV) based services; the ability to support both access services and conventional cloud service to support for multiple security domains in the network; and the "end-to-end glue" needed to make CORD both extensible and controllable. The XOS metro-net Service is a dynamically on boarded XOS Service used for creating/managing/deleting Ethernet Services at the XOS level. [8]

5.7.2 Prerequisites

For Ethernet connection at the network level MEF SCA API is responsible.

5.7.3 Tutorial

Installing and activating the Service

Procedure:

There is one 'root' repo of interest, from it; scripts will download service and the base repo. Here is how you setup them up, generate containers.

Service Profile:

- git clone https://gerrit.opencord.org/service-profile

Create Containers

- cd service-profile/metro network
- make local_containers
- make

Open the GUI:

- http://myhostname:9999 , type on your browser.
- Account and credentials are typical XOS default

Metro network Video Demo Source:
https://www.youtube.com/watch?v=XHFiLA-XYFE

More Information:
https://wiki.opencord.org/display/CORD/XOS+metro-net+Service

Conclusion:

Enterprise CORD is the new network architecture in combination with SDN, NFV and cloud. It is bringing business benefits such as the self-service IT, lower operational expenses, security, and compliance with scale and agility. The requirements for enterprise networks may require different policies depending on whether they are accessed via LAN or WAN. The current state of the enterprise network is closed proprietary boxes with high operational cost and it is very difficult to create new services. As now with the help of ECORD it can be possible to provide a virtual network on demand, it allows customers to manage their network with all the controls. And the software stack enables services provider to create services according to customers need. ECORD environment also helps in managing the network easily and make it fast as it used infrastructure with commodity and open source building blocks. ECORD helps us to provide customer to observe, control and adapt features. In turn, enterprise customers can use E-CORD to create on-demand networks at all endpoints or for the company branches. With the use of ECORD services enterprise can make their network safe and they can spend less on their infrastructure component.

The E-CORD source code is available in cord 4.1 release and the official release of ECORD1.0has gone through testing for automated build and deploy phase. The basic enterprise services are rapidly growing and a lot of company like china mobile, China Unicom, NTT, Nokia are working on the environment of ECORD and china mobile has successfully deployed E-CORD pod in their labs. In the latest market survey the cord market is to grow to 300 billion dollars annually and 80% of the companies are planning to deploy cord services.

References:

Chapter 1: CORD

[1] Delivering Software based Network Infrastructure. Krish Prabhu. AT&T Labs (October 2015).
[2] Atrium: A Complete SDN Distribution from ONF.http://onosproject.org/wp-content/uploads/2015/06/PoC_atrium.pdf (2016).
[3] OpenStack: Open Source Cloud Computing Software. https://www.openstack.org/ (2016).
[4] Docker: Build, Ship, Run Any App, Anywhere. https://www.docker.com/ (2016).
[5] ONOS: Towards an Open, Distributed SDN OS. HotSDN 2014. P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar (August 2014).
[6] XOS: An Extensible Cloud Operating System. ACM BigSystems 2015. L.Peterson, S. Baker, A. Bavier S. Bhatia J. Nelson, M. Wawrzoniak, M. De Leeneer, and J. Hartman (June 2015).
[7] CORD: Re-inventing Central Offices for Efficiency and Agility. http://opencord.org (2016).
[8] Network Functions Virtualization—An Introductory White Paper. SDN and OpenFlow World Congress (October 2012).
[9]: https://www.commscope.com/Docs/How_CORD_will_impact_your_central_office_WP-112415-EN.pdf
[10]:https://wiki.opencord.org/display/CORD/CORD+Monitoring+Service
[11]: https://onosproject.org/
[12]: https://wiki.onosproject.org/display/ONOS/Wiki+Home
[13]: https://wiki.opencord.org/display/CORD/CORD+Reference+Implementation
[14]: Central Office Re-architected as a Datacenter (CORD). CORD Design Notes (June 3, 2015).
[15]:CORD Network Infrastructure: Switching Fabric, Overlay Virtualization, and Service Composition. CORD Design Notes (March 2016).
[16]:Service Assembly and Composition in CORD. CORD Design Notes (February 2016).
[17]:Security in CORD. CORD Design Notes (March 2016).
[18]:Virtual OLT (vOLT). CORD Design Notes (March 2016).
[19]:Virtual Subscriber Gateway (vSG). CORD Design Notes (March 2016).
[20]:Virtual Router (vRouter). CORD Design Notes (March 2016).
[21]:CORD Monitoring Service. CORD Design Notes (November 17, 2015)
[22]: https://wiki.opencord.org/display/CORD/User+Interfaces
[23]: https://wiki.opencord.org/display/CORD/Trellis%3A+CORD+Network+Infrastructure
[24]: https://www.sdxcentral.com/networking/nfv/mano-lso/definitions/whats-a-virtual-

