Verizon Network Infrastructure Planning

SDN-NFV Reference Architecture

Version 1.0
February 2016
Legal Notice

This document contains information regarding the development and evolution of SDN and NFV. The views expressed in this document are not final, and are subject to change at any time. Although this document contains contributions from multiple parties, it is not a consensus document and does not necessarily reflect the views of any or all contributors. Verizon reserves all rights with respect to its technology and intellectual property that may be disclosed, covered, or reflected in this document. The only rights granted to you are to copy, print, and publicly display this document. All trademarks, service marks, trade names, trade dress, product names and logos appearing in this document are the property of their respective owners, including, in some instances, Verizon. All company, product and service names used in this document are for identification purposes only. Use of these names, logos, and brands does not imply endorsement. Neither Verizon nor any contributor to this document shall be responsible for any action taken by any person in reliance on anything contained herein.

© 2016 Verizon. All rights reserved.
Executive Summary

The intersection of telecommunications, Internet and IT networking paradigms combined with advances in hardware and software technologies has created an environment that is ripe for rapid innovations and disruptions. This document is aimed at laying a foundation for enabling various industries to benefit from applications and devices that change how industries are run. ‘Software defined networking’ (SDN) and ‘network function virtualization’ (NFV) are two promising concepts developed in the telecommunications industry. SDN is based on the separation of control and media planes. The control plane in turn lays the foundation for dynamically orchestrating the media flows in real-time. NFV, on the other hand, separates software from hardware enabling flexible network deployment and dynamic operation.

Advances in hardware technologies, software technologies, cloud computing, and the advent of DevOps have led to agile software development in the IT and web application industries. These same methodologies can be used to enable the transformation of the telecommunications network to simplify operations, administration, maintenance and provisioning (OAM&P) and also to lay the network foundation for new access technologies (e.g., 5G).

Several industry bodies are working on various aspects of SDN and NFV. However, operator network transformation needs to get started now. This architecture consolidates and enhances existing SDN, NFV and orchestration into an implementable and operable framework. This document enhances the industry developed architectures to address several operating scenarios that are critical for network evolution.

- Standalone NFV architecture where virtual network functions (VNFs) can be deployed: This scenario is to migrate network functions that continue to grow and require continued capital investment. Some operational changes like DevOps, automation of workflows, and orchestration of services can be initiated while continuing to leverage existing backend systems as needed.
- Hybrid NFV where physical network functions (PNFs) and virtual network functions co-exist.
- Standalone SDN with new SDN controllers and white boxes: This scenario covers Data Center and Inter-DC use cases initially and could later expand to wide area networks (WANs).
- Hybrid SDN where new SDN controllers will work with existing forwarding boxes and optionally vendor-specific domain controllers.
- SDN-NFV networks.

The architecture extends the concepts of NFV to maintain SDN catalogs in addition to VNF catalogs. Concepts of NFV network services based on forwarding graphs of VNFs are extended to combo SDN-NFV network services. End-to-End Orchestration (EEO) extends the concepts of NFV orchestration to support orchestration of SDN services (transport/WAN/enterprise) and end-to-end network services.

The document covers all aspects of network operation like provisioning, capacity planning, KPIs and service assurance of infrastructure and software; charging; and security. It is intended to provide Verizon planning and operations teams an understanding of the end-to-end architecture.

This Architecture document has been co-authored with several Verizon industry participants – Cisco, Ericsson, Hewlett Packard Enterprise, Intel, Nokia, Red Hat and Samsung.
## Contributors and Acknowledgements

<table>
<thead>
<tr>
<th>Organization</th>
<th>Contributors</th>
<th>Acknowledgements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verizon</td>
<td>Kalyani Bogineni, Fellow</td>
<td>Dave McDysan, Anil Guntupalli, Girish Nair, Mike Ryan, Luay Jalil, Phil Ritter,</td>
</tr>
<tr>
<td></td>
<td>Denny Davidson, Document Editor</td>
<td>Mike Altland, Sankaran Ramanathan,</td>
</tr>
<tr>
<td></td>
<td>Antonia Slauson, Project Manager</td>
<td>Andrea Caldini, Chris Emmons, Gagan Puranik, Fernando Oliveira, Raquel</td>
</tr>
<tr>
<td></td>
<td>Lorraine Molocznik, Graphic Artist</td>
<td>Morera Sempere, and their teams.</td>
</tr>
<tr>
<td>Cisco</td>
<td>Christian Martin, Kirk McBean</td>
<td>Ken Gray, Ravi Guntupalli, Humberto LaRoche, Scott Wainner, Mike Geller</td>
</tr>
<tr>
<td>Ericsson</td>
<td>Doug Brayshaw, Torbjorn Cagenius</td>
<td>Mehul Shah, Tarmo Kuningas, Francesco Caruso</td>
</tr>
<tr>
<td>HPE</td>
<td>Tariq Khan, Kevin Cramer</td>
<td>Stinson Mathai, Noah Williams, Ajay Sahai, Raju Rajan, David Lenrow, Paul Burke,</td>
</tr>
<tr>
<td>Intel</td>
<td>Kevin Smith, Joseph Gasparakis</td>
<td>Jonas Arndt, Marie-Paule Odini</td>
</tr>
<tr>
<td>Nokia</td>
<td>Peter Busschbach, Nabeel Cocker, Jukka Hongisto, Steve Scarlett</td>
<td>Peter Kim, Volker Mendisch, Tuomas Niemelä, Marton Rozman, Norbert Mersch, Hui-Lan</td>
</tr>
<tr>
<td>Red Hat</td>
<td>Rimma Iontel, Gordon Keegan</td>
<td>Lu</td>
</tr>
<tr>
<td>Samsung</td>
<td>Ray Yuhanna, Nurig Anter</td>
<td>Bret Luango, Stephen Bates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Robbie Martinez</td>
</tr>
</tbody>
</table>
This page intentionally left blank.
# Table of Contents

**Executive Summary** .................................................................................................................. 3

1. **Introduction** .......................................................................................................................... 8

2. **Architecture Framework** ....................................................................................................... 14

3. **NFV Infrastructure** ............................................................................................................... 18

4. **VNF Manager** ....................................................................................................................... 35

5. **VNF Descriptors** ................................................................................................................... 39

6. **End-to-End Orchestration** .................................................................................................... 45

7. **End-to-End Network Service Descriptors** .......................................................................... 50

8. **Policy Framework** ............................................................................................................... 51

9. **SDN Controller Framework** ............................................................................................... 54

10. **Interfaces** ............................................................................................................................ 64

11. **Architectural Considerations** ............................................................................................. 72

12. **VNF considerations for NFVI** ............................................................................................ 78

13. **Reliability** ........................................................................................................................... 90

14. **IMS Functions** ..................................................................................................................... 93

15. **EPC Functions** ................................................................................................................... 104

16. **L1 – L3 Functions** ................................................................................................................ 113

17. **SGi-LAN Architecture** ....................................................................................................... 117

18. **Charging Architecture** ....................................................................................................... 131

19. **Service Assurance** .............................................................................................................. 138

20. **Key Performance Indicators** ............................................................................................... 145

21. **Security** ............................................................................................................................... 157

22. **Devices and Applications** .................................................................................................... 173

Annex A: **Intent-Based Networking** .......................................................................................... 192

Annex B: **Federated Inter-Domain Controller** .......................................................................... 194

Annex C: **Segment Routing** ...................................................................................................... 196

Annex D: **SDN Controllers** ...................................................................................................... 203

Annex E: **IMS VNF Management Example** ............................................................................. 205

References ....................................................................................................................................... 210

Acronyms ......................................................................................................................................... 213
This page intentionally left blank.
1 Introduction

Traditional networks have been designed with purpose-built network equipment (e.g., Routers, Ethernet switches, EPC hardware, firewalls, load balancers, etc.) based on vendor-specific hardware and software platforms. Deploying these monolithic network elements has resulted in long development and installation cycles (slowing time-to-market for new products and services), overly complex lifecycle management practices (adding to operational inefficiency and overhead) and increasing levels of investment driven by escalating customer demand at a rate that exceeds revenue growth.

An operator’s network is currently composed of a variety of “bundled” network elements, where the control, management, and data (user data traffic) functions are physically tied together and the bundled network elements are each provided by the same supplier. Deployment of new services or upgrades or modifications to existing services must be done on an element-by-element basis and requires tight coordination of internal and external resources. This bundling limits operational flexibility while increasing reliance on proprietary solutions from suppliers.

The goals of the SDN-NFV program in Verizon are the following:

1. Operational Efficiencies
   - Elastic, scalable, network-wide capabilities
   - Automated OAM&P; limited human touch
   - Dynamic traffic steering and service chaining

2. Business Transformation
   - Time-to-market improvements; elimination of point solutions
   - Agile service creation and rapid provisioning
   - Improved customer satisfaction

**Verizon SDN-NFV Based Network**

The following are key features of the network based on SDN and NFV:
- Separation of control and data plane;
- Virtualization of network functions;
- Programmatic control of network;
- Programmatic control of computational resources using orchestration;
- Standards-based configuration protocols;
- A single mechanism for hardware resource management and allocation;
- Automation of control, deployment, and business processes; and
- Automated resource orchestration in response to application/function needs.

Combining these techniques facilitates dynamic adaptation of the network based on the application, increases operational flexibility, and simplifies service development. Functions may be dynamically scaled to meet fluctuations in demand.

SDN and NFV together change the traditional networking paradigms. This significant shift in how an operator designs, develops, manages and delivers products and services brings with it a variety of technological and operational efficiencies. These benefits are aimed at fundamentally redefining the cost structure and operational processes, enabling the rapid development of flexible, on-demand services and...
Enhancements to the Software Defined Networking (SDN) Concept

The fundamental concept of ‘Software Defined Networking’ (SDN) changes the current network design paradigm by introducing network programmability and abstraction. In its initial, narrow definition, SDN is about separating the network control and data planes in L1, L2, L3, and L4 switches (Figure 1-1). This enables independent scaling of control plane resources and data plane resources, maximizing utilization of hardware resources. In addition, control plane centralization reduces the number of managed control plane instances, simplifies operations, and enables orchestration.

The idea of centralized network control can be generalized, resulting in the broader definition of SDN: the introduction of standard protocols and data models that enable logically centralized control across multi-vendor and multi-layer networks. Such SDN Controllers expose abstracted topology and service data models towards northbound systems, simplifying orchestration of end-to-end services and enabling the introduction of innovative applications that rely on network programmability.

Enhancements to Network Functions Virtualization

The second concept is network function virtualization (NFV). This concept is based on the use of commercial off-the-shelf (COTS) hardware for general purpose compute, storage and network. Software functions (implementations of physical network functions) necessary for running the network are now decoupled from the hardware (NFV infrastructure).

NFV enables deployment of virtual network functions (VNFs) as well as network services described as NF Forwarding Graphs of interconnected NFs and end points within a single operator network or between different operator networks.

VNFs can be deployed in networks that already have corresponding physical network functions (PNFs) and can be deployed in networks that do not have corresponding PNFs. The proposed architecture enables service assurance for NFV in the latter scenario and enables data collection for KPIs from the hardware and software components of NFV.

End-to-End Orchestration

This section provides a high-level description of the Management and Control aspects of the SDN and NFV architecture. This serves as an introduction to the more detailed description of the architecture shown in Figure 1-2 below.
Figure 1-2 shows the high-level management and control architecture. The figure shows a network infrastructure composed of virtualized and physical network functions. The virtualized network functions (VNFs) run on the NFV Infrastructure (NFVI). The management and control complex has three main building blocks:

**NFV MANO** - Manages the NFVI and has responsibility for life-cycle management of the VNFs. Key functions:

- Allocation and release of NFVI resources (compute, storage, network connectivity, network bandwidth, memory, hardware accelerators)
- Management of networking between Virtual Machines (VMs) and VNFs, i.e., Data Center SDN Control
- Instantiation, scaling, healing, upgrade and deletion of VNFs
- Alarm and performance monitoring related to the NFVI

**WAN SDN Control** - Represents one or more logically centralized SDN Controllers that manage connectivity services across multi-vendor and multi-technology domains. WAN SDN Control manages connectivity services across legacy networks and new PNFs, but can also control virtualized forwarding functions, such as virtualized Provider Edge routers (vPE).

**End-to-End Orchestration (EEO)** - Responsible for allocating, instantiating and activating the network functions (resources) that are required for an end-to-end service. It interfaces with:

- NFV MANO to request instantiation of VNFs
- WAN SDN Control to request connectivity through the WAN
- PNFs and VNFs for service provisioning and activation

EEO and NFV MANO are shown as overlapping. This is because the ETSI NFV definition of MANO includes a Network Service Orchestration (NSO) function, which is responsible for a sub-set of functions that are required for end-to-end orchestration as performed by EEO.

There is a fundamental difference between NFV MANO on the one hand and WAN SDN Control on the other. NFV MANO is responsible for the operational aspects of NFV Infrastructure Management and VNF lifecycle management. As such, NFV MANO can instantiate VNFs without any awareness of the functions performed by that VNF. However, once a VNF has been instantiated it functions just like its physical
counterparts. For example, an operator may deploy both virtualized and physical PGWs. Other network functions (e.g. MME and SGW) and management and control systems should not see any difference in the external behavior of the two incarnations. Therefore, the service provisioning and activation actions performed by WAN SDN Control and EEO are the same, whether a network function is virtualized or not. To put it succinctly: NFV MANO knows whether a network function is virtualized without knowing what it does. WAN SDN Control knows what a network function does, without knowing whether it is virtualized.

**Interworking with legacy systems**

In today’s networks, services are managed through OSS and BSS systems that may interface with Element Management Systems (EMS) to configure network elements. Due to standardization of control protocols and data models, EMS systems will be gradually replaced by new systems that work across vendor and domain boundaries, such as SDN Controllers and generic VNF Management systems.

The high-level architecture shown in Figure 1-2 will not immediately replace existing systems and procedures. Instead, it will be introduced gradually. For example, WAN SDN Control can be introduced in specific network domains or for specific services, such as Data Center Interconnect. Over time, its scope will grow. Similarly, EEO can be introduced to orchestrate specific services that rely heavily on virtualized functions, such as SGI-LAN services, while existing services continue to be managed through existing OSS, BSS and EMS systems. Over time, EEO can be used for more and more services.

**Abstraction, automation and standardization**

The desire for service automation is not new, but in the past it was difficult to achieve. Due to a lack of standard interfaces and data models, OSS systems required fairly detailed knowledge of the vendor and technology domains that they had to stitch together. Several recent developments significantly lighten that burden.

![Abstraction, Automation and Standardization enable Service Orchestration](image)

**Automation** - The introduction of virtualization and associated management tools that are part of NFV MANO enable automated, template driven instantiation of VNFs, groups of VNFs and the networking between them.

**Standardization** - Work is underway in several standards organizations and open-source communities to create (de-facto) standard data models that describe devices and services. Using new protocols like NETCONF, OpenFlow, OPCONFIG and a data model (e.g. YANG), SDN Controllers can provision services across vendor and technology domains because they can use the same data models to provision different network functions in a standard way.
**Abstraction** - Since standardization enables SDN Controllers to manage services across vendor domains, these SDN Controllers can provide abstracted topology and service models to northbound systems, such as EEO. Therefore, EEO does not need the same detailed knowledge about the network as OSS required previously. Similarly, NFV MANO hides the details of the NFV Infrastructure from EEO. As a consequence of these abstractions, it is much easier to create End-to-End Network Service Descriptors that capture end-to-end workflows and much easier to implement an orchestration engine that can act on those descriptors and manage end-to-end service instantiation and activation.

**Specification Structure**

The specification is organized as follows:

- Section 2 provides an overview of the architecture along with a short description of the functions and interfaces.
- Section 3 describes the Network Function Virtualization Infrastructure (NFVI) covering platform aspects and CPU/Chipset aspects.
- Section 4 describes Virtual Network Function Manager (VNFM) and how it interfaces with other functions like VIM and NFVO.
- Section 5 describes the baseline VNFD and the additional informational elements needed for various VNF categories.
- Section 6 describes End-to-End Orchestration including functionality and interfaces.
- Section 7 describes the baseline NSD and the variations needed for different kinds of SDN and VNF based network services.
- Section 8 describes the Policy Framework for the target architecture.
- Section 9 describes SDN and the different kinds of SDN controllers.
- Section 10 provides short description of all the Interfaces in the Architecture described in Section 2.
- Section 11 provides architectural considerations.
- Section 12 describes various implementation considerations for VNFs on NFVI.
- Section 13 outlines reliability aspects for the architecture.
- Section 14, 15, 16, 17 cover various VNF categories: IMS functions, EPC functions, L1-L3 functions, and SGI-LAN functions, respectively.
- Section 18 describes the Charging Architecture for SDN and NFV.
- Section 19 describes Service Assurance for the target architecture.
- Section 20 lists the key performance indicators for all components of the architecture.
- Section 21 covers Security aspects of SDN and NFV.
- Section 22 addresses device and application aspects.
2 Architecture Framework

Verizon SDN-NFV architecture is based on SDN and NFV concepts developed in the industry. The architecture supports network function virtualization, software defined networking, and service and network orchestration. Figure 2-1 below shows the high-level architecture and identifies the functional blocks.

The functional blocks are listed below and are described in detail in later chapters of this specification.

- **Network Function Virtualization Infrastructure (NFVI)** includes all hardware and software components on which VNFs are deployed.

- **Virtualized Network Function (VNF)** is a software implementation of a network function, which is capable of running on the NFVI.

- **Physical Network Function (PNF)** is an implementation of a network function that relies on dedicated hardware and software for part of its functionality.

- **Virtualized Infrastructure Manager (VIM)** is responsible for controlling and managing NFVI compute, storage and network resources. It also includes the Physical Infrastructure Manager (PIM).

- **VNF Manager (VNFM)** is responsible for VNF lifecycle management (e.g. instantiation, upgrade, scaling, healing, termination).

- **End-to-End Orchestration (EEO)** is the function responsible for lifecycle management of network services. The orchestration function has a number of sub-components related to different aspects of that orchestration functionality. VNF orchestration is done by the NFV Orchestrator (NFVO).

- **Catalogs/Repositories** is the collection of descriptor files, workflow templates, provisioning scripts, etc. that are used by EEO, VNFM, and SA to manage VNFs and NFV/SDN/End-to-end network services.

- **Service Assurance (SA)** collects alarm and monitoring data. Applications within SA or interfacing with SA can then use this data for fault correlation, root cause analysis, service impact analysis, SLA management, security monitoring and analytics, etc.
• **Data Center SDN Controller** (DC SDN Controller) is responsible for managing network connectivity within a data center.

• **WAN SDN Controller** is responsible for control of connectivity services in the Wide Area Network (WAN).

• **Access SDN Controller** is responsible for control of wireline or wireless access domains.

• **Domain/Vendor-Specific Controller** are optional controllers that may be required to handle specific vendors or technology domains in the absence of standard interfaces or for scalability purposes.

• **Service Orchestration** is the customer facing function responsible for providing services catalog to the portal.

• **Portals** include customer portal where customer can order, modify and monitor their services and the Ops portal.

• **Element Management System** (EMS) is a legacy system responsible for the management of specific network elements.

• **Operations Support Systems** and **Business Support Systems** (OSS/BSS) are responsible for a variety of functions such as order entry, service fulfillment and assurance, billing, trouble ticketing, helpdesk support, etc.

The different kinds of catalogs are as follows.

- C1 - Catalog of individual VNFs (e.g., BNG, PCRF)
- C2 - Catalog of SDN based network services (e.g., IP VPN service, E-line service, lambda service)
- C3 - Catalog of network services (e.g., VNF based services, End-to-End Network services)

C1 is used by the VNFM for lifecycle management of VNFs. The VNF descriptor files tell the VNFM how to construct the VNF, i.e., it identifies the VMs, the order in which they are to be instantiated, the required network connectivity, the scaling properties, etc.

C2 above is used by the SDN controller. Knowledge of how to manage the service using the data models is embedded in the controller itself.

C3 above is used by EEO. The end-to-end network service descriptor may contain information about VNF forwarding graphs and associated descriptors, virtual links and associated descriptors, WAN connectivity aspects, PNF selection, and configuration scripts required for network service activation.

Note that service definitions can be used in a recursive manner. For example, a service exposed by EEO may rely on a connection service exposed by the WAN SDN Controller and published in the SDN service catalog, and on VNF functionality published in the VNF catalog.

There are various repositories in the architecture. Repositories are updated based on the activities of EEO, VNFM, VIM and SA.

- R1. VNF Instances
- R2. VNF Resources
- R3. Network Service Instances and Resources
- R4. Inventory information such as topology information
- R5. Service Assurance related repository of alarm and performance information

Figure 2-2 below shows the detailed architecture with interfaces.

Figure 2-2: Verizon SDN-NFV Detailed Architecture

The reference points in the architecture are listed below and described in Chapter 10.
<table>
<thead>
<tr>
<th>Reference Point Name</th>
<th>Description</th>
<th>Implementation Example(s)</th>
</tr>
</thead>
</table>
Libvirt drivers: [https://libvirt.org/drivers.html](https://libvirt.org/drivers.html) |
Libvirt drivers: [https://libvirt.org/drivers.html](https://libvirt.org/drivers.html) |
| NI-Vi               | NFVI - VIM | OpenStack Plugins:  
| Or-Vi               | VIM - Orchestrator | OpenStack API: [http://developer.openstack.org/api-ref.html](http://developer.openstack.org/api-ref.html) |
| Vi-Vnfm             | VIM - VNFM | OpenStack API: [http://developer.openstack.org/api-ref.html](http://developer.openstack.org/api-ref.html) |
| Ve-Vnfm             | VNF-VNFM | NetConf*, ReST*, Proprietary CLI |
TOSCA: [http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html](http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html)  
Yang/Netconf: [http://www.yang-central.org/](http://www.yang-central.org/) |
TOSCA: [http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html](http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html)  
Yang/Netconf: [http://www.yang-central.org/](http://www.yang-central.org/) |
TOSCA: [http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html](http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html)  
Yang/Netconf: [http://www.yang-central.org/](http://www.yang-central.org/) |
| Os-Ma               | OSS/BSS - Orchestrator | Proprietary |
| Or-Sa               | Orchestrator - SA | NetConf*, ReST* |
| Or-Emis             | Orchestrator - EMS | Proprietary |
| Ve-Sa               | VNF - SA | sFTP |
| Vnfm-Sa             | VNFM - SA | ReST* |
| Vi-Sa               | VIM - SA | ReST* |
| Sdnc-Sa             | SDN Controller - SA | ReST* |
| Nfiv-Sa             | NFVI - SA | NetConf*, ReST*, XMPP* |
| Sdnc-Vi             | SDN Controller - VIM | ReST* |
| Or-Sdnc             | Orchestrator - SDN Controller | ReST*, ReSTConf*, OpenFlow, OpenDayLight |
| Or-Nf               | Orchestrator - PNF/VNF | NetConf*, ReST*, Proprietary CLI |
| Sdnc-Nf             | SDN Controller - PNF/VNF | NetConf*, OpenFlow, PCEP |
| Sdnc-Net            | SDN Controller - Network | Published APIs, Object Models, Data Models, CLI |
| Cl-N                | Network - Collection Function | Port Forwarding |
| Cl-Sa               | Collection Function - SA | ReST* |
| Dsc-Nf              | Domain Specific Controller - PNF | Proprietary |

* with standard device and service data models

Table 2-3: Architecture Reference Points
3 NFV Infrastructure

Verizon NFV Infrastructure (NFVI) is aligned with the ETSI NFV definition, which comprises all hardware and software components building up the environment in which VNFs are deployed, managed and executed.

This chapter describes the Network Functions Virtualization infrastructure (NFVI) block shown in Figure 3-1 that includes hardware resources, virtualization layer & virtualized resources, CPU/chipset, and forwarding box and their associated interfaces. The figure below shows the NFVI domains in the architecture.

![Figure 3-1: NFVI Domains within the SDN-NFV Architecture](image)

3.1 Platform Aspects of the NFVI

3.1.1 Introduction & Definitions

The NFVI functional block in the Architecture has three domains:

- **Compute Domain** that includes compute and storage resources that are commonly pooled.
- **Virtualization Domain** that includes the virtualization layer for VMs and containers that abstracts the hardware resources and decouples the VNF from the underlying hardware.
- **Infrastructure Networking Domain** that includes both physical and virtual networking resources that interconnect the computing and storage resources.

Verizon’s NFVI will host VNFs from multiple vendors and needs to support the requirements for real-time applications like VoLTE. Application may be hosted in multiple data centers connected through wide-area networks. The NFVI needs to support different requirements for application latency. The architecture supports deployment of both green-field VNFs and VNFs of existing PNFs. The NFVI includes NEBS compliant and non-NEBS hardware.

**Compute Domain**

This domain includes the computing & storage resources that provide processing & storage to VNFs through the virtualization layer (e.g. hypervisor). Computing hardware is assumed to be COTS. Storage resources can be differentiated between shared storage (NAS or SAN) and storage that resides on the server itself.
A more detailed discussion on CPU and Chipset aspects of the compute domain is available in section 3.2. This domain interfaces with the hypervisor domain using the Vi-Ha interface described in section 3.2.5.

COTS hardware platforms provide an optimum mix of value and performance. These hardware resources are characterized by high performance, non-blocking architectures suitable for the most demanding network applications. Examples of these configurations are a larger number of processor cores, large memory and support for I/O capabilities like SR-IOV and DPDK enabled NICs. Section 3.2 provides details on some of these capabilities. This also requires the hardware to be configured with redundant & field replaceable components (like power supplies, fans, NIC’s, management processors, enclosure switching modules, etc.) that eliminate all hardware related single points of failure.

Some of the considerations and guiding principles for this domain are as follows:

- Modular & extendable hardware that share communication fabrics, power supplies, cooling units & enclosures
- Redundant & highly available components with no single point of failure for
  - Communication fabric (NICs & enclosure interconnect)
  - Power Supplies
  - Cooling Units / Fans
  - Management Processors
- Support a non-blocking communication fabric configuration
- Out of band management
- Support for advanced network I/O capabilities like (additional detail in Section 3.2)
  - Single Root I/O Virtualization (SR-IOV)
  - Data Plane Development Kit (DPDK)
- Plug-in card support for workload specific advanced capabilities like (additional detail in Section 3.2)
  - Compression acceleration
  - Media specific compute instructions
  - Transcoding acceleration & graphics processing

Virtualization Domain

The virtualization layer ensures VNFs are decoupled from hardware resources and therefore the software can be deployed on different physical hardware resources. Typically, this type of functionality is provided for computing and storage resources in the form of hypervisors and VMs. A VNF is envisioned to be deployed in one or several VMs. In some cases, VMs may have direct access to hardware resources (e.g. network interface cards or other acceleration technologies) for better performance. VMs or containers will always provide standard ways of abstracting hardware resources without restricting its instantiation or dependence on specific hardware components.

This domain provides the execution environment for the VNFs that are exposed using the Vn-Nf interface and implemented in OpenStack and Linux environments using libvirt.

The hypervisor domain is characterized by Linux and KVM/libvirt. For signalling and forwarding applications, the hypervisor domain has to enable predictable performance (low jitter) and low interrupt
latency by utilizing Linux distributions that include real-time extensions and a pre-emptible kernel. The requirements needed by the telecommunication industry for a Linux operating system are captured in the Linux Foundation's Carrier Grade Specification Release 5:

[REFERENCE: http://www.linuxfoundation.org/collaborate/workgroups/cgl]. They provide a set of core capabilities that progressively reduce or eliminate the dependency on proprietary systems.

Infrastructure Networking Domain

Infrastructure Networking Domain primarily includes the network resources that are comprised of switching and routing functions. Example components include Top of Rack (TOR) Switches, routers, wired or wireless links that interconnect the compute and storage resources within NFVI, etc. The following two types of networks within this domain are of relevance to Verizon:

- **NFVI network** – A network that interconnect the computing and storage resources contained in an NFVI instance. It also includes specific switching and routing devices to allow external connectivity.

- **Transport network** - A network that interconnects different NFVI instances or to other network appliances or terminals not contained within the NFVI instance.

This Infrastructure Networking domain exposes network resources to the hypervisor domain using the Vi-Ha interface with the compute domain. It also uses the ETSI NFV Ex-Nf reference point (not shown in Figure 2-2) to interface with existing and/or non-virtualized network resources.

The forwarding boxes and whitebox networking strategies required for this Networking domain are described in the SDN chapter.

The infrastructure networking domain is characterised by mainstream physical and virtual switches. This infrastructure also provides a non-blocking architecture for the physical network, and provides an I/O fast-path using a combination of:

- **User space Data Plane Development Kit (DPDK) enabled vSwitch** - This enables the network packets to bypass the host kernel space reducing the latency associated with an additional copy to the kernel user space, which results in significant I/O acceleration. Near line-speed is possible by using DPDK poll mode drivers in the guest VM. This allows the VMs to leverage all the capabilities of virtual switch like VM mobility & network virtualization. Information about DPDK libraries, installation and usage can be found at www.intel.com or www.dpdk.org.

- **Single Root I/O Virtualization (SR-IOV)** - This enables the VM to attach a Virtual Function on the NIC, which directly bypasses the host to provide line speed I/O in the VM. However, since this bypasses the host OS, VMs are not able to leverage the capabilities afforded by vSwitch like VM mobility & virtual switching. Information about SR-IOV overview and usage can be found at www.intel.com.

- **PCI-Passthrough** - This is similar to SR-IOV, but in this case, the entire PCI bus is made visible to the VM. This allows line speed I/O but limits the scalability (number of VMs per host) and flexibility.
Guiding Principles

High-level requirements for the NFV Infrastructure to host data plane sensitive (forwarding & signaling VNFs) applications can be characterized by:

- **Availability and Reliability**
  - 5x9's VNF availability & reliability
  - Advanced self-healing of OpenStack Control Plane nodes like Controller/Schedulers/API Servers

- **Performance**
  - Workload placement based upon NUMA/Memory/Networking Requirements
  - High Performance Networking with DPDK-enabled vSwitches or Switch bypass using technologies like SR-IOV

- **Manageability**
  - In-Service upgrade capabilities
  - Live migration of NFV workloads

3.1.2 VIM, PIM & SDN Considerations

This section describes the Virtual Infrastructure Manager (VIM), which is responsible for controlling and managing the NFVI compute, storage and network resources, the Physical infrastructure Manager (PIM), and the SDN considerations within the NFVI functional block. ETSI NFV defines VIM but does not define PIM, which is an important functional block.

Virtual Infrastructure Management Specifications

The following list expresses the set of functions performed by the VIM. These functionalities may be exposed by means of interfaces consumed by VNFM and NFVO or by authorised external entities:

- Orchestrating the allocation/upgrade/release/reclamation of NFVI resources, and managing the association of the virtualized resources to the physical compute, storage, networking resources.

- Supporting the management of VNF Forwarding Graphs, e.g., by creating and maintaining Virtual Links, virtual networks, subnets, and ports, as well as the management of security group policies to ensure network/traffic access control.

- Managing repository-based inventory-related information for NFVI hardware resources and software resources, and discovery of the capabilities and features of such resources.

- Management of the virtualized resource capacity and forwarding of information related to NFVI resource capacity and usage reporting.

- Management of software images as requested by other NFV-MANO functional blocks.

- Collection of performance and fault information for hardware, software, and virtualized resources; and forwarding of performance measurement results and faults/events information relative to virtualized resources.

- Management of catalogs of virtualized resources that can be consumed in the NFVI.
VIM will evaluate the placement policies defined in the VNF Descriptor and chose the appropriate resource pool, which may include general-purpose compute resource pools of data plane optimized compute resource pools.

VIM exposes the capabilities of the NFVI using the APIs to upstream systems like the NFV Orchestrator (NFVO using Or-Vi interface) and VNF Manager (VNFM using Vi-Vnfm interface). Since signaling and forwarding functions are characterized by high throughput and predictable performance (consistent forwarding latency, available throughput, and jitter), the combination of NFVI & VIM is required to provides a fine-grained control on various aspects of the platform, including:

- High performance accelerated Virtual Switch
- Simple and programmatic approach
- Easy to provision and manage networks
- Overlay networks (VLAN, VXLAN, GRE) to extend domains between IP domains
- Accelerated distributed virtual routers
- Network link protection that offloads link protection from VMs
- Rapid link failover times
- Support for PCI-Passthrough and SRIOV

The infrastructure exposes advanced capabilities using OpenStack API's that can be leveraged by VNF Managers and NFV Orchestrators. The fine grained VM placement control imposes requirements on the VM scheduler & API, including:

- Mixed ‘dedicated’ and ‘shared’ CPU model – Allows an optimal mix of different CPU pinning technologies without the need for assigning dedicated compute nodes
- Specification of Linux scheduler policy and priority of vCPU threads – Allows API driven control
- Specification of required CPU model – Allows VM to be scheduled based on CPU model
- Specification of required NUMA node – Allows VMs to be scheduled based on capacity available on the NUMA node
- Provider network access verification – Allows VMs to be scheduled based on access to specific provider network(s)
- Network load balancing across NUMA nodes – Allows VMs to be scheduled so VMs are balanced across different NUMA nodes
- Hyperthread-sibling affinity – Allows VMs to be scheduled where hyperthread sibling affinity is required

Advanced VM scheduling along with the capabilities of the underlying infrastructure (carrier-grade Linux & real-time KVM extensions, etc.) allows predictable performance with advanced resiliency. These capabilities can be leveraged by the VNF Managers and NFV Orchestrators to make optimum policy-based deployment decisions by matching the VNFs with appropriate operating environments. Some of the characteristics of the resiliency framework are:

- No cascading failures between layers
- Layers detect and recover from failures in adjacent layers
- Live Migration of VMs
  - For maintenance and/or orchestration procedures
Physical Infrastructure Management Specifications

The Physical infrastructure Manager (PIM) is an important part of network management and orchestration, which is part of the Virtual Infrastructure Management (VIM) block. These are covered by VI-Ha and Nf-Vi interfaces. These interfaces provide the options for programmatic control of the NFVI hardware capabilities such as:

- **Low Latency BIOS Settings** – Modern rack-mount servers and server blades come with a number of options that can be customized in firmware or BIOS that directly impact latency. The infrastructure must offer northbound APIs to control these settings so that the VIM or other orchestration entities can manage this higher up in the stack.

- **Firmware Management** – Since modern servers have a number of components that have their associated firmware, PIM will need to be able to manage the firmware, and be able to flag (and update, as needed) if the required firmware versions are not detected. PIM needs to offer northbound APIs that provide programmatic and fine grained control that can be utilized by VIM and upstream orchestrators to automate the entire workflow.

- **Power Management** – Programmatic management of the power lifecycle of a server is required for providing the resiliency required by Telco applications. The primary use case is isolation of faults detected by the availability management framework (also called fencing). This allows components to be taken out of service while the repair actions are underway.

- **Physical Infrastructure Health** – Most modern systems have sophisticated fault detection and isolation capabilities, including predictive detection of faults by monitoring and analyzing a number of internal server metrics like temperature, bad disk sectors, excessive memory faults, etc. A programmatic interface is required so upstream systems can take advantage of these alerts and take action quickly or before the faults impact the service.

- **Physical Infrastructure Metrics** – While most of the physical infrastructure metrics are exposed to the operating systems, collecting of these metrics impose an overhead on the system. A number of modern servers provide an out-of-band mechanism for collecting some of these metrics, which offloads their collection and disbursement to upstream systems.

- **Platform lifecycle management** – Since most modern servers have a number of sub-components, the platform needs to support the lifecycle management of their subcomponents and be able to upgrade the firmware of redundant components in a rolling manner with minimal or no impact to service availability.

- **Enclosure Switch Life Cycle** – Most bladed environments include multiple switches that are typically installed in redundant pairs. The PIM needs to be able to address lifecycle actions (update, upgrade, repair, etc.) associated with these switches without impact to the service. For active/passive deployments, PIM needs to be able to handle transitions seamlessly and also be able to alert the upstream systems of state changes.

- **Enclosure Manager Life Cycle** – Since modern blade enclosures have active components, it is common to have a 2N enclosure manager. The PIM needs the ability to monitor their health and manage their lifecycle and provide lifecycle actions without disruption.
Platform SDN Considerations

Network changes are initiated by the VIM to provide VNFC connectivity and establish service chains between VNFs as instructed by EEO and described in later Chapters. The VIM and the DC SDN controller communicate using common interfaces to implement the necessary networking changes. Specific Data Center SDN controller considerations are provided in the SDN discussion in Chapter 9.

- **Scope of Control – SDN controller vs. other NFV system elements:**

  The interface between End-to-End Orchestration, non-network controllers (like non-network OpenStack services) and the network controller is important for interoperability. There are efforts currently underway in ONF, Open Daylight, ONOS, OpenStack, IETF, and elsewhere to implement a common Northbound Interface (NBI) to support operator critical use cases.

  The primary purpose of the SDN controller is to (re)establish some clean layering between subsystems that exclusively control their respective resources. The DC SDN controller subsystem (VIM + SDN Controller) is responsible for controlling connectivity, QoS, path selection, selection of VNF network placement, inter-VNF connections, etc.

  In the ETSI definition of Forwarding Graphs (FG), the FG passed to the network controller only specifies the types/flavors of VNFs to use for specific subscriber traffic and the order in which to apply VNFs of various types/flavors. This allows the SDN controller to have latitude in deciding how best to satisfy the functional requirements of a given forwarding graph. In addition, this clean layering of decision making allows for network changes and faults that would be service impacting to be healed within the network control plane without requiring interaction and direction from the orchestration layer. The software systems outside of the SDN controller can’t possibly understand the state of network resources and their usage enough to make the best decisions about network proximity, network utilization, interface speeds, protocol limitations, etc. and thus are forbidden from specifying network specific instances and paths. The design for the Service Function Chaining (SFC) interface (also called SGi-LAN) must include allowing the SDN controller to choose specific instances of VNFs for assignment and to choose the network paths which are used to forward traffic along the chain of chosen instances. The term “intent-based networking” has recently been adopted to describe this concept (refer to Annex A).

- **Network Virtualization:**

  In addition to the subscriber oriented services delivered by an NFV solution, the NFV developer and operator is expected to have diverse multi-tenant (possible MVNO) virtual networking requirements that must be supported by a controller-based NetVirt implementation. Shared infrastructure ultimately needs logical separation of tenant networks and like cloud providers, NFV operators can be expected to require support for a scale-out NetVirt solution based on a single logical controller and large numbers of high-performance virtual networks with highly customized network behaviors.

- **Extensible SDN controller:**

  A multi-service DC SDN controller platform that can integrate and manage resource access among diverse, complimentary SDN services is critical to a successful NFVI. A NetVirt-only controller and NBI are insufficient, just as an SFC-only controller is insufficient. A controller architecture where multiple, independently developed services share access to forwarding tables by cooperating at the flow-rules level is chaotic and unmanageable. Therefore, an extensible
3.2 CPU and Chipset Aspects of the NFVI

3.2.1 Introduction

ETSI NFV defines the compute domain consists of a server, processor, chipset, peripherals, network interface controller (NIC), accelerator, storage, rack, and any associated components within the rack; including the physical aspects of a networking switch and all other physical components within NFVI. The below figure from the ETSI NFV Infrastructure Compute Domain document shown identifies these components.

In addition to standard compute domain architecture, the compute domain may include the use of accelerator, encryption and compression technologies for security, networking, and packet processing built into and/or around processors. As shown in the figure below, these infrastructure attributes are used by applications to optimize resource utilization and provide optimal application performance in a virtualized environment.

Section 3.3 highlights common chipset capabilities that should be considered to support the target SDN-NFV Architecture.
The typical two socket server architectures are designed with CPU along with a Peripheral Chipset that supports a variety of features and capabilities. The figures below provide a representation of the fundamental server architecture and the peripheral device support from an accompanying chipset.

**Figure 3-2: CPU Architecture**

Figure 3-2 provides a representation of the typical compute complex associated with a two-socket or dual processor server. In this case, the CPU provides both the integrated memory controller and integrated I/O capability. Also, a mechanism for inter-processor communication for shared cache and memory is provided via a high speed, highly reliable interconnect.
Figure 3-3 provides a representation of a typical Peripheral Chipset or Platform Controller Hub, sometimes referred to as the Southbridge. The features found in the PCH can vary from OEM to OEM depending on the design requirements. The example shown above is typical for current Xeon E5v3 generation-based servers.

For NFV, there is a need to ensure performance in high throughput data plane workloads by tightly coupling processing, memory, and I/O functions to achieve the required performance in a cost effective way. This can be achieved on standard server architectures as long as the orchestrator has detailed knowledge of the server hardware configuration, including the presence of plug-in hardware acceleration cards. To enable optimal resource utilization, information models are exposed that enable VNFs and orchestrators to request specific infrastructure capabilities.

### 3.2.2 CPU and Chipset Considerations for Security and Trust

SDN-NFV introduces new security challenges associated with virtualization that require new layers of security, attestation and domain isolation. In addition to runtime security, security layers include platform root-of-trust, interface security, application security and transport security. The items below highlight security related chipset capability considerations.

- **Security Encryption**
  - Chipsets support standard encryption and compression algorithms
  - Encryption of Public Key exchanges
  - Instruction set extensions for faster, more secure implementations

- **Trusted Platform / Trusted Boot Attestation** for platform and SW stack allowing:
  - Ability of the Orchestrator to demand “secure” processing resources from the VIM, such as use of enhanced platform awareness to select infrastructure that includes “Trusted Execution Technology” (TXT) features ensuring that VNF software images have not been altered
- Logging capability, so that any operations undertaken by the Orchestrator are HW recorded with details of what (or who) initiated the change, which enables traceability in the event of fraudulent or other malicious activity

3.2.3 CPU and Chipset considerations for Networking

NFV enables dynamic provisioning of network services as Virtual Network Functions (VNFs) on high-volume servers as virtual machines. The items below highlight infrastructure considerations related to network and interface requirements for Verizon’s SND-NFV architecture.

- **SDN and Network Overlays**
  - NVGRE, IPinGRE, VXLAN, MACinUDP
  - VXLAN-GPE (Generic Protocol Extension for VXLAN)
  - Future Consideration: NSH (Network Service Header)

- **Ethernet and Converged Network Adapters**
  - Connection speed: 10GbE / 40GbE / 100Gbe (future)
  - Cabling type: QSFP+ Direct Attached Twin Axial Cabling up to 10 m
  - Ports: Single and Dual Port
  - Supported Slot Height(s): Low Profile and Full Height
  - DPDK optimized: Within a DPDK runtime application the network adapter drivers are optimized for highest performance. Thus, an application utilizing DPDK will get the best overall performance
  - Optimized platform support for Flexible Filters (RSS, Flow Director)
    - Note: Certain NICs from Intel such as Fortville support both Flow Direction and RSS to the SR-IOV virtual function, other cards such as Intel’s Niantic may not support this functionality. As an example, Intel Fortville supports directing packets to a Virtual Function using the Flow Director, and then will perform RSS on all the queues associated with a given Virtual Function

- **Port Partitioning**
  - Multi-function PCIe
  - Scale for combination of physical functions and virtual functions

- **Converged Networking**
  - LAN, SAN (FCoE, iSCSI)
  - Remote SAN boot with data path intelligent offload

- **Standards Based Virtualization**
  - SR-IOV, EVB/802.1Qbg
  - VEB (Virtual Ethernet Bridge)
  - VTD, DDIO

- **Management and CPU Frequency Scaling**
  - Customer-defined hardware personalities
  - <10W NIC device power (2.4W typical for 1x 40G)
  - IPMI
3.2.4 CPU and Chipset considerations for Performance

The specific performance and memory capabilities of a chipset may vary based on processor and platform generation. The server topology and network architecture will ultimately determine the performance and memory necessary to support the required capacity.

Some examples of the plug-in card capabilities that enable optimal resource utilization for the specific workloads include:

- Compression Acceleration
- Media-specific Compute Instructions
- Transcoding Acceleration and Graphics Processing

In addition to add-in cards, the use of native chipset capabilities to optimize performance is critical to optimal resource utilization for NFV.

3.2.5 NFVI Capability Discovery

The ETSI-defined interface between the VIM and the infrastructure network domain (Nf-Vi) provides for the ability to discover the capabilities of the infrastructure.

EEO can use this information from the VIM to determine appropriate placement of VNFs based on correlation of the VNFD with the infrastructure capabilities. For example, the OpenStack Nova libvirt driver provides a mechanism for discovering CPU instruction set capabilities and sharing this with the Nova Scheduler.

Aligning the requirements of the VNF (based on the VNFD information model) with the infrastructure capability provides the means to achieve optimal VNF efficiency, and when combined with intelligent orchestration, provides for infrastructure efficiency.

Detailed NFVI Capability Discovery

This section describes in more detail how add-in cards and native chipset and CPU capabilities are communicated to the VIM. When there are particular examples, reference is provided to Linux/OpenStack model, but other Operating Systems and/or VIMs should behave similarly. The following description provides additional detail to the description of interfaces provided in Chapter 10.

VI-Ha

The Operating System (OS) of the host in the NFVI identifies the right device driver for the card or chipset by scanning the PCIe bus and matching the PCI IDs of the devices with the right driver. The OS together with the device driver advertise the capabilities of these devices to the virtualization layer via several standard Linux tools (like ip tool, ethtool, etc.) and/or via the pseudo file system in /proc or /sys. Also, the topology of all the devices on the PCIe bus (including Non-Uniform Memory Access topologies) and any virtualization technology are also shared via the same or similar means.

Nf-Vi

The VIM in the case of OpenStack has several services running in the OS of the host (Nova, Neutron, Ceph etc). These services, together with the virtual switch (if there is one) are consuming and processing the information provided by the hardware resources via the Vi-Ha in order to take advantage of available capabilities. As an example, Nova (the compute agent of OpenStack) will identify the PCIe and NUMA
layout of the host and will use it in order to efficiently spawn the VNF(s) when requested by the VIM to do so.

It might be worth noting that some of the hardware resource capabilities do not need to be advertised further to the VIM via its services, as they can be directly used by the services layer, thus saving the VIM or any other of the upper layers from having to make low level decisions. As an example, the stateless offloads for in-the-NIC acceleration of inner and outer VXLAN checksums will be used without intervention from the VIM if they are enabled directly from the NIC driver (default behavior).

Vn-Nf

The main Vn-Nf implementation in OpenStack and Linux deployments is through libvirt. This library provides an abstraction of the physical hardware that allows policy implementation, while through V1-Ha interface; it will identify and enable any special capabilities that lie underneath it in the hardware resource layer. Libvirt also provides APIs to enumerate and monitor resources such as CPUs, memory, storage, networking etc.

3.2.6 NFVI related capabilities and technologies

Below are some common infrastructure attributes that should be considered to support the SDN-NFV Architecture. As mentioned before, some of them will need to be added to the VNFD so the VIM can identify the right placement of the VNFs:

NFVI Capability Attributes exposed to the VIM

- Non-Uniform Memory Access (NUMA) CPU & Memory configuration (co-located memory and socket)
- NUMA I/O Device Locality configuration (co-located PCI device and socket) - Openstack will locate VM and PCI device on the same NUMA node, but only for VMs with explicitly defined NUMA topology
- CPU Pinning – Openstack does CPU pinning if requested
- Encryption and Compression acceleration (Quick Assist Technology)
- Trusted Platform / Trusted Boot
- AES-NI, AVX, SSE4.2, RD RAND (Instruction Set Extensions) – ComputeCapabilitiesFilter allows requests to land a VM on a host that has specific CPU feature
- vSwitches (type, capability) - OVS specified, with or without either DPDK/HWOA with some DPDK awareness, not at scheduler level. Selecting vhost-user ports on supporting platforms happens behind the scenes with no HWOA awareness
- Memory (size, page sizes, NUMA allocation policy) - Openstack is aware of the amount of memory and page sizes present in a platform, NUMA allocation policy is supported
- Huge Page Support (2MB/1GB)
- I/O Pass-through (Full PCIe pass-through of the I/O device to the guest – Requires bootstrapping of the node in VNFI)
- I/O Pass-through (Virtual Function (SR-IOV) pass-through of the I/O device to the guest – Requires bootstrapping of the node in VNFI)
NFVI Capability Attributes not needed to be exposed to the VIM

- CPU DDIO (Direct Data I/O)
- Network cards (interface capabilities such as LSO, LRO etc.; interface BW, DPDK support)

NFVI Capability Attributes that are under development to be exposed in the VIM

- In hardware geolocation (based on Trusted Platform)
- Standard HW Acceleration API (e.g. DPDK-AE)
- LLC utilization (CTM)
- Cache Allocation (CAT)
- CPU SMT

Cache Allocation Technology (Future)

Processors that use memory or I/O on the same die may cause cache thrashing. One method to optimize this footprint is to use Cache Allocation Technology (CAT). The fundamental goal of Cache Allocation Technology is to enable resource allocation based on application priority or Class of Service (COS or CLOS). The processor exposes a set of Classes of Service into which applications (or individual threads) can be assigned. Cache allocation for the respective applications or threads is then restricted based on the class with which they are associated. Each Class of Service can be configured using bitmasks, which represent capacity and indicate the degree of overlap and isolation between classes.

This technology is not directly exposed to the VIM today, apart from the fact that it can be identified if present by the CPU type. The nodes in the VNFI, however, can be configured during the bootstrap process to enable it by default for their packet processing related processes (such as the vSwitch – with or without DPDK) and provide dedicated cache, preventing thrashing and allowing better determinism and in some cases better performance.

Cache Monitoring Technology (CMT) & Memory Bandwidth Monitoring (MBM)

Cache Monitoring Technology (CMT) allows an Operating System, Hypervisor or similar system management agent to determine the usage of cache by applications running on the platform. The initial implementation is directed at L3 cache monitoring (currently the last level cache in most server platforms).

Memory Bandwidth Monitoring (MBM) builds on the CMT infrastructure to allow monitoring of bandwidth from one level of the cache hierarchy to the next - in this case focusing on the L3 cache, which is typically backed directly by system memory. As a result of this implementation, memory bandwidth can be monitored.

This is upcoming technology and therefore it is not exposed to the VIM yet.

Performance Monitoring, and Performance Monitoring Events Capabilities

NFV deployments should consider event and performance monitoring with a set of model-specific performance-monitoring counters. These counters permit selection of processor performance parameters to be monitored and measured. The information obtained from these counters can be used for tuning system and compiler performance. Each micro-architecture has its performance monitoring events that can be monitored.
All the above performance monitoring technologies are not currently exposed to the VIM in the case of OpenStack, but they are available to the performance tools of Linux. Making these technologies available to the VIM layer or above would require development efforts today.

**PCle Bandwidth**

Saturation of the PCle bus can affect I/O, memory transactions, and ultimately CPU related functions. Therefore, the PCle bandwidth (platform dependent) should be considered for the NFVI as a potential impact on CPU.

**Reliability, Availability, Serviceability (RAS)**

Server reliability, availability, and serviceability (RAS) are crucial issues to deliver mission-critical applications. Any system downtimes that impact resource pools and/or the ability to deliver services in a most efficient and optimal manner are extremely costly. Furthermore, the likelihood of such failures increases statistically with the size of the servers, data, and memory required for these deployments.

RAS Features Include: Error detection, correction and containment, recovery in all processors, memory, and I/O data paths. For additional details:


**DPDK – Data Plane Development Kit**

DPDK is an optimized library for high speed network processing. The library is compatible with all Intel processors, and other processor architectures. The DPDK library is a single software-programming model that can accommodate vastly different networking operations and system configurations. Intel and other third party companies contribute to the project via the http://dpdk.org website. DPDK is a set of libraries that can be used in the Linux or BSD user spaces. It offers low processing overhead alternatives to operating system calls. This enables the creation of user space applications that scale beyond what native OS applications can offer.
Libraries for network application development on Intel Platforms speeds up networking functions, enables user space application development and facilitates both run-to-completion and pipeline models. DPDK is Free, Open-sourced, Community driven BSD Licensed (www.intel.com/go/dpdk and Git: http://dpdk.org/git/dpdk).

- **Ecosystem Support**

  Multiple CPU architectures supported (on dpdk.org) including Intel x86_64, ia32, targeting Intel® Xeon® to Intel® Atom™, Power 7/8 (Added by DPDK community members) Tilera TileGx (Added by DPDK community members). Multiple vendor NICs supported open source including Intel, Mellanox, and Cisco (Added by DPDK community members). The latest OS Support includes the following: Fedora release 20, Ubuntu 12.04 LTS, Wind River Linux 5, Red Hat Enterprise Linux 6.3, SUSE Enterprise Linux 11 SP2, FreeBSD 9.2, CentOS 6.4.

  Programs using DPDK include Virtio (http://www.linux-kvm.org/page/Virtio) and OVS (http://www.openvswitch.org/)

- **Speeds up networking functions**

  DPDK speeds up networking functions by offering an optimized memory library, poll mode network interface drivers, low lock cost queueing and optimized libraries. Memory routines are optimized to take advantage of operating system huge pages and allocate memory in large chunks of contiguous memory. Contiguous memory reduces the allocation and deallocation costs. Poll mode drivers for Intel and other party NICs are provided to offer the optimum packet movement to and from a network.
interface card. Polling the interface eliminates the large cost of servicing an interrupt on the platform. Low cost locking queues for inter-process communication are provided. These queues utilize atomic operations, which are less costly than a traditional system spinning lock. These core features are coupled with optimized libraries for packet search (ACL, Exact Match, LPM, HASH), packet scheduling (SCHED, Meter), a pipeline application development library in Packet Processing Framework, and an optimized packet load balancer in DISTRIB.

- **Enables user space application development**

User space application development is advantageous for two key reasons. First DPDK is a BSD based library, and this is not subject the GPL licensing requirements. This enables users to develop proprietary applications without having to be concerned with intellectual property rights. Second development in the user space is much easier, allowing developers to take advantage of standard compiler and debugging utilities.

- **Facilitates both run-to-completion and pipeline models**

An application developer can develop according to either a run-to-completion or a pipeline packet-processing model. DPDK offers a very efficient threading mechanism that in its simplest terms is a looping operation. This allows for a run-to-completion in that all operations that would be performed upon network traffic would occur on that single looping thread. Many run to completion threads can be created, each processing different ports in the system. DPDK offers a simplified framework for developing pipelined applications by using either the packet-processing framework by itself, or in conjunction with the DISTRIB library. In certain cases a pipeline application model will scale beyond a run-to-completion model. The pipeline model allows for finer grained application dissemination, and utilization of queuing models to balance the complexity of operations on network traffic amongst the available cores. The pipeline model enables more complex processing operations upon packets.
4 VNF Manager

Role & Overview of VNF Manager

VNF Manager (VNFM) is an entity that is part of the NFV architectural framework defined by ETSI NFV that interacts with the VIM (Virtualized Infrastructure Manager), the NFVO (NFV Orchestrator), the VNFs (Virtualized Network Functions), and the Service Assurance (SA) as shown in the figure below:

The VNF Manager is responsible for the lifecycle management of VNF instances. Each VNF instance is associated with the VNF Manager. The VNF manager is assigned the management of all VNF instances. Most of the VNF Manager functions are assumed to be generic common functions applicable to any type of VNF. However, the NFV-MANO architectural framework needs to also support cases where VNF instances need specific functionality for their lifecycle management, and such functionality may be specified in the VNF Package.

The following list expresses the non-exhaustive set of functions performed by the VNF Manager function. These functionalities may be exposed by means of interfaces and consumed by other NFV-MANO functional blocks or by authorized external entities. Operations include:

- VNF instantiation, including VNF configuration if required by the VNF deployment template (e.g., VNF initial configuration with IP addresses before completion of the VNF instantiation operation)
- VNF instantiation feasibility checking, if required
- VNF instance software update/upgrade
- VNF instance modification
- VNF instance scaling out/in and up/down
- VNF instance-related collection of NFVI performance measurement results and faults/events information
- Correlation to VNF instance-related events/faults
- VNF instance assisted or automated healing
- VNF instance termination
- VNF lifecycle management change notification
- Management of the integrity of the VNF instance through its lifecycle
- Overall coordination and adaptation role for configuration and event reporting between the VIM and the EM

The deployment and operational behavior of each VNF is captured in a template called Virtualized Network Function Descriptor (VNFD) that is stored in the VNF catalog. NFV-MANO uses a VNFD to create instances of the VNF it represents, and to manage the lifecycle of those instances. A VNFD has a one-to-one correspondence with a VNF Package, and it fully describes the attributes and requirements necessary to realize such a VNF. NFVI resources are assigned to a VNF based on the requirements captured in the VNFD (containing resource allocation criteria, among others), but also taking into consideration specific requirements, constraints, and policies that have been pre-provisioned or are accompanying the request for instantiation and may override certain requirements in the VNFD (e.g. operator policies, geo-location placement, affinity/anti-affinity rules, local regulations).

The information elements to be handled by the NFV-MANO, including the VNFD, need to guarantee the flexible deployment and portability of VNF instances on multi-vendor and diverse NFVI environments, e.g. with diverse computing resource generations, diverse virtual network technologies, etc. To achieve this, hardware resources need to be properly abstracted and VNF requirements must be described in terms of such abstractions.

The VNF Manager has access to a repository of available VNF Packages and different versions of them, all represented via their associated VNFDs. Different versions of a VNF Package may correspond to different implementations of the same function, different versions to run in different execution environments (e.g. on hypervisors or containers, dependent on NFVI resources availability information, etc.), or different release versions of the same software. The repository may be maintained by the NFVO or another external entity.

The VNFM also has the overall coordination and adaptation role between the VIM and the SA. The VNFM exposes interfaces for lifecycle management and infrastructure resource monitoring towards the VNF. The VNF may in turn expose application management interfaces (FM, PM, CM) related specifically to the VNF lifecycle management operations.

The target architecture will use a Generic VNF Manager interacting with multiple VNFs as shown in Figure 4-1. However, for initial deployments an interim solution may be to use Specific-VNFMs (S-VNFMs) for some categories of advanced VNFs, like EPC and IMS. The S-VNFMs may be part of the VNF or EMS, meaning the Ve-Vnfm interface may not be exposed. There will be no change to the Or-Vnfm, Vnfm-Sa and Vi-Vnfm interfaces. A S-VNFM can support more complex lifecycle management cases where application level aspects can be more tightly integrated with the VNFM lifecycle operations. Considering an S-VNFM can cater for multiple of a vendor’s VNF’s, this could simplify the operational aspects and limit the testing to a single endpoint per S-VNFM.

A generic VNFM can serve multiple VNFs of different types and/or originating from different providers. The operation of generic VNFM has no dependency on the VNFs it manages and it should be able to accommodate the VNF-specific scripts defined in the VNF package. Plug-ins may be used to accommodate vendor variations.
• Standard interfaces are exposed by the VIM and the NFVO at the Vi-Vnfm and Or-Vnfm reference points (respectively).
• Standard interfaces are exposed by the VNF at the Ve-Vnfm.
• The VNFM provides VNF lifecycle management procedures for each VNF it manages.
• The Generic VNFM is provided by VZ and does operations such as:
  - Instantiate VNF
  - Configure VNF
  - Start, Pause, Resume, Stop, Terminate
  - Get State
  - Scale-In / Scale-Out

This Generic VNF Manager supports OpenStack and the following interfaces are specified:
• Vi-Vnfm
• Or-Vnfm
• I/F with VNF Catalog
• Ve-Vnfm
• Vnf-Sa

Deployment is performed with a number of steps:
• Definition of VNF Descriptors
• Definition of script templates for VNF lifecycle operations
• Develop VNF & VNFC images
• Pre-validation Test
• Deploy
• Run time test

**VIM Resource allocation Alternatives**

The target architecture can use the following two resource allocation strategies:

**VNFM-driven** - The NFVO shares the resource allocation responsibility with the VNFM through a reservation and granting mechanism where traffic and network conditions drive scaling. This approach offers a VNF provider a way to handle the virtual resource allocation in isolation with a specialized VNF strategy, which can be essential for mission critical VNFs.

**NFVO-driven** – The NFVO is fully responsible for the VIM resource allocation for usecases where service orchestration drives VNF instantiation or scaling. This approach typically guarantees more optimal resource utilization, avoiding pre-allocation and potential under-utilization of resources. Resource reservation can be avoided as it could be implicitly assumed by the resource requested by the VNFM, thus leading to simplified workflows. This approach offers simpler access to all globally available resources due to logically centralized resource management. NFVO also offers a logical placement for global resource management such as consolidated IP address management over multiple VIMs. Basically the NFVO provides an abstraction of VIM services irrespective of VIM type, version, and geographic location. In a multivendor, multi-technology platform, NFVI environment, NFVO will act as adaptation-isolation layer towards the VIMs, reducing the number of integrations and interfaces.

Supporting both allocation strategies provides the flexibility to address different levels of federation and autonomy for the VIM and NFVI execution environments.
Figure 4-2 below shows VNFM driven interactions while the Figure 4-3 shows NFVO driven interactions:

Open Source VNF Manager

A number of Open source initiative have been launched lately, primarily with Telefonica putting its software prototype OpenMANO in github (https://github.com/nfvlabs/openmano), and with the Tacker (https://wiki.openstack.org/wiki/Tacker) initiative hosted on the OpenStack platform.

Open source initiatives (especially OpenStack Tacker) are promising since they have the potential of federating inputs from multiple organizations, including operators, VNF vendors and integrators. These are still in the early stages and will take time before they become viable to be used in production.
5 VNF Descriptors

In existing physical network functions, the relationship between internal PNF software components is performed on a backplane, which is typically hidden from the operator and managed by the equipment vendor. In the NFVI, inter-component communication is exposed and supported by the NFVI; therefore the intra-VNF virtual links must be defined as part of the VNFD to ensure VNFs operate properly.

A VNF Descriptor (VNFD) is a deployment template which describes a VNF in terms of deployment and operational behaviour requirements. The VNFD also contains connectivity, interface and KPIs requirements that can be used by NFV-MANO functional blocks to establish appropriate Virtual Links within the NFVI between VNFC instances, or between a VNF instance and the endpoint interface to other Network Functions. VNFDs are used by the VNFM to perform lifecycle management operations on the VIM and VNFs on the Vi-Vnfm and Ve-Vnfm interfaces, respectively, shown in Figure 5-1 below.

Figure 5-1: Ve-Vnfm and Vi-Vnfm Interfaces.

In addition to data models, descriptor files are important to the end-to-end architecture. Data models describe how to manage a function or service in terms of provisioning and monitoring. Descriptor files describe how to build, scale, heal, and upgrade a VNF and/or Network Service. Descriptor files are created by the network service architect or VNF designer. Descriptor files only capture the information required at each level of the orchestration process. For example, the NSD associated with the SGi-LAN identifies that a firewall needs to be instantiated, but it does not provide details about the internals of that firewall. The VNFD associated with the firewall captures the internal architecture of the VNFCs within the VNF.

The VNFM uses the VNFD to perform lifecycle management functions (e.g. instantiation, upgrade, scaling, healing, termination) using the VNFD template.
The **Vi-Vnf** interface is used to define the required platform and network characteristics required for each VNFC. The VNFM lifecycle management functions generate specific API calls to the VIM using information elements present in the VNFD so that the VIM may assign optimized and compatible compute/network/storage resources to the VNFC, as well as establish virtual links between the VNFCs.

The **Ve-Vnfm** interface is used to perform additional configuration and post-processing tasks on the VNFCs. The VNFM must execute additional post-processing VNFC configuration scripts in the VNFD. Using the output from previously performed lifecycle management functions, such as dynamic IP address assignment of VNFC network interfaces by the VIM, the VNFM performs post-processing application-layer associations of the VNFCs.

Once the VNFC is activated and configured, additional procedures are likely required to complete standard VNF lifecycle events. Such workflows are complex, are implementation specific, and require VNFC application-level configuration. These procedures may extend to additional elements beyond the VNFC andNFVI. VNFM s may perform these procedures differently depending on the NFVI and VIM functionality. Examples include:

- Application-layer load sharing & load balancing configuration
- Redundancy mode configuration
- External logical interface provision (e.g. configuration of a Diameter peer on a EPC or IMS element or addition of a DNS entry)
- Networking changes (physical link activation or route peering configuration) external to the VNFC

These additional operations are often performed using scripts or plugins with variable arguments and inputs on the Ve-Vnfm interface to apply the necessary configuration changes.

**Platform VNF Descriptors**

A minimum representative subset of VNFD elements considered necessary to on-board the VNF have been identified by ETSI NFV. The following set of informational elements is the minimal set of common platform parameters that are defined on a per VNFC basis. These elements allow the EEO, VNFM, and VIM to identify and assign the required resources to each VNFC so that the VNFC may run in its optimal environment. Once the VNFC is instantiated, additional post-processing scripts are applied; informational elements for these additional sections are summarized in the SGI-LAN, IMS and EPC specific sections of this chapter.

High-level platform VNFD informational element categories include:

- CPU types
- Memory requirements
- Storage
- NIC
- Security
- Policy
- Image information
- Licensing information
- Affinity/anti-affinity rules
- Guest OS/Hypervisor

The format of the VNF Descriptor (VNFD) is implementation specific and varies depending on the VIM, VNFM, and NFVO selected. Industry lifecycle management functions typically use an XML-based VNFD.
The Table below outlines detailed platform-specific VNFD informational elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNF</td>
<td>VNF Name &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>VNF Provider &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>VNF Description &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>VNF Version &lt;Value&gt;</td>
</tr>
<tr>
<td>VNF-C</td>
<td>(one of more per VNF)</td>
</tr>
<tr>
<td></td>
<td>VNFC Name &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>VNFC Provider &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>VNFC Description &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>Sequence Number For VNFC Deployment &lt;Value&gt;</td>
</tr>
<tr>
<td></td>
<td>VNFC Image Name &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>VNFC Version &lt;Value&gt;</td>
</tr>
<tr>
<td></td>
<td>Package Formatting OVF</td>
</tr>
<tr>
<td></td>
<td>Image Format Qcow2</td>
</tr>
<tr>
<td></td>
<td>QCOW2 Conversion Required Yes</td>
</tr>
<tr>
<td></td>
<td>VNF URL &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>Generate a Security Key Yes</td>
</tr>
<tr>
<td></td>
<td>Internet Access Required Yes</td>
</tr>
<tr>
<td></td>
<td>Floating IP Required Yes</td>
</tr>
<tr>
<td></td>
<td>Live Migration Supported Yes</td>
</tr>
<tr>
<td></td>
<td>Object Storage Required Yes</td>
</tr>
<tr>
<td></td>
<td>Boot Volume From Cinder</td>
</tr>
<tr>
<td></td>
<td>PostProcessing Required Yes</td>
</tr>
<tr>
<td></td>
<td>Cloud Enabled Yes</td>
</tr>
<tr>
<td>Guest OS</td>
<td>(one per VNF-C)</td>
</tr>
<tr>
<td></td>
<td>OS Vendor Ubuntu</td>
</tr>
<tr>
<td></td>
<td>OS Version &lt;Value&gt;</td>
</tr>
<tr>
<td></td>
<td>OS Architecture 32-bit</td>
</tr>
<tr>
<td>Hypervisor</td>
<td>(one per VNF-C)</td>
</tr>
<tr>
<td></td>
<td>Hypervisor Name KVM</td>
</tr>
<tr>
<td>Affinity/Anti-Affinity</td>
<td>(one per VNF-C)</td>
</tr>
<tr>
<td></td>
<td>Datacenter Affinity</td>
</tr>
<tr>
<td></td>
<td>Rack Affinity</td>
</tr>
<tr>
<td></td>
<td>Node Affinity</td>
</tr>
<tr>
<td></td>
<td>Network Affinity</td>
</tr>
<tr>
<td>License</td>
<td>(one per VNF-C)</td>
</tr>
<tr>
<td></td>
<td>License Required Yes</td>
</tr>
<tr>
<td></td>
<td>Perform Autoscaling Yes</td>
</tr>
<tr>
<td></td>
<td>License URL &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>License automation method REST</td>
</tr>
<tr>
<td></td>
<td>Additional URL Parameters &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>Contact Person Name &lt;String&gt;</td>
</tr>
<tr>
<td></td>
<td>Contact Person EmailId &lt;String&gt;</td>
</tr>
<tr>
<td>CPU</td>
<td>(One per VNF-C)</td>
</tr>
<tr>
<td></td>
<td>Number of CPUs &lt;Value&gt;</td>
</tr>
<tr>
<td></td>
<td>vCPU-pCPU Pinning Method Don't Care</td>
</tr>
<tr>
<td>Memory</td>
<td>(One per VNF-C)</td>
</tr>
<tr>
<td></td>
<td>RAM Required (MB) &lt;Value&gt;</td>
</tr>
<tr>
<td></td>
<td>HugePages Required Yes</td>
</tr>
<tr>
<td></td>
<td>Number of HugePages &lt;Value&gt;</td>
</tr>
<tr>
<td></td>
<td>HugePage Size &lt;Value&gt;</td>
</tr>
</tbody>
</table>

Table 5-2: VNFD Informational Elements (Part 1)
### Table 5-3: VNFD Informational Elements (Part 2)

In addition to the above platform-specific VNFD informational elements, there will be vendor-specific VNFD informational elements and operator specific VNFD informational elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>(one or more)</td>
</tr>
<tr>
<td>Block Storage</td>
<td></td>
</tr>
<tr>
<td>Bootable Volume</td>
<td>Yes</td>
</tr>
<tr>
<td>Storage Size</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>Minimum IOPS</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>RAID Level</td>
<td>Don’t Care</td>
</tr>
<tr>
<td>Snapshots Required</td>
<td>Yes</td>
</tr>
<tr>
<td>Snapshot Frequency</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>Snapshot Unit</td>
<td>Don’t Care</td>
</tr>
<tr>
<td>Encryption Support Required</td>
<td>Don’t Care</td>
</tr>
<tr>
<td>RDMA Support Required</td>
<td>Don’t Care</td>
</tr>
<tr>
<td>Object Storage</td>
<td>Yes</td>
</tr>
<tr>
<td>NIC</td>
<td>(one or more)</td>
</tr>
<tr>
<td>SRIOV Required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Supported Vendors</td>
<td>Intel</td>
</tr>
<tr>
<td>DPDK Required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Network Range for deployment</td>
<td>tenant</td>
</tr>
<tr>
<td>Acceptable limit for Jitter (ms)</td>
<td>Don’t Care</td>
</tr>
<tr>
<td>Acceptable limit for latency (ms)</td>
<td>Don’t Care</td>
</tr>
<tr>
<td>DPDK Required?</td>
<td>Yes</td>
</tr>
<tr>
<td>DPDK enabled vNIC Drivers included in VM</td>
<td>Yes</td>
</tr>
<tr>
<td>DPDK Driver Type</td>
<td>&lt;String&gt;</td>
</tr>
<tr>
<td>DPDK Driver Version</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>Can these be added at boot time</td>
<td>Yes</td>
</tr>
<tr>
<td>DPDK Poll Mode Driver (PMD) included in VM</td>
<td>Yes</td>
</tr>
<tr>
<td>DPDK PMD Driver Type</td>
<td>&lt;String&gt;</td>
</tr>
<tr>
<td>DPDK PMD Driver Version</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>Supported/Validated NICs</td>
<td>Yes</td>
</tr>
<tr>
<td>Network Type</td>
<td>VLAN</td>
</tr>
<tr>
<td>Inbound Bandwidth</td>
<td></td>
</tr>
<tr>
<td>Minimum Bandwidth (Mbps)</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>Maximum Bandwidth (Mbps)</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>Outbound Bandwidth</td>
<td></td>
</tr>
<tr>
<td>Minimum Bandwidth (Mbps)</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>Maximum Bandwidth (Mbps)</td>
<td>&lt;Value&gt;</td>
</tr>
<tr>
<td>Security</td>
<td>(once per VNF-C)</td>
</tr>
<tr>
<td>Firewall Required</td>
<td>Yes</td>
</tr>
<tr>
<td>Intel TXT secure Boot Required</td>
<td>Yes</td>
</tr>
<tr>
<td>Enable SSL Certificate</td>
<td>Yes</td>
</tr>
<tr>
<td>SSL File Location</td>
<td>&lt;String&gt;</td>
</tr>
<tr>
<td>Policy</td>
<td>(Once per VNF-C)</td>
</tr>
<tr>
<td>Auto Scaling</td>
<td>Yes</td>
</tr>
<tr>
<td>Load Balancing method</td>
<td>LBaaS</td>
</tr>
<tr>
<td>Scaling trigger</td>
<td>CPU</td>
</tr>
<tr>
<td>External LB supported?</td>
<td>Yes</td>
</tr>
<tr>
<td>Pass metadata to VNFC’s</td>
<td>Yes</td>
</tr>
<tr>
<td>OpenStack Metadata</td>
<td>&lt;String&gt;</td>
</tr>
</tbody>
</table>

Copyright © 2016 Verizon. All rights reserved
L1-L3 VNFD Considerations

VNF Descriptors (VNFDs) contain virtual link descriptions. Internal virtual links connect VNFCs within a VNF. QoS and throughput are defined on each virtual link. Virtual links include three connectivity types:

- **E-Line** - For a simple point to point connection between a VNFC and the existing network
- **E-LAN** - When a VNFC instance needs to exchange information with all other VNFC instances within a Network Service to ensure synchronization
- **E-Tree** - When traffic from a trunk interface needs to be routed to a specific branch, such as in a load balancing application or management system connections

The NFVI virtual networking components should provide intra-VNF connectivity between VNFCs within the NFVI for the three connectivity types listed above while also supplying the required throughput and QoS to the virtual link.

Virtual links are also used in NSDs between VNFs as part of a VNFFG.

SGi-LAN VNFD Considerations

Below are two possible ways in which the SGI-LAN architecture can be deployed:

- The operator can elect to build their SGI-LAN from discrete components, using the Orchestration to automate the coordination of the various Service Functions, each of which is delivered as a separate VNF.
- The operator may choose to deploy a SGI-LAN platform which is delivered as a single composite VNF (in the sense of a VNF made-up of multiple VNF components or VNFCs).

In the former approach, the operator may decide it has more flexibility to architect its data center. In the latter, the composite VNF achieves information hiding to the Orchestration. This means that specific capabilities associated to the domain requirements can be included as features of the composite VNF. A good example is the HA feature which in the first approach is delivered as an orchestration solution attribute whereas in the second approach it is actually a feature self-contained in the composite VNF.