router-vrouter/
[25]: http://opencord.org/wp-content/uploads/2016/03/Virtual-OLT.pdf
[26]: http://opencord.org/wp-content/uploads/2016/03/VirtualSubscriberGateway.pdf
[27]: https://www.opennetworking.org/trellis/

[28]: https://www.opennetworking.org/

[29]: https://onosproject.org/

Chapter 2: SDN:

[1] https://tools.ietf.org/html/rfc5440
[2] https://tools.ietf.org/html/rfc5575
[3] https://tools.ietf.org/html/rfc6241
[4] https://tools.ietf.org/html/draft-ietf-netconf-restconf-05
[5] https://tools.ietf.org/html/rfc6566
[6] https://tools.ietf.org/html/rfc4054
[7] http://openroadm.org/home.html
[8] https://www.opennetworking.org/images/stories/downloads/SDN-resources/technical-reports/oif-p0105_031_18.pdf
[9] http://www.oiforum.com/public/documents/OFC_SDN_panel_lyo.pdf
[10] https://tools.ietf.org/html/draft-ietf-pce-remote-initiated-gmpls-lsp-02
[11] https://tools.ietf.org/html/rfc6566
[12] https://www.opendaylight.org/odlbe
[13] https://www.opendaylight.org/ecosystem
[14] https://www.sdxcentral.com/cisco/datacenter/definitions/cisco-opflex/
[15] http://www.opencontrail.org/wp-content/uploads/2014/10/Figure01.png
[16]: http://onosproject.org/wp-content/uploads/.../Technical-Whitepaper-CORD.pdf

Chapter 3:NFV

[1] http://www.etsi.org/index.php/news-events/news/644-2013-01-isg-nfv-created
[2] http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf
[3] https://portal.etsi.org/NFV/NFV_White_Paper.pdf
[4] http://www.researchandmarkets.com/research/l3cw7s/network_functions
[5]https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/enterprise-network-functions-virtualization-nfv/q-and-a-c67-736831.pdf
[6]https://www.nojitter.com/how-simplify-branch-networks-nfv

Chapter 4: Integrated NFV/SDN Architecture

[1] ETSI. 2012. Network Functions Virtualisation - An Introduction, Benets, Enablers, Challenges and Call for Action. White Paper (Out. 2012).
[2] ETSI. 2015. Network Functions Virtualisation (NFV) - Network Operator Perspectives on Industry Progress. White Paper (Jan. 2015).
[3] http://dx.doi.org/10.1109/
[4] http://dx.doi.org/10.1145/1355734.1355746
[5] http://dx.doi.org/10.1109/JPROC.2014.

[6]dx.doi.org/10.1109/COMST.2015.2477041

[7]http://dx.doi.org/10.1186/s13174-015-0035-3

[8]http://dx.doi.org/10.1109/TLA.2016.7437249

[9] http://dx.doi.org/10.1109/ACCESS.2015.2499271

[10] http://dx.doi.org/10.1049/iet-net.2014.0117

[11]http://dx.doi.org/10.1016/j.infsof.2008.09.009 Special Section - Most Cited Articles

[12]
http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.02.01_60/gs_nfv002v010201p.pdf

[14] https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wpcontent/uploads/2014/10/open-ow-switch-v1.3.5.pdf.