Additional VNFD Informational Elements required for SGI-LAN VNFCs

The SGI-LAN VNF category requires the following informational elements in addition to the platform elements:

- Associated PGWs (VNFs or PNFs) with a particular set of SGI-LAN interfaces to determine the first hop of the SGI-LAN service
- Associated routers to determine the last hop of the SGI-LAN service
- 3GPP Logical interface association for Sd, St, Gzn/Rf, and Gy
- OpenFlow rule and throughput capacity of the network fabric and individual network fabric endpoints (such as the vswitch and ToR)
- Traffic distribution configuration

Many of these SGI-LAN information elements apply only when SGI-LAN services are all part of a single VNF where each VNFC is a micro service versus when SGI-LAN is implemented as a network service using forwarding graphs between multiple VNF microservices.

IMS & EPC VNFD Considerations

IMS and EPC VNFDs include information about the following:
- The VMs that belong to the VNF
- The name of the Images to be deployed for the VNF
- The start-order for the VMs
- The affinity rules, e.g. ensuring that the CSCF Server VM that belongs to a cluster runs on different blades
- The vNICs and networks used by the VMs of the VNF
- The virtual CPU, memory and virtual disk information to be allocated for each VM
- Management and Configuration
6 End-to-End Orchestration

Introduction and definitions

End-to-End Orchestration (EEO) is responsible for lifecycle management of end-to-end network services.

The objective of EEO is to realize zero-touch provisioning: a service instantiation request -- from the operations crew or from the customer, through a self-service portal -- results in an automatically executed workflow that triggers VNF instantiation, connectivity establishment and service activation.

TMForum distinguishes between Service Management and Resource Management and in that context uses the terms customer-facing services and resource-facing services. Service Management and customer-facing services are related to the service as perceived by the end-user. The customer-facing service aspects include order entry, billing, trouble ticketing, self-service portals, helpdesk support, etc. The resource-facing service consists of the network functions/resources required to deliver the customer-facing service. Resource Management includes bandwidth allocation, QoS management, protocol handling, etc. Using the TMForum definitions, EEO is responsible for resource management. All customer-facing service aspects are handled by other OSS and BSS systems. The network services alluded to in the first paragraph are what TMForum calls resource-facing services.

The NFV Management and Orchestration (MANO) architecture defined by ETSI NFV identifies the NFV Orchestrator (NFVO). In its definition, NFVO is responsible for lifecycle management of “network services”. In the ETSI definition, a “network service” consists of a group of VNFs, the connections between them and the identification of Physical Network Functions (PNFs) with which they interface.

EEO as defined in this architecture document includes the NFVO functionality but supports additional functions as well. Specifically:

- EEO manages end-to-end network services, including connectivity through the network infrastructure outside the data center, which typically consists of PNFs. NFVO does not cover network segments that consist of PNFs.
- EEO allocates resources, triggers instantiation of VNFs and configures network functions in order to activate the end-to-end service. NFVO covers instantiation of VNFs but does not cover activation.

To understand the importance of activation as a necessary part of service management automation, consider the following example. When a virtualized PGW has been instantiated, its address needs to be configured in the HSS, because the HSS identifies the SGW and PGW used for a UE at attachment. Without configuring the address in the HSS, the PGW will never receive any user traffic. Since EEO is the only element that oversees all network resources required for a service, it is the task of EEO to execute certain configuration actions required to activate a service.

Functionality and interfaces

EEO executes service-related requests from northbound systems, such as OSS systems or other components that are responsible for customer-facing service orchestration. It uses End-to-End Network Service Descriptor (EENSD) information to determine the required actions. This includes workflow aspects, since actions need to be executed in a certain order. For example, a VNF needs to be instantiated before it can be activated.
EEO allocates resources. To do so, it needs a high-level understanding of the network topology and an inventory of available resources. For example, it may need to decide which of several datacenters should be used to instantiate certain network functions.

Finally, after instantiating VNFs, allocating PNFs and establishing the required connectivity, EEO needs to perform configuration actions to activate the end-to-end network service. In summary, the key functions performed by EEO are:

- Resource allocation and orchestration
- Workflow execution
- Topology and Inventory Management
- Configuration and activation

The actions and decisions of EEO are largely determined by external triggers and information contained in End-to-End Network Service Descriptor files, but in many cases EEO still needs to select one of many options. Hence, a final aspect of EEO is the fact that it is policy-driven, which enables the operator to tailor EEO’s decisions according to policy rules.

Figure 6-1 is a distillation of the overall architecture depicted Figure 2-2. EEO interfaces with the following functional blocks:

- **OSS/BSS** - The interface enables OSS/BSS systems to trigger lifecycle management actions related to network services
- **End-to-End Network Service Descriptor (EENS)D** - Contains workflow templates, connectivity models, provisioning scripts, etc. related to specific end-to-end network services. The interface enables EEO to retrieve this information
- **VNF Manager (VNFM)** - Executes VNF lifecycle management actions. The interface enables EEO to trigger these actions, e.g. it can request instantiation of a new VNF
- **Virtualized Infrastructure Manager (VIM)** - Requests from the VNFM to the VIM may be relayed through EEO so that it can make resource allocation decisions
- **SDN Controllers** - The interface enables EEO to request establishment of connectivity services. For example, EEO can request connectivity between an enterprise location and VNFs that are hosted in a data center.

- **VNFs and PNFs** - To activate services, EEO may need to configure parameters on VNFs and PNFs.

- **EMS** - If PNFs or VNFs do not support standard data models and configuration protocols, EEO may need to interface with the EMS, which in turn configures the network function.

- **Service Assurance** - The interface enables EEO to provide service related data to the Service Assurance function. In return, the Service Assurance function may trigger EEO in case it detects that a network service has failed or is not performing according to the agreed-upon SLAs.

Note that the interfaces between EEO-PNF and EEO-VNF are labeled the same. The reason is that from a functional perspective, there is no difference between a VNF and a PNF. Their implementations differ and virtualization allows for automated lifecycle management, but once a VNF has been instantiated, it behaves in the same way as its non-virtualized counterpart. Hence, service provisioning is identical on both the virtualized and non-virtualized forms of a particular network function.

### Abstraction and delegation

EEO is responsible for topology and resource management. It is important to note that EEO is not an omnipotent and omniscient machine that has full visibility into every detail of the end-to-end network. Instead, the overall management and control architecture relies on a division of functionality, where lower-level systems such as the VIM or domain-specific SDN Controllers maintain detailed information and present abstracted information to higher-level systems.

EEO has a high-level, abstracted view of the network topology and network inventory. This information enables it to make allocation decisions. These high-level decisions are translated by SDN Controllers and VIM to more detailed allocation decisions. For example, EEO can decide to instantiate a VNF in a particular section of a data center. The VIM that is responsible for that section subsequently decides which VMs need to be instantiated on which servers. Similarly, EEO can decide to establish an Ethernet connection between an enterprise site and a pair of PE routers. The WAN SDN Controller subsequently decides on specific aspects such as the encapsulation technology used (e.g. MPLS, GRE) and the optical connections over which to route the Ethernet connection.

For troubleshooting purposes it may be necessary to drill down into the lowest levels of topology and inventory. This process starts at the Orchestration level, but EEO subsequently requests more detailed information from the lower-level systems that manage the resources in question.

### OSS/BSS Evolution Considerations

The introduction of EEO does not imply a wholesale replacement of existing OSS and BSS systems. Initially, the operator will continue to manage most of its services using existing systems and procedures. It may decide to introduce EEO for specific new end-to-end services that rely heavily on virtualized network functions. Over time, one can expect that more and more services will be managed through EEO.

### Example call flow

This section illustrates the roles and responsibilities of the entities identified in the management and control solution with respect to the activation of a virtual CPE service for an enterprise customer.
The figure below shows the management and control systems superimposed on a depiction of the network resources required to provide the Enterprise vCPE service.

Assumptions:

- The customer has an IP VPN service that connects its enterprise sites. This VPN needs to be extended to the data center where the vCPE service functions are hosted.
- The service chain consists of virtualized network functions: a firewall (FW), and Intrusion Detection System (IDS) and a Network Address Translation (NAT) device.
- The Firewall VNF consists of multiple Virtual Machines (VMs).
- All VNFs are located in the same data center.

The following step-wise approach illustrates the roles and responsibilities of the control and management entities. Note that the order of steps may differ since various actions could happen in parallel. Also note that this example is over-simplified to serve its illustrative purpose.

1. The customer signs up via a Web portal. Via the portal, the customer can choose from services that are listed in the services catalog. Once it has selected a vCPE service, the customer can further customize the service to its needs.
2. An OSS system is responsible for orchestration of the customer-facing service aspects. For example, it interfaces with the billing system to register the user and its service choice. The OSS triggers EEO to line up all the network resources required for delivering the service.
3. EEO retrieves the End-to-End Network Service Descriptor information associated with the requested service. Based on this, it determines which resources are required and decides where to place new VNFs.

4. EEO interfaces with the VNFM and requests instantiation of the required VNFs.

5. For VNFs that consist of multiple VMs, the VNFM interfaces with the Data Center SDN Controller to create connectivity between the VMs within that VNF.

6. EEO interfaces with the DC SDN Controller to establish connectivity between VNFs. In the case of a vCPE, this entails creating a service chain that traverses the VNFs in the desired order.

7. EEO interfaces with the WAN SDN Controller to extend the customer’s VPN and connect to the newly established service chain in the data center.

8. EEO provisions the network elements in order to provide the end-to-end service requested by the customer. For example, the customer may have selected a specific security profile that needs to be enforced by the firewall. Therefore, EEO needs to make sure that the right profile is provisioned in the firewall. The IDS system may need to alert the customer when an intrusion is detected. EEO needs to provision the contact information (email address or otherwise) in the IDS system, etc. EEO could interface directly with the network functions or do so through an EMS.
End-to-End Network Service Descriptors (EENSD) capture the end-to-end infrastructure construction to deliver specific services that can be executed automatically by the EEO in order to achieve zero-touch service creation and activation. EENSDs generally include:

- Service templates that identify the network functions required to deliver the end-to-end services, the connections between them and the connections to the customer locations and data centers that are affected
- Work flow information, as some decisions need to be made before others in order to streamline and optimize service creation and activation
- Scripts, e.g. for the provisioning of relevant features

Apart from some work in ETSI/NFV, which will be discussed below, there has not been much progress in the industry on standardizing EENSDs. That is not considered an impediment, for the following reasons:

1. EEO functionality and sophistication will improve over time
2. Operators can start using EEO solutions in the absence of standard EENSDs

With reference to (1) above: In a sense, EENSDs act as a programming language for the orchestration engine. The more functionality there is in EEO, the simpler the EENSDs can become. One could compare it to the difference between programming in assembly or in a higher order programming language or with the difference between IaaS and PaaS. Efforts to nail down the content and format of EENSDs today could prove fruitless in the future when more sophisticated EEO solutions would require different types of input.

With reference to (2) above: EENSDs are created by the operator’s service architects. They reflect the services offered by the operator and are tightly linked to the operator’s network infrastructure. It is unlikely that the service architect would share its EENSDs with other operators or would be able to use existing work from others. The penalty for starting EEO for specific service in the absence of standard EENSDs is therefore not significant.

There is work going on in ETSI/NFV to define data models that capture how to construct and interconnect groups of VNFs required for a specific service. It specifies that a network service descriptor needs to identify the VNFs required to deliver the service, the PNFs with which they interface, the topology that describes how they are interconnected (i.e. the VNF Forwarding Graph or VNFFG) and a specification of the attributes of specific connections between VNFs (Virtual Links).

ETSI/NFV definitions don’t cover PNFs beyond those that interface directly with a group of NFVs. An EENSD therefore needs to be extended with information that captures connectivity across the end-to-end network. In general, this will involve the identification of required services that are controlled through WAN SDN Controllers.

More importantly, the ETSI/NFV definition covers which VNFs need to be instantiated and connected, but does not address how to activate the service. For example, it is one thing to instantiate an SGW and PGW as the EPC Core for a specific M2M application. However, to get a working service, EEO needs to identify the Access Point Names (APNs) that these gateways will serve and it needs to update the HSS so that UEs that attach are directed towards these new gateways. Actions like these will be captured in provisioning scripts that are part of the EENSD.
8 Policy Framework

8.1 Introduction

Policy does not exist in isolation. It coexists with, depends upon and supports other functions such as orchestration, analytics and control functions as well as service provider-wide issues like business models and objectives. In a sense, policy can be seen as externalizing, delegating and abstracting parts of the control logic of network functions, including business support and corporate strategies.

Thus, policies support automation (including automated coordination of service provider strategies), defined as the process of achieving specified behavior expressed by formalizing and structuring decisions in a stringent and unambiguous way. Whereas traditional polices facilitate automation, we can extend and advance policy concepts to make them adaptive, creating an important basis for achieving (sub)-system autonomic behavior.

The SDN-NFV architecture brings forth structure and organizational challenges that must be reflected in the policy framework. In this increasingly automated environment with shared resources, the role of security plays an important role. Hence, security must be an integral part of the policy framework at all levels.

In order to ensure that any architecture can be described in a consistent manner and be understood not only by the architects, it is necessary to lay down the key terminology. There are many ways to define policy; however in this document policy is defined as a function that governs the choices in behavior of a system.

Specifically, this can take the form of a set of rules which, when triggered by explicit or implicit events, evaluate conditions and generate appropriate actions. Note that policy does not implement the behaviors, it only selects among the possible behaviors.

The term “rules” in this context is used rather loosely; there are a number of technologies available that deviate from the classical rule engine model. In some emerging policy concepts, the focus is on policy as a statement of intent.

The inputs, decisions and actions that define a policy are context dependent and their nature depends very much on where they are applied. The following clarifications need to be made:

- Rule: this is not just classical if-then-else statements, but is to be interpreted rather loosely. It also covers other kinds of policies, such as intent or goal-oriented variants. The statements are phrased in terms that make sense in the jurisdiction (the area of influence of a given set of policy statements), i.e., the language chosen needs to be adapted to the context.
- Conditions and events: both are defined in terms of the information model that is relevant to the context. The work of identifying and condensing raw system data and event flows is typically allocated to the analytics function.
- Actions: essentially, this is defined by the services provided by the controlled system(s) in the jurisdiction (i.e., affected by this policy system).

8.2 Policy categories

As the scope of policy applicability is very wide, it is helpful to define the following three top level policy categories.
Business policy

This policy states the intent of the service provider in terms of business goals and the service level ambitions towards customers, owners and other stakeholders. This includes policies that express strategies intended to reach business targets such as profitability levels and market capitalization.

Service policy

This policy is related to what an administrative domain promises to its consumers (internal or external). This policy is used to set characteristics and control the behavior of service instances (deliveries, individual sessions), including how individual service instances should be handled in the face of resource constraints (including underlying resource failures). These policies typically form part of the definition of the service. Service policies relate to resource policies in that that the service must be defined in accordance with the bounds established by the resource policies defined in the affected domains.

The service policies may list applications that need to run on the host and define a policy that offers different levels of services at different prices. For example there may be basic, medium and high levels of service and price each level accordingly.

Resource policy

This policy is related to how a set of system resources stays within its defined constraints, and how it optimizes the delivery capabilities given those constraints. A resource policy is used to specify corrective behavior when the constraints (and/or optimization criteria) cannot be met. Examples include the scaling policies for VNFs, the control of resources allocation for tenants in the cloud, end-users access steering for access network optimization, traffic routing policies for transport optimization, policies for the control of rights to access platform services in the cloud, etc.

Resource policy involves a wide range of applications on different domains, like QoS marking, shaping, and queuing per traffic rules, firewall/authentication rules, routing and tunneling policies, L2 policy, and service chaining policy.

Business policies are traceable to the other subgroups as well, in particular for the service characteristics. The intent (the promise) is stated in e.g. marketing. It is restated in the customer contract and is
transformed into SLA supervision data and network state constraining the service’s behavior. In the same way, business requirements on availability can be transformed into resource constraints, given that the policy author (supported by the validation function) has access to structural and performance information of the system.

8.3 Separation of concern

Introducing policy driven governance of a large system as the service provider network may create additional complexity if based on a monolithic policy function to control the entire network. Instead, the complexity can be reduced by considering the network as a set of operational domains, or administrative domains, within a service provider. This subdivision narrows down the interaction between domains to the essentials, reducing the degrees of freedom and making the system more manageable. Some policies apply to the inter-domain interfaces, and define for example how one domain can use services from another domain. Other policies, like those that control how specific resources are used within a domain, are strictly defined and applied within an administrative domain.

As an example, the business policies may be defined at the highest level in three different contexts; Wireline, Wireless and Enterprise. As described above, these business policies can then be mapped to different service policies per administrative domains. Examples of these domains could be a transport domain (WAN), a cloud infrastructure (NFV) domain or a core network domain (EPC & IMS).

One of the key aspects of applying policy technologies is that it must allow the domain experts to state their policies in terms that make sense to them and the context. This means that the policy languages used to create statements will be different, as they serve different purposes. This does not preclude that the entire set of policies can be edited, maintained and validated (to the extent that this is possible pre-runtime) in a coherent fashion.

A system will contain many different kinds of policies, affecting system behavior in different ways. The main structuring concept is the notion of policy jurisdictions: i.e., the set of resources over which a given set of related policies can assert authority. This means that the jurisdiction is tightly coupled to ownership of, accountability for and responsibility for a set of resources. A policy jurisdiction may cover an entire administrative domain, or the domain may very well be subdivided. The subdivision can be along different dimensions: geographical regions, vendors, functionality etc. One jurisdiction may influence another via well-defined and agreed interfaces (this includes “cascaded” and “adjacent” jurisdictions).
9 SDN Controller Framework

9.1 Management and Control Architecture

Network operators have traditionally managed networking systems via standard management interfaces from one or more element management systems (EMS) toward the network elements within their purview. OSS/BSS systems leverage APIs and interfaces provided by these EMS systems to drive business operations into the network (e.g., provisioning, service management, and billing). Many of these interfaces are proprietary to each EMS. In some cases, the OSS/BSS layer interacts directly with the network elements themselves, often with proprietary interfaces, but in many cases through standards-based approaches. Further, networking protocols and protocol stacks, while written based on defined standards, are difficult to enhance without support from the vendor, which may contribute to elongated enhancement cycles impacting service revenue.

The goal of SDN is to separate the management and control planes from the data plane networking functions themselves. A key element of this goal is that the interfaces between the planes must be well documented in standards, to ensure interoperability across the ecosystems. These interfaces and the general constructs of an SDN control hierarchy are described in the remainder of this chapter.

9.2 SDN Control Architecture

The original SDN network architecture proposed decoupling of network control from packet forwarding and a direct programmability of the network control function. Network intelligence is (logically) centralized in software-based SDN controllers, which maintain a global view of the network. SDN Controllers, in turn, are responsible for programming forwarding rules in the network forwarding functions. In this existing definition of SDN, the forwarding functions do not support routing protocols themselves. Their entire behavior is controlled by the SDN Controllers. The forwarding functions are therefore called “white boxes”.

The architecture in this document has the following characteristics and is based on simultaneous existence of white boxes and forwarding boxes in the network.

- Logically centralized control
- Abstraction and network programmability
- Multi-vendor support through existing and new protocols
- Service deployment capability and activation instead of individual product components

Annex D contains the description of generic SDN Controllers.
Through the End-to-End Orchestrator: This form of orchestration applies when the domains are largely independent. For example, a Virtual CPE service may require an IP VPN service in the WAN that connects to a service chain in the data center to steer traffic through various virtualized functions such as a firewall and NAT. The service requirements in the WAN and data center can be managed independently of each other. The Network Resource Orchestrator that ties the two together only needs to identify the handoff point between the two (e.g. a Data Center Border Router) and the association between their respective identifiers (e.g. link VPN instance #31 to service chain #5). Another example is the scenario where an SDN domain is part of a service driven through hybrid control. Consider a WAN SDN controller laying down MPLS resources across a core network that will be used as a substrate for an L3 VPN. In this scenario, the L3 VPN instantiation points run on classic IP/MPLS PE routers that run BGP signaling between each other for service construction. The EEO uses NETCONF to configure the BGP and MPLS components of the PE, while the SDN controller pushes label entries into the PE and P routers.

Through an SDN Controller: This form of cross-domain control applies when there is a need for end-to-end resource management. One example is management of end-to-end optical connections that traverses multi-vendor domains. Current standards are insufficient to enable third-party controllers to efficiently manage resources within a vendor-specific optical domain. In the near term, therefore, domain-specific controllers are required for path calculation and connection establishment within a vendor-specific optical domain. The cross-domain WAN SDN controller is responsible for management of end-to-end connections. It receives high-level topology information from the domain controllers and uses that to determine the optimal end-to-end path. It subsequently instructs the domain controllers to establish the connection segments for each domain. Another example is the distribution of IP/MPLS control across multiple regions or administrative segments. The network may be partitioned for scale and operational flexibility. Local controllers may have deep visibility to local resources but present an abstract model northbound, where a cross-domain controller acts to create a path across multiple sub-domains.
The End-to-End Orchestration is a superset of classical provisioning and may or may not include activation, depending on the characteristics of a particular service. In the traditional network, provisioning occurs on inventoried elements as part of a workflow. If an element doesn’t exist or lacks sufficient capacity to meet the service requirements, provisioning stops and construction or upgrades commence. However, in the NFV world, the instantiation of new capacity is fundamental to the NFVI and the ability to spin up new resources “just in time” is one of the main reasons to virtualize network functions in the first place. Once the capacity is available, it must be plumbed into the rest of the network. The manual and labor-intensive process is now completely automated, dynamic, and optimized.

The following sections define the functionality of the various controllers in more detail. Interfaces, protocols and other standards are discussed in the context of each controller.

### 9.3 Data Center SDN Controllers

With the introduction of virtualization, where multiple Virtual Machines (VMs) could reside on the same server and where VM creation and migration introduces scalability and dynamicity requirements not seen before, new connectivity solutions are required. One example of a deployment option is shown in Figure 9-2:

![Figure 9-2: Data Center SDN Solutions](image)

The solution revolves around a possibly federated SDN Controller, which controls forwarding functions on servers through open protocols like OpenFlow, OVSDB, and NETCONF. Depending on the use case, these protocols require extensions, or additional adaptations, or the actual choice of the protocol may itself vary. The forwarding function on each server or VNF could either consist of native functionality alone, i.e. the “vSwitch”, or could be augmented by an additional forwarding agent, which may be controlled via the same protocols or via a vendor published API. It is important to note that the SDN controller may also be required to control the forwarding function on a physical switch. For example, in the case of a bare metal server or an appliance connected to a top of rack (ToR) switch, the SDN controller may be required to create an overlay tunnel between the ToR and the virtual forwarding function.

The Northbound interface (Or-Sdnc) is defined from the controller perspective and is presented as APIs towards Orchestration layers to pass the appropriate service primitives to provision the SDN controller.
based on the current use case. Towards northbound systems, the DC SDN Controller provides APIs that enable definition of connectivity requirements in abstract terms, typically using a REST API, an Openstack Neutron API or a vSphere API. Specifically, in the context of NFV MANO, these connectivity requirements may be formulated in YANG or TOSCA models that complement Network Service and VNF templates.

The Southbound interface (Sdnc-Net and Sdnc-Vi) is defined from the controller perspective and is utilized to communicate specific network element instructions. A southbound interface provides a mechanism to provision or program downstream network elements using a common set of parameters with extensions as needed per use case. The interface should have extensibility to support adding new functionality without backward compatibility concerns. Additional details on the Southbound Interface are provided in the subsequent sections. Note that the southbound interface can also be utilized for the network elements to communicate any metrics or KPIs. Southbound, the DC Controller uses Openflow, OVSDB, NETCONF, or Vendor Published APIs to program the network connectivity. When a VM is created, the SDN Controller applies the connectivity requirements and creates overlay tunnels between the VM and all other VMs that it needs to connect to. These overlay tunnels generally use VXLAN or GRE encapsulations.

As an example use case, consider the case of virtual services (firewall, load-balancers, etc). The DC Controller is responsible for instantiating the connectivity to attach the virtual service to the network per the connectivity requirements. The connectivity requirements could represent a completely virtual service or it could represent a service requiring connectivity to the physical network. When the connectivity requirements call for connecting to the physical network, the DC Controller is responsible for programming the gateway between the physical and virtual network (eg. VXLAN to VRF mapping).

Another example is a simple network slicing use case. In such a scenario, an operator wishes to create multiple network slices, or virtualized networks, that are interconnected via a common L2 closed user group over an IP-based underlay. VLANs are the classical method of creating such an environment, but due to 4K VLAN limitations, and given the wide proliferation of L3 capable DC switches, an IP-based overlay, such as VXLAN, provides the ability to slice a DC. Here, the DC controller may provision VXLAN IDs into VTEPs on TORs or on the hosts themselves. OF can be used to provision the rules for forwarding MAC frames into VXLAN tunnels with the assigned VXLAN IDs. BGP EVPN can be used to communicate MAC address reachability to the controller, which may program this reachability via OF, OVSDB, or via propagation of EVPN NLRI to interested hosts or DCI routers.

Federation between DC SDN Controllers is generally based on BGP and variations thereof, such as Multi-Protocol BGP (MP-BGP), while more comprehensive protocols are explored. This is described in Annex B.

The overlay network created by the DC SDN Controller can be independent of the underlay network. The underlay network could consist of classical data center routers and switches or it could consist of white boxes under control of an “Underlay SDN controller”. If stricter performance guarantees are warranted, the underlay and overlay can be bound using segment routing capabilities if the underlay supports MPLS or IPv6 Segment Routing. If there is no requirement for overlay performance guarantees, the overlay DC SDN controller can establish overlay tunnels without interaction with the underlay network. However, for troubleshooting, it is desirable that the operator can correlate problems in the overlay network with failures in the underlay network. There are several ways to create assurance solutions that cover both underlay and overlay. One such method leverages SR OAM capabilities (see Segment Routing in the Annex C).
### 9.4 Architectural Considerations for SDN/NFV

NFV requires a networking solution that is SW-defined and API-driven that allows operators to use virtual networks that are dynamically created, modified, scaled and deleted along with the VNFs. This virtual networking system also needs to provide a flexible service chaining mechanism for creating virtual topologies with specific rules for forwarding traffic along pre-defined service chains.

A clear strategy for datacenter SDN is to develop a highly scalable, multi-tenant architecture that is application-centric and policy-driven.

Key characteristics of the DC SDN strategy include:
- Simplified automation by an application-driven policy model
- Centralized visibility with real-time, application health monitoring
- Open software flexibility for DevOps teams and ecosystem partner integration
- Scalable performance and multi-tenancy using SW or HW virtual overlays

The ability to leverage software defined networking methods is predicated on a contractual relationship between the resource supplier and the resource consumer. The programmatic capabilities are dependent upon the resources offered and methods of consumption. Much like the WAN infrastructure has fixed resource capabilities; the data center also has fixed resource capabilities. For Virtualized Network Functions, an important characteristic that is defined as a part of the service level agreement is the resource availability, and resource availability has many attributes that may be part of the NSD, the VNFD, or both\(^1\). The many attributes may include the following:

- Availability
- Dedicated – Resources permanently assigned
- Scheduled – Resources assigned during scheduled window
- Priority On-Demand – Resource dynamically assigned with priority
- Best-Effort On-Demand – Resource dynamically assigned no priority
- Resiliency Protection Method
- Active:Active – Full redundancy
- Active:Active-N – Elastic full redundancy
- Active:Standby – Assigned recovery resources
- Active:Recovery – Unassigned recover resources
- Affinity Rules
- Affinity Co-location – Resources assigned from same asset
- Affinity Diversity – Resources assigned from separate assets
- Affinity Geo-Diversity – Resource assigned from geo-diverse assets
- Bandwidth
- Dedicated Bandwidth
- Shared Bandwidth with Known Maximum
- Dedicated Bandwidth with Variable Maximum
- Shared Bandwidth with Variable Maximum
- Quality of Service
- Throughput
- Prioritization Queues
- Latency Limits
- Jitter Limits

---

\(^1\) Attributes - Note that many of these attributes and descriptors apply to the physical network elements themselves (i.e., bandwidth out of the DC).
9.5 WAN SDN Controller

Data Center SDN solutions are well defined, clearly bounded solutions that meet the scalability, dynamicity and multi-tenancy needs in data centers. WAN SDN solutions are emerging as the next major application of SDN and a growing number of deployments have been announced. In general, a WAN SDN controller must accomplish the following objectives:

- Northbound APIs (Or-Sdnc) for network abstraction and programmability to be used by customer-facing orchestration systems and third-party applications
- Model-driven Adaptation Layer that allows network service models to help drive north bound and south bound mappings
- Multi-layer and multi-vendor control
- Automated management of end-to-end services
- Resource optimization
- Topology discovery and rectification
- Statistics collection and processing
- Provide a control interface to hybrid/classical control mechanisms

Figure 9-3 depicts the primary functions of a WAN SDN Controller:

![Figure 9-3: Functions of a WAN SDN Controller](image)

A goal of this document is to clarify the boundaries and functions of the WAN SDN Controller. The boundaries of a WAN SDN Controller are not well defined. For example, some consider resource optimization an integral part of an SDN Controller, while others consider it a separate function or an app that runs on top of an SDN Controller. The same is true for service management. The dotted line in the Figure 9-4 below conveys that certain functions may or may not be part of the core controller:
There are various standardization efforts (e.g. in IETF and ONF) as well as open source initiatives (e.g. OpenDaylight and ONOS) that are defining the internal data models and APIs that enable mixing and matching of functional elements. The functional blocks shown in the diagram above can be aggregated or disaggregated.

Before discussing the various functions in more detail, it is necessary to define the difference between a “service” and a “resource” in this context:

- A WAN connectivity service is defined by the information carried over the service, the high-level routing and connectivity structure and associated attributes, such as QoS, security and protection schemes.
- A resource is a networking construct on which WAN connectivity services are built: MPLS LSPs, Ethernet connection, lambdas, etc.

For example, an MPLS LSP with a guaranteed bandwidth is a resource. That LSP can be used to carry point-to-point Ethernet traffic (using pseudowire encapsulation) or IP VPN traffic. Clearly, these are two distinct services that could make use of the same resource. To a large extent, resources and services can be managed independently. For example, as a consequence of an optimization action, an LSP could be rerouted without affecting the service. On the other hand, the service could be modified (e.g. encryption can be switched on or off) without affecting the LSP.

**Service Management**

The Service Management function is responsible for all aspects of service instantiation and management in the network. The northbound API enables a high-level, abstract definition of service instances. For example, a service could be defined by:

- **Protocol** - Defines what information is carried over the service: IP, Ethernet, OTN, alien lambdas, etc (ITU-T G.809 calls this “characteristic information”)
• **Structure** - Point-to-point (line), point-to-multi-point (tree), multipoint-to-multipoint (vpn), chain and composite (i.e. combinations of the previous, e.g. a point-to-point service as on-ramp to a VPN).

• **Path attributes** (segmented, loose, strict, expanded, inter-area, inter-domain) - These inform the orchestration system of available path characteristics that can be leveraged as part of a service. This includes the attributes below. For example, an enterprise may wish to order a VPN service where certain traffic takes a low-latency path.

• **End-points** - Which consumers, mobile units, NFs, enterprise locations, hosts/VMs and/or data centers are connected through the service

• **Additional attributes** - QoS (bandwidth, latency, prioritization, shaping); security, availability and protection requirements, etc

Service Management as defined in this manner is a form of Resource Orchestration of WAN connectivity services, which covers multiple network functions rather than a single network function. As a consequence, northbound resource orchestration functions don’t need to know the details of the WAN. They can simply rely on the abstract definitions provided by the Service Manager to create more complex services that combine WAN connectivity services with other services (such as IMS, Gi-LAN, or CDN services).

**Resource Management and Optimization**

This function enables northbound systems to create, modify or delete resources. It provides functions like optimal path selection, tunnel load balancing, bulk optimization, traffic reroute for maintenance actions, etc. The Resource Management function can use the statistics collected by the SDN Controller itself or it can interface with standalone analytics solutions that may trigger optimization actions.

**Statistics and Topology**

An SDN Controller provides several basic functions, such as statistics collection (per service and/or resource) and topology discovery. This information is used as input to the Resource Management function and can also be exposed to northbound systems via ReSTful APIs.

**Forwarding Rules Manager**

When the network contains white boxes, a Forwarding Rules Function is required to define the exact set of forwarding rules to be pushed to those white boxes. The Forwarding Rules Manager takes into account topology information and the service and resource related inputs from the Service Management and Resource Management functions.

**Adaptation**

The adaption function translates abstract, device-independent data models that are used by the Service Management and Resource Management functions to device-specific data models. These device independent data models are service-oriented models that are rendered in YANG.

**South-bound protocol stacks**

Depending on the network element and the required action, different protocols may be used. OpenFlow is one of the protocols of choice for managing white boxes. The choice of the south bound protocol depends on the use case and will also require additional adaptations / extensions. NETCONF (RFC 6241) is
Emerging is the primary candidate for provisioning network functions in general. PCE Protocol (PCEP, RFC 5440) is the primary protocol for path setup in RSVP-TE MPLS networks. PCEP, OF, and NETCONF are also being leveraged for Segment Routing-based MPLS networks. BGP-LS (still in the Internet Draft stage) can be used for topology discovery. In addition, certain network functions may require proprietary protocols.

In summary:

- **Southbound** (Sdnc-net, Sdnc-Vi, Sdnc-Nf or Proprietary) - Towards network functions. WAN SDN Controllers support a variety of different protocols, including OpenFlow and NETCONF, as discussed above.

- **Northbound** (Or-Sdnc) - WAN SDN Controllers use ReSTful APIs to enable service management and resource optimization based on abstract, simple data models. For example, a YANG model that describes a segment-routed, label-switched path can be used to generate a ReST NB API. This allows a portal or OSS subsystem to direct the establishment of a path across multiple domains in the network.

### 9.6 Domain-specific controllers

Existing Domain-specific controllers are not necessarily different from WAN SDN Controllers as discussed above. Even if a controller is only able to manage a vendor-specific domain today, with the introduction of new standards for protocols and data models, that same controller may be able to manage multi-vendor networks tomorrow. That said, a domain-specific controller exposes more limited functionality than a higher-level WAN SDN Controller. This is illustrated in the following figure:

![Figure 9-5: WAN SDN Controller Plus Domain-Specific Controllers](image)

**Note**: Figure 9-5 shows only a subset of functional blocks of the WAN SDN Controller.

Domain-specific SDN Controllers expose high-level topology information to the WAN SDN Controller. Based on this information the WAN SDN Controller interacts with the Domain-specific Controllers to
define a possible (or the optimal) path. It then instructs them to establish the segments that traverse their respective domains and to provision the service characteristics at the end-points.

Figure 9-5 above shows establishment of an optical connection across optical domains. However, a similar construct could be used to manage end-to-end IP/MPLS domains that are split into independent domains for scalability reasons or it could be used for combined IP-Optical optimization where the higher-level WAN SDN Controller could perform IP/MPLS optimization and interface with the Domain-specific Controller for optimization in the optical domain.

The interfaces between higher-level WAN SDN Controllers and Domain-specific Controllers are currently being discussed in various forums. These interfaces will likely be based on a combination of ReSTful APIs, PCEP and NETCONF.

9.7 Access SDN Controllers

Access SDN will be addressed at a later date.
10 Interfaces

This chapter describes all the interfaces in the target SDN-NFV architecture. Some of them are based on ETSI NFV, while others are introduced in this architecture.

Vi-Ha (Virtualization Layer – Hardware Resources)

Interfacing the virtualization layer to hardware resources to create an execution environment for VNFs, and collect relevant hardware resource state information for managing the VNFs without being dependent on any hardware platform.

Vi-Ha interface enables the abstraction of the hardware, BIOS, Drivers, I/O (NICs), Accelerators, Memory and the network domain. This interface is part of the execution environment and is internal to NFVI. For OpenStack and Linux/KVM based deployments, Linux Operating System, KVM (part of Linux kernel) & QEMU provide this interface. Libvirt includes drivers to interface with different virtualization & storage technologies.

Some implementation examples of this interface are:

- Libvirt drivers: [https://libvirt.org/drivers.html](https://libvirt.org/drivers.html)

Vn-Nf Interface (VNF - NFVI)

This interface represents the execution environment provided by the NFVI to the VNF. It does not assume any specific control protocol and guarantees hardware independent lifecycle, performance and portability requirements of the VNF. The Vn-Nf interface provides a VM shell that interfaces with the underlying hypervisor and provides transparent network services to VNFs. This interface is used to interconnect the following:

- VNFCs to other VNFCs within the same or another VNF.
- VNFCs to storage.
- VNFs to PNFs and external endpoints

For OpenStack based deployments, this interface is realized by OpenStack flavors that provide VM sizes (CPUs, memory & storage) and OpenStack glance images that provide the guest OS and associated drivers. OpenStack Neutron provides the capabilities of creating network ports that can be attached to virtual network interfaces within VNFCs.

Some implementation examples of this interface are:

- OpenStack compute flavors: [http://docs.openstack.org/admin-guide-cloud/compute-flavors.html](http://docs.openstack.org/admin-guide-cloud/compute-flavors.html)
- OpenStack Glance images: [http://docs.openstack.org/developer/glance/](http://docs.openstack.org/developer/glance/)

Nf-Vi Interface (NFVI - VIM)

This is the interface between the management and orchestration agents in the infrastructure network domain and the management and orchestration functions in the virtual infrastructure management (VIM).
Orchestration and management of NFVI is strictly via the Nf-Vi interface and includes the following:

- Specific assignment of virtualized resources in response to resource allocation requests
- Forwarding of virtualized resources state information
- Hardware resource configuration and state information (e.g. events) exchange

Nf-Vi provides the interface between the management and orchestration agents in compute domain and the management and orchestration functions in the Virtual Infrastructure Management (VIM) and is used by the VIM to manage the compute and storage portion of the NFVI. An example implementation for Nf-Vi is using libvirt/KVM plugin to OpenStack Nova.

Nf-Vi also provides the interface that is used by the hypervisor to send monitoring information of the infrastructure to the VIM. All necessary commands, configurations, alerts, policies, responses and updates go through this interface.

This interface is also used for NFVI capability discovery described in Section 3.2.5

In case of OpenStack, a number of services run in the host (Nova, Neutron, etc.). These services, together with the virtual switch (if there is any) are consuming and processing information provided by the hardware resources using the Vi-Ha interface. OpenStack uses a plugin approach that is leveraged by NFVI component vendors to develop their device specific plugins. For OpenStack based deployments, plugins for the respective OpenStack projects provide the relevant interfaces & APIs.

Some implementation examples & guidelines are:

- Neutron: https://wiki.openstack.org/wiki/Neutron_Plugins_and_Drivers
- Nova: http://docs.openstack.org/developer/nova/devref/api_plugins.html
- Cinder: https://wiki.openstack.org/wiki/CinderSupportMatrix

In addition to the above NFVI-VIM interfaces specified by ETSI, some additional interfaces are required that provide and interface to manage the physical infrastructure as described in Section 3.1.2. While it is desirable to have a common standard for interfacing with hardware from different vendors, most of these capabilities are exposed using vendor specific APIs. Some efforts like DMTF’s Redfish (http://dmtf.org/standards/redfish) look promising and have the potential to providing a common interface to most, if not all, capabilities required for physical infrastructure management. Until such time, it is expected that vendors will provide a vendor specific API that should align with modern specifications like the use of ReSTful APIs.

Or-Vi (NFVO – VIM)

This interface is used for exchanges between the NFV Orchestrator and the VIM to request resources and VNFC instantiations and for the VIM to report the characteristics, availability, and status of infrastructure resources.

- Resource reservation and/or allocation requests by the NFV Orchestrator
- Virtualized hardware resource configuration and state information (e.g. events) exchange:
  - NFVI resource reservation/release
  - NFVI resource allocation/release/update
  - VNF software image addition/deletion/update
  - Forwarding of configuration information, events, measurement results, and usage records regarding NFVI resources to the NFV Orchestrator
Or-Vi interface supports the following functions:

- VNF software image management
- Virtualized resources catalogue management
- Virtualized resources capacity management
- Virtualized resources management
- Virtualized resources performance management
- Virtualized resources fault management
- Policy administration interface
- NFP management interface

For OpenStack based deployments, the OpenStack API realizes this interface. Please note that each OpenStack project exposes its own API that can either be accessed directly or using OpenStack Heat orchestration templates.

Implementation example:

- OpenStack API: [http://developer.openstack.org/api-ref.html](http://developer.openstack.org/api-ref.html)

**Vi-Vnfm Interface (VIM - VNFM)**

This interface is used by the VNF Manager to request and/or for the VIM to report the characteristics, availability, and status of infrastructure resources

- Resource allocation requests by the VNF Manager
- Virtualized hardware resource configuration and state information (e.g. events) exchange
  - NFVI resources reservation information retrieval
  - NFVI resources allocation/release
  - Exchanges of configuration information between reference point peers, and forwarding to the VNF Manager such information for which the VNFM has subscribed to (e.g. events, measurement results, and usage records regarding NFVI resources used by a VNF)

Vi-Vnfm interface supports the following functions:

- VNF software image management
- Virtualized resources catalogue management
- Virtualized resources management
- Virtualized resources performance management
- Virtualized resources fault management

For OpenStack based deployments, this interface is also realized by the OpenStack API either directly or using the Heat orchestration templates.

Implementation example

- OpenStack API: [http://developer.openstack.org/api-ref.html](http://developer.openstack.org/api-ref.html)

**Ve-Vnfm Interface (VNF/EM - VNFM)**

This interface is used for exchanges between the VNF, Element Management Systems (EMS) & the VNF Manager. This interface can further classified into the following two interfaces:
Ve-Vnfm-em, a reference point between EMS and VNFM. This is likely to be vendor specific and will be dependent on the capabilities & interfaces exposed for the Element Managers.

Ve-Vnfm-vnf, a reference point between VNF and VNFM. This will likely be vendor specific in the near term, but can benefit from a common or standard interface when used with OpenStack based deployments.

These reference points are used for exchanges between VNF/EM and VNFM and needs to support the following:

- Requests for VNF lifecycle management
- Exchanging configuration information
- Exchanging state information necessary for network service lifecycle management

While this interface is defined by ETSI, there are no standards or standard implementations that govern this. The target architecture will only use Ve-Vnfm-vnf.

Ve-Vnfm interfaces support the following functions:

- VNF configuration
- VNF performance management
- VNF fault management

Or-Vnfm Interface (NFVO - VNFM)

This interface is used for exchanges between NFV Orchestrator and VNF Manager, and needs to support the following:

- Resource related requests, e.g. authorization, validation, reservation, and allocation to the VNF Manager(s)
- Sending configuration information to the VNF manager, so that the VNF can be configured appropriately to function within the VNF Forwarding Graph in the network service
- Collecting state information of the VNF necessary for network service lifecycle management
- ETSI has defined the following actions that need to be supported by this interface:
  - NFVI resources authorization/validation/reservation/release for a VNF
  - NFVI resources allocation/release request for a VNF
  - VNF instantiation
  - VNF instance query (e.g. retrieve any run-time information)
  - VNF instance update (e.g. update configuration)
  - VNF instance scaling out/in, and up/down
  - VNF instance termination
  - VNF package query

This interface also supports the forwarding of events, and other state information about the VNF that may also impact the Network Service instance.

Or-Vnfm interfaces support the following functions:

- VNF package management
- VNF lifecycle operation granting
- VNF lifecycle management
VNF lifecycle change notification
VNF performance management
VNF fault management
Virtualized resources management
Policy administration interface

While this interface is defined by ETSI, there are no standards or standard implementations that govern this and the VNF (or VNF Manager) vendors are developing custom interfaces, which are then used by Orchestrators. This interface will benefit from a common or standard API, especially when used with OpenStack, which can be facilitated by generic VNF Manager efforts like that of Tacker described earlier.

- OpenStack Tacker: https://wiki.openstack.org/wiki/Tacker

Se-Ma (Catalogs/Repositories and Orchestration)

Service, VNF and Infrastructure Description schemas provide information regarding the VNF deployment template, VNF Forwarding Graph, service-related information, and NFV infrastructure information models. These templates/descriptors are used within NFV Management and Orchestration. The NFV Management and Orchestration functional blocks handle information contained in the templates/descriptors and may expose (subsets of) such information to applicable functional blocks, as needed.

While ETSI has defined the interface, there is no consensus for implementation, different options being discussed and considered are OpenStack Heat, TOSCA and Yang/NETCONF.

- TOSCA: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html
- Yang/NETCONF: http://www.yang-central.org/

Re-Sa (Repositories and Service Assurance)

This interface is used by SA to access service assurance data in Repository R5 as described in Chapter 2.

- TOSCA: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html
- Yang/NETCONF: http://www.yang-central.org/

Ca-Vnfm (Catalogs and VNFM)

This interface is used by VNFM for the lifecycle management of VNFs.

- TOSCA: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html
- Yang/NETCONF: http://www.yang-central.org/

Os-Ma (OSS/BSS – NFVM & Orchestration)

The Os-Ma reference point is used for exchanges between NFV Orchestration and the existing OSS/BSS systems. Os-Ma-nfvo reference point which is derived from Os-Ma needs to support the following:
Network Service Descriptor and VNF package management

Network Service instance lifecycle management:
- Network Service instantiation
- Network Service instance update (e.g. update a VNF instance that is comprised in the Network Service instance)
- Network Service instance query (e.g. retrieving summarized information about NFVI resources associated to the Network Service instance, or to a VNF instance within the Network Service instance)
- Network Service instance scaling
- Network Service instance termination

VNF lifecycle management:
- For VNF lifecycle management, the NFV Orchestrator identifies the VNF Manager and forwards such requests (see Or-Vnfm description)

Policy management and/or enforcement for Network Service instances, VNF instances and NFVI resources (for authorization/access control, resource reservation/placement/allocation, etc.).

Querying relevant Network Service instance and VNF instance information from the OSS/BSS.

Forwarding of events, accounting and usage records and performance measurement results regarding Network Service instances, VNF instances, and NFVI resources to OSS/BSS, as well as information about the associations between those instances and NFVI resources, e.g. number of VMs used by a certain VNF instance.

Os-Ma interface supports the following functions:
- NS interfaces
- VNF package management
- VNF software image management
- VNF lifecycle management
- VNF lifecycle change notification
- Policy administration interface

While this interface is defined by ETSI, there are no standards or standard implementations that govern this. For the near term, this will be vendor specific and defined by the existing OSS/BSS solutions and the NFV Orchestrators will need to provide a way to interface with them.

Or-Sa (NFVO – Service Assurance)

This interface has not been defined by ETSI and is expected that the NFV Orchestrator provide a flexible and extensible way to interface with Service Assurance systems. REST APIs are required on this interface. NETCONF can be used with VNFs where a NETCONF agent is available and where a YANG model is used to set the Service Assurance configuration details. In the medium term, translators could be developed that can convert YANG models to proprietary interfaces.

Or-EMS (NFVO – Element Management Systems)

This interface has not been defined by ETSI and will likely be vendor specific in the near term. It is expected that the NFV Orchestrator provide a flexible and extensible way to interface with the existing Element Management Systems either directly or using the VNF Manager.
Ve-Sa (Virtual Element – Service Assurance)

This interface has not been defined by ETSI and will likely be vendor specific in the near term. It is expected that Virtual Elements will continue to expose the metrics of their physical counterparts. Over time it is expected that Virtual Elements will start leveraging the common infrastructure type metrics provided by the VIM and NFVI in addition the existing vendor specific metrics.

Vnfm-Sa (VNF Manager – Service Assurance)

This interface has not been defined by ETSI and will likely be vendor specific in the near term. It is expected that Virtual Elements will continue to expose the metrics of their physical counterparts. Over time it is expected that VNF Managers will start leveraging the common infrastructure type metrics provided by the VIM and NFVI in addition the existing vendor specific metrics.

For VNF's that can be managed by a generic VNF Manager, it is expected that a common interface will be used that aligns with the metrics available through VIM and NFVI. REST APIs are required on this interface. NETCONF can be used where a NETCONF agent is available and where a YANG model is used to drive SA configuration details.

Vi-Sa (VIM – Service Assurance)

This interface has not been defined by ETSI, and will likely be specific to each VIM in the near term. REST APIs are required on this interface. NETCONF can be used where a NETCONF agent is available and where a YANG model is used to drive SA configuration details. It is expected that Service Assurance provides a flexible and extensible way to interface with the existing and evolving metrics and APIs that are being developed.

For OpenStack based implementations, following options are available:

- OpenStack Ceilometer: (https://wiki.openstack.org/wiki/Ceilometer)
- OpenStack Monasca: (https://wiki.openstack.org/wiki/Monasca)

Nfvi-Sa (Network Function – Service Assurance)

This interface has not been defined by ETSI and will likely be vendor specific in the near term. It is expected that Network Functions will continue to expose the existing metrics, such as SNMP, SYSLOG, and NETCONF. Over time it is expected that Network Functions will also start leveraging the common infrastructure type metrics provided by the VIM and NFVI in addition the existing vendor specific metrics.

Or-Nf (Orchestrator – Network Function)

This interface will be a combination of NETCONF, RESTCONF, and vendor specific methods (such as CLI or custom SDKs). NETCONF and RESTCONF are defined by IETF.

Or-Sdnc (Orchestrator – SDN Controller)

This interface has not yet been defined by ETSI. End to end service orchestration and NFVI orchestration typically use REST API (for OpenDaylight and others) or other SDN-Controller specific interfaces to make network changes to enable end-to-end network service provisioning across data centers and WAN domains that are likely controlled by multiple SDN controllers. NETCONF and RESTCONF are also used, where the capabilities of the SDN controller have been specified in YANG Models.
Sdnc-Nf (SDN Controller – Network Function)

This interface has not yet been defined by ETSI. Network functions, like software forwarding boxes and virtualized PE/CE routers often require interaction with WAN and DC SDN controllers. OpenFlow, XMPP, NETCONF, and other interfaces are commonly used on this interface.

Vi-Sdnc (VIM to SDN Controller)

This interface has not yet been defined by ETSI. The VIM must signal to the datacenter SDN controller that new virtual machines are available and need to be assigned to specific subnets. The OpenStack Neutron API is commonly used on this interface, but it lacks a number of critical networking functions required for scalability, topology management, and service chaining. SDN controller vendor-specific Neutron plugins are available as are Modular Layer 2 (ML2) https://wiki.openstack.org/wiki/Neutron/ML2 to address some of the gaps in Neutron networking.

Sdnc-Net (SDN Controller - Networks)

This interface has not yet been defined by ETSI. The DC SDN controller controls the NFVI networks via this interface. Northbound interfaces on the SDN controller to the VIM and Orchestration define high-level intent-based changes. The SDN controller then implements these changes on the networks, typically using an OpenStack Neutron agent, OpenFlow, XMPP, NETCONF, or another southbound protocol.

Dsc-Nf (Domain Specific Controller – Network Functions)

This interface has not yet been defined by ETSI. Domain specific controller interfaces will leverage a mix of proprietary extensions and standard protocols like NETCONF and OF.

Sdnc-Sa (SDN Controller – Service Assurance)

This interface has not been defined by ETSI. SDN Controllers gather statistics related to the services they manage. They will pass these statistics, potentially in a filtered and aggregate form, on to the Service Assurance function via the Sdnc-Sa interface. The statistics will generally cover network utilization and performance aspects (load, latency, packet loss, etc).

Cf-N (Collection Function - NFVI)

This interface has not been defined by ETSI. This is a specific interface that was designed for diagnostic purpose and captures messages passing between VNFs and sent to an off board processing platform which collects, correlates and displays the relevant call flows and session information, from the data collected.

Cf-Sa (Collection Function – Service Assurance)

This interface has not been defined by ETSI. This interface is used to trigger the capture of network information during disaster or abnormal conditions.
11  Architectural Considerations

11.1 Software Considerations

When constructing the high-level software architecture using VNFs, there are five basic functional handling areas, which should be considered depending on the VNF:

- VNF O&M Handling (VNFC which may talk to the Element Manager)
- Control Plane Handling
- User Plane Handling
- IP routing/Interface Handling
- Database Management Handling

VNF O&M handling can encompass the FCAPS capabilities and VNF element supervision. Control plane handling can involve managing the UE and the associated signaling. User plane handling can contain all the functions involving the forwarding of UE traffic. IP routing and interface handling can be comprised of the functions needed to handle the 3GPP logical interfaces and forwarding of user traffic. Database management handling can cover the subscriber information handling. By separating the VNFs into these functional handling areas, it allows for the independent scaling of each area on an as-needed basis.

Once the functional areas are defined, there are a few functional considerations in the NFV application implementation, which should be taken into account:

- FCAPS – Fault, Configuration, Accounting, Performance & Security
- High Availability
- Resilience
- Load Balancing and Scaling
- Software Handling (Upgrade, Patching, Backup, Restore)

FCAPS

There are five basic pillars to operating and maintaining a network element: Fault Management, Configuration Management, Accounting Management, Performance Management and Security Management.

Fault Management is the sending and managing of alarms for network elements. Configuration Management involves the necessary basics to configure a network element. Accounting Management governs the billing functionality (if applicable) for a network element. Performance Management entails the collection and utilization of counter based statistics, which provides insight into the performance of the network element. Security Management involves user and command level access to a network element.

High Availability & Resilience

Integral to any telecommunications network element is the ability to withstand faults and prevent the existence of a single point of failure in a network element. The high availability and resilience design is fundamental to the architecture of the network element. When considering the resilience architecture for a network element, there are two key aspects to consider: network element internal high availability and resilience, and high availability and resilience on a network level.

Network Element internal high availability needs to account for three fundamental areas:

- Interfaces
- Call Processing
Subscriber Handling

The VNF should have redundant interfaces to ensure that communication to and from the network to other elements can withstand a single network failure. This could be accomplished using the following mechanisms:

- 1+1 Protection Switching Schemes
- Use of Layer 2 Redundancy Mechanisms
- Use of Layer 3 Dynamic Routing Protocols

The ability to support redundant call processing is also vital to a resilient design. The network element needs the ability to continue processing calls if one of the call processing functions (e.g. Control Plane or User Plane handling) were to suffer a software or hardware failure. Ideally, the design should support a hitless architecture, which prevents the end user devices from realizing a fault has occurred in the network. Call processing typically works in conjunction with subscriber handling in most EPC elements and can be accomplished by stateful or stateless subscriber handling.

Once the internal resilience foundations have been established, a decision must be made as to how much of it will be extended to other similar network elements (example: MME backing up other MME’s) in the network in a geographically redundant fashion. Typically, this involves the sharing of call processing between network elements. An example of shared resilience between similar network elements is the MME in Pool concept described in 3GPP. This allows the RAN to be connected to multiple MMEs at the same time in a load sharing and resilient fashion. This concept can be extended to the sharing of subscriber state information using the Geo-Redundant MME Pool functionality. Careful consideration should be taken when determining how much of the internal resilience functionality is complemented or duplicated externally because it can have tradeoffs such as increased network traffic, increased processing load, reduced individual node capacity and increased latency.

Network architecture and the network platform should allow flexible IMS software configuration. For example, CSCF VNF may have one or more logical functions either as a combined VM or separated VMs that provide the CSCF services. Each CSCF Subscriber is assigned to a certain CSCF VM and a traffic dispatcher can be used to ensure that all messages for a subscriber are forwarded to the same VM.

Load Balancing and Scaling

Load balancing in VNFs is an essential component in allowing the element to scale by distributing traffic across multiple compute platforms. In IP-based networks, some load balancing techniques are centered around the use of OSI layer 2 and layer 3 protocols such as ECMP, OSPF or BGP. Other techniques can utilize the use of Domain Name Service (DNS) or specialized hashing algorithms, which key on a particular field or value to determine how to spread the traffic. One example might be to use a combination of IP Address and Layer 4 port numbers to distribute traffic.

Software Handling

All software implementations need the ability to evolve and recover. Provisions in the applications should be made to support version upgrades and patching in as seamless a manner as possible. This means supporting backwards compatibility while at the same time adding new functionality. The upgrade and patching process should also ideally support a modular approach so that only the segments of code that require patching or upgrading will be changed. This should be complimented by some form of in service software updating (ISSU) to minimize the impact to existing subscribers.
Applications also need the ability to backup and restore software and configuration information. The backup of essential configuration information, parameters and software elements requires some form of persistent storage, which can be accessed as needed.

All of these functional considerations should be inherent to the VNF, but how they are implemented depends on vendor architecture choices. For example, a typical PGW today might implement the collection of performance statistical information differently if the choice is to have a split control plane and user plane architecture using forwarding boxes. The chosen architecture of a particular VNF could also affect the way high availability and resilience is handled.

11.2 Virtualization and SDN Considerations

11.2.1 Introduction

When virtualizing the VNFs, there are several considerations to take into account. Some of the general principles include having the same feature set for virtualized and non-virtualized resources, no impact on surrounding network nodes or VNF to device communication, and 3GPP-complaint multivendor support. Beyond that there are other categories to consider such as Availability, Performance, Network Separation, Security & Integrity, and Scalability. Highlighted below are some of the functions and capabilities that should be considered for those categories. This is not intended to be an all-inclusive list; rather, it is a highlighting of some of the more prominent capabilities.

11.2.2 Availability

Much like the traditional PNF functions, availability is an important aspect in virtualized applications. Some of the fundamental traits of availability include:

- Elimination of single points of failure
- Load balancing / dispatching internal to VNF
- Session replication and failovers between network functions
- Geo-redundancy / inter-site failovers between network functions

The goal of each of these traits is to ensure resilience to the five-nines availability required in telecommunications networks. It begins with assuring that there are no single points of failure in the design. This can be accomplished by having multiple functions performing the same task so that the failure of one function cannot result in the loss of service. When you have multiple functions performing similar tasks it is important to incorporate load balancing mechanisms to distribute the traffic in a symmetric way across the like functions. Depending on the implementation technique it is also possible for the load balancing mechanism to detect failures and redirect traffic to a working function. This, in turn, triggers the need for a session replication capability so that user information can be seamlessly transitioned without the end user or other network elements being aware of the transition. Finally, the applications should be deployed in geographically diverse locations to prevent complete site outage or site isolation from providing availability to the end users. Typically, geographic redundancy is accomplished by having another working version of the application in the second geographic location. Generic network load balancers or domain name service lookups then could control the load between the two different sites, for example.
11.2.3 Performance

Currently, most NFs are implemented in purpose-built hardware with dedicated backplanes, bandwidth, and resources only for that NF. When these applications are transitioned into the cloud, a VNF will typically span multiple x86 compute platforms connected by a shared cloud-networking infrastructure. This means the application can no longer take for granted the benefits gained from the dedicated platform. As a result, this introduces some new design aspects, which must be considered when virtualizing:

- Maintaining low latency between VNFCs with a VNF
- Utilizing cloud and infrastructure techniques to maintain high throughput
- Capacity scaling using a combination of mechanisms

Low latency between VNFCs could be accomplished by a number of techniques including the use of affinity and anti-affinity rules. These rules could help ensure that VMs are not instantiated in different data centers even though they are part of the same VNF.

Ensuring the required throughput could be accomplished by the use of software techniques which allow access to hardware acceleration such as DPDK and SR-IOV. There are also techniques involving the use of an optimized open virtual switch in the orchestration layer to ensure proper scheduling of network hardware resources.

Scaling of the VNF can typically be performed either vertically or horizontally. Vertical scaling, sometimes referred to as a “scale up”, involves the allocation of more compute resources to a particular VM. Horizontal scaling, sometimes referred to as a “scale out” involves the allocation of additional VMs to the VNF to add capacity.

Scale up can be used in VNF to provide additional CPU processing power to an individual VM. This could increase capacity of the VM without the need to worry about need to replicate any state information. Scale up, however, does not address the issue when more networking resources are required. This can occur when the amount of traffic entering and exiting the VM is close to exceeding the physical line rate available from the Ethernet card.

Scale out can be used to address the need for additional networking resources by creating an additional VM on new compute resources with available networking capacity. Scaling out does introduce the potential need for the replication of state or session information to the new VM. This is dependent on the vendor’s implementation of handling state and session information.

11.2.4 Network Separation

In a virtualized world, there are two aspects of network separation which must be accounted for:

- Multi-Tenancy
- Separation of traffic interfaces within a VNF

Virtualization introduces the concept of multi-tenancy. This means that a VNF could occupy the same compute, network, and storage resources as another VNF. In order to ensure proper security and performance of each VNF in the cloud, the cloud system must maintain separation of VNFs. In addition, NFs have multiple logical interfaces which require separation from each other either for security or performance reasons (Example: MME has S1-MME, S11, Gom, S6a and others). Network separation in
each case can be accomplished via several techniques, including, but not limited to, the use of affinity rules, open virtual switch use of vNICs, and networking techniques such as VxLAN. Different NFs might require different combinations of techniques in order to meet application requirements.

### 11.2.5 Security & Integrity

This section will discuss the security and integrity as it relates to the VNFs. Security of the underlying orchestration infrastructure will be covered in the security section of this document. Some of the security aspects to consider for the NFVs include:

- VNF User Security
- Logical Interface Security
- Guest OS Hardening
- Lawful Intercept
- VNF Package integrity

VNF user security contains the policies and procedures surrounding the handling of management users of the EPC element. It includes the creation/deletion of users, password management, and command group/level access.

Logical interface security encompasses the securing of the VNF logical interfaces and the data across them. This is typically handled today by a combination of access control lists and secure protocols such as IPSec as well as being supplemented by external firewall appliances in the network. It is also important for the cloud infrastructure to provide the necessary tenant separation so there are not any gaps in the security of the VNF interfaces.

In the virtualized world with hypervisors, a VNF is built upon x86 hardware, with a host operating system, a hypervisor, and then a guest operating system. This is portrayed in the Figure 11-1.

Security of the x86 hardware, guest operating system and hypervisor is the responsibility of the orchestration infrastructure; however, security of the guest operating system is the responsibility of the VNF which requires it. Hardening of the guest operating systems typically involves the removal or deactivation of services not necessary for the VNF as well as the use of secure protocols such as SSH where applicable. Detailed requirements for hardening of the guest operating system will likely vary between different operators.
Lawful Intercept is a mandatory requirement from 3GPP and law enforcement authorities in various countries. Lawful Intercept involves the sending of a request or warrant from the law enforcement agency to the relevant EPC elements. Those elements then make copies of the appropriate messages or user plane data and forward them to the lawful intercept server.

VNF Package integrity involves the process of ensuring the integrity of components of a VNF, from the VM image to the OVF. Methodologies used to ensure integrity of those packages could include the use of checksum matching and X.509 certificates.

11.2.6 APIs

In addition, there are a few primary API categories, which should be considered when virtualizing the 3GPP NFs. Rather than lock into specific APIs for a particular implementation, such as OpenStack for example, listed below are a few of the categories for consideration.

- Networking and Interfaces
- Compute
- Storage
- VIM Management

11.2.7 SDN Integration

Finally, in order to evolve the virtualized functions towards more of a fully automated implementation, automatic configuration of the logical interface paths (S1-MME, S5, Gx, etc) to neighboring nodes via SDN after VNF established should be considered. This could be accomplished via the implementation of the networking and interface APIs from the orchestration or management environment to the SDN controller. There are significant implications on the forwarding box and the message rate with the SDN controller, which need to be taken into account when considering tight integration to the SDN control network.
12 VNF considerations for NFVI

This chapter describes VNF design considerations. Specifically the following two aspects are covered.

a) Network services implemented with particular network functions, and service availability and reliability requirements can influence the choice of NFVI resources.

b) VNF design patterns influence the NFVI layer design and implementation, and factors like network service availability. VNF design and deployment methods need to be considered in light of the desired level of NFVI resource utilization, elasticity models, automation, agility etc. This section looks at VNF design and deployment patterns of VNFs, and finally lists a set of factors to consider when designing and placing NFVI (or VNFs).

12.1 Network Service Availability requirements and NFVI environment

A key design consideration for choosing the NFVI environment for any VNF is the associated network service availability & reliability requirements. These requirements influence decisions like desired VNF redundancy mechanisms, and locality, adjacency and availability requirements. Other factors like throughput and elasticity also need to be considered before a decision can be made about the environment for VNFs.

12.2 VNF design patterns

The OPNFV High Availability project has identified the following design patterns in VNFs and the following classification of VNFs is based upon this work (refer to:


- VNFs that provide in built in resiliency support for constituent components (VNFCs):
  - VNFs where each VNFC maintains its state information locally and keeps a redundant counterpart VNFC synced (ACT:STBY)
  - VNFs with more than one VNF Component, where VNFCs may individually keep state information and where more than one VNFC may (or may not) sync a counterpart component with the state information they maintain individually. In any case, this VNFC can take over functionality in case of fault conditions (M:1)

- VNFs that do not provide in built resiliency support:
  - **Stateful VNFs** – VNFs that may be comprised of single or multiple VNFCs, with the VNFC(s) keeping state but the overall VNF having a single point of failure (no redundant VNFCs inside)
  - **Stateless VNFs** – VNFs that may be comprised of single or multiple VNFCs with no individual state in the VNFC but having a single point of failure (no redundant built-in VNFCs)
The options described above can be summarized in Table 12-1 below:

<table>
<thead>
<tr>
<th></th>
<th>VNFs provide built-in redundancy</th>
<th>VNFs with no built-in redundancy</th>
</tr>
</thead>
</table>
| Stateful (at least one VNF component keeps state information – can be session driven state) | • VNF with Active: Standby VNFCs  
• M:1 redundant VNFCs | • VNF – No built-in redundant VNFC |
| Stateless (no VNF component keeps state information) | • M:1 redundant VNFCs | • VNF – No built-in redundant VNFC |

Table 12-1: VNF Redundancy Options

It should be pointed out that it VNFs with built-in redundancy (stateful or stateless) are by definition “complex” and comprised of more than one VNFC. VNFs with no redundancy can consist of more than one VNFC but may also be limited to single VM component depending on the VNF implementation. For purposes of considering VNFs from the design of NFVI or the deployment of VNFs on NFVI we thus simplify VNFs into the following three VNF design patterns:

- VNFs with Active/Standby Components (VNFCs)
- VNFs with M:1 Active Components
- VNFs with no redundancy (limited to a single component)

### 12.2.1 VNFs with Active Standby Components:

In this VNF design the VNF consists of >1 VNFC (VMs) and a given VNFC has a corresponding VNFC that can fully function in its place. Consider an example of a VNF with a total of 4 VNFCs: 2 Active and Standby VNFCs pairs.

- VNF1 Active & VFC1' (Standby)
- VNF2 Active & VFC2' (Standby)

A very high-level view of how an ACT:SBY VNF can be deployed is shown in Figure 12-2 below:
Internal and External VNF Networking:

The internal communication path between the VNFCs in this design needs to be fault resistant. The same applies to communication paths to EMS (e.g. a path may be needed to the EMS entities from all or a subset of the VNFCs). Similarly, the external traffic path should also be redundant, (i.e. paths to/from external entities to VNFC1 and VNFC1'). Very often these designs require the VNFCs to share an interface IP address with the active entity acquiring the same address at the time it becomes active.

Storage:

Typically in this design the VNFCs can be expected to keep state information locally and replicate it with their counterpart, so there may be no special needs from the storage network(s) to provide fast storage/recovery of state object information.

Scalability:

High performance may require the infrastructure to provide capabilities like SR-IOV, DPDK and NUMA vCPU pinning for these kinds of VNFs. Clearly, the redundancy requirements will also need to be met, so identical capabilities have to be provided in multiple platforms on which the VNFCs might fail.

Elasticity:

Scale-up and scale-down refers to increasing/decreasing capacity within the VNF (i.e. increase/decrease VNFCs or increase/decrease capacity VNFCs) and these types of procedures are possible with the ACT:SBY. Naturally, the former need to work within the affinity/anti-affinity constraints, and capacity needs to be available on all nodes when scaling up or scaling down is needed. This will typically require addition or removal of compute or memory resources available to a VNFC. Whether this can be done at runtime or requires something like a restart of the VNFC is implementation dependent.

Scale-in and scale-out refers to adding or removing instances at the VNF (as opposed to the VNFC) level. By adding/removing capacity in smaller steps it is possible to provide capacity elastically, but this comes
with the challenge that the traffic must be divided up between the available (currently deployed) instances of the VNF, which has to be done outside the VNF.

Resource Utilization:

In this VNF design up to 50% of the deployed resources may not be used at any given time. This applies to compute, memory and even networking links within and to/from the VNF. So if the VNFC are scaled too high then overall resource utilization may suffer, particularly if it is not straight forward to scale down a VNFC. Using a mix of scale-up/down and scale-in/out capability may present a more optimal design.

12.2.2 VNF design with M:1 redundancy for VNFCs

Overview:

In this VNF design the VNF consists of >1 VNFC (VMs) where for a given set of VNFCs a VNFC is a designated backup that can fully function in place of the one of the members of the set of VNFCs. Consider an example of a VNF with a total of 6 VNFCs:

- VNFC1, VNFC2 provide line card capability (one or both can be active)
- VNFCw, VNFC x, VNFC y and VNFC z provide higher layer processing functions (any 3 can be active)

A very high-level view of how an M: 1 VNF can be deployed is shown in Figure 12-3 below:

![Figure 12-3: VNF with Built-In M:1 Redundancy](image)

Networking:

The internal communication path between the VNFCs in this design needs to be fault resistant as any combination of 2 VNFCs from the set (L1,L2) and (w, x, y, z) may be active. Furthermore, the external traffic may be somehow divided up between the two VNFC sets (e.g. L1 and L2) and at times of failure one of them can take over any one of the other. This may have impact on the internal networking towards (w, x, y, z).
Storage:

Typically in this design the active VNFCs sets can be expected to keep state information locally and also replicate it with the designated counterpart (which then naturally keep state from more than one set). Depending on the amount of storage there maybe support required from the storage infrastructure to provide fast storage/recovery of state object information.

Scalability:

High performance may require the infrastructure to provide capabilities like SR-IOV, DPDK and NUMA vCPU pinning for these kinds of VNFs. Clearly, the redundancy requirements will also need to be met, so identical capabilities have to be provided for active and backup components.

Elasticity:

Scale-up and scale-down refers to increasing/decreasing capacity within the VNF (i.e. increase/decrease VNFCs or increase/decrease capacity VNFCs) and these types of procedures are possible with the ACT:SBY. Naturally the former need to work within the affinity/anti-affinity constraints, and further capacity needs to available when needed. The latter will typically require addition or removal of compute, memory resources available to VNFCs. Whether this can be done at runtime or requires something like a restart of the VNFC is implementation dependent.

Scale-in and scale-out refers to adding or removing instances at the VNF (as opposed to the VNFC) level. By adding/removing capacity in smaller steps it is possible to provide capacity elastically, but this comes with the challenge that the traffic must be divided up between the available (currently deployed) instances of the VNF.

Resource Utilization:

In this VNF design the utilization of deployed resources is higher than the active/standby model. This applies to compute, memory and even networking links within and to/from the VNF, but the traffic distribution between sets of VNFCs may not be simple and in some cases require internal or external load balancers. Using a mix of scale-up/down and scale-in/out capability may present a more optimal design.

12.2.3 VNF design with no built-in redundancy for VNFCs

Overview:

In this VNF design the VNF consists of one or more VNFCs (VMs) but there is no built-in redundancy inside the VNF. This does not, however, mean that redundancy cannot be provided; it’s just that it is not handled inside the VNF itself. Clearly this places requirements on the infrastructure and MANO stack. Here we consider the example of a VNF that is comprised of a single VNFC (VM).

In this example the requirements for networking within the VNF are nil. Redundancy and capacity can be provided by instantiating additional VNFs. Traffic may be equally distributed amongst the current deployed instances (each operating at say 40% capacity) and when an error happens it should be redistributed to the other instances, which until the fault is repaired will operate at 60% capacity. It is possible to build redundancy with anti-affinity rules for placement of the VNF. Since this kind of VNF design has no built-in fault detection capabilities, fault detection and recovery must be provided by the platform layers (NFVI and VIM). Further traffic handling capabilities are needed in the infrastructure to distribute the traffic; however, these may be needed anyway.
From a state point of view the information can be stored within the VNF, so loss of the VNF may result in some loss of state, but failure domains can be kept smaller.

A very high-level view of how a VNF with single component be deployed is below:

**Networking:**

The internal communication path in the VNF is purely within the VNF and no external networking services are needed. Furthermore, the external traffic may have to be divided up between the available VNF instances and at times of failure of one, say on HW1, the traffic can be taken over by any one of the other two.

**Storage:**

Depending on the amount of state information, support may be required from the storage infrastructure to provide fast storage/recovery of state object information.

**Scalability:**

High performance may require the infrastructure to provide capabilities like SR-IOV, DPDK and NUMA vCPU pinning for these kinds of VNFs. Clearly, the redundancy requirements will also need to be met, so identical capabilities have to be provided for active and backup components.

**Elasticity:**

Scale-up and scale-down refers to increasing/decreasing capacity within the VNF (i.e. increase/decrease VNFCs or increase/decrease capacity of VNFCs) and these types of procedures are possible. Whether this can be done runtime or requires something like a restart of the VNFC is implementation dependent.

Scale-in and scale-out refers to adding capacity by adding removing instances at the VNF (as opposed to the VNFC) level. By adding/removing capacity in smaller steps it is possible to provide capacity elastically, but this comes with the challenge that the traffic must be divided up between the available (currently deployed) instances of the VNF.
Resource Utilization:

Naturally in this VNF design the utilization of deployed resources is higher than the other models. This applies to compute, memory and even networking links within and to/from the VNF, but the traffic distribution between sets of VNFs requires smart infrastructure.

12.3 VNF deployment and NFVI connectivity considerations

Several VNFs may be involved in the traffic path or directly manage or control the traffic path. These VNFs are connected to each other to provide some network service. These interconnections between VNFs can be within a NFVI PoP or remote. These connectivity requirements are driven by service needs and availability requirements, which have to be balanced against requirements for elasticity, agility etc. Effective control and management of compute, storage and networking resources has a direct impact on the extent to which the SDN-NFV program can achieve its goals of increasing operational efficiencies, support business transformation and provide financial benefits.