[15] http://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_nfv-eve005v010101p.pdf

[16] http://docbox.etsi.org/ISG/NFV/Closed_WGs/PER/05-CONTRIBUTIONS/2014/NFVPER(14)000064_CloudNFV_Open_NFV_Framework_POC_1_Interim_Report.docx

[17]http://nfvwiki.etsi.org/images/NFVPER%2814%29000004r2_NFV_ISG_PoC_Proposal_Service_Chaining_for_NW_Function_Select.pdf

[18]http://docbox.etsi.org/ISG/NFV/Closed_WGs/PER/05CONTRIBUTIONS/2014//NFVPER(14)000080__NFV_ISG_PoC_Report__Automated_Network_Orchestration.docx

[19]https://docbox.etsi.org/ISG/NFV/TST/05CONTRIBUTIONS/2016//NFVTST(16)000019_Final_Report_for_PoC_13__Multi-Layered_Trac_Steering_for_.docx

[20] http://docbox.etsi.org/ISG/NFV/Closed_WGs/PER/05-CONTRIBUTIONS/2014/NFVPER%2814%29000108_
[21] https://docbox.etsi.org/ISG/NFV/TST/05

CONTRIBUTIONS/2015//NFVTST(15)000111r1_PoC_21__Network_Intensive_and_Compute_Intensive_Hardware_Acc.docx

[22]http://docbox.etsi.org/ISG/NFV/Closed_WGs/PER/05-CONTRIBUTIONS/2014/NFVPER(14)000113_Proof-of-Concept__23_Final_Report.docx

[23]http://docbox.etsi.org/ISG/NFV/TST/05CONTRIBUTIONS/2015/NFVTST(15)000116r3_PoC_26__nal_report.docx

[24]http://nfvwiki.etsi.org/images/NFVTST%2815%29000137_Final_Report_PoC_27_VoLTE_Service_based_on_vEPC_and_vIMS_Arc.pdf

[25] http://nfvwiki.etsi.org/images/PoC_28_report.pdf

[26]http://nfvwiki.etsi.org/images/NFVTST%2815%29000006_NFV_ISG_PoC_Proposal_SDN_Enabled_EPC_Gwy_r2_was_PER114.pdf

[27] http://nfvwiki.etsi.org/images/NFV_ISG_PoC38_Report.pdf

[28] http://dx.doi.org/10.1109/MNET.2015.7113222

[29] http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[30]http://dx.doi.org/10.1109/UCC.2013.89

[31] http://dx.doi.org/10.1109/EWSDN.2014.18

[32] http://www.t-nova.eu, Accessed: 2016-07-25.

[33] https://www.openstack.org/.

(2016). Accessed: 2016-07-25.

[34]. http://www.opendaylight.org. (2016).

[35] dx.doi.org/10.1109/GLOCOM.2015.7417869

[36dx.doi.org/10.1109/ICC.2015.7249414

[37] www.openSDNcore.org/. (2016). Accessed: 2016-07-25.

[38] http://dx.doi.org/10.1145/2342356.2342359

[39] http://dx.doi.org/10.1109/ISCC.2015.7405550

[40] http://dx.doi.org/10.1109/SDN4FNS.2013.6702546

[41] http://dx.doi.org/10.1145/2774993.2775002

[42]:http://dx.doi.org/10.1109/MNET.2015.7113225

[43]:http://dx.doi.org/10.1109/NFV-SDN.2015.7387414

[44] http://dx.doi.org/10.1109/NFV-SDN.2015.7387419

[45] http://dx.doi.org/10.1109/NFV-SDN.2015.7387398

[46]http://dx.doi.org/10.1109/ICTON.2015.7193561

[47] http://dx.doi.org/10.1109/MCOM.2017.1601039

[48] https://docs.docker.com/. (2016). Accessed: 2016-07-25.