Within an NFVI-PoP, Layer 2 (or VxLAN) connectivity is provided to VNFs for internal connectivity. The traffic to/from a VNF may have to go to another VNF and sometimes this may be based on various criteria. It is important that the networking between VNFs can take these factors into consideration.

VNFs connected across PoPs need to be load balanced and geo-redundant across PoPs. In these cases different configurations of WAN interconnects are needed at the NFVI-PoP and WAN boundary.

12.4 Factors to consider when determining platform needs related to VNF/VNFC

<table>
<thead>
<tr>
<th>VNFC/Workload characteristics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Availability requirements &amp; Remediation strategies of VNFs</td>
<td>Service availability drives fault detection times at both the VNF and VNF-C level. These can be:</td>
</tr>
<tr>
<td></td>
<td>- &lt; 1-5 second</td>
</tr>
<tr>
<td></td>
<td>- &lt; 1 minute</td>
</tr>
<tr>
<td></td>
<td>- or &gt; 1 minute?</td>
</tr>
<tr>
<td></td>
<td>VNF availability characteristics:</td>
</tr>
<tr>
<td></td>
<td>- 1:1 VNFCs</td>
</tr>
<tr>
<td></td>
<td>- M:1 VNFCs</td>
</tr>
<tr>
<td></td>
<td>- Simple VNF with no recovery capability</td>
</tr>
<tr>
<td>VM/VNF Failure detection and recovery</td>
<td>Fault detection and recovery capabilities of the VNF:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fault types detected:</td>
</tr>
<tr>
<td></td>
<td>- VNFC software</td>
</tr>
<tr>
<td></td>
<td>- Hypervisor or hardware</td>
</tr>
<tr>
<td></td>
<td>- Networking</td>
</tr>
<tr>
<td></td>
<td>- Storage</td>
</tr>
<tr>
<td></td>
<td>VNF in built fault recovery behavior vs. types of faults</td>
</tr>
<tr>
<td>VNFC/Workload characteristics</td>
<td>Platform and host recovery</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Networking strategies to deal with fault conditions:</td>
<td>Platform fault detection and recovery capabilities to reduce VNF fault recovery times:</td>
</tr>
<tr>
<td>- Floating IPs (Gratuitous ARP)</td>
<td>- Platform Fault and VM health detection but no built-in automatic recovery from VIM/NFVI:</td>
</tr>
<tr>
<td>- Virtual IPs (External Load balancers)</td>
<td>- Detection, recovery based on external software (VNFM)</td>
</tr>
<tr>
<td></td>
<td>- Sub section detection and recovery</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 12-5: Factors Impacting VNF/VNFC Platform Requirements**

ETSI has defined the information elements for various descriptors – VNF, NF, VL & VNFFG at a relatively high-level and require operators to refine these and define the specifics based on their design choices, e.g. OpenStack as the VIM, Linux/KVM in NFVI, etc.

Onboarding VNFs in a factory approach can enable the acceleration of the onboarding process and assist with validation and characterization of the VNF’s on Verizon’s platform. This approach may include the following steps:

1. Logical Design of the VNF (function definition)
2. Physical Design of the VNF (implementation definition)
3. Data Model definition (VNF Descriptor)
4. Creation of templates
5. Import the various templates
6. Testing of deployment Scenarios

As VNF’s are on-boarded and the details for each of these information elements are captured, the VNFs (and associated VNFC’s) can be mapped to the operating environment that provides the optimum capabilities.

12.5 Geo-redundancy Configurations for an Example VNF

Resiliency is a requirement for the Service Provider network. There are various aspects that should be considered as part of the architecture. The design of the VNF will determine the specific nature of the deployment but there are some foundational principles and practices that should be followed.

The VNF can be deployed in a few configurations. As an example if the IaaS function is being served by Openstack then below is a description of how various configurations can be setup. In a non-geo redundant scenario, a VNF can be setup in a specific tenant space. Within this tenant space the VNF can be deployed in a redundant mode. This is depicted in the Figure 12-5 where the example VNF has one VNFC. The VNF (ex: PGW) is deployed in Data Center A, on a specific NFVI/VIM, in Tenant Space A, as a VNF which has an active and standby VNFC. This means that the VNFCs will be redundant in either active-active or active-standby modes within this tenant space. This makes the VNF redundant within that tenant space.

As next step, the VNF can be deployed in two tenant spaces within the same NFVI/VIM. In this case the modes of the VNFCs (active-active etc.) will be determined by the specific VNF design. This configuration can ensure that the VNF is installed in separate compute pools ensuring a further level of infrastructure separation and redundancy. This is depicted in Figure 12-6 where the example active VNFC of the VNF (ex: PGW) is deployed in Data Center A, on a specific NFVI/VIM, in Tenant Space A, while the standby VNFC of the VNF (ex: PGW) is deployed in Data Center A, on the same NFVI/VIM, in Tenant Space B. This means that the VNFCs will be redundant in either active-active or active-standby modes between the two tenant spaces. This makes the VNF redundant between those two tenant spaces. This concept can be extended to Openstack availability zones as well.

Figure 12-5: VNF with Active:Standby VNFCs on same Tenant Space

Figure 12-6: VNF with Active:Standby VNFCs on different Tenant Spaces
The next step would be to deploy the VNF in two tenant spaces BUT under separate VIMs. Similarly in this case the modes of the VNFCs (active-active etc.) will be determined by the specific VNF design. This configuration can ensure that the VNF is installed in separate compute pools ensuring a further level of infrastructure separation and redundancy. This is depicted in Figure 12-7 where the example active VNFC of the VNF (ex: PGW) is deployed in Data Center A, on a specific NFVI/VIM, in Tenant Space A, while the standby VNFC of the VNF (ex: PGW) is deployed in Data Center A, on a different NFVI/VIM, in Tenant Space C. This means that the VNFCs will be redundant in either active-active or active-standby modes between the two tenant spaces as well as be on different NFVI/VIM ensuring some physical separation and resource management. This makes the VNF redundant between those two tenant spaces and infrastructure spaces.

A fully geo-redundant scenario is that the VIM could be in separate Data Centers. This is depicted in Figure 12-8 where the example active VNFC of the VNF (ex: PGW) is deployed in Data Center A, on a specific NFVI/VIM, in Tenant Space A, while the standby VNFC of the VNF (ex: PGW) is deployed in Data Center B, on a different NFVI/VIM, in Tenant Space X. This means that the VNFCs will be redundant in either active-active or active-standby modes between the two tenant spaces as well as be on different NFVI/VIM ensuring some physical separation and resource management. Being in different Data Centers also ensures geographical separation. This makes the VNF geo-redundant.
A further redundancy scenario is one in which an instance of a VNF resides in one Data Center and another instance of the VNF resides in a different Data Center.

The VNFs may be configured to support one of several redundancy schemes:

- **Active / Standby Cold (No State Replication)** – The active VNF processes all service requests. Upon failure, the standby VNF becomes active, taking over processing for all service requests. At switchover, the state of the active service requests are not preserved, they are torn down and reestablished on the standby, now active, VNF. A failure detection mechanism is required to detect the failure of the active VNF. The instantiation of a switchover event depends on the type of failure as defined by the VNF redundancy policy. For example, the failure of a single VNFC may not necessarily trigger a VNF switchover.

- **Active / Standby Hot (State Replication)** – The active VNF processes all service requests. Upon failure the standby VNF becomes active and takes over processing for all service requests. The state of the active service requests are preserved and maintained across the switchover. In addition to a failure detection mechanism, a mechanism is required to synchronize the state of the active service requests from the active VNF to the standby VNF. The mechanism for state
synchronization should also monitor the role of the VNFs to determine whether the standby VNF should become the active VNF. Not all VNF failures may trigger a switchover.

- **Active / Active** – In this redundancy scheme all VNFs are active and each process a percentage of all service requests. This configuration reduces stranded resources. It allows for the functionality of the standby VNF to be verified, before an actual switchover event occurs. Service requests can temporarily burst above the resource capacity of any of the individual VNF, while additional resources and VNFs are provisioned.

![Diagram of VNF with Active:Standby VNFCs in Different Tenant Spaces, Different NFVI/VIM, Different Data Centers](image)

Figure 12-10: VNF with Active:Standby VNFCs in Different Tenant Spaces, Different NFVI/VIM, Different Data Centers

In an Active/Active configuration, service requests may be distributed across active VNFs spanning multiple data centers. Each VNF should be allocated enough resources to support 100% of the total service requests, even though in steady state it will only process a percentage of the total load. A load-balancing method is required to distribute service requests across the active VNFs. It could be manually provisioned, utilize round robin allocation or some other algorithmic allocation method.

Within this configuration there are additional considerations for redundancy:

- a. **No State Replication** – Service requests are not preserved during switchover. They are torn down and must be reestablished on a VNF capable of supporting the service request load.

- b. **State Replication** – Service requests are synchronized and updated between all active/active VNFs. Upon failure, service requests are switched to another VNF, capable of handling the load, without service interruption.

In either case, not all VNF failures would trigger a switchover. Policy should be defined, indicating which individual failure or group of failures, are considered a failure event that triggers a redundancy switchover.
13 Reliability

Introduction

With traditional, non-virtualized network elements, hardware, software and backplane are bundled as one entity. The vendor guarantees performance and reliability of the entire system. With the introduction of virtualization that model changes. Hardware and software are acquired from different vendors and the various VNF components, residing on different servers, are connected via a network of switches from yet another vendor.

In the new model, the operator is ultimately responsible for assuring end-to-end reliability and performance. This Chapter discusses the issue in more detail.

Better or worse?

The disaggregation of hardware and software definitely complicates matters for the operator. The next section lists several issues that determine end-to-end reliability. The fact that there are so many moving parts may give the impression that virtualization has a negative impact on reliability. The opposite is likely to be true, though. Due to the automation of lifecycle management, fast service restoration after failures may actually increase reliability. That said, the industry will have to go through a learning phase where operators and vendors need to work together to understand the interworking and dependencies between the various aspects.

A fail-over scenario

Before this Chapter addresses the factors that affect reliability, consider a typical fail-over scenario. Note that this is just one of many possible implementations.

A particular critical VNFC, e.g. the VNFC that provide data plane functionality in a vPGW, will in many implementations be instantiated in a 1+1 configuration. That means that during VNF instantiation, two copies of the VNFC were created where one is active and the other standby. The two VNFCs may have access to a common data base where state information is stored, or the active VNFC may update the standby VNFC whenever a state change occurs. In any case, at any given point in time, the standby VNFC is ready to continue where the active one left off.

To check for failures, there is usually some form of life-check between the two VNFCs. For example, in case very fast failure detection is required, the two VNFCs may send each other heartbeat messages every 10 ms. When the standby VNFC does not receive three consecutive heartbeat messages, it considers the active one failed and it becomes active itself. In this case, service is restored within 50 ms after a failure.

At this point, though, the now active VNFC is unprotected. It is necessary to restore the failed VNFC so that it can now become the standby to protect its partner. Restoration is performed by the VNFM, which will receive alarms related to the VNFC failure (either from the now active VNFC, from the VIM which may have detected a server failure that cause the VNFC to fail, or from other elements that may have detected the failure). In response, the VNFM interfaces with VIM to create a new VNFC instance.
Factors that influence reliability

This section lists factors that influence reliability.

- **Server and hypervisor failures**
  
  Server and hypervisor failures render a VNFC inoperable. In general, VNFs will have protection mechanisms in place to switch over to a standby VNFC. However, the main cause of unavailability is often not the initial failure, but the failure of the subsequent switch-over procedure. Switchovers generally happen within seconds and contribute little to overall unavailability, but a failed switch-over impacts service for a much longer time, generally until service is restored manually.

  Robustness of the physical infrastructure is a key ingredient in NFV reliability.

- **Network failures**
  
  Network failures impact the communication between VNFCs or between a VNF and other VNFs. In general, network connections between servers are protected and network failures therefore generally have less impact than server failures. Nevertheless, reliability of the network infrastructure is a second key ingredient in NFV reliability.

- **VNF reliability**
  
  VNF reliability is the third main pillar of NFV reliability. VNF software may have bugs that result in VNF unavailability.

- **Management and control systems**
  
  VNFM and VIM are involved in restoring a failed VNFC. The VNFM is generally responsible for triggering the creation or restoration of a failed VNFC. VIM subsequently executes those actions. If either the VNFM or the VIM is unavailable, restoration will fail and as a consequence, the VNF will temporarily not have the required protection level. For example, if a particular VNFC is protected based on 1+1 redundancy, the failure of one of the two VNFCs will leave the VNF in an unprotected state. A subsequent failure of the second VNFC may render the VNF unavailable.

  VNFM and VIM availability are important for the overall NFV reliability.

- **Speed of VNFC restoration**
  
  The speed of VNFC restoration is related to the previous point. As long as a failed VNFC has not been restored, the VNFC is in an unprotected state. Even if VNFM and VIM are available, the entire procedure to detect a failure, allocate a new server, create the VM, download and bootstrap the software image etc. may take a while.

  That said, in non-virtualized networks, it could take hours before the operations crew had swapped a failed circuit pack for a new one. In NFV, full restoration of a VNFC is a matter of a few minutes or less. Whether VNFC restoration is completed in a few seconds or a few minutes is therefore not expected to have a significant impact on NFV reliability.

---

2 This section often uses text that suggests that some form of 1+1 protection is being used. Of course, there are many other ways to achieve redundancy and increase reliability. The text in this section should be interpreted liberally to cover those alternative implementations as well.


- **Latency between VNFCs**

To enable a switchover to a redundant VNFC in case of a failure, a standby VNFC needs to have a way to detect failure of the active one. This generally involves some form of life-check message. The time that it takes to detect a failure depends on the frequency of such messages, but also on the latency between the VNFCs. More latency results in a higher unavailability. Of course, latency is expected to be in the order of milliseconds, so the impact on overall NFV reliability is likely to be small.

- **Failure domains**

In NFV, multiple VNFCs, belonging to different VNFs may reside in the same “failure domain”, i.e. the group of VNFCs impacted by a single failure. This may impact overall reliability in ways that did not exist in the non-virtualized world. For example, in traditional networks, double failures are considered extremely rare. Therefore, in the case of two interfacing network elements, A and B, the protection mechanisms implemented in A dealt with failures in A and assumed that B was stable and not going through a protection switching action at the exact same time (unless A and B were both reacting to a failure in the link between them). In a virtualized world, the probability that multiple VNFs are going through a protection switching procedure at the exact same time increases. This may cause instability and potentially result in unavailability.

In many data center implementations, a single failure will bring down at most one server. In that case, the impact on overall reliability is small, since only a handful of VNFCs reside on that one server. However, if a single failure could take down an entire rack of servers, the situation becomes different. It is very well possible that each VNFC impacted by a rack failure is protected by VNFCs residing in other racks, but the simultaneous execution of tens or hundreds of protection switching actions at the same time significantly increase the probability of subsequent failures and could result in unavailability of a VNF.

**Structured Process**

The industry will have to go through a learning process with respect to NFV reliability. The following steps could be considered by an Operator to structure that process.

1. Quantify the robustness of the infrastructure.
   - Expected failure rates of servers and hypervisors
   - Expected failure rates of network connections
   - Expected latency in a fully loaded infrastructure?
   - Are there failures that could bring down more than a single server at a time?

2. Assess and test the reliability of critical control systems:
   - Expected availability of VNFM, VIM, SDN Controllers

3. Based on the quantified failure rates and control system availability, vendors can assess the theoretical availability of their solutions. For example, such an analysis could reveal that 6:1 redundancy is inadequate and that 4:1 redundancy is called for. Similarly, such an analysis could reveal that latency between VNFCs is a critical parameter and needs to be bounded to, say, 50 ms.

4. Test:
   - Run VNFs on the actual infrastructure. Insert failures and verify that all fail-over procedures work.
   - Run the network for long periods of time and verify that the assumed availability rates hold. Etc.
14 IMS Functions

IMS architecture has basic requirements, which guide how it has been created and how it should evolve. The following functions define the baseline for the IMS architecture:

- IP connectivity and roaming
- IP Multimedia Sessions
- Access independence and Layered Design
- Quality of Service and IP Policy Control
- Service Control
- Interworking with other networks
- Charging
- Secure communication

Figure 14-1 below shows IMS control plane elements connected via dotted lines and user plane elements connected via solid lines:

Figure 14-1: IMS Architecture as shown in 3GPP TS 23.002
14.1 Functions

IMS entities and key functions can be roughly classified into five main categories:

- Session management and routing (CSCF)
- Services (Application server, MRFC, MRFP, MRB)
- Interworking functions (BGCF, MGCF, IMS-MGW)
- Support functions (IBCF, TrGW, IMS Access Gateway, SEG, LRF, HSS, AAA, SDM)
- Charging (see Chapter 18)

Session management and routing

There are four different kinds of Call Session Control Functions (CSCF): Proxy-CSCF (P-CSCF), Serving-CSCF (S-CSCF), Interrogating-CSCF (I-CSCF) and Emergency-CSCF (E-CSCF). Each CSCF has its own special tasks, but common to P-CSCF, S-CSCF and I-CSCF is that they all have a role during registration and session establishment and form the SIP routing machinery. Moreover, all functions are able to send charging data to an offline charging function.

Services

IMS services consist of Multimedia Resource Function Controller (MRFC), Multimedia Resource Function Processor (MRFP), Media Resource Broker (MRB) and Application Server (AS). Application Servers are entities that provide value-added multimedia services in the IMS, such as Telephony, Presence and Supplementary Services. MRFC and MRFP together provide mechanisms for bearer-related services such as conferencing, announcements to a user or bearer transcoding in the IMS architecture. MRB supports the sharing of a pool of heterogeneous MRF resources by multiple heterogeneous applications. The MRB assigns specific suitable MRF resources to calls when being addressed by S-CSCF or AS.

Interworking functions

Interworking functions are needed to enable voice, video and SMS interworking between IMS and the CS core network. For breaking out the S-CSCF sends a SIP session request to the Breakout Gateway Control Function (BGCF), which further chooses where a breakout to the CS domain occurs. BGCF selects a Media Gateway Control Function (MGCF) to handle the session towards CS network. The MGCF also controls the IMS Media Gateway (IMS-MGW) which provides the user-plane link between CS networks and the IMS.

Support functions

Several functions can be categorized as support functions such as Interconnection Border Control Function (IBCF), Transition Gateway (TrGW), IMS Access Gateway, Security Gateway (SEG), Location Retrieval Function (LRF), Home Subscriber Server (HSS), Authentication Authorization and Accounting (AAA), and Subscriber Data Management (SDM).

Below is the list of IMS functions, which can also be shown as standalone VNFs:
Table 14-2: IMS Functions and Standalone VNFs

<table>
<thead>
<tr>
<th>Function</th>
<th>Control Plane</th>
<th>User Plane</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-CSCF (Proxy-Call Session Control Function)</td>
<td>√</td>
<td></td>
<td>SIP/SDP, Diameter</td>
</tr>
<tr>
<td>S-CSCF (Serving-Call Session Control Function)</td>
<td>√</td>
<td></td>
<td>SIP/SDP, Diameter</td>
</tr>
<tr>
<td>I-CSCF (Interrogating-Call Session Control Function)</td>
<td>√</td>
<td></td>
<td>SIP/SDP, Diameter</td>
</tr>
<tr>
<td>TAS (Telephone Application Server)</td>
<td>√</td>
<td></td>
<td>SIP/SDP, Diameter, XCAP</td>
</tr>
<tr>
<td>PS (Presence Server)</td>
<td>√</td>
<td></td>
<td>SIP/SDP, Diameter, XCAP</td>
</tr>
<tr>
<td>SBC (Session Border Controller)</td>
<td>√</td>
<td>√</td>
<td>SIP, SIP-I, H.323, MGCP, H.248</td>
</tr>
<tr>
<td>SCG (Service Continuity Gateway)</td>
<td>√</td>
<td></td>
<td>SIP, MAP, Diameter, Camel</td>
</tr>
<tr>
<td>VMS (Voice Mail Server)</td>
<td>√</td>
<td></td>
<td>SIP/RTP</td>
</tr>
<tr>
<td>MRF-P (Media Resource Function Processor)</td>
<td></td>
<td>√</td>
<td>RTP/RTCP, H.248</td>
</tr>
<tr>
<td>MRF-C (Media Resource Function Controller)</td>
<td>√</td>
<td></td>
<td>SIP/SDP, XML, H248</td>
</tr>
<tr>
<td>MGW (Media Gateway)</td>
<td></td>
<td></td>
<td>RTP, TDM, H.248</td>
</tr>
<tr>
<td>IMS PBX (IMS Private Branch Exchange)</td>
<td>√</td>
<td>√</td>
<td>SIP, H243, RTP</td>
</tr>
<tr>
<td>IP-SM GW (IP Short Message Gateway)</td>
<td>√</td>
<td></td>
<td>SIP, MAP, Diameter</td>
</tr>
<tr>
<td>RCS/CPM (Rich Communication Services/Converged IP Messaging)</td>
<td>√</td>
<td></td>
<td>SIP, MRSP</td>
</tr>
<tr>
<td>DSC/DRA (Diameter Signal Controller/Diameter Routing Agent)</td>
<td>√</td>
<td></td>
<td>Diameter</td>
</tr>
<tr>
<td>WebRTC GW (RTC-Real Time Communication)</td>
<td>√</td>
<td>√</td>
<td>WebRTC, SIP/RTP</td>
</tr>
<tr>
<td>IP SMSC (Short Message Service Center)</td>
<td>√</td>
<td></td>
<td>SMPP, MAP</td>
</tr>
<tr>
<td>EIR (Equipment Identity Register)</td>
<td>√</td>
<td></td>
<td>Diameter</td>
</tr>
<tr>
<td>MMSC (Multimedia Message Service Center)</td>
<td>√</td>
<td></td>
<td>WAP, HTTP</td>
</tr>
</tbody>
</table>

IMS User Plane

For user plane functions three different options can be consider: Traditional, VNF-based or SDN-based. For all options the H.248 protocol is used between the control plane and user plane.

Since media over packet functions require very low latency and jitter, support in software has been challenging because of the I/O structure of a traditional user space and operating system design where interrupts, process scheduling and shared packet I/O creates latency and jitter. The traditional software design is also inefficient in terms of CPU usage for media processing. However, as described in section 3.2, X.86 technology (e.g., SR-IOV with VF queues assigned to media) and changes in the OS could provide acceptable quality and economical performance for some media functions (e.g., basic packet forwarding, NAT, firewall). However, some media plane processing (e.g. transcoding) remains a challenge to achieve acceptable latency and achieve economical performance and may require acceleration hardware. The table below compares some attributes of the traditional (ASIC) and software (NFV) options.
Requirements for mass market services such as VoLTE / Residential / Fixed VoIP, PSTN / CS interworking and Enterprise customer services might be different based on the number of subscribers and degree of customization.
<table>
<thead>
<tr>
<th>User Plane function</th>
<th>Traditional</th>
<th>VNF-based</th>
</tr>
</thead>
</table>
| **SBC**             | Combined or separated options possible. In separated option:  
  - Standardized H.248 interface between the Control plane and the User plane.  
  - Independent scaling of BCF and BGF HW.  
  - Different locations for the BCF and the BGF deployment.  
  Optimized hardware for media functions:  
  - DSP for Media Processing.  
  - Packet Processor for user plane interfaces.  
  - x86 CPU for general processing and control plane. | Combined or separated options possible. In separated option:  
  - Standardized H.248 interface between the Control plan and the User plan (H.248).  
  - Independent scaling of BCF and BGF HW.  
  - Different locations for the BCF and the BGF deployment.  
  Optimizations for media functions:  
  - Acceleration by DPDK fast path technology.  
  - SR-IOV (Single Root I/O Virtualization) to provide a short cut from VM to physical NIC.  
  - Latest generation CPUs, and SW components optimized for media handling. |
| **MRF-P**           | Standardized H.248 interface used between the Control plane and the User plane. Transcoding is supported for normal calls, emergency calls, early media, LTE – LTE calls and for PSTN breakout or break-in calls.  
  Active and Re-Active Mode Transcoding alternatives:  
  - Re-active mode: The MRF is only included in the call after a codec mismatch has been detected.  
  - Active mode: The MRF is included in every call based on rules in Transcoding Control. | Combined or separated options possible. In separated option:  
  - Standardized H.248 interface between the Control plan and the User plan (H.248).  
  - Different locations for the MRF-P (MRF) and control plane to optimize media path.  
  Optimizations for media functions:  
  - Acceleration by DPDK fast path technology.  
  - SR-IOV (Single Root I/O Virtualization) to provide a short cut from VM to physical NIC.  
  - Latest generation CPUs, and SW components optimized for media handling. |
## User Plane Functions - Traditional vs. VNF

<table>
<thead>
<tr>
<th>User Plane function</th>
<th>Traditional</th>
<th>VNF-based</th>
</tr>
</thead>
</table>
| **MGW**             | Standardized H.248 interface used between the Control plane and the User plane. Many functions can run on x86 bare metal. Media processing and speech enhancements done on DSP. | TDM support. Optimizations for media functions:  
  - Acceleration by DPDK fast path technology.  
  - SR-IOV (Single Root I/O Virtualization) to provide a short cut from VM to physical NIC.  
  - Latest generation CPUs, and SW components optimized for media handling. Speech centric application services like MGW and BGW gets benefits of optimizations on IPP speech/audio codecs in CPU architecture. |
| **IMS PBX**         | Low number of subscribers. | |
| **WebRTC GW**       | WebRTC eP-CSCF functionality used for signaling interworking and eIMS-AGW functionality for the user plane interworking. Standardized H.248 interface used between the Control plan and the User plan. | WebRTC GW has additional requirement for video transcoding. |

Table 14-3: User Plane Functions - Traditional vs. VNF
14.2 Interfaces/APIs

The IMS architecture figure (shown in the beginning of this chapter) illustrates reference points for IMS. Those interfaces will be the same on the PNF and the VNF deployments, but SDN-NFV will introduce new interfaces, which are described in more detailed elsewhere in this document.

<table>
<thead>
<tr>
<th>CSCF Interface</th>
<th>CSCF Peer Function</th>
<th>SDN/NFV aspects (changes from traditional interfaces and protocols)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ve-Vnfm</td>
<td>VNFM</td>
<td>VNF lifecycle management, Exchanging configuration information, State information for network service lifecycle management</td>
</tr>
<tr>
<td>Vn-NF</td>
<td>NFVI</td>
<td>Virtual Machine (VM) interface which is the execution environment of a single VNF</td>
</tr>
<tr>
<td>Or-Vnfm</td>
<td>Network Orchestrator</td>
<td>Interface is between CSCF VNF and NO</td>
</tr>
<tr>
<td>Gm</td>
<td>UE</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>FIS</td>
<td>FIS Server</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>ISC</td>
<td>TAS, SMSC, RCS</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>Mw</td>
<td>CSCF</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>Mj, Mg, Mk</td>
<td>I-SBC/MGCF</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>Cx</td>
<td>DRA/HSS</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>Rx</td>
<td>PCRF</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>Rf</td>
<td>CCF/GGF</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>DNS</td>
<td>DNS Server</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>ENUM</td>
<td>ENUM Server</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>FEE ISC</td>
<td>CSCF</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>FEE Mr'</td>
<td>MRF</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>FEE DNS</td>
<td>DNS Server</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>NTP</td>
<td>NTP on TIAMS</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>OAM</td>
<td>VNF-EMS</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>Cluster Interconnect</td>
<td>CSCF peer node</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>Installation, Admin</td>
<td>TIAMS, HP Admin Tools</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>X1</td>
<td>SS8 LIMS</td>
<td>Unchanged from PNF Implementation</td>
</tr>
<tr>
<td>X2</td>
<td>SS8 LIMS</td>
<td>Unchanged from PNF Implementation</td>
</tr>
</tbody>
</table>

Figure 14-4: CSCF Interfaces, Peer Functions and SDN/NFV Aspects

14.3 Functional Evolution

The Core network is getting more complex as there will be more and more elements in the network. Billions of devices are coming, which introduces new requirements for scaling, cost, robustness and management mechanisms. Services are converging and they will be used in this multi-device environment. It is also clear that service introduction must become faster, more agile and more cost efficient with lower TCO/OPEX.
With the migration of the IMS Core to a NFV-SDN cloud-based environment, there are opportunities to further enhance the solution. Unlike EPC elements, there is limited impact of the control-plane and user-plane split (with the minor exception of media gateway functions), but there are opportunities to address other areas. Specifically:

1. Decreasing time to market (TTM) for new features, patches and fixes
2. Network simplification

Specific to decreasing TTM, there is an opportunity for adopting a more flexible and nimble model of delivering features and fixes. Leveraging the concepts pioneered in the enterprise and web environments means reorganizing the software to become more “DevOps” aligned to facilitate the more rapid and continuous deployment of features and functions. The key to this potential area of improvement is taking full consideration of the end-to-end delivery value proposition in the Verizon environment. Modifying a software element without ensuring the ability to properly hand it off into the network creates minimal value (issues to be considered include automated test procedures, operations hand-offs, care contract language, etc).

To address network simplification, there is the opportunity to identify “platform” services embedded in various NFV functions (not limited to CSCF) and pulling them out of the larger software model to allow them to be unified and/or managed separately from the application services that are specific to the functional unit. A simple example of this is the database function that is common to many of the Telecom platforms today – this could be offered as a platform service in the cloud to avoid a duplication of effort and licensing issues associated with embedding it in a larger function. There are many other such “platform services” that can be identified and explored for separation.

Decomposition is one direction for IMS application evolution. Decomposition can be divided into two parts; IMS Application Services and IMS Platform Services. Decomposition offers reduced feature testing efforts since only specific services need to be tested. Also, this supports faster CSCF feature deployment since only specific services need to be upgraded, e.g. only the Transcoding CSCF Application, or only the 911/NENA CSCF Application is upgraded.

This model also brings requirements like Multi-Service hosting VMs, which are needed to reduce the VM induced overhead. The software delivery process has to become agile and continuous, and the software testing/management process has to become automated for the VNFs.

The figure below shows an example of IMS application services and platform services:
VoLTE/IMS e911 and CALEA requirements are done at the application layer with related security implementations like IPSec etc. Lawful interception can be separated using VLANs or optionally with additional Lawful Intercept Virtual Machine Network Interface Cards (NICs).

14.3.1 Platform and Application Services

When exploring the decomposition of an IMS core, or any other network function, there are two distinct types of functions that are typically identified. Those that are unique to the function and that give it the unique capability set are called application services. For an IMS core, examples of such application services include (but are not limited to):

- ATCF
- P-CSCF
- S-CSCF
- IBCF
- BGCF

In addition to these application services, there are common functions, which typically are imbedded in many other similar network functions, which may be good candidates to be provided by the platform. Examples of platform services include (but are not limited to):

- DNS Handler
- Diameter Load Balancer
- SIP Load Balancer
- Database

The key in examining the value of separating out platform services is understanding the performance requirements of the application and the SLAs offered by the Platform as a Service (PaaS). It may not
always be the correct answer to fully decompose every function. In particular, protection schemes such as geo-redundancy may be impacted by decomposition.

14.3.2 Voice Applications (Enterprise and Consumer)

For various voice services, one or more Telephony Application Servers (TAS) can be deployed on the IMS core. In the IMS architecture figure above, the AS or Application Server is an example of a primary VoLTE TAS server. This TAS is either deployed as a PNF or a VNF depending on the situation. Additional TAS servers can be added to provide unique enterprise services, video services or additional types or classes of consumer voice services. When the IMS client attaches to the network, the user profile in the HSS will indicate which type of AS should be utilized. The benefit of deploying VNF-based TAS platforms is that small, private enterprise solutions can easily and cost effectively be added to the network offering, and the TAS could even be hosted in the Enterprise Cloud domain by Verizon Enterprise Solutions.

14.4 Virtualization and SDN Considerations

IMS VNF includes a variety of Virtual Machines (VM’s) that provide the CSCF and other services. A single CSCF VNF may support millions of subscribers and larger deployments will require additional CSCF VNFs. Every scalable VM (CSCF VM) supplies its capacity utilization information to the Elasticity Manager (EM), and if the EM recognizes that the existing VMs of the Deployment unit are running out of capacity, or the resources are not optimally utilized, it could decide to scale out/in the CSCF VM. After scale-out the CSCF VM installs & configures itself; it does a dynamic registration of itself with the CSCF load distributor. After scale-in the CSCF VM pair(s) de-registers itself from the CSCF load distributor and the load distributor moves the subscriber pools from the scaled-in VM pair(s) to the remaining VMs.

The IMS functions have certain transport infrastructure requirements, e.g. RTT delay, bandwidth, VLAN support, DSCP support, etc. that are assumed to be supported in an SDN controlled network in a similar manner as today.

When application cluster elements are located on different servers the RTT delay should be below 10ms.

Bandwidth requirements depend on the traffic model and are operator specific, but typically control plane traffic is at the Gbit/s level.

VLANs are used for traffic separation in all IMS deployments. Some operators also require physical interface-based traffic separation.

DSCP Marking and Traffic Prioritization is done to differentiate VoLTE signaling traffic from other traffic (e.g. DSCP=32 can used for signaling). DSCP can also be used to indicate a terminating VoLTE call to the Serving GW, which can prioritize a different paging profile for voice calls and other IMS services like SMS.

IMS application Virtual Machines (VMs) are managed by the VNFM. Below is an overview of CSCF applications in separated Virtual Machines.
High availability support is complemented by the fast repair capabilities of the NFVI. Based on the VNFD of the application, the VNFM can instantiate the application or VNF instances with different redundancy, for example, 1+1 redundancy on different servers. When a server outage occurs, the Application reacts by switching the service to the standby VM instance, which ensures carrier-grade service availability.

The NFVI responds to the server outage by detecting the lost VMs and by automatically restarting replacements for the lost application instances. This is performed within minutes and restores the capacity and application HA.

Live Migration allows relocation of applications/VMs in live operation (e.g. a CSCF with active Registrations/Sessions) from one server to another server without service disruption. This is basically an on-the-fly snapshot / cloning, including processes, memory content (even storage, if needed) and is performed without service disruption. Live Migration is used for planned maintenance work, e.g. when a server needs to be upgraded/replaced.
15 EPC Functions

15.1 Functions

The Evolved Packet Core (EPC) architecture consists of a number of core functions all interconnected using an IP infrastructure to provide packet data services to a Long Term Evolution (LTE) Radio Access Network (RAN). These functions will be present in both the traditional and virtualized world. The core functions include the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Control Plane</th>
<th>User Plane</th>
<th>Signaling</th>
<th>Session Awareness</th>
</tr>
</thead>
<tbody>
<tr>
<td>MME</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SGW</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGW</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PCRF</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>HSS</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Femto Gateway</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ePDG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BM-SC</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>eMBMS Gateway</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Figure 15-1: EPC Functions

Definitions used in table:

- **Control Plane** - Communications that provides the transfer of control and management information
- **User Plane** - Communications that provides the transfer of user data, along with associated controls
- **Signaling** - The exchange of information specifically concerned with the establishment and control of connections, and with management, in a telecommunications network
- **Session awareness** - A network function that keeps the state of a session or subscriber

Following is a short description of each functional element. For a more detailed list of the functions these elements perform, please refer to 3GPP TS 23.401 or 23.402.

**MME**

From a Core Network perspective, the MME is the main node for control of the LTE access network. It acts as the primary signaling node in the EPC network and is principally responsible the mobility management and authentication of the UEs.
SGW

The SGW is responsible for terminating the User Plane interface towards the E-UTRAN. The SGW is assigned to a particular UE by the MME based on the E-UTRAN tracking area location information. Each UE is assigned to only one SGW for the forwarding of end-user data packets.

PGW

The PDN GW provides connectivity to external PDNs for the UE, functioning as the entry and exit point for the UE data traffic and is assigned to a particular UE bearer by the MME based on APN name resolution. A UE may be connected to more than one PDN GW if it needs to access more than one PDN. The PDN GW also allocates an IP address to the UE. In addition, the Policy and Charging Enforcement Function (PCEF) is typically integrated into the PGW implementation.

PCRF

The Policy and Charging Rules Function encompasses the policy control decision and flow-based charging control functionalities. It provides network-based control related to service data flow detection, gating, QoS and flow-based charging towards the PCEF.

HSS

The Home Subscriber Server manages user’s subscription information and the associated logical procedures in the EPC network.

Femto Gateway

A Femto Gateway or Home eNodeB gateway acts as an S1 signaling aggregation point for small cells in an LTE network. Multiple small cells connect to this single gateway, which then presents a single S1 interface towards the core network. Femto Gateways can also provide the ability to offload UE traffic directly via the use of the Local IP Access (LIPA) and the Selected Internet IP Traffic Offload (SIPTO) functionalities.

ePDG

The ePDG allows for the convergence between the WiFi access domain and the LTE domain. It enables seamless mobility between WiFi and LTE via the S2b interface to the PGW.

Broadcast-Multicast Service Center (BM-SC)

The BMSC handles eMBMS sessions (start, stop) and to deliver user plane media to the MBMS-GW. The BMSC provides a number of service layer functions such as Service Announcement and Forward Error Correction and File Repair.

eMBMS Gateway

The eMBMS Gateway provides functionality for sending/broadcasting of MBMS packets to each eNB transmitting the service. The MBMS-GW uses IP Multicast as the means of forwarding MBMS user plane data to the eNB's. The MBMS GW performs MBMS Session Control Signaling (Session start/stop) towards the E-UTRAN via MME.
15.2 Interfaces

The communication between different elements in the EPC is outlined by 3GPP. The 3GPP standards assign interface label names for the logical connections between those elements. The figure below provides a pictorial representation of those interfaces.

Figure 15-2: EPC and CDMA Interworking Diagram.

For a complete description of each of the 3GPP logical interface, see the applicable 3GPP specification.

NFV-MANO interfaces from the NFV to the orchestration and management domain are addressed in Chapter 4.

15.3 Functional Evolution

15.3.1 Introduction

Migrating EPC nodes to a NFV deployment model as VNFs enables a multitude of benefits already with existing 3GPP network element definitions. It is possible to leverage independent scaling of control and user plane functions, and to benefit from more dynamic life cycle management processes. An additional opportunity is grouping of multiple VNFs into a single aggregated VNF deliverable, realized as a network slice. This enables simplified interop testing and opens up possibilities for optimizing these network slices for different service categories.

Another possibility when moving to NFV is to decompose the network functions as specified by 3GPP. This would provide increased flexibility by addressing exact network needs but those benefits must be balanced against the potential complexity of managing many smaller components and associated interop testing.
The 3GPP EPC functional architecture is under constant evolution to adapt to new service requirements and the emerging opportunities with a NFV deployment model as discussed above. Some studies worth mentioning are DECOR (Dedicated Core Network), which is part of 3GPP Rel 13 and others just starting up like eDECOR and CUPS (Control and User Plane Separation), targeting initial results by mid-2016.

In addition, there are ongoing pre-standardization activities that address the core network architecture in a 5G context with its vast requirements space supporting a range of different use cases. NGMN has released a 5G white paper outlining both business and technical requirements of future 5G networks. Also 3GPP SA1 has started work on 5G.

3GPP Dedicated Core Networks (DECOR), TR 23.707

http://www.3gpp.org/ftp/specs/archive/23_series/23.707/23707-d00.zip

3GPP enhanced Dedicated Core Networks (eDECOR)

http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_110_Dubrovnik/Docs/S2-152661.zip

3GPP Control and User Plane Separation (CUPS)

http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_110_Dubrovnik/Docs/S2-152709.zip

NGMN 5G white paper

https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf

The following sections outline different aspects of the above topics for deploying EPC in a NFV environment.

15.3.2 Network Slicing

With the continued increase in MBB traffic driven primarily by video in combination with new market segments and verticals like IoT and emerging 5G use cases, a more modular approach to the 3GPP core network architecture needs to be evaluated. The new use cases are expected to come with a high diversity of requirements on the network. For example, there will different requirements on functionality such as charging, policy control, security, mobility etc for different use cases. Some use cases such as MBB may require e.g. application specific charging and policy control while other use cases can efficiently be handled with simpler charging or policies. The use cases will also have different performance attributes; from delay-tolerant applications to ultra-low latency scenarios, from high-speed entertainment applications in a vehicle to mobility on demand for connected objects, and from best effort applications to ultra-reliable applications for health and safety. Furthermore, the use cases will involve a wide range of terminals such as smartphones, MTC devices, wearable’s, industry equipment etc.

In order to handle the multitude of segments and verticals in a robust way, there is also a need to isolate the different segments from each other allowing different organizations to operate each segment independently. For example, a scenario where a huge amount of electricity meters are misbehaving in the network should not negatively impact the MBB users or the health and safety applications. In addition, with new verticals supported by the 3GPP community, there will also be a need for independent management and orchestration of segments, as well as providing analytics and service exposure functionality that is tailored to the needs of each vertical or segment. The isolation should not be restricted to isolation between different segments but also allow an operator to deploy multiple instances of the same network partition.
Network slicing is a key concept that allows partitioning (or slicing) of the network in a way that enables efficient support of different and diverse segments and use cases as well as operational isolation. The figure below provides a high level illustration of the concept. A slice in this case is composed of a collection of logical network functions that supports the service requirements and performance of a particular use case. It shall be possible to direct terminals to slices in a way that fulfill operator needs, e.g. based on subscription or terminal type. The network slicing targets an end-to-end partition of the network.

Figure 15-3: Illustration of Core Network Slices that Cater for Different Use Cases

Network slicing is a far-reaching concept proposed to address the wide range of 5G use cases. But already in current networks we have the notion of existing and emerging network slicing. One obvious example is the use of different APNs for different services. These APNs may even be allocated to dedicated (physical) PGWs to provide additional security and resource separation between the “slices”.

DECOR is a feature being standardized for 3GPP Rel 13. It provides additional separation of EPC functions by allocating a complete dedicated core network including the MME, still within a single PLMN. The DECOR feature does not require any modification or configuration of the UE. The selection is based on an operator configured subscriber parameter provided by the HSS to the MME. The MME evaluates this parameter and if needed the UE is re-directed to an MME that is part of the dedicated core network. The re-direction is conducted before authentication and registration is performed in the core network.

The Rel-13 DECOR solution was not allowed to impact the UE. As mentioned above, there is a proposed 3GPP study item to enhance DECOR to allow some UE impacts. The objective of the study is to enhance the DECOR selection mechanism by providing assistance information from the UE. This can reduce the signaling required to register and also improve isolation between dedicated core networks since there is no need for redirecting between different Dedicated Core Networks.

15.3.3 Operational Aspects & Use Cases

Generic life cycle management tasks of the network functions also apply when they are virtualized. However, some tasks are split between the Element Manager of the network function and the NFV MANO functions, which manage the virtualized resources required by the VNF. The latter functionality is handled by the VNF Manager function in the NFV MANO architecture. Many operational tasks can be simplified or automated by leveraging the cloud deployment model of NFV:

- Reduced TTM
- VNF upgrade procedures using parallel VNFs with graceful introduction into production networks
- Scale in or out of VNF capacity
- Increased support for DevOps with faster test and deployment cycles

From a functional evolution perspective of the EPC architecture, there are additional opportunities to leverage the possibilities of a cloud deployment. Examples include the following:

- Multiple VNFs can be grouped into an aggregated VNF, which simplifies some of the management tasks
- Managing one aggregated VNF in contrast to multiple VNFs simplifies many operational tasks
- VNF upgrade is handled within the aggregated VNF by the vendor, reducing interop testing in the operator network
- Reduced signaling load between VNFs
- Vendor separation into different network slices
- Different configurations optimized for different service segments
- Consolidation of monitoring (probes, taps) requirements (e.g. a tap point may only exist inside the aggregated VNF)

Applying the aggregated VNFs in the context of network slicing with separation of logical network resources and operational isolation provides additional benefits:

- Operational isolation between network slices enabling e.g. individual network management and statistics per slice
- Business expansion by low initial investment and risk due to minimal/no impact on existing services
- Allow for different architectures or functional splits in different network slices, based on a subset of EPC or the future 5G-core network

The following figure shows some of the possible use cases:

![Figure 15-4: Examples of vEPC Use Cases](image-url)
15.3.4 Topology Aspects

The common management and virtualization infrastructure across the network topology as outlined in the Architecture Framework provides opportunities for more flexible deployment alternatives than in current model with dedicated hardware per network function. Figure 15-5 shows sample network architecture with national, regional and local data centers. The local and regional DCs may be collapsed based on the geography and latency characteristics of the network.

![Figure 15-5: Topological Distribution of Data Centers in Sample Network Architecture](image)

The deployment of network functions can to a large extent be driven by service and traffic needs. For example, the EPC can be deployed closer to peering points where a large amount of traffic is received from the Internet. Or another example, a new service launched in a specific region or market could require a specific service-optimized EPC deployed locally. These examples are still based on traditional network dimensioning and planning before deployment but give significant benefits in flexibility and shorter time-to-market (TTM). In a longer-term perspective, there is an opportunity to more dynamically deploy and change configurations in network functions, supported by E2E orchestration.

Given the fairly distributed nature of EPC today in the operator network, initial gains are likely to be seen in a centralization of EPC. This could be complemented with a more distributed deployment for specific service needs (e.g. efficient delivery of cached Internet or video traffic), dependent on the business case and availability of local DCs (e.g. driven by video and cache for fixed access).

15.3.5 Future Opportunities for Functional Evolution

The above sections describe the evolution of the current 3GPP architecture and possibilities when it is deployed in a NFV environment. The 3GPP-standardized architecture is maintained including well-defined interfaces. This provides multiple benefits including feature parity with existing EPC networks, support for (multi-vendor) interop testing, field proven software with telecom grade performance and availability at minimal risk exposure.

However, once in a cloud environment, there are additional technologies and ways of working (both in development of the network functions and in the operation of the network, AKA DevOps) that can be leveraged. These technologies and methods typically originate from the IT domain. Hence, it is worth noting that all may not be applicable to Telco network functions. Network functions are typically large in nature with high performance requirements in latency and throughput, and a requirement to maintain session state while IT applications have less stringent requirements. So when applying these technologies to network functions, careful analysis of the impact on both technical characteristics and added value are required.
Software architecture aspects:

- **Decomposition**
  - Concept of decomposing network functions into smaller components, enabling re-use of same functionality across several applications. Commonly used in SW architecture development internally by vendors with many network functions. Possibility to move common functions to the platform.

- **Micros-services**
  - Concept of decomposing an application into smaller components, where each component provides only one function (i.e. a “micro service”). (Special case of decomposition).

- **PaaS**
  - The cloud platform provides additional services to the application beyond IaaS. Requires a PaaS environment to be defined prior to application development, e.g. what PaaS services an application can rely on.

- **Containers**
  - A technology for deploying applications in a virtualized environment. Suitable for small workloads, offering a low overhead since the application is running on the host OS (which creates a dependency). Work is ongoing to enhance networking capabilities of containers to support Telco network functions.

- **Separation of control plane and user plane**
  - Concept of an architecture split of existing user plane intensive 3GPP network functions with the expectation to achieve more optimized user plane functions to meet increased capacity needs, e.g. by independent scaling of CP/UP. Challenges include how to maintain policy and charging control, session management, lawful intercept etc.
  - Note: The objective of individual scaling of CP and UP functions can be achieved already with today’s VNF architecture, even though physically separated deployments over different sites is not feasible.

Operational methods:

- **Micro-services**
  - A method in SW development where the tasks are divided into well-defined and separated modules as defined above. From an organizational perspective, the purpose is to keep the design teams small and agile (c.f. the “2 pizza” term = 12 persons).

- **Continuous Integration**
  - Method of continuously integrate and test SW components during development to minimize re-work after final testing and enable Continuous Delivery.

- **Continuous Delivery**
  - Method to frequently, e.g. monthly or weekly, deliver SW updates from vendor to service provider.

- **Continuous Deployment**
  - Method of service provider including automated acceptance testing etc. to enable more frequent deployment of SW updates in a commercial network.

The operational methods listed above are rather orthogonal to the software architecture aspects listed, even if they can be seen as enablers. The operational methods are more related to ways of working and the operational procedures in the vendor’s SW development organization and the service provider’s network operations.
15.4 EPC Network Function Requirements

The following outlines high level requirements towards the NFVI in relation to the individual EPC elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Control Plane</th>
<th>User Plane</th>
<th>Possible Optimizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MME</td>
<td>X</td>
<td></td>
<td>DPDK, SR-IOV</td>
</tr>
<tr>
<td>SGW</td>
<td>X</td>
<td>DPDK, SR-IOV</td>
<td></td>
</tr>
<tr>
<td>PGW</td>
<td>X</td>
<td>DPDK, SR-IOV</td>
<td></td>
</tr>
<tr>
<td>PCRF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSS</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Femto Gateway</td>
<td></td>
<td>DPDK, SR-IOV</td>
<td></td>
</tr>
<tr>
<td>ePDG</td>
<td>X</td>
<td>DPDK, SR-IOV, encryption acceleration</td>
<td></td>
</tr>
<tr>
<td>BM-SC</td>
<td>X</td>
<td>DPDK, SR-IOV</td>
<td></td>
</tr>
<tr>
<td>eMBMS Gateway</td>
<td>X</td>
<td>DPDK, SR-IOV</td>
<td></td>
</tr>
</tbody>
</table>

Figure 15-6: High-Level Requirements for NFVI in Relation to the Individual EPC Elements

Assumptions on generic requirements on the NFVI:

- Fast failure detection cannot be ensured; hence failover must be solved at the application level.
- NFVI must provide multiple availability zones to ensure redundancy for both CP and UP VNFs or VNFCs.
- Multi-tunnel support (e.g. VLAN, GRE) per vNIC or multiple vNICs per host required for all VNFs.
16 L1 – L3 Functions

Layer 1 – Layer 3 functions form the underlying transport infrastructure, providing connectivity for applications and other core network functions. Shown below are the various layers of a modern transport network. Some of these functions may be virtualized; however, many components of Layer 1 and Layer 2 functions are not suitable for virtualization.

A Dense Wave Division Multiplexer (DWDM) supports the simultaneous transmission of multiple optical signals over a single fiber by provisioning a specific wavelength of the light spectrum for each signal. This has allowed for significant increase in fiber capacity. DWDM systems are commonly configured as reconfigurable optical add-drop multiplexers (ROADM) and are primarily the popular choice for metro optical and long haul optical networks.

Optical Transport Network (OTN) is an ITU standard that extends the concepts of SONET/SDH to higher data rates (100Gbps). The OTN convergence layer has evolved over the layer 1 DWDM network for enabling optimal convergence of traditional SONET and newer high-speed data services. It is designed to effectively map different protocols and rates into the same 10G or 100G uplink. Using the OTN layer, any protocol carried over the network, whether SONET or Ethernet, has similar deterministic latency, bandwidth and performance monitoring. The OTN infrastructure also enables transport over longer distances with less regeneration by utilizing forward error correction (FEC) mechanisms embedded in the OTN layer.

A ROADM and an OTN switch may not be suitable for virtualization.

Packet devices, like routers, operate at layer 3 and are able to forward data packets between source and destination based on OSI network layer information. Routers are essentially highly specialized computers, and in some cases are well suited for virtualization. Generally, routers are logically composed of a control plane and a forwarding plane (also known as a data plane). The control plane is responsible for handling signaling and network awareness. It does so by utilizing several mechanisms:

1. Routing protocols for distributing and gathering reachability information
2. Routing algorithms for path determination
3. Routing databases for storing paths determined by the routing algorithm
These functions are compute intensive and are traditionally performed on the CPU of the route processor (RP) in a physical network function.

The forwarding plane is responsible for rapidly switching data packets through the router platform. Router forwarding plane architectures vary depending on router, but can generally be divided into three categories; shared memory, distributed, and cross bar. The main difference between the architectures is in the data path and switching decision location.

Router virtualization can take several forms. The control plane and data plane can be packaged together into a standalone VNF as shown below:

In this configuration the router is ported as an application running on the NFVI. It is desirable in most cases that the forwarding plane of the router is accelerated. A high performance forwarding plane makes the best use of underlying NFVI resources to accelerate packet processing. This is necessary for cases where a standalone router VNF is replacing a physical network functions such as MPLS Provider Edge router. For control plane only functions, such as a virtual route reflector, accelerated packet processing is usually not required.

Another form of router virtualization is a hybrid of virtual and physical functions as shown below:
In this configuration the router control plane is virtualized and runs on the NFVI. This allows CPU intensive control plane functions to take advantage of the NFVI CPU and memory resources. Improved scale and processing performance can be achieved. The router’s forwarding plane remains in a physical network function. High-speed custom or merchant silicon ASIC technology may be utilized to achieve accelerated and specialized packet processing. The virtual control plane and physical forwarding plane communicate via a standard or proprietary interface. Together, the virtualized control plane and the physical forwarding plane form a unified routing platform.

Below is a table of Layer 3 (and above) network elements which can also be deployed as standalone VNFs. This is a representative list of network functions and is not intendend to be exhaustive. These VNFs are typically comprised of a vendor’s network operating system running as a virtual machine on a hypervisor. Within this environment a virtual switch (vSwitch) is responsible for switching traffic between the core network and a VNF or between multiple VNFs. The performance of the vSwitch directly impacts the performance of the VNF, particularly VNFs requiring a high speed forwarding plane. As result, multiple approaches have been adopted to help boost VNF performance. The table indicates which VNFs may benefit from additional performance optimization.

<table>
<thead>
<tr>
<th>Network Function</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route Reflector</td>
<td></td>
</tr>
<tr>
<td>Broadband Network Gateway (BNG)</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>IP Router</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>IP Multicast Router</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>L3 VPN Router</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>L2 VPN Router</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>IP/MPLS Router</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>MPLS Segment Router</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>MPLS Multicast Router</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>Ethernet Switch</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>Firewall</td>
<td>Container solutions may also be considered</td>
</tr>
<tr>
<td>Traffic Analysis</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>L3-L7 Monitoring</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>Network Function</td>
<td>Comment</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>NAT</td>
<td></td>
</tr>
<tr>
<td>Load Balancer</td>
<td></td>
</tr>
<tr>
<td>IPSEC VPN</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>CPE</td>
<td>Container solutions may also be considered</td>
</tr>
<tr>
<td>Media Streamer</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>Media Transcoding</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>Media Encapsulation</td>
<td></td>
</tr>
<tr>
<td>Media Encryption</td>
<td>HW acceleration may be required</td>
</tr>
<tr>
<td>Media Origination</td>
<td></td>
</tr>
<tr>
<td>Media Vault</td>
<td></td>
</tr>
<tr>
<td>Media Cache-Node</td>
<td></td>
</tr>
</tbody>
</table>

Table 16-4: L1 – L3 Network Elements
17 SGi-LAN Architecture

17.1 Introduction

Today, Mobility Service Providers enhance and monetize content delivery through various services deployed with middle-box functions such as traffic analysis (TA), Firewalls, NA(P)T, HTTP traffic optimization, video and web optimization and more. Given that most of these services are deployed between the mobile network and the Internet and because the 3GPP interface to the mobile network is termed the “Gi” interface in 3GPP and “SGi” in LTE, these service functions are said to be hosted on the “SGi-LAN” and are called “SGi-LAN service functions”.

In today’s Mobile Networks, SGi-LAN service functions are built from physical devices, which are interconnected so that each can analyze, manipulate, control, enhance or report on mobile data. Packets are routed from the mobile network to the Internet through “hardwired” service chains, so called because they rely on physical cabling and static routing mechanisms (such as “policy-based routing”). Any segregation of subscribers is via APN.

This is shown in the diagram below:

SGi-LAN functions can be virtualized following ETSI NFV principles with VMs interconnected using SDN. Doing so solves the “rigidity” problem associated with physical SGi-LANs and enables the operator to capture the benefits of automation. As each service function is now a microservice VM where scaling to the 100’s of Gbps proceeds via replication of the VMs comprising the system (e.g. as system capacity grows, then more VMs are added).

The purpose of this chapter is to identify the functional architecture of the SGi-LAN domain. The term “service-function chaining” has emerged to describe composite services that are constructed from one or more service functions. A service chain steers traffic through a set of functions in a set order. The reverse path traverses the same service functions as the forward path but in reverse. Reverse-path affinity is the requirement that the same instances of service functions traversed in the forward direction are traversed in the reverse direction, in reverse order.

In networking today, there are three types of services that rely on service chaining: (1) SGi-LAN, the topic of this chapter; (2) Residential vCPE which is akin to SGi-LAN except for fixed access (i.e. residential vCPE); and (3) Enterprise vCPE which is focused on delivering managed services to enterprise customers. The difference between these is that (1) and (2) may map multiple subscriber flows onto a single chain, whereas in (3) the chain is often dedicated to precisely one customer (an enterprise).
chapter focuses on the SGi-LAN service with some reference to the residential vCPE use case as it is related to fixed/mobile convergence.

The chapter begins with a review of relevant emerging standards covering the SGi-LAN as well as “de facto” interfaces derived from the open-source community. The emerging standards include activities in 3GPP and IETF. The open source de facto interfaces result from developments in, e.g., OpenStack and OpenDaylight or ONOS™. Also, we discuss the implications of software forwarding in the context of the NFV infrastructure (NFVI). We follow with a discussion of the functional architecture including target use cases and automation options as well as HA options. The final section deals with major evolution considerations, including container support, fixed/mobile convergence, and managing the challenges of encrypted traffic.

17.2 Standards Relevant to the SGi-LAN NFV-SDN Solution

This section reviews the work done in 3GPP around the topic of service function chaining to establish language, and to establish the direction of the industry. The 3GPP PCC framework and the PCRF defined as a part of it are enhanced to support SDN and NFV. An SDN steering rule is conveniently interpreted as being derived from a PCC rule in the PCC architecture. This concept is extremely powerful and is the foundation of many use cases.

The IETF also has done work in the area of service functions. The IETF addresses the problem of “force steering” flows in a service chain, independent of service function networking configuration and independent of protocol. The IETF work has an additional virtue in that it supports metadata exchange which is useful in terms of allowing nodes on the service chain to communicate.

17.2.1 Flexible Mobile Services Steering and 3GPP

3GPP work on the topic of service function chaining has resulted in two technical reports which are TR 22.808 and TR 23.718, covering the topic of “Flexible Mobile Services Steering”. Release 13 work is expected to impact the PCC procedures in TS 29.212 as well as the PCC architecture.
Figure 17-2 depicts the 3GPP Release 13 PCC architecture. The primary policy and charging functions Policy and Charging Rules Function (PCRF), Online Charging System (OCS) and Offline Charging System (OFCS) are discussed in more detail in chapter 18.

The user plane functions are Bearer Binding and Event Reporting Function (BBERF), the Policy and Charging Enforcement Function (PCEF), and the Traffic Detection Function (TDF).

- **PCEF** is the enforcement functionality in the gateway that acts as the anchor point of the user traffic (in the context of LTE, the PGW).
- **BBERF** is an additional enforcement point (think SGW or ePDG) that needs to be managed through the PCRF in case Proxy Mobile IP (PMIP) is used instead of GPRS Tunneling Protocol (GTP).
- **TDF** is a traffic analysis function that identifies specific application flows.

Other functions in the PCC architecture:

- **The Subscriber Repository (SPR)** contains subscriber information related to policy and charging.
- **Application Functions (AF)** are related to specific applications that may interface with the PCRF to request QoS. An example is IMS, which may interface with the PCRF to request QoS support for VoIP calls.
- **RAN Congestion Awareness Function (RCAF)** emerged as part of the UPCON (“User Plane Congestion) work and is a functional entity which reports RAN User Plane Congestion Information to the PCRF to enable the PCRF to take the RAN user plane congestion status into account for policy decisions.
To support service chaining, 3GPP introduced the Traffic Steering Support Function or TSSF. The TSSF is a function that receives traffic steering control information from the PCRF and ensures that the related traffic steering policy is enforced in the SGi-LAN.

The 3GPP approach has the following features:

- A functional architecture for services steering. In the 3GPP architecture view, the PCRF is used to create/modify/remove traffic steering policies in a PGW or a TDF, defined with the granularity of subscriber, application or service data flow.

- The TSSF is the control plane part of the TDF without the charging related aspects of the TDF, which are unnecessary for services steering. The TSSF can be deployed as a modular component for further control-data plane separation. The role of the TSSF is to provide “classification” control.

- Identify interfaces in the PCRF that can be used to create/modify/remove traffic steering policies defined granularity of subscriber, application, and service data flow. In this respect 3GPP has: (1) enhanced the “Gx” if steering rules are implemented in the PGW, (2) enhanced the “Sd” if they are implemented in the TDF, and (3) introduced a new “St” interface in case the TSSF is identified as a separate control function. The latter has two flavors, one of which is based on Diameter and is basically a subset of the “Sd” and the other based on REST.

A traffic steering policy rule is pre-configured in PGW, TDF or TSSF (as a PCC rule) and can be referenced for Service Data Flows (SDF) in the uplink, downlink or for both directions. Activation of a steering rule is via interaction with the PCRF, which can be either “pull” or “push” modes. Should there be a change in policy rules, the re-authorization request procedures can be used to update policy (Diameter RAA message).

### 17.2.2 Service Function Chaining and the IETF

In today’s data center networks, it is possible to create service chains in the virtualized environment using a technique known as V(x)LAN stitching. In this approach, L2 segments are maintained by the SDN controller and concatenated to create the service chain in a manner analogous to the way one would create the service chain with physical cables. It is also possible to use other tunnel encapsulations (GRE, for example). Nevertheless, V(x)LAN stitching has certain limitations, as explained in RFC 7498.

An alternative is to use the Network Service Header or NSH, as defined by the IETF SFC working group, which allows the separation of the forwarding path from the logical topology imposed by the chaining concept. The NSH is currently being standardized in the IETF with broad industry support along with an identified set of mobile network use cases. NSH is a data-plane protocol introducing an encapsulation header that contains an identification of the service path in the network and provides a common service plane fully orchestrated top to bottom, which is specifically designed to support dynamic service function chains. The path information is akin to a subway map: it tells the packets where to go without requiring per flow configuration. The header may also contain metadata, which is information about the packets, and which can be consumed or injected by service functions, hence this can enable policy. NSH allows for the separation of the forwarding path, e.g. the service chain itself, from the physical connectivity. It is this attribute that makes the use of NSH invaluable in the data center.
Figure 17-3: IETF Service Function Chaining (SFC) Architecture with 3GPP Control Options

Note: One of Gx, Sd, or St can be used. In the latter case, the classifier is further decomposed into separate user and control planes.

The NSH is added to packet via a classifier that mediates between NSH and non-NSH domains. Included in the NSH is the 24 bit SPI (Service Path Identifier) field, which is used by forwarders to define the disposition of forwarding. In NSH, a service chain is an association of service functions defined by a common SPI.

NSH is carried along the chain to services and intermediate nodes that do not need to be NSH-aware since a proxy can take over the NSH function so that non-NSH enabled services are supported. The addition of metadata via the NSH header in the IETF SFC protocol enables the evolution of service creation from a very simple set of non-related building blocks to composeable services that can pass and react to imputed data and mid-service computational results. Thus, beyond simple linear graphs, any kind of directed graph can be supported with a dynamic component whereby a graph node can, via policy, make an autonomous determination to re-steer a flow. Additionally, the presence of programmable metadata allows new opportunities for compliance and assurance.

For reference, the NSH fields are depicted below:
17.2.3 Use Cases

This section covers specific virtualized SGi-LAN use cases. The essential idea of the use cases is that groupings of subscribers, called “classes”, can be formed in the SPR so all members of the class receive a specific treatment in terms of assignment to service chains. The means of managing which value added services are traversed by a subscriber can be done on a single APN, the Internet APN. Normally, to achieve the same objective, subscribers are assigned to different APNs. In this latter scenario, service chains are “hardwired” into the APN interface on the PGW and SIMs on subscriber devices must be re-provisioned and re-assigned to match the APN with the associated configuration of the subscriber on the HSS. In the proposed approach, by contrast, the operational functions reduce to: (1) creating the class in the SPR and (2) configuring the service chain definitions in the orchestration system.

17.2.3.1 Mobile Enterprise Access VPN

A virtualized SGi-LAN has the potential to improve the way services are introduced, with additional speed and flexibility. In this use case, the marketing department in the operator conceptualizes a new type of service for an enterprise called XYZ. The service comprises application acceleration appliances for Outlook email, SalesForce, and other enterprise utilities. In addition, the “XYZ Service” provides access to the Internet but only through the enterprise network. Hence, a VPN router is provided as well. As a service, it is to be applied to a set of subscribers, which have been identified to the carrier by the XYZ Corporation. The service is delivered as virtualized service functions interconnected using a specified way on the SGi-LAN. The marketing department creates the “directed graph” representing the sequencing of primitive services to be applied to subscribers that use it. These service definitions are loaded onto the orchestration systems as a service description. Concurrent with this activity, the operator creates a class of users that will consume the service. Membership of a subscriber to the “XYZ Service” class is determined by an association of the subscriber identity (MSISDN for example) to the service in the form of a service steering rule, identified by reference to a name in the SPR.
An initial instance of service chains is brought into existence. The first subscriber can access the XYZ Service and as the number of subscribers grows, the orchestration system is closely monitoring the consumption of resources. When a pre-specified high water mark is reached, additional relevant service chains are automatically created.

17.2.3.2 Consumer Internet Access

The most common use case would be in support of “Consumer Internet Access”. A service chain comprising this service would include an optimization appliance (TCP optimization, for example), a traffic shaping appliance to enforce fair use policies, if applicable, maybe a caching server, and then finally, a NAT/Firewall.

Since the subscriber session record contains reference to the device IMEI, an operator could partition services associated to Consumer Internet Access also by device type. For example, an operator could have service chains specialized to Apple devices, and others specialized to Android devices. Such service chains could include optimizations explicit to the device type. For example, video servers in the chain could stream video at a resolution appropriate to the devices. Alternatively, operating system updates could be cached so they are specific to the device type.

17.2.3.3 Clean Pipe Internet Access

The simplest possible service chain offers a “clean pipe service” where perhaps a NAT/Firewall is the only appliance on the service chain. A policy rule is installed so that IMEIs associated with PC dongles or LTE routers are selectively routed into clean pipe service chains.

17.2.3.4 Named Community of Interest

In “Named Community of Interest” use cases, a subscriber may self-identify to the network with the objective of becoming a member of a class that uses specific service chains. The self-identification into the class is conveniently accomplished through an app and allows the user to specifically “opt-in” to a set of services. From a design perspective, the app works through an agent in the operator’s policy layer that effectively provisions the SPR on the back-end with the relevant information binding the user to the service chains associated to the selected class.

Possibilities are unlimited. The operator could create community of interest classes associated to anything (sports teams, special interests, etc.) and could create value added services that host specialized content.
17.2.3.5 Parental Controls

One specific example of a “named community of interest” is a parental controls community. This class of users has restricted access to certain web sites for which the content is deemed inappropriate. The service chain contains a service function with the ability to filter sites based on the IP address of the site or on the DNS name of the site (determined through reverse DNS).

Parental control filters operating on destination IP addresses work even on encrypted SSL/TLS web flows but are not capable of discriminating between appropriate content pages and inappropriate content pages issued from the same server. They are, however, independent of the app used to access the content (browser type, social network app, etc.). In addition, to prevent users from installing proxy servers or VPNs that allow the by-pass of the parental controls, the user device must be “locked-down” to restrict installation of any capability that allows bypassing the parental controls filter.

17.2.3.6 TCP Optimization

Optimization via transcoding is becoming more difficult as the volume of encrypted traffic in mobile networks increases. However, it is still possible to optimize TCP transport with a middlebox appliance designed and tested for that purpose. It is well understood that Reno and Reno-derived TCPs interpret packet loss and delay as the consequence of a buffer overflow event at a location in the path where there is a throughput bottleneck. On the radio, loss and delay is most frequently the consequence of radio effects such random bit errors (which cause the radio to re-transmit) and delay spikes (caused by retransmissions). Retransmissions can also be at the LTE link layer to correct for packet losses and can also cause delay spikes in the TCP. These delay spikes cause spurious RTO time-outs that are not related to packet loss but force the TCP sender to back-off. For these reasons, TCP is not particularly well suited for the radio.

A TCP optimization appliance can be a valuable tool to enhance QoE when deployed in a service chain.

17.3 Management aspects

Chapter 4 described a generic VNF Manager based on Openstack Tacker. Currently an important aspect of management is data model driven programmability. Models capture the config and operational state of the system. These models are rendered into a running system. The elements of data modeling and template-driven orchestration for the new SDN and NFV management and control solutions as well implications for SGI-LAN delivery are described below:
The left-hand side of the figure above shows the increasing level of abstraction from bottom to top.

- Network functions (whether virtualized or not) have their associated data models. Every network function could have its own data model, but there are several efforts underway in the industry to standardize these models (e.g. standardization work in IETF and the work done by the OpenConfig [22] consortium of operators). Systems like the WAN SDN Controller and the Resource Orchestrator rely on NF data models to manage the associated network functions.

- To the extent that NFs don’t use standardized models, SDN Controllers and Orchestration functions generally abstract the specialized models to generalized models. For example, the adaptation layer in the WAN SDN Controller performs this translation function. Thus, the higher layer functions within SDN Controllers deal with abstracted models of the network equipment.

- The WAN SDN Controller in turn exposes service data models to higher layer systems. These models describe services like IP VPNs or E-lines. The Network Resource Orchestrator, therefore, does not need to have a detailed understanding of the network itself. It simply requests establishment of services between specific end-points.

- The Network Resource Orchestrator itself exposes service models to the systems above it. In the example discussed in this chapter, it exposes SGi-LAN service models to the Service Orchestrator. The service models exposed by the Network Resource Orchestrator may build on the service models exposed by the SDN Controllers.
In addition to data models, descriptor files are important. Where data models describe how to manage a function or service in terms of provisioning and monitoring, descriptor files describe how to build, scale, heal and upgrade a VNF or Network Service. Descriptor files are created by the network service architect or VNF designer. Descriptor files only capture the information required at each level of the orchestration process. For example, the NSD associated with the SGi-LAN identifies that a firewall needs to be instantiated, but it does not provide details about the internals of that firewall. The VNFD associated with the firewall captures the internal architecture.

There are two ways in which the orchestration architecture can be deployed.

- The operator can elect to build its SGi-LAN from discrete components, using the Resource Orchestrator to automate the coordination of the various Service Functions each of which is delivered as a separate VNF.

- Or, on the other hand, it can choose to deploy a SGi-LAN platform, which is delivered as a single composite VNF (in the sense of a VNF made-up of multiple VNF components or VNFCs).

In the former approach, the operator may decide it has more flexibility to architect its data center. In the latter, the composite VNF achieves information hiding to the Network Resource Orchestrator. This means that specific capabilities associated to the domain requirements can be included as features of the composite VNF. A good example is the HA feature, which in the first approach is delivered as an orchestration solution attribute whereas in the second approach is actually a feature self-contained in the composite VNF. The latter approach may be more replicable (it involves less customization and tailoring to an operator’s environment), which is an important consideration as it reduces overall solution integration cost.

**Session establishment Call Flow**

An example of a session establishment call flow is shown below.

**Preconditions:**

- The initial set of service chains has been configured. At least one instance of every type of service chains is in existence. Note that no direct user manipulation of the SDN controller should be required.

- The subscriber policies mapping each subscriber to a service chain type have also been configured.

- The call flow shown is for one user who is assumed to be attached. The case where the initial packet in the flow is an uplink is shown.

**Call Flow Sequence:**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>As part of the session establishment, the SGW sends a GTP-c Create Session Request message to the PGW.</td>
</tr>
<tr>
<td>2.</td>
<td>PGW initiates IP-CAN session creation procedure by sending CC Request to PCRF server as per APN configuration.</td>
</tr>
<tr>
<td>3.</td>
<td>PCRF queries Subscriber Profile Repository to obtain subscriber policies for the particular user.</td>
</tr>
<tr>
<td>4.</td>
<td>PCRF provisions the classifier (the TDF) with steering policies for all the flows associated to the specific user.</td>
</tr>
<tr>
<td>5.</td>
<td>Create Session Request Response.</td>
</tr>
<tr>
<td>6.</td>
<td>When the uplink user packets arrive at the classifier, it encapsulates them</td>
</tr>
</tbody>
</table>
17.4 Functional Needs

The basic requirements for the SGi-LAN solution are covered in this section.

17.4.1 Open Architecture

Operators have traditionally deployed “best-in-class” physical network elements in the SGi-LAN. It stands to reason that the virtualized SGi-LAN must similarly be able to on-board service functions from any vendor that adheres to on-boarding guidelines. This implies:

- Support for the operator selected hypervisor
- Adherence to “industry standard” architectures for performance acceleration
- Clear and well defined practices and procedures for integrating the service function into the orchestration environment
- Support for configuration management interfaces

17.4.2 High Availability functional needs

High Availability (HA) functional requirements on the SGi-LAN must be examined very carefully prior to the design stage.

One question concerns the way the system reacts in the event of a failure of a service function. There are two extreme viewpoints:

- The orchestration system is continuously monitoring the health of a service function. When a service function fails, the event is detected and the failed service function is simply recreated elsewhere. There is no state preservation post-failure. State is recreated on the new service function. This approach is service interrupting.
HA is an attribute (feature) of the service function. The service function is deployed in a “mated pair” configuration and an overlay link is used to convey instantaneous state from a primary service function to a back-up service function. On a failure, the back-up service function picks up the load while preserving state. No intervention of the orchestration system is required to implement the fail-over; however, the orchestration system is expected to “clean-up” post-facto. This scheme is entirely analogous to how HA is implemented across line cards in a physical network function today.

Other schemes are possible that are "in between" these two extremes. For example, HA is possible at the service-chain level meaning that back-up and primary service chains can be deployed. However, synchronizing state between the two service chains (difficult with TCP protocol) is an engineering challenge and will carry an extra cost in terms of (1) operational complexity, (2) duplication, and (3) extra physical equipment. Additional considerations involve the size of the failure group. An example of an important consideration: if the service function supports a small enough number of subscribers, does its failure constitute an FCC reportable event?

### 17.4.3 Scalability

Scalability considerations will vary depending on implementation. In any event, scalability of the system is via replication. When new bandwidth demands arise that cannot be fulfilled in the current system, the orchestration system takes notice and creates additional resources by replicating services chains. Along with each, classifiers must be replicated as well.