[49] http://www.noxrepo.org/pox/about-pox/. (2016). Accessed

[50]
http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf

[51]http://dx.doi.org/10.1016/j.comnet.2015.09.028

[52] http://www.rad.com/template.MEDIA

[53] http://dx.doi.org/10.1109/CloudNet.2014.6969010

[54] //dx.doi.org/10.1109/MCOM.2015.7045397

[55] DOI:http://dx.doi.org/10.1109/TNSM.2017.2744807

[56] DOI:http://dx.doi.org/10.1145/2342441.2342465

[57]dx.doi.org/10.1109/NETSOFT.2015.7116164

[58] http://dx.doi.org/10.1109/GLOCOMW.2012.6477567

[59]http://dx.doi.org/10.1186/s13638-015-0450-y

[60] (March 2017). http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.pdf

[61] http://dx.doi.org/10.1109/ACCESS.2015.2461602

[62] 2017. ITU towards “IMT for 2020 and beyond”. (2017). http://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2020/Pages/default.aspx, Accessed: 2017-09-28.

[63] 2017. Verizon 5G Technical Forum. (2017). http://www.5gtf.org/, Accessed: 2017-09-28.

[64] 2017. 5G Infrastructure Public Private Partnership–5G PPP. (2017). https://5g-ppp.eu/, Accessed: 2017-09-28.

[65] https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf

[66] https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-use-cases-and-performance-evaluation-modeling_v1.0.pdf

[67] http://dx.doi.org/10.1109/MCOM.2014.6815890

[68]http://carrier.huawei.com/~/media/CNBG/Downloads/Program/5g_nework_architecture_whitepaper_en.pdf

[69] https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP_SoftNets_WG_whitepaper_v20.pdf

[70] http://dx.doi.org/10.1109/MCOM.2015.7263353

[71] http://dx.doi.org/10.1109/EWSDN.2014.27

[72] http://link.springer.com/chapter/10.1007/978-3-319-16292-8_5DOI: 10.1007/978-3-319-16292-8_5.

[73]http://dx.doi.org/10.1109/ISWCS.2014.6933481

[74] DOI:http://dx.doi.org/10.1109/EuCNC.2015.7194059

[75] http://dx.doi.org/10.1109/NETSOFT.2015.7116177
[77] http://dx.doi.org/10.1109/ICC.2017.7997391

[78]//dx.doi.org/10.1109/GLOCOM.2014.7037150

[79] http://dx.doi.org/10.1109/EuCNC.2015.7194108

[80] http://dx.doi.org/10.1109/MCOM.2017.1600935

[81]http://dx.doi.org/10.1364/JOCN.7.000B62

[82]http://dx.doi.org/10.1109/NETSOFT.2015.7116150

[84] http://dx.doi.org/10.1109/ECOC.2015.7341931

[85]http://dx.doi.org/10.1109/JLT.2015.2508044

[87] http://dx.doi.org/10.1016/j.comnet.2015.10.013

[88] http://dx.doi.org/10.1109/NETSOFT.2015.7116187

[89] http://dx.doi.org/10.1109/ONDM.2016.7494060

[92] http://dx.doi.org/10.1364/JOCN.9.00A135

[93] http://dx.doi.org/10.1109/EuCNC.2017.7980775

[94] www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf

[96]
http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf

[97] http://arxiv.org/abs/1602.00484

[98] https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf

[99] http://www.selfnet-5g.eu

[100] http://dx.doi.org/10.1155/2016/2897479

[101] http://dx.doi.org/10.1016/j.csi.2016.12.008SI: Standardization SDN& NFV.

[102] http://innovation.verizon.com/content/dam/vic/PDF/Verizon_SDN
NFV_Reference_Architecture.pdf

[103] https://github.com/nfvlabs/openmano. (2016). [104] http://openbaton.github.io/. (2016).
[105] http://www.project-oodlight.org/-oodlight/. (2016).

[106] https://osrg.github.io/ryu/. (2016 [108]//dx.doi.org/10.1109/MCOM.

2015.7045399

[109] http://dx.doi.org/10.1109/JLT.2015.

[110] http://dx.doi.org/10.1109/NFV-SDN.2015.7387422

[111] http://dx.doi.org/10.1109/NETSOFT.2015.7116179

[112]http://dx.doi.org/10.1109/MCOM.2015.7081091

[113] http://dx.doi.org/10.1109/MNET.2016.7579021

[114] https://www.xenproject.org/. (2016). Accessed: 2016-07-25.