The primary design issue is “spreading” a multi-gigabit load of traffic across service chains.

### 17.4.4 IPv4 Address Overlap Issues

Carriers often use overlapping IPv4 address ranges for their UEs on their Internet APN. For example, PGWs might be assigning from the same 10/24 IP address pool. In order to uniquely resolve an IP address on the SGi-LAN to a subscriber identity, 3GPP Release 13 introduces an IP-Domain-ID AVP, which when combined with the IPv4 address, is unique.

### 17.5 Evolution Considerations

This section covers SGi-LAN evolution considerations. This is partitioned into three categories which are: (1) container support, (2) fixed/mobile convergence, and (3) encrypted traffic.

#### 17.5.1 Container support

Linux containers are logical partitions within the Linux operating system. These partitions provide an environment for operating Linux applications in an isolated fashion. The major advantage of containers over VMs is that they are lightweight and can boot up quickly. That means that container management systems can be more agile and dynamic than cloud management systems that use VMs.

There are several disadvantages to containers though. For one, applications must be containerized, which means they must be ported to a supported Linux environment. Not all applications Verizon uses are on Linux. For another, container-based virtual networking is not as mature as VM technology.
Containers are a powerful technology and the SGi-LAN system is expected to integrate Linux container support over time.

17.5.2 Fixed/Mobile Convergence

The SGi-LAN can support Fixed/Mobile Convergence via the virtual CPE mechanisms. Devices that attach to the fixed wireline network can make use of services that are also delivered as part of the SGi-LAN.

The context is a tunnel interface such as an S2a or an S2b from TS 23.402 extending from the mobile core to a device supporting the fixed/mobile convergence capability. Such a device could use split tunneling to access local resources (e.g. a printer) but would make use of converged services delivered from the mobile core.

![Fixed/Mobile Convergence Use Case](image)

In a virtual CPE approach, the CPE functions are pulled into the operator’s network domain. In this way we believe we can realize the vision of common personalized services available to any subscriber on any network.

Examples of such personalized services include:

- Unified, access independent, parental controls
- Common environment for home automation
- Seamless IMS communications anywhere
- Cross access mobility

The essential solution component for fixed/mobile convergence is provided by access, via tunnels, into the mobile core from a non-3GPP network. Such access is enabled via TS 23.402 mechanisms.

17.5.3 Virtualized SGi-LAN Service Functions and Encrypted Traffic

At the time of writing, over 50% of mobile network traffic is encrypted on a per session basis using TLS or SSL [24]. The percentage by volume is expected to rise to near 100% with the deployment of HTTP/2 consistent with the IAB recommendations and the increasing ease of obtaining X.509 certificates. One consequence is that the proper functioning of the existing SGi-LAN is compromised by encryption. Smith and also Moriarty and Morton have documented the impact in the IETF Internet Drafts.
SGi-LAN service functions are deployed to optimize delivery of packets to subscribers. Some of these trans-rate or compress the content reducing the volume of bytes delivered over the air interface. Others still will accelerate the user experience by caching content closer to the subscriber, reducing RTT and WAN congestion. Per session encryption via SSL or TLS will render caching service functions inoperable unless a content publisher has explicitly contracted with a CDN operator for PKI services. Trans-rating/compression of content will not work if the content is not visible. Other types of SGi-LAN service functions will offer HTTP enriched headers that can be consumed by advertising customers. The traffic analysis (TA) service function is frequently deployed to enable path management functions and fair use policies, throttling traffic based on the identified application type. Neither HTTP header enrichment nor TA will work on encrypted traffic.

Careful consideration must be given to the deployment of service functions in the SGi-LAN as encryption is changing the impact these might have. Please see relevant industry references on this topic.

The following set of service functions (as well as others) can have a continued role on the virtualized SGi-LAN even with 100% encrypted traffic:

- **NAT and Firewall.** To protect the IP address of the mobile device and to provide translation of the mobile device IP address to one that is publicly routable. Other security functions can be supported (IDS, DDoS mitigation, etc.).

- **VPN router service function.** To provide connectivity to an enterprise domain.

- **Service functions that provide traffic shaping or limiting** can be useful in connection with HTTP Adaptive Streaming. Particularly, if they can heuristically infer the nature of the traffic with a high enough probability.

- **Analytics collectors** that operate at the five-tuple level and collect flow usage information can provide useful information.

- **Parental controls** operate on the basis of destination IP addresses to filter content.

- **CDN caching nodes** operated by the mobile operator on content provided by cooperating publishers.

- **TCP optimization middleboxes,** since they operate at a level below TLS.
18 Charging Architecture

18.1 Introduction

To monetize services, an operator gathers relevant information and turns this into bills. Different services use different types of data. For example, data services can be billed based on the amount of transmitted and received traffic. On-demand video is usually billed per instance and may be differentiated based on the type of content. Cloud services can be billed based on the amount of compute capacity, memory and/or transmitted data. Telephony services can be billed based on location information (e.g. domestic vs international) and call duration. Furthermore, in the case of mobile telephony, there are enhancements such as pre-paid billing that offer additional ways to provide attractive payment options to end-users.

This chapter discusses the high-level charging and billing architecture and examines how SDN and NFV might affect that architecture.

18.2 Billing and charging architecture

The high-level charging and billing architecture is depicted in this diagram:

The process starts when a customer selects a service from the service catalog and signs up. This can often be done in multiple ways, for example by calling the service provider customer service department or through a web portal. As result, the service orchestration system interfaces with the billing system to set up the billing mechanisms for this particular user.

Depending on the type of data that is used for billing a particular service, a data collection mechanism is used to collect the right information and associate it with a specific customer or service instance. That collection system subsequently forwards information to the billing system in the form of Charging Data Records (CDRs), which the billing system translates into monetary values based on the subscriber's service contract. Since different services use different types of data, the various collection solutions are independent and different.
Note that billing does not necessarily require the collection of low-level information from network functions. Requests made by the customer through the web portal or API Gateway could themselves be the events that are used by the billing system to bill the customer.

Most of the processing related to billing and charging takes place in the OSS/BSS layer. The actions within the network and the resource management layer are predominantly focused on data collection.

### 18.3 Charging and billing for mobile and IMS services

For mobile and IMS services a special charging solution has been developed that meets the requirements of that specific domain. Billing and charging architectures for LTE and IMS are covered by 3GPP specs 23.203 (Policy and Charging Control), 32.251 (PS domain charging) and 32.260 (IMS Charging) respectively. Figure 17-2 in the previous chapter showed the 3GPP PCC architecture for TS 23.203.

The charging architecture for mobile services recognizes two functions:

- **The Offline Charging System (OFCS)**, which collects information about chargeable events (e.g. session start and stop times, amount of bytes consumed in uplink and downlink directions for packet data sessions, etc.) from the network functions and passes this information on to the billing system in the form of Charging Data Records (CDRs).
- **The Online Charging System (OCS)**, which keeps track of network utilization in real time. Once a user exhausts his or her available credit, the OCS interferes, e.g. by blocking further calls.

These functions act in concert with the Policy and Charging Rules Function (PCRF).

- **Policy and Charging Rules Function**
  
  The PCRF interfaces with network elements that enforce policy and charging rules, specifically the SGW, PGW and the Traffic Detection Function (TDF), an optional function which performs traffic analysis and can be used for application-specific charging, application-based QoS enforcement and traffic redirection. The PCRF acts based on information contained in the Subscriber Policy Repository (SPR) and based on inputs from Application Functions (AF) such as IMS and RAN Congestion Aware Functions (RACF) that trigger actions based on RAN congestion state. See Chapter 13 for more information on the role the PCRF plays in the case of Gi-LAN services.

- **Offline Charging system**

  In the Evolved Packet Core (EPC), the OFCS interfaces with the SGW, PGW and the TDF. In the IMS architecture, the OFCS has interfaces with several functions (xCSCF, AS, BGCF, MGCF, MRFC, etc). The interface between network functions and the OFCS is denoted as the Gz reference point in the LTE specs and as Rf in the IMS specs. It is a Diameter interface.

- **Online Charging System**

  In contrast to the OFCS, the OCS interfaces with a smaller set of Network Functions, as these functions need to support dedicated procedures in order to work in concert with the OCS. In EPC, those functions are the PGW and TDF. In IMS that function is the SIP Application Server (a generic name, which covers, among other things, the Telephony Application Server or TAS). Other IMS functions that may interface with the OCS are the MRFC and the S-CSCF. The
interface between network functions and the OCS is denoted as the Gy reference point in the LTE specs and as Ro in the IMS specs. It too is a Diameter interface.

In the 3GPP PCC architecture, an interface exists between the OCS and the PCRF. This enables the OCS to request the PCRF to enforce new policy rules in case the user exceeds available credit. PCRF, PGW and OCS form a finely tuned collection of systems that enable the operator to enforce a wide variety of policies and apply sophisticated charging schemes in the highly dynamic world of LTE services.

### 18.4 Assumptions about future charging and billing solutions

When considering the evolution of charging and billing in the context of SDN and NFV, several assumptions need to be made:

1. Charging and billing solutions for mobile and IMS services will need to be preserved. Options like pre-paid services will continue to exist while LTE evolves and 5G is being deployed.

2. With respect to mobile and IMS services, the existing charging interfaces (i.e. between the network functions and OFCS and OCS) cover a wide array of charging options. These interfaces will continue to be used in future years.

3. The variety of billing options will continue to grow. Dynamic pricing, tiered pricing, spot pricing, early reservations, pay-as-you-go, coupons, location-based pricing, zero-rating, 1-800 options, service bundles, family plans, sponsored service, etc, may be introduced and evolve. This will require evolution of existing OCS and OFCS solutions as well as the introduction of new billing innovations. The latter will often be provided through solutions in the OSS and BSS space that use collected usage information in creative ways.

4. Given the wide variety of future billing option, it is impossible to accurately predict the new features in network element or resource control systems or new protocols that may be required to support all potential future developments. The impact of new billing options on the network functions themselves is likely to be small, but to the extent that there is impact, it is better to wait with specifying the associated requirements and implementing the necessary features until there is a clear plan for the introduction of those new billing options. Otherwise, the effort is premature, often wasted and likely to miss the mark anyway.

### 18.5 Charging and billing in the SDN & NFV architecture

Figure 18-2 shows charging and billing in the SDN and NFV architecture:
Compared to today’s architecture, SDN and NFV introduce two major sources of usage data:

1. Network utilization is collected through the WAN SDN Controller or through a data collection solution. The WAN SDN Controller associates it with a particular network connectivity service and forwards the relevant CDRs to the billing system.

2. NFVI utilization (compute resources, processing cycles and memory utilization) is collected through a solution like Ceilometer, which forwards the data to the VNFM and EEO. The VNFM associates the data with specific VNFs. The EEO associates it with specific Network Services. Either one can then forward the relevant CDRs to the billing system.

3. For many new services, such as Bandwidth-on-Demand, it is likely that billing is based on the attributes associated with the service (e.g. guaranteed bandwidth) instead of actual resource utilization. In those cases, billing events can be generated directly by the OSS function.

4. Network Functions themselves may generate service specific information that is collected through dedicated systems and forwarded to the billing system. Below, this document discusses further details on charging related to mobility services.

With respect to statistics collection, there are two options:

a. Collection can be done in the WAN SDN Controller itself. Statistics collection has been identified as one of the components of WAN SDN Controllers.

b. Collection can be done in a separate data collection system, such as the Service Assurance function. In that case, there needs to be an interface between the WAN
The advantage of the latter approach is that the data collection function could be part of a broader analytics function, which could facilitate a more extensive set of billing options, for example, by combining network statistics with other types of information. Similar considerations apply to NFVI data collection.

18.5.1 Monetization options

To see the benefits of the billing and charging architecture proposed above, consider a service like virtual CPE (vCPE) where functions that traditionally reside in CPE equipment are hosted in the core network. The end-user only needs a thin CPE device. Additional functions are instantiated as VNFs and service chaining is used to route traffic through those functions. Billing for such a service could be based on a fixed price per vCPE function, but it could also be based on the NFVI usage of those functions. For example, if the vCPE service uses a virtualized firewall, the compute capacity and memory used by the firewall instance could be taken into account in the billing process. The architecture depicted above has the right hooks so that the usage information can be associated with the service and filter up to the billing system.

The solution can also be used to introduce forms of dynamic pricing where the price of a service depends on the overall utilization of the infrastructure. The NFVO has an overall view of the utilization of the NFV Infrastructure and can expose that information to OSS systems, which in turn can define the price to be paid for resources at that specific point in time.

18.6 Virtualization and SDN impact on charging

As we assume that existing charging mechanisms need to remain in place during the evolution of LTE and the transition to 5G, this section discusses in detail the impact of SDN and virtualization on the 3GPP Policy and Charging solution.
Figure 18-3 provides a broader view of the relation between billing and charging systems on the one hand and the rest of the management and control infrastructure on the other. As discussed above, OCS and OFCS interface with specific network elements to provide real-time control and collect charging information respectively. In turn, OCS and OFCS provide CDRs to the billing system.

Most of the actions of the OCS and OFCS are determined by provisioning actions from the customer-facing service orchestration functions. For example, if an LTE subscriber signs up for a family plan, all related information (e.g. which UEs are covered by the plan) is provisioned directly to the OCS. In addition, OCS actions may be governed by charging rules that the PCRF pushes to the PGW. The charging rules are provisioned in the Subscriber Policy Repository, which in turn is provisioned by the customer-facing service management systems.

From a functional perspective, virtualization has no impact on the charging systems. PCRF, OCS and OFCS will continue to function in the same way as their physical incarnations, using the same interfaces.

The impact of SDN on mobility and IMS is twofold:

- SDN may be used to control connectivity service between network functions that are part of RAN and/or EPC. In this case, SDN does not affect the mobility or IMS services themselves and therefore does not affect PCC.
- SDN will be used for service chaining on the Gi-LAN. To the extent that billing for these services relies on network usage data, it seems existing data collection solutions in PGW and TDF will suffice, mediated through OCS and OFCS.
The emergence of the Internet-of-Things will result in parallel core networks with different connectivity options. One example is the control-plane-only option. This option applies to devices that have so little data to send that it makes sense to convey all information over the control interface to the MME, without establishing a dedicated data bearer via SGW and PGW. In that case, the MME is the only system that has information about chargeable events. Therefore, the MME will need to support an interface with the charging systems, most likely with the OFCS (which is already covered in 32.251 for SMS usage recording).
19 Service Assurance

Operators have a need for providing Service Assurance (in the form of NFVI SLAs to customers and VNF/NFVI KPIs for internal metrics) when deploying network services in an SDN-NFV environment. However, traditional service assurance mechanisms, which largely comprise a combination of NMS, EMS, and logging functions, is lacking in several areas. Today, service assurance is poorly defined by industry and inconsistently implemented, leading to point solutions with varying degrees of efficacy. For example, SA solutions today are expensive to implement and operate, tend to be statically defined, and are generally isolated from provisioning. Orchestration of workloads and network services, business process management, and virtualization all add new complexity and only serve to exacerbate the problem.

However, virtualization, along with automation and orchestration, provide opportunities for real-time service provisioning to be finally realized. At the same time, open source big data and data science/analytics technologies are beginning to be deployed successfully across the industry (outside of big Web companies) for service and performance management, network assurance, and operational optimization.

This chapter covers service assurance of VNFs and NFVI. Security monitoring and analytics will be addressed at a later date though some aspects are described in Chapter 21.

19.1 Functional View

Figure 19-1 depicts a high level view of the architecture with the Service Assurance component and the other architectural components that it interacts with.
In the figure, EEO is responsible for service provisioning and pushes state to the infrastructure. This is the “C” in FCAPS. SA is responsible for collecting data from the infrastructure, monitoring the infrastructure and analysis of the infrastructure’s performance, security, and availability. This is the “F_APS” in FCAPS. The two components are related as loosely coupled systems that are tightly integrated via APIs and interfaces. For example, part of the EEO function is to activate SA functions to provide service assurance for the network service being orchestrated. In the opposite direction, the EEO function may act on data retrieved from the SA function in order to adjust the service to meet SLA or KPI objectives (ie, reroute traffic across the network).

Data is collected from the network, the NSDs, the VNFDs, and data models that describe the VNFs and PNFs in the network. Examples of these functions include:

- **Service assurance:**
  - Fault management
  - Performance management
  - Incident and problem management

- **Log search**

- **Security and threat analytics**

- **Capacity management**
19.2 Service Assurance Lifecycle

Service Assurance comprises the management and operations functions that are used to ensure that the required service levels and KPIs are met through day-0, day-1 and day-2.

The key notion here is the lifecycle of day-0 (initial provisioning and first sign of life), day-1 (activation and billing commencement), and day-2 (ongoing operations, MACD/CRUD operations, and lifecycle termination). Assuring an SLA is a lifecycle process, as depicted in Figure 19-2 below:

QoE can be maximized by monitoring the service not only for availability, but for QoE. QoE can be measured objectively using analytical methods run across a consistent dataset. In the new network such a dataset will be inherently larger and more distributed, and new tooling is needed to consume, process, and produce information that can be used to measure QoE (an approach to solving this new big data problem is addressed later in the section.) A service is defined as unavailable if QoE falls below thresholds, and steps taken to maximize resulting service availability through proactive problem detection, problem isolation and prioritization according to stack ranking criteria, and proactive resolutions to problems in the network that may in the future affect QoE.

The cost of development is minimized by leveraging open source big data capabilities over a common dataset, collected minimally, via both push and pull models.

The cost of operations can be minimized by making this data available via APIs to a variety of systems with appropriate access controls, thereby allowing for faster MTTR via shorter troubleshooting cycles.
19.3 Service Assurance System Fundamental Principles

A service assurance system is effective only if it meets the following key objectives. It must be measurable. KPIs of the service assurance system must be tracked in terms of how it serves operations. Some key examples of this measurement are:

- Percentage of service impacting issues first identified by SA system – target 100%
  - Percentage of service impacting issues first identified by customer – target 0%
  - Percentage of customer reported faults for which there is a corresponding system detected fault – target 100%

- Percentage of different tickets for which there is a single underlying cause – target 0%
- Percentage of junked tickets, i.e. tickets raised for which there was no action required – target 0%
- Percentage of repeat incidents – target 0%
- Amount of Configuration + customization required
  - Ideally no configuration or customization should be required after day 1, i.e. the system should be self-adapting to changes in infrastructure and services

- Be able to track availability, MTBF, MTTR, and MTTD

A service assurance system must enable operational proactivity. The SA system should know of issues before the customer, and should provide a means to keep the customer informed of status. This is typically handled by ticketing systems, which, by and large, require swivel-chair data entry (i.e., the operator opens a ticket manually and copies data in from the NMS, logs, etc). Additionally, the service assurance system must provide real-time visibility of the status of services, to both the operator, and (with controls) to the customer. Moreover, the SA system should be dynamically adaptable to new service development without redeployment of the system. For example, if a new service is onboarded, the SA system should be able to readily support this new service through updates to data models and schemas, rather than extensive new code. Leveraging YANG data models in concert with the orchestration system that defines assurance parameters, methods, and procedures (including existing interfaces, APIs, and protocols), minimizes new code development while leveraging present capabilities. These data models indeed will be represented in VNFDs (in the case of NFV-driven services) and in the NSD (particularly when classical or white box PNFs are involved in the service delivery). Finally, the service assurance interface should leverage, where possible, a single pane of glass, thus allowing the operator to assure services from one view, minimizing context switching, multiple views and interfaces, and onerous copy/paste data entry sequences via the swiveling chair.
19.4 Primary Functions of a Service Assurance System

This section defines both the primary and secondary functions of SA. A service assurance system, in order to be effective, must perform the following primary baseline functions:

<table>
<thead>
<tr>
<th>Primary Function</th>
<th>Mechanism</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Management</td>
<td>SLA Reporting</td>
<td>Monitor/report whether SLAs are being met (minimize MTTD)</td>
</tr>
<tr>
<td></td>
<td>Performance Analytics</td>
<td>Automatically detect performance anomalies</td>
</tr>
<tr>
<td>Fault Management</td>
<td>Cause Isolation</td>
<td>Identify the underlying causes of faults</td>
</tr>
<tr>
<td></td>
<td>Service Impact Analysis</td>
<td>Identify which services are impacted to prioritize fault resolution and minimize service impact</td>
</tr>
<tr>
<td>Incident and Problem Management</td>
<td>Mapping</td>
<td>Automatically mapping problems to trouble tickets so an engineer can fix them</td>
</tr>
<tr>
<td></td>
<td>Prioritization</td>
<td>Prioritize trouble tickets to target operations efforts based on service impact</td>
</tr>
<tr>
<td></td>
<td>Enrichment</td>
<td>Ensure that operations engineers are informationally equipped to minimize MTTR</td>
</tr>
<tr>
<td></td>
<td>Expertise</td>
<td>Capture expert knowledge to try and prevent future incidents</td>
</tr>
</tbody>
</table>

Table 19-3: Primary Functions of a Service Assurance System

19.5 Functional Architecture

The service assurance functional architecture can be realized by way of several key capabilities. First, it should be cross-domain – rather than siloes for each domain. This is especially important in an NFV context, as services will span domains from the start. It should combine bottom-up and top-down inputs, such that it uses infrastructure level information (e.g. events, logs) to try and build a picture of what’s broken, i.e. through fault management, and, uses application and service level information to determine service health (e.g. through active and passive monitoring, customer issues, social media).

Additionally, the SA solution should automate mapping of customer reported issues to infrastructure issues. Moreover, the SA system leverages both event data and metric data. Event data (traps, logs, etc) records the occurrence of discrete events, and the system needs to cope with both structured and unstructured data, for example, through log parsing, pattern matching, and machine learning methods. Metric data records how a measured variable changes over successive measurements e.g. Utilization of IO / memory / CPU, link utilization / errors, etc. Over time, this data allows the SA system to leverage analytics to minimize dependency on rules and thresholds. Lastly, the solution must maintain a complete, real-time inventory of all functional elements, instead of partial, static or discovered models, and it should
integrate assurance with incident management. The functional architecture for a SA system is shown in Figure 19-4 below:

![Service Assurance Functional Architecture Diagram](image)

Figure 19-4: Service Assurance Functional Architecture

The secondary service assurance functions use analytics to determine anomalies. Different functions apply different analytics to different subsets of data. Performance analytics, for example, automatically detect performance anomalies. Route analytics determine changes in infrastructure and topology that can be used to determine impacts to performance and KPIs. Security analytics can address impacts to customer and system security through threat analytics. There are many other supporting functions. Figure 19-5 below shows an amended view of the functional architecture with example secondary functions.
19.6 Orchestrating Service Assurance

The target SDN-NFV architecture places some emphasis on the role of the orchestration function for both network and resource services. This end to end architecture includes interfaces between EEO and the Service Assurance system. These interfaces are ultimately needed to both orchestrate service assurance parameters and methods as a result of SLA/KPI measures expected of an quantified in the NSD, but also allowing for consumption of events and data by the orchestrator that allow for reactivity to events that can allow for redeployment, optimization, or refactoring of the service. A high level view of this interplay is shown in Figure 19-6 below:
# 20 Key Performance Indicators

Many of the same KPIs that are used today with the traditional PNF functions will be exactly the same on the VNF functions. This is due to the fact that virtualization of the PNF functions only changes how the PNF functions are implemented and deployed. It does not change the fundamental adherence to the 3GPP specifications, which govern the communication between the different network entities. Virtualization, however, does introduce some potentially new KPIs, which measure how well the VNF is utilizing the NFVI resources it has been allocated.

The counter information used to calculate the KPI formulas for each of the VNFs must be delivered to the Service Assurance function so that the KPIs can be calculated and monitored. According to the architecture outlined in this document, the VNFs should deliver their counter information via the Ve-Sa interface described in the interface section of this document. The NFVI KPI information should be delivered via the Cf-N interface described in the interface section of this document.

Information of the various logical interfaces in the network is the key to troubleshooting issues in the core network. When the current PNF elements evolve to VNF elements, those logical interfaces will still be present in the network and need to be accessible for information collection. Probing of the VNF logical interfaces will occur at the networking part of the NFVI via the mirroring of the ports. The mirrored data will be forwarded to the Collection Function and Analytics via the Cf-N interface of Figure 2-2 for processing of the data.

The counters from the VNFs are typically delivered via one of three types of mechanisms:
- Pull from the VNF by the Service Assurance function.
- Push from the VNF to the Service Assurance function.
- Near real-time streaming of data from the VFN to the Service Assurance function.

The pull and push mechanisms can be accomplished via a variety of techniques including but not limited to SNMP, FTP/SFTP, SSH, and Remote Procedure Calls (RPC). Near real-time streaming usually involves the setup of an HTTP, UDP or TCP session between the VNF and Service Assurance function, which allows the VNF to send counter information over an established socket-like connection. Implementation of this functionality varies from vendor to vendor.

The following sections will provide additional detail on KPIs collected in the EPC, IMS, L1-L3, SGi-LAN, EEO, VNFM, VIM, and NFVI domains.

## 20.1 EPC Key Performance Indicators

This section will focus on the KPIs relevant to the EPC VNF applications. In the EPC, the KPIs can be categorized into three basic categories:
- Accessibility
- Retainability
- Traffic Utilization

In the sections below a brief description of each category along with some example KPIs are provided. Each operator has their own perspective on which KPIs are important. In addition, each vendor of EPC elements has different levels of counter and statistical information, which may be collected from their EPC elements. Taking both of those factors into account, the examples provided in the following sections are some of the common KPIs but do not represent all of the KPIs possible. 3GPP 32.426 provide some of the specification defined counters, which measure some of these KPIs.
Accessibility

Accessibility refers to the ability of a service to be obtained within a specified tolerance when requested by the user. Typically, these KPIs focus on the ability to access the individual EPC elements either via session establishment or UE reachability. Examples of accessibility KPIs with a brief description is provided in the table below:

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Failure Ratio</td>
<td>The probability an Attach fails</td>
</tr>
<tr>
<td>Dedicated Bearer Activation Failure Ratio</td>
<td>Probability that a Dedicated Bearer Activation Fails</td>
</tr>
<tr>
<td>Service Request Failure Ratio</td>
<td>Probability that a Service Request Fails</td>
</tr>
<tr>
<td>CS Fallback Failure Ratio</td>
<td>Probability of failure of a CS Fallback</td>
</tr>
<tr>
<td>Modify Bearer Failure Ratio</td>
<td>Probability of failure of a Modify Bearer</td>
</tr>
<tr>
<td>Gx Session Failure Ratio</td>
<td>Probability of failure of a Gx session establishment</td>
</tr>
<tr>
<td>Sx Session Failure Ratio</td>
<td>Probability of failure of a Sx session establishment</td>
</tr>
<tr>
<td>SMS Paging Failure Ratio</td>
<td>Probability of failure of a Page for SMS</td>
</tr>
</tbody>
</table>

Table 20-1: Accessibility KPIs.

Retainability

Retainability refers to the probability that a connection is retained for a communication under a given condition and for a given time duration. Typically these KPIs focus on the success or failure of various Mobility Management procedures found in the MME. Examples of retainability KPIs with a brief description is provided in the table below:

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAU Failure Ratio</td>
<td>Percentage of failed Tracking Area Update Procedures</td>
</tr>
<tr>
<td>Paging Failure Ratio</td>
<td>Probability that a paging procedure fails</td>
</tr>
<tr>
<td>S1-Handover Failure Ratio</td>
<td>Probability that an S1 Handover fails</td>
</tr>
<tr>
<td>X2-Handover Failure Ratio</td>
<td>Probability that an X2 Handover fails</td>
</tr>
</tbody>
</table>

Table 20-2: Retainability KPIs.

Traffic Utilization

Traffic Utilization measurements provide snapshots into the current utilization of an EPC element. Examples of traffic utilization KPIs with a brief description is provided in the table below:
This section describes KPIs for IMS VNF applications. KPIs can be classified for three different groups:

- KPI’s for monitoring the IMS Registration
- KPI’s for monitoring the IMS Sessions (including the handover to CS Voice)
- KPI’s for monitoring the Session quality like VoLTE Call Quality

### IMS Registration

The user needs to successfully register in the IMS before he can use IMS services. Part of the IMS registration is the (USIM/ISIM based) authentication of the user. After the registration of the user, the CSCF sends a 3rd party IMS registration to the AS.

The UE registration in the IMS Core is "always on", and refreshed periodically by the UE. The KPIs for monitoring the IMS registration are listed in the table below:

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attached Users</td>
<td>Number of Currently Attached Users</td>
</tr>
<tr>
<td>PDN Connections</td>
<td>Number of Current PDN Connections</td>
</tr>
<tr>
<td>Bearers</td>
<td>Number of active EPS Bearers</td>
</tr>
<tr>
<td>Throughput</td>
<td>Amount of traffic per logical interface</td>
</tr>
<tr>
<td>Gx Sessions</td>
<td>Number of active Gx Session in PCRF</td>
</tr>
<tr>
<td>CPU utilization</td>
<td>Percentage of assigned processing power a VNFC is using</td>
</tr>
<tr>
<td>Memory utilization</td>
<td>Utilization of assigned memory per VNFC</td>
</tr>
<tr>
<td>Disk space utilization</td>
<td>Utilization of assigned disk space</td>
</tr>
</tbody>
</table>

Table 20-4: IMS Registration KPIs.

### IMS Sessions

Key KPIs for monitoring IMS sessions are listed in the table below. For VoLTE service there is need to link IMS session setup to the setup/drop of the dedicated bearer with GBR QoS (and SRVCC handover to CS voice). These are described in the EPC KPIs section.

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP Initial Registration Success Rate</td>
<td>Rate of successful SIP registration requests versus the total number of SIP registration requests.</td>
</tr>
<tr>
<td>SIP Re-registration Success Rate</td>
<td>Rate of successful SIP registration requests versus the total number of SIP registration requests.</td>
</tr>
<tr>
<td>Currently Fully Registered Subscribers on the CSCF</td>
<td>Total number of the registered subscribers</td>
</tr>
</tbody>
</table>

Table 20-3: Traffic Utilization KPIs.
### KPI Name

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP Session Setup Success Rate</td>
<td>Rate of sent SIP Invite requests for which a corresponding 180 Ringing (or 200OK) was received versus the total number of SIP Invite requests.</td>
</tr>
<tr>
<td>SIP Session Setup Answer Rate</td>
<td>Rate of sent SIP Invite requests for which a corresponding 200OK was received versus the total number of initial SIP Invite requests sent.</td>
</tr>
<tr>
<td>Number of Simultaneous Dialogs on the CSCF</td>
<td>The total number of currently active SIP Invite initiated dialogs maintained on the CSCF.</td>
</tr>
</tbody>
</table>

**Table 20-5: IMS Session KPIs.**

### Session quality

Key KPIs for monitoring IMS session quality are listed in the table below. In addition to control plane quality KPIs there is also Mean Opinion Score (MOS) estimation for user plane quality measurement.

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Session Duration</td>
<td>Average duration of successful sessions in seconds.</td>
</tr>
<tr>
<td>Average MT Session Setup Time</td>
<td>The average duration of the terminating session setup duration in seconds.</td>
</tr>
<tr>
<td>Average MO Session Setup Time</td>
<td>The average duration of the originating session setup duration in seconds.</td>
</tr>
</tbody>
</table>

**Table 20-6: Session Quality KPIs.**

### VoLTE and SMS over IMS

The VoLTE KPIs are collected from several network elements to show end-to-end view. This combination includes the VoLTE session setup, the setup of the dedicated bearer with GBR QoS for voice and the drop rate of the dedicated bearer. In addition, selected KPIs are typically used e.g. the Average SIP Session Duration and the Average SIP Session Setup Time. All elements have more KPIs for VoLTE services, which can be either added to end-to-end monitoring or keep on the element level only. The KPIs for monitoring the VoLTE service are listed in the table below:

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eRAB Setup Success Ratio - QCI1</td>
<td>Rate of successful QCI1 GBR bearer setup in eNB versus the total number of QCI1 GBR bearer setup.</td>
</tr>
<tr>
<td>eRAB Drop Ratio - QCI1</td>
<td>Rate of dropped QCI1 GBR bearers in eNB versus the total number of QCI1 GBR bearers setup.</td>
</tr>
<tr>
<td>Dedicated EPS Bearer Activation Success Ratio</td>
<td>Rate of successful QCI1 GBR bearer setup in MME versus the total number of QCI1 GBR bearer setup.</td>
</tr>
<tr>
<td>SIP Session Setup Success Rate</td>
<td>Rate of sent SIP Invite requests for which a corresponding 180 Ringing (or 200OK) was received versus the total number of SIP Invite requests.</td>
</tr>
<tr>
<td>SIP Session Setup Answer Rate</td>
<td>Rate of sent SIP Invite requests for which a corresponding 180 Ringing (or 200OK) was received versus the total number of SIP Invite requests.</td>
</tr>
</tbody>
</table>
KPI Name | Description
---|---
Number of Simultaneous Dialogs on the CSCF | The total number of currently active SIP Invite initiated dialogs maintained on the CSCF.
Mean Opinion Score (MOS) | Perceptual Evaluation of Speech Quality in the MGW or BGW

Table 20-7: VoLTE and SMS Over IMS KPIs.

Similarly the SMS KPIs are collected from several network elements to show end-to-end view. The KPIs for monitoring the VoLTE service are listed in the table below:

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default EPS Bearer Activation Success Ratio</td>
<td>Rate of successful QCI5 non-GBR bearer setup for SMS in MME versus the total number of QCI5 non-GBR bearer setup for SMS.</td>
</tr>
<tr>
<td>SIP Message (SMS submit) Success Rate</td>
<td>Rate of sent SIP Message requests for which a corresponding SIP message was received versus the total number of SIP Message requests.</td>
</tr>
<tr>
<td>Number of MO/MT SMSs sent on the CSCF</td>
<td>The total number of MO/MT SMS sent on the CSCF.</td>
</tr>
</tbody>
</table>

Table 20-8: SMS KPIs.

### 20.3 SGi-LAN Key Performance Indicators

A Service Chain consists of a set of service functions and the order that must be applied to packets along a service function path. A service function path may be unidirectional or bidirectional. KPI related to SGi-LAN can be viewed in terms of the availability and the performance of the service chain. Table 19-9 lists these KPIs may be used to monitor the performance of the service chain.

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Instance Availability</td>
<td>The availability of the service chain instance itself.</td>
</tr>
<tr>
<td>Packet Loss Ratio</td>
<td>Packet loss across the service chain.</td>
</tr>
<tr>
<td>Average Delay</td>
<td>Average time it takes for a packet to traverse the service function path.</td>
</tr>
<tr>
<td>Average Throughput</td>
<td>Average throughput for traffic traversing a service function path.</td>
</tr>
<tr>
<td>Flows</td>
<td>Total concurrent flows.</td>
</tr>
<tr>
<td>Subscribers</td>
<td>Total concurrent subscribers.</td>
</tr>
</tbody>
</table>

Table 20-9: SGi-LAN KPIs.
20.4 L1-L3 Key Performance Indicators

For any L1 – L3 specific type of VNF that is deployed, there are fixed and limited resources that must be tracked and trended in order to understand the capacity and performance of those functions. In doing so, we must enable monitoring of how those functions are performing on a day to day basis, and whether those devices have the necessary capacity to continue functioning properly. This section explains the various resources that assurance service is aimed at monitoring and trending. Those resources include CPU utilization, memory utilization, buffers and statistics. In this section, we will cover CPU, Memory, and Buffer statistics.

**CPU Utilization**

This KPI is used to monitor and trend CPU consumption on the individual VNF:

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Total Physical</td>
<td>CPU usage of a specific VNF</td>
</tr>
<tr>
<td>CPU Total 5min</td>
<td>The overall CPU busy percentage in the last 5-minute period.</td>
</tr>
<tr>
<td>CPU Process Execution Runtime</td>
<td>The amount of CPU time the process has used,</td>
</tr>
</tbody>
</table>

Table 20-10: CPU Utilization KPIs.

**Memory Utilization**

This KPI is used to provide performance data on memory consumption. For memory consumption of the VNF, we are concerned with processor/application memory due to the fact that this memory space is where all running processes consumption occurs. When the VNF runs out of processor memory, it essentially has reached its scaling limits and is in danger of function failure.

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Pool Type</td>
<td>Determines the type of buffer, whether it is processor Memory or i/o Memory</td>
</tr>
<tr>
<td>Memory Pool Name</td>
<td>Describes whether the memory pool is related to processor or i/o</td>
</tr>
<tr>
<td>Memory Pool Used</td>
<td>Processor memory used and i/o memory used</td>
</tr>
<tr>
<td>Memory Pool Free</td>
<td>Processor memory free and i/o memory free</td>
</tr>
<tr>
<td>Memory Pool Utilization 5 Min</td>
<td>Overall CPU busy percentage in the last 5 minute period.</td>
</tr>
</tbody>
</table>

Table 20-11: Memory Utilization KPIs.

**Buffer**

Buffer KPIs are used to help track and trend the availability of buffers on a particular VNF. The two areas covered are buffer utilization and buffer failures. Trending these two areas will determine buffer leaks or packet drops due to buffer failures.

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Memory Buffer Total</td>
<td>Small memory buffer pool</td>
</tr>
</tbody>
</table>
### Interface Statistics

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Packet Rate</td>
<td>Five minute exponentially decayed moving average of inbound packet rate for this interface.</td>
</tr>
<tr>
<td>Input Octet Rate</td>
<td>Five minute exponentially decayed moving average of inbound octet rate for this interface.</td>
</tr>
<tr>
<td>Output Packet Rate</td>
<td>Five minute exponentially decayed moving average of outbound packet rate for this interface.</td>
</tr>
<tr>
<td>Output Octet Rate</td>
<td>Five minute exponentially decayed moving average of outbound octet rate for this interface.</td>
</tr>
<tr>
<td>Input Runt Errors</td>
<td>The number of packets input on a particular physical interface which were dropped as they were smaller than the minimum allowable physical media limit</td>
</tr>
<tr>
<td>Input Giant Errors</td>
<td>The number of input packets on a particular physical interface, which were dropped as they were larger than the interface MTU (largest permitted size of a packet which can be sent/received on an interface).</td>
</tr>
<tr>
<td>Input Framing Errors</td>
<td>The number of input packets on a physical interface, which were misaligned or had framing errors. This happens when the format of the incoming packet on a physical interface is incorrect.</td>
</tr>
<tr>
<td>Input Overrun Errors</td>
<td>The number of input packets which arrived on a particular physical interface which were too quick for the hardware to receive and hence the receiver ran out of buffers.</td>
</tr>
<tr>
<td>Input Ignore</td>
<td>The number of input packets which were simply ignored by this physical interface due to insufficient resources to handle the incoming packets.</td>
</tr>
<tr>
<td>Input Aborts</td>
<td>Number of input packets, which were dropped because the receiver aborted. Examples of this could be when an abort sequence aborted the input frame or when there is a collision in an ethernet segment.</td>
</tr>
<tr>
<td>Input Queue Drops</td>
<td>The number of input packets, which were dropped. Some reasons why this object could be incremented are: 0 Input queue is full.</td>
</tr>
<tr>
<td>Output Queue Drops</td>
<td>This object indicates the number of output packets dropped by the interface even though no error had been detected to prevent them being transmitted</td>
</tr>
<tr>
<td>Reset count</td>
<td>The number of times the interface was internally reset and brought up</td>
</tr>
<tr>
<td>Carrier Transition Count</td>
<td>Number of times interface saw the carrier signal transition.</td>
</tr>
</tbody>
</table>
### Table 20-13: Interface Statistics KPIs.

<table>
<thead>
<tr>
<th>KPI Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Multicast Packet</td>
<td>The number of packets, delivered by this sub-layer to a higher (sub)layer, which were addressed to a multicast address at this sub-layer.</td>
</tr>
<tr>
<td>Input Broadcast Packets</td>
<td>The number of packets, delivered by this sub-layer to a higher (sub)layer, which were addressed to a broadcast address at this sub-layer.</td>
</tr>
<tr>
<td>Output Multicast Packets</td>
<td>The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent.</td>
</tr>
<tr>
<td>Output Broadcast Packets</td>
<td>The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded or not sent.</td>
</tr>
<tr>
<td>Interface Speed</td>
<td>An estimate of the interface's current bandwidth in bits per second. For interfaces, which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth.</td>
</tr>
<tr>
<td>Interface OperationalStatus</td>
<td>The current operational state of the interface.</td>
</tr>
<tr>
<td>Interface last Change</td>
<td>The value of system Up Time at the time the interface entered its current operational state.</td>
</tr>
<tr>
<td>Input Octets</td>
<td>The total number of octets received on the interface, including framing characters.</td>
</tr>
<tr>
<td>Input Unicast Packets</td>
<td>The number of packets, delivered by this sub-layer to a higher (sub)layer, which were not addressed to a multicast or broadcast address at this sub-layer.</td>
</tr>
<tr>
<td>Input Discards</td>
<td>The number of inbound packets, which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.</td>
</tr>
<tr>
<td>Input Errors</td>
<td>For packet-oriented interfaces, the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.</td>
</tr>
<tr>
<td>Output Octets</td>
<td>The total number of octets transmitted out of the interface, including framing characters.</td>
</tr>
<tr>
<td>Output Unicast Packets</td>
<td>The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.</td>
</tr>
<tr>
<td>Output Discards</td>
<td>The number of outbound packets, which were chosen to be discarded even though no errors had been detected to prevent their being transmitted.</td>
</tr>
<tr>
<td>Output Errors</td>
<td>For packet-oriented interfaces, the number of outbound packets that could not be transmitted because of errors.</td>
</tr>
<tr>
<td>No Buffer</td>
<td>The total number of output packets dropped due to buffer exhaustion.</td>
</tr>
</tbody>
</table>

#### 20.5 NFVI Key Performance Indicators

For any VMs that are deployed on the NFVI, there are fixed and limited resources available from the NFVI that must be tracked and trended in order to understand the capacity and performance to understand how those functions are performing on a day to day basis, and whether those devices have the necessary
capacity to continue functioning properly. This section details the various resources – both for VMs and host servers, that assurance service is aimed at monitoring and trending.

**VM Level Metrics**

These KPIs are used to monitor and trend the performance of the individual VMs running on the NFVI:

<table>
<thead>
<tr>
<th><strong>KPI Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of vCPUs</td>
<td>Number of virtual CPUs allocated to the instance</td>
</tr>
<tr>
<td>CPU Utilization</td>
<td>Average CPU utilisation</td>
</tr>
<tr>
<td>Disk Capacity</td>
<td>The amount of disk that the instance can see</td>
</tr>
<tr>
<td>Disk Usage</td>
<td>The physical size in bytes of the image container on the host</td>
</tr>
<tr>
<td>Ephemeral Disk Size</td>
<td>Size of ephemeral disk</td>
</tr>
<tr>
<td>Disk IOPS</td>
<td>Average disk iops</td>
</tr>
<tr>
<td>Disk Latency</td>
<td>Average disk latency</td>
</tr>
<tr>
<td>Disk Read Rate</td>
<td>Average rate of reads</td>
</tr>
<tr>
<td>Disk Read Request Rate</td>
<td>Average rate of read requests</td>
</tr>
<tr>
<td>Disk Write Rate</td>
<td>Average rate of writes</td>
</tr>
<tr>
<td>Disk Write Request Rate</td>
<td>Average rate of write requests</td>
</tr>
<tr>
<td>Memory</td>
<td>Volume of RAM allocated to the instance</td>
</tr>
<tr>
<td>Memory Resident</td>
<td>Volume of RAM used by the instance on the physical machine</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>Volume of RAM used by the instance from the amount of its allocated memory</td>
</tr>
<tr>
<td>Network Incoming Byte Rate</td>
<td>Average rate of incoming bytes</td>
</tr>
<tr>
<td>Network Incoming Packet Rate</td>
<td>Average rate of incoming packets</td>
</tr>
<tr>
<td>Network Outgoing Byte Rate</td>
<td>Average rate of outgoing bytes</td>
</tr>
<tr>
<td>Network Outgoing Packet Rate</td>
<td>Average rate of outgoing packets</td>
</tr>
</tbody>
</table>

Table 20-14: VM-Level KPIs

**Host Level Metrics**

These KPIs are used to monitor and trend the performance of the host servers. When used along with the VM level metrics, these metrics provide a complete view.

<table>
<thead>
<tr>
<th><strong>KPI Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Frequency</td>
<td>CPU frequency</td>
</tr>
<tr>
<td>CPU Idle Time</td>
<td>CPU idle time</td>
</tr>
<tr>
<td>CPU I/O Wait Time</td>
<td>CPU I/O wait time</td>
</tr>
<tr>
<td>CPU Kernel Time</td>
<td>CPU kernel time</td>
</tr>
<tr>
<td>CPU User Time</td>
<td>CPU user mode time</td>
</tr>
<tr>
<td>Disk Size Total</td>
<td>Total disk size</td>
</tr>
<tr>
<td>Disk Size Used</td>
<td>Used disk size</td>
</tr>
</tbody>
</table>
### KPI Name | Description
--- | ---
Memory Total | Total physical memory size
Memory Used | Used physical memory size
Memory Buffer | Physical memory buffer size
Memory Cached | Cached physical memory size
Memory Swap Available | Available swap space size
Memory Swap Total | Total swap space size
Network Incoming Bytes Rate | Bytes received by network interface
Network Incoming Packet Rate | Number of received datagrams
Network Incoming Errors | Receiving error of network interface
Network Outgoing Byte Rate | Number of sent datagrams
Network Outgoing Packet Rate | Bytes sent by network interface
Network Outgoing Errors | Sending error of network interface
I/O Blocks incoming | Aggregated number of blocks received to block device
I/O Blocks Outgoing | Aggregated number of blocks sent to block device

Table 20-15: Host-Level KPIs

### 20.6 EEO Key Performance Indicators

Key performance indicators in the EEO are derived from the API transactions performed by EEO, VNFM, and VIM components. Additional KPIs like CPU/Memory/Disk utilization also apply to these components, as these components are deployed on standard compute platforms. Each component has KPIs that should be monitored by service assurance functions.

The EEO has a number of interfaces (Or-Vi, Or-Vnfm, Or-Sdnc, Or-Ems, Or-Nf Or-Sa, Vi-VNFM) to other components, each which needs to be monitored to collect KPIs.

KPIs collected on each interface should include:
- Interface transaction rate (by API transaction type)
- Interface availability (successful transactions / total transactions)
- Interface transaction completion time (by API transaction type)

EEO KPIs should include KPIs specific to the EEO software:
- CPU utilization (per software component)
- Memory utilization (per software component)
- Database operations monitoring
  - Create, read, update, and delete transaction rates
  - Create, read, update, and delete average response times
  - Availability (successes/total transactions)

### 20.7 VNF Manager Key Performance Indicators

The transactional interface and software specific KPIs that apply to the EEO also apply to the VNFM and its interfaces (Or-Vnfm, Vi-Vnfm, and Ve-Vnfm). In order to provide lifecycle management functions, the
VNFM also monitors the performance and availability of each VNFC and the virtual links between, which can be tracked as KPIs:

- Number of objects (VNFCs, Virtual Links) under management
- Number of VNFC KPIs that monitored by the VNF-M

The VNFM will likely provide VNF-specific KPIs to other components like service assurance. These are application specific and not supplied in this section.

## 20.8 VIM Key Performance Indicators

In an OpenStack environment, VIM transactions and communications with local agents are performed by the OpenStack controller cluster. CPU and memory utilization are high-level KPIs that can be easily monitored and collected:

- OpenStack Controller CPU Utilization
- OpenStack Controller Memory Utilization

The VIM receives commands from its northbound interfaces to VNFM and EEO to create, read, update, and modify platform-level parameters. The quantity, rate, transaction time, and success rate all indicate overall system health, and should be included as KPIs.

- OpenStack transaction rate (by command type e.g. Nova, Neutron)
- OpenStack availability (Successful transactions / total transactions)
- OpenStack transaction completion time (by command type Nova, Neutron, etc)

![Figure 20-16: OpenStack transaction completion time.](https://wiki.openstack.org/wiki/Rally)

Two OpenStack projects to enable operators to characterize a platform’s performance capabilities to establish a measurable benchmark as modifications are made to the platform over time.

- **OpenStack Tempest** ([http://docs.openstack.org/developer/tempest/overview.html#](http://docs.openstack.org/developer/tempest/overview.html#)) is a set of API validation and scenario-based test cases and test engine to functionally verify an OpenStack platform.
OpenStack Rally (https://wiki.openstack.org/wiki/Rally) is a benchmarking tool that allows operators to benchmark an OpenStack deployment using simulated loads.

OpenStack Rally is typically only used for benchmarking. Rally benchmarks may be run periodically to ensure that SLAs are met. However, in order to provide operators with service-provider level availability monitoring of the VIM, additional service assurance tools are required to monitor critical VIM KPIs over time.
21 Security

21.1 Introduction

Applications are written to be resilient to security issues such as proper discarding of packets when a DDOS attack is detected. Care is also taken to not be a source of security issues inadvertently. From the configuration and management of the application and the infrastructure point of view, role-based access control, attestation, and entitlements defined with varying degrees of configuration privileges. For a virtualization environment, establishing a root of trust is necessary to insure trustworthiness of the platform and the provisioned workloads and functional components in the environment. As an example, configuration of the networking elements is a tightly controlled access because of the scope of the impact that can manifest if an error is made in the configuration or deploying a suspect software image. The sections below describe the various levels and best practices that are used for securing the various layers of the architecture.

In today's networks we deliver application outcomes and those applications need to securely ask the “network” for a specific set of behaviors. The process of securing the application’s requests to the network, treating the network programmatically to deliver an outcome, is detailed in the graphic above. There are 8 layers of the threat vectors associated with securely delivering a service in a network evolved to include SDN, NFV and Virtualization. This is not meant to be a comprehensive list of threats, but more a group of related categories that represent them. Some of these threat vectors are covered in the discussion below. However some are being considered for future study.

![Multiple layers of security to protect Software Defined Network](image-url)

Figure 21-1: Multiple layers of security to protect Software Defined Network
21.2 Platform Security

21.2.1 Domain security considerations

The attack surface for a virtualized, SDN-enabled infrastructure element is different from that of a dedicated network element. For one thing, a workload is no longer tied to a particular server but has the flexibility to run on any server.

Hardware-based Root-of-Trust technologies can be used to ensure the integrity of a compute server, by measuring and validating its firmware and software components before their execution. Only validated components can be executed. Any attempts to inject unauthorized changes through physical (or remote) access may result in an alert and any attempts to run unauthorized firmware and software will fail.

![Figure 21-2: Chain of Trust](image)

21.2.2 Hardware Security Assistance

The table below shows some common hardware technologies related to security and their use cases.

<table>
<thead>
<tr>
<th>Hardware feature</th>
<th>Description</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXT (Trusted Execution Technology)</td>
<td>Processor and chipset enhancements in support of trusted computing</td>
<td>Limit workloads to the servers that are attested secure and located at authorized locations</td>
</tr>
<tr>
<td>TPM (Trusted Platform Module)</td>
<td>A microchip that supports cryptographic operations and shielded storage as specified by the Trusted Computing Group</td>
<td>TPM functions defined by TCG, including Assists in measurement, validation of platform firmware, OS, and workloads at boot. If a SaaS model where underlying hardware is abstracted, virtual TPM should be investigated to trust and attest for the virtual service.</td>
</tr>
</tbody>
</table>
### Table 21-3: Hardware Security Assistance

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fTPM (Firmware TPM)</td>
<td>Ability to use a firmware TPM application running in a trusted execution environment</td>
</tr>
<tr>
<td>UEFI Secure Boot</td>
<td>Platform UEFI BIOS secure boot mechanism.</td>
</tr>
<tr>
<td>AES-NI</td>
<td>A set of instructions specific to AES primitives</td>
</tr>
<tr>
<td>GCM</td>
<td>Set of instructions for accelerating the Galois Counter Mode (PCLMULQDQ)</td>
</tr>
<tr>
<td>QAT</td>
<td>Quick Assist Technology</td>
</tr>
<tr>
<td>DPDK</td>
<td>Packet processing acceleration</td>
</tr>
<tr>
<td>VT (Virtualization Technology)</td>
<td>CPU full-virtualization</td>
</tr>
<tr>
<td>SGX (Software Guard Extensions)</td>
<td>Set of processor instructions that provide hardware based, per-application protected execution environment</td>
</tr>
<tr>
<td>Hardware Root Of Trust</td>
<td>Hardware anchor for providing trust for platform owners</td>
</tr>
</tbody>
</table>

#### 21.2.3 Server UEFI BIOS

At the server level, there are a number of UEFI BIOS settings that should be configured to help secure against unauthorized changes or updates. BIOS settings and capabilities are manufacturer and model specific, so where available the following settings should be configured:

1. **Supervisor password** – This restricts unauthorized access to settings like date & time, boot order and priority, and network settings.
2. **Lock BIOS settings** – To prevent changes to BIOS settings without supervisor password.
3. **Lock BIOS updates** – To prevent BIOS upgrades/downgrades, potentially resetting BIOS to factory settings. (A separate procedure will need to be in place to handle genuine BIOS updates). Protecting BIOS (and, other platform FW, SW) from upgrade/downgrade attacks can be done by performing sign verification of images and preforming security anti-rollback checks at boot.

4. **Disable any unused ports and devices** – Any NICs, USB ports, WiFi, Bluetooth, or any other devices not actually in use should be disabled.

5. **Lock boot order** – To prevent booting from unauthorized sources.

6. **Disable boot device list** – To prevent interactively selecting a boot device from the console.

7. **Verify all BIOS Firmware components at boot** – Use UEFI Secure Boot to verify signatures of all UEFI BIOS firmware components before they are loaded.

### 21.2.4 Installation and Maintenance

Where the OpenStack install process or creation of a tenant calls for a new operating system installation, a best practice is to install the minimum package set necessary to provide the foundation for the applications or services that will run on top of it. In the case of OpenStack, this is achieved by selecting the "Minimal Install" option during a manual installation or by specifying the '@core' package group as part of an automated Kickstart-based installation. Once the minimal operating system is in place, then proceeding with the OpenStack installation process will automatically install only the packages needed and no additional ones. Avoiding the installation of unnecessary packages reduces the attack vector(s) available in the case of security vulnerabilities.

Maintenance will need to be addressed at two levels – regularly scheduled updates, upgrades, and patches that are installed during regular maintenance windows, and unscheduled updates that typically involve errata that address security issues, which must be addressed urgently, usually within a few days of being released. Maintenance windows for scheduled patching or upgrading should be established by the operations team, either quarterly or biannually. These updates should follow well-established change management procedures, including full testing of all updates in a non-production environment before deploying to production, and notification and approvals from all stakeholders prior to any changes to the production environment. Unscheduled security updates will also need a change management process, and additionally will need a person or team monitoring the applicable security advisories. This is necessary in order to learn when security updates become available, their criticality level(s), and then be in a position to make an assessment as to whether a security update is of a high enough criticality to warrant deploying into production outside of the regular maintenance schedule. In both cases, appropriate actions should be taken to avoid any downtime for the VNFs.

There is an ongoing work upstream in the OpenStack community to enable a CI/CD update and an upgrade methodology built on top of tools such as Ansible, Puppet, Gerrit and Jenkins. This work has not yet been formalized into an OpenStack blueprint but should be monitored so that when this capability becomes available it can be utilized. It might be integrated as an OpenStack component or installation and lifecycle management tool.

In all cases, the package installer integrated with the operating system must be configured to verify package signatures by default. For example, Linux and OpenStack packages can be signed with the GPG key or any other security key management system used by the infrastructure owners. If the signature verification fails, it indicates that the package has been altered and should not be installed. Once the new packages have been verified and installed, a best practice is to reboot the server. This ensures that not only kernel and core library updates take effect but also that any processes that would
otherwise have remained in memory using old, potentially insecure versions of updated libraries are cleaned up and restarted.

At both the operating system level and the OpenStack level, Role Based Access Control (RBAC) should be implemented in order to limit the access of any given account to only those actions that are needed by that account. For instance, the “root” account is required to update or upgrade the operating system and the OpenStack infrastructure and should be closely controlled. Remote login as root should be disabled, and the use of the “sudo” facility, which enables execution of specifically-configured operations as root, should be utilized not only to restrict specific access, but also to allow better logging of activity.

In addition to verifying the signatures of the OpenStack and operating system packages, the VNF images to be upgraded or installed should also be verified, where possible. This verification is not only directed at the binary VNF but also at the request to upgrade or install the VNF. Similar RBAC rules should be configured so that only specific accounts are allowed to request that new VNFs be installed or upgraded.

Application accounts will be needed, and their type, number and access will vary depending on the VNF implementation. There may be one account that has supervisor or admin-level access to the entire tenant and all of the VNFs, networks, security groups, and other components. There may also be individual accounts that only have access to specific VNFs. There may also be a service account that is used for programmatic control of some or all of the tenant or project. These accounts should be carefully considered from an access perspective when defining the VNF implementation.

21.2.5 Nova Compute Server Security

Section 3.2.2 covered security and trust aspects for CPUs and chipsets. This section covers security aspects of Compute Servers.

Certain workloads may be required to run only on compute servers that are proven secure. To this end, a pool of servers supporting trusted computing and remote attestation can be set up in an OpenStack-based NFVI. Specifically, each server is enabled with Intel TXT and a TPM, among other things. Intel TXT provides Root of Trust for Measurement (RTM) to take measurements (pertaining to the integrity and authenticity of various firmware and software components). The TPM provides Root of Trust for Storage (RTS) to securely store data (e.g. signing keys and measurements) and Root of Trust for Reporting (RTR) to attest to and report on certain non-confidential contents of RTS. These hardware-based roots of trust are inherently trusted.

When a trusted server boots, various firmware and software components (e.g. the BIOS, PCI ROM, boot loader, and hypervisor) as well as their configurations are verified for digital signatures to ensure they are authorized to install by platform owners, measured and measurements verified against tampering step by step. A server will boot successfully with full functionality only if all the components pass digital signature checks up the chain of trust. In addition, measurement in each step must pass verification. So the server, once up and running, is assured to be in a known good state, with a chain of trust rooted in hardware. The chain of trust may be dynamic so that the hypervisor (or operating system) can initiate measurements, independent of booting. It may also be extended to cover VMs. Here a key building block is an attestation authority that can validate measurements against Verizon’s policy. It goes without saying that the attestation authority also needs to be trusted.

In general, the OAMP aspects of a large-scale, carrier-grade deployment of trusted computing and remote attestation technologies are still not well understood and require further study. A particular challenge is to manage known good measurements and policy in step with the latest firmware and
software updates. The slow performance of typical TPMs should also be taken into account. This may limit TPM usage in SDN/NFV. Finally, the NFVI should support TPM 2.0 rather than its predecessor TPM 1.2. To say the least, TPM 1.2 supports only SHA-1 and RSA, while TPM 2.0 supports, in addition, a list of more robust algorithms (e.g. SHA256 and ECC) as well as cryptographic agility.

The UEFI Secure Boot is defined by TCG, and a stable implementation is currently in open source. UEFI Secure Boot has been installed and shipping broadly by OEMs and ODMs on millions of devices.

### 21.3 Network Communication

The security of internal network traffic is first considered. This includes communication between the OpenStack services on the controller nodes (i.e. Keystone, Glance, Cinder, Neutron, etc.), as well as database network access, messaging queues, and traffic between tenants on private networks. A number of steps should be taken to protect this internal traffic in order to minimize the damage an attacker can do should they compromise a server (physical or virtual) and gain access to the internal network.

There are the following types of internal traffic that need to be protected:

1. **Messaging** – Some OpenStack services communicate with each other using a message broker and queue. Communication with the broker should be secured using TLS (Transport Layer Security) to prevent eavesdropping or impersonation on the network. In addition, the OpenStack environment should utilize a message broker implementation that supports X.509 client certificates to authenticate to the broker, which is strongly preferred over password authentication.

2. **Database** – OpenStack services use underlying databases for persistent data and metadata storage. These communications should also be configured to use TLS and client X.509 certificates to prevent eavesdropping and password hacking.

3. **Libvirt** – Access across the internal network by this facility is needed during live migration operations. If enabled, this communication should also be configured to use TLS and either Kerberos or X.509 client certificates for authentication.

4. **API endpoints** – In addition to messaging, some OpenStack services communicate with each other using internal HTTP endpoints. All OpenStack API endpoints should be configured to use TLS in order to secure their contents.

5. **Tenants and Projects** – Internal tenant traffic is logically isolated through the use of tunnel encapsulation (VxLAN, VLAN, GRE.) In addition, Security Groups should be used in order to limit network access into or out of a tenant VNF to specific port(s) or other tenant(s) in the same project. When a Security Group is created, firewall rules are put in place that track network traffic at the kernel level and only allow traffic specified by the Security Group.

6. **Inter-Compute Node and Inter-VNF Traffic Protection**: In certain deployment scenarios, it might be required to provide confidentiality and integrity protection to all traffic between various compute nodes, and in some scenarios protect all traffic between (including tenant’s) VNFs. There is ongoing effort in OpenStack and ETSI NFV to extend VPNaaS to address such use cases.

One type of external network access can come from an OpenStack user connecting to OpenStack services. This could be connecting to a Horizon dashboard via a web browser, or perhaps command-line tools to review configuration settings, or API endpoints, which could be called by anything from an in-
house custom reporting program to a full blown Cloud Management Platform. These are all done via HTTP API calls and it is critical that these be configured to use TLS since this traffic flows to and from the external network. Users send authentication credentials and authorization tokens over these HTTP connections, so a lack of encryption would make it easy to intercept this critical data. A common OpenStack configuration uses TLS termination at a load balancer that sits in front of the OpenStack services, however, best practice from a security perspective is to have the TLS travel all the way to the OpenStack receiving service. In addition, secure web browser access should be augmented by HTTP Strict Transport Security (HSTS), which provides protection from some men-in-the-middle types of attacks not secured against by HTTPS alone.

The other type of external network access comes from tenant VNFs that send or receive data to/from the outside world. It is up to the individual VNF owners and creators to ensure that the contents of these communications are secured. The table below lists the current VNFs and their secure communication methods:

3GPP calls for the use of IPsec on any logical interface in which traffic needs to be secured. Many vendors also supplement this with the use of access lists which can be setup to allow or deny profiles of traffic on a per interface basis. Interfaces are then secured by perimeter security mechanisms such as firewalls, routing VRFs, and access lists on both the routers and then sometimes the network elements themselves.

One common method for virtual machines in an OpenStack environment to communicate externally is through floating IP addresses. Steps should be taken through account access and tenant configuration to ensure that only those VNFs that require external communication are allowed to associate floating IP addresses.

### 21.4 Network Security Monitoring and Management

Network security measures such as live network security monitoring, real time network behavioural analysis, intrusion detection/protection and external firewalls may also be implemented in the network. Network security monitoring allows the infrastructure provider and/or tenants to monitor live traffic across the VNFs as well as at Management, Control and Data planes and gain "visibility" into the dynamic virtual overlay networks in NFV deployments.

Security Monitoring may require securely delivering monitoring policy into a Security Monitoring Agent (running as part of a VM or Container) on the platform, and securely exposing per-policy traffic to the appropriate network traffic analytics engines. Sflow, Netflow/IPFIX and raw packets formats may be used, along with some meta-data monitoring specific to the analytics engines.

In addition to security monitoring agents, IDS/IPS and firewalls may also be considered per security policy when using SR-IOV and PCI passthrough capabilities for tenant traffic, since internal kernel and OpenStack security measures are bypassed.

Security management includes security lifecycle management, which includes security planning and enforcement. This ensures that security policies are consistently designed and provisioned across virtual, physical and infrastructure security functions. The security policies can also be tuned dynamically based on the intelligence provided through security monitoring and policy compliance can be assured end to end.
Big Data and analytics may be used to actively correlate NFV traffic patterns. This enables the Control and Management plane policies to be correlated with Data traffic, and helps identify anomalies in the NFV networks. Analytics engines may analyze, among other things, aggregated network logs to search for attacks or other hostile patterns. Big Data analytics have the potential to significantly reduce the time to correlate and consolidate diverse network information and provide a more rapid path to actionable security intelligence. This is most easily accomplished when central logging is implemented (see Audit and Logging section). On-going industry activities such as ETSI NFV Security Working Group work will be considered in developing implementation architectures.

21.5 Authentication and Access Management

Users of an OpenStack cloud authenticate to the Identity service (Keystone), which gives them an authorization token that grants them access to the rest of the OpenStack services that they are allowed to use for a limited period of time. The authentication mechanism used is typically local passwords stored in Keystone. Keystone should therefore be configured to use an LDAP server, which in turn should be configured to enforce strict password policies (syntax & dictionary checking, history, rotation, special characters, etc.). While Keystone can store passwords, it has no mechanism or support for any type of password policy.

Keystone is not restricted to using an LDAP server as the only method of external authorization verification. It can also federate to a SAML 2 identity provider, allowing enterprises that already have a SAML 2 infrastructure to leverage the existing authentication and access policies in place.

It is also important to evaluate the policy files for all deployed OpenStack services to ensure that they meet the needs of the particular deployment. The policy files determine what API endpoints a given token can access as well as what operations it is allowed to perform. The default policies are designed to be secure for the average OpenStack deployment, but should be reviewed and customized for the particular SDN-NFV environment.

Available throughout in the OpenStack environment is a mandatory access control facility called SELinux, which is short for Security Enhanced Linux. SELinux checks for allowed access and operations after the usual discretionary access controls are checked. This allows for much tighter security and access, since all files, directories, processes, and even objects like network ports have a SELinux context and associated policies that determine allowed access at a system level. In addition, there is a companion facility called sVirt that extends SELinux specifically to a virtualized environment to contain and isolate the virtual machines on a hypervisor. In the event of vulnerability, like the recent VENOM which allowed a guest virtual machine to break out to the hypervisor, sVirt and SELinux prevent non-authorized access from the guest out to the hypervisor and kernel as well as preventing access to other guest virtual machines. At the very least, SELinux should be configured to use the “Enforcing” mode on all of the Nova compute nodes in the OpenStack environment, if not all of the controller nodes as well. Other Mandatory Access Control facilities such as AppArmor may also be encountered and should be similarly configured.

21.6 Security in the policy framework

Each layer of the architecture has a related security policy data model and control. Security policy in an NFV & SDN context is deployed as a controller of controllers. An example of this is where Openstack has an identity control in Keystone. For a service and all of its trust boundaries, overall policy control occurs when the policy control for each element like Openstack with Keystone is treated as a part of an aggregate system and that system in protected by an overall policy control. That policy control can take
different forms. The first form is a system that delivers policy control for the NFV/SDN solution. The second form is where each part of the system delivers control to a "parent" system via the use of an API. Deployment of the policy control is use case dependent. The expectation is that each VNF, management functions, orchestration components, VM Lifecycle Managers and other components will all have an API exposed for the "manager of managers" for policy to use.

Analytics plays an important role in the overall security architecture shown in figure 21-1. Analytics allows the operator to gather information about the behavior of the network so it can be baselined to determine what "normal" behaviour looks like. Anomalous events can then be determined by indentation of behaviour (either learned or known via signature) that is outside of policy, perhaps a certain number of standard deviations from the “norm.” One of the key focus areas of NFV is orchestrating the placement of security controls at the right time, at the right place, as fast as possible to mitigate a threat as close to the source as possible, minimizing collateral damage. There is a bi-directional relationship between analytics and policy. As analytics learns of an “anomalous” event, policy is then triggered to mitigate the threat (or to decide if the threat is serious enough to warrant further action). After the attack is mitigated, the policy database is updated so the system is better capable of handling the threat the next time.

Security policy in the context of the SDN-NFV architecture faces structure and organizational challenges. In many networks, policy is controlled in administrative domains. For example an IT domain, a mobility domain, a fixed line domain and perhaps there’s a business services domain. The shared resource provided by data centers, networks and clouds that are supporting these domains face a formidable challenge in integrating these domains into a single policy framework. One of the benefits of SDN-NFV is elastic deployment of security controls across the entire data center or cloud fabric-inclusive of the WAN’s that connect them. Workload placement is made possible by a consistent, converged, multi-domain policy layer.

21.7 Stored Data Security

The first aspect of data security is operational – define a comprehensive backup and restore policy and process. Regular backups of critical data should be made, stored in a secure facility locally, and rotated out to a remote facility on a regular basis. Periodic testing should be done to verify that the backups are viable and can be restored as expected. This capability is vital to the ongoing success of the environment. Whether data is compromised by a cyber-attack, hardware failure, user error, or any other reason, valid backups are critical to being able to restore the SDN-NFV OpenStack environment to a fully functioning state quickly. While the VNFS and other VMs in the environment may or may not have any persistent data, there will be critical data from the infrastructure side of the OpenStack environment if not from the application/tenant side. It is important to identify these data and review periodically to ensure that all necessary data are being included.

Another component of data security is confidentiality protection. If the VNF tenants do need to store persistent data or even need temporary disk/volume storage, those storage locations should be encrypted. Encryption can happen at a number of levels for different storage types. Entire volumes can be encrypted at creation time, supported by a back-end key store for enhanced security. Volume data within iSCSI packets can be encrypted. Object Storage can also be encrypted, but only at the disk level at this time, not individual objects or per-object encryption. Key management for encrypted storage is currently a manual process, and access to the key file must be tightly controlled. There is an OpenStack project called Barbican that can eventually act as both a keeper of encryption keys and X.509 certificates. As of the Liberty Design Summit it is still outside of the main tree and not an integrated project, but is making progress.
When removing or deleting volumes, steps must be taken to ensure that sensitive information is not left behind for potential unauthorized access. For unencrypted volumes, Cinder provides two methods for wiping the underlying volume. The first uses the Linux utility “shred” to write random data over the volume three times. The other uses the Linux “dd” utility to write zeros over the volume. Encrypting the underlying volume reduces the need for either of these methods, since the volume erasure is now reduced to the action of securely deleting the encryption keys.

It is important to note that data deletion needs to take into account the type of the storage medium. Common methods (e.g., the one based on “shred”) assume that data are written in place. They are ineffective for solid-state drives where write operations are distributed evenly to avoid localized wearing. Nevertheless, given their high I/O throughput, solid state drives are of particular relevance to NFV. To further complicate the matter is that NFV applications are subject to strict privacy and security regulations. Hence, the OpenStack security study by the ETSI NFV ISG has identified the need for supporting secure deletion in OpenStack. Secure deletion can provide assurance that deleted data in shared storage cannot be recovered, regardless of the type of the storage medium.

21.8 Audit and Logging

Each OpenStack server has multiple logs that are constantly being written to as requests are submitted, actions taken, and events or errors occurring. The OpenStack environment should be configured to log to a central logging server. This provides two distinct benefits. First, from an operational perspective, if there is an incident it becomes much easier to investigate and piece together exactly what happened if all of the logs are located on the same server instead of having to hunt across several or possibly several dozen servers. Second, if an attacker is able to gain access to a server, they may be able to alter or remove the local logs but there would still be logs on the central server. The attacker would have to successfully attack and gain access to the log server in order to manipulate or remove logs there.

21.9 Container Security Considerations

Containers present a tremendous opportunity to an SDN/NFV environment for a number of reasons. To name a few, they are small and lightweight, not requiring the overhead of full virtualization, they are much faster to instantiate, and allow for a greater density per server. However, containers are still an emerging technology and are not ready for a mission-critical, production deployment. There are still security aspects that can be taken into consideration today to prepare for their deployment in the future.

The first is to improve User ID and Group ID (UID/GID) security by mapping the UIDs and GIDs in a container to a different range than the host system. For example, map UIDs 0 – 1000 on the host system to 60,000 – 61,000 in the container. In this case, UID 0 (root) in the container would be seen as 60,000 on the host system. Multiple ranges of UIDs and GIDs could be configured for multiple containers to avoid overlap, or could be configured identically for containers that required identical access.

Additionally, where SELinux is used, a single container and all of the services running inside have a single SELinux context. This means that if one service were to be compromised, any other services running inside the container would also be accessible and not protected by SELinux as would be the case with normal processes. For this reason, each Container should run only a single service.

Also, from an access control standpoint, RBAC access should be defined so that only certain roles can perform certain actions. For example, Admin A can only start/stop certain containers, Admin B can create containers anywhere in a project, and Admin C can create containers anywhere in the environment. Like
other activity within the OpenStack environment, all of these actions should be logged both locally and to a remote logging server.

For Docker containers, another security option is to use Seccomp. This is a facility originally developed by Google for removing system calls from a process. There are several system calls that are seldom or never called from a container, which are candidates for being removed from even being accessible using Seccomp.

VM security is discussed in section 11.2.5.

21.10 Lawful Intercept Considerations

Lawful Intercept (LI) requirements and standards for NFV are still under consideration and are not fully developed. In addition, there are national specifics that will have to be taken into account when implementing LI. This section is intended to provide a summary of the current state of LI definition for NFV.

Lawful Intercept implications on NFV are addressed in the draft report ETSI GS SEC 004 and currently the LI architecture is under development at ETSI NFV SEC 011. The following points are applicable to the current document:

1. There are no NFV-specific LI obligations; the Network Operator is obligated to collect required data, Intercept Related Information (IRI) and Content of Communication (CC), as defined by ETSI TS 101 331 and hand it over to LEA using Handover Interface as defined by ETSI TS 101 671.
2. The definition of a target identity might need to be specified in the NFV context.
3. IRI and CC might need to be defined in the context of virtualized functions and their management, where virtualized functions are part of the offered services.
4. Point of Interception (PoI) needs to be identified for every component of the virtualized infrastructure.
   a. Root of trust for LI needs to be compatible with general NFV infrastructure to allow instantiation of LI specific functions.
5. Capabilities need to be defined and developed to accept and process interception orders and to invoke and deactivate interception per the details of received order. This will require:
   a. Infrastructure support for verification and dissemination of LI certificates.
   b. Existence of an authorized user capable of instantiating LI service.
      i. The user would be different from the traditional admin or root user.
      ii. RBAC rules have to allow explicit exclusion of traditional superuser, “admin” or “root” users, from access to LI-related data, resources and services.
   c. An ability to identify all communication pertaining to the target identity.
   d. Ability to secure LI service related audit and log records with access control allowed only to the authorized user.
   e. Ability to isolate LI service from all other services so it does not introduce any modifications