[115] http://dx.doi.org/10.1109/NETSOFT.2015.7116155

[116]//dx.doi.org/10.1109/NETSOFT.2015.7116145

[117] http://dx.doi.org/10.1109/MNET.2015.7064898

[119] http://dx.doi.org/10.1109/MCOM.2015.7081089

[120] http://dx.doi.org/10.1109/ICC.2017.7997431

[121http://dx.doi.org/10.1007/s10922-014-9319-3

[122http://www.rfc-editor.org/rfc/rfc5810.txt

[123] http://dx.doi.org/10.1109/MCOM.2015.7081093

[124] http://dx.doi.org/10.1016/j.comnet.2015.09.015

[125] http://dx.doi.org/10.1109/JLT.2015.2509476

[126http://dx.doi.org/10.1016/j.future.2014.12.004 Special Section: Advanced Architectures for the Future Generationof Software-Intensive Systems.

[127] https://www.linux-kvm.org/page/Main_Page. (2016).

[128] http://dl.acm.org/citation.cfm?id=2616448.2616491

[129] https://datatracker.ietf.org/doc/html/draft-natarajan-nfvrg-containers-for-nfv-03 Work in Progress.

[130] http://dpdk.org/doc. (2016

[131] https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2013/04/open-ow-spec-v1.0.0.pdf.

[133] http://dx.doi.org/10.17487/rfc7047

[134] http://dx.doi.org/10.1109/NETSOFT.2016.7502474

[135] http://dx.doi.org/10.1109/TNSM.2017.2723477

[136] Open Network Foundation (ONF). 2017. The CORD Project. http://opencord.org/. (2017). Accessed: 2017-10-02.

[138] http://dx.doi.org/10.1109/

[140]http://dx.doi.org/10.1145/2342441.2342451
[141] :http://dx.doi.org/10.1145/2034773.2034812
[142http://dx.doi.org/10.1109/TNSM.2015.2401568

[143]www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/004/01.01.01_60/gs_NFV-EVE004v010101p.pdf

[145 http://eurosys2013.tudos.org/wp-content/uploads/2013/paper/Schwarzkopf.pdf

[146] http://dl.acm.org/citation.cfm?id=1972457.1972488

[147]http://dx.doi.org/10.1109/MCOM.2017.1600951

[148] http://dx.doi.org/10.1109/MIC.2017.3481355

Chapter 5 ECORD:

[1]: https://wiki.opencord.org/display/CORD/Enterprise+CORD

[2]: https://www.opennetworking.org/e-cord/

[3]: https://guide.opencord.org/cord-4.1/profiles/ECORD/overview.html

[4]: https://opencord.org/wp-content/uploads/2018/01/CORD-Build-9Nov17-Layer-2-Monitoring-in-E-CORD-CFM-SOAM.pdf

[5]: https://wiki.onosproject.org/display/ONOS/Carrier+Ethernet+Application

[6]: https://wiki.opencord.org/display/CORD/Development+Environment

[7]: https://wiki.opencord.org/display/CORD/The+E-CORD+Pod

[8]: https://wiki.opencord.org/display/CORD/XOS+metro-net+Service

[9]: https://schd.ws/hosted_files/cordbuild2017/77/2017-11-09%20Enterprise%20CORD.pptx

[10]: https://www.fujitsu.com/us/Images/ONOS-E-CORD-use-case.pdf

[11]: https://opencord.org/

[12]: https://guide.opencord.org/

[13]:https://wiki.opencord.org/display/CORD/Roadmap?preview=%2F1278702%2F1279426%2F2016-07-29_Enterprise_CORD_Roadmap.pdf

[14]: https://wiki.opencord.org/display/CORD/CORD+Summit+--+July+29%2C+2016?preview=%2F1278537%2F1279396%2F2016-07-29_Enterprise_CORD_overview-Ayaka.pdf

[15]: https://en.wikipedia.org/wiki/IEEE_802.1ag

[16]:https://github.com/opencord/cord/blob/master/docs/quickstart_physical.md

[17]: https://www.mef.net/resources/technical-specifications

[18]:(https://github.com/open-cloud/xos/tree/master/containers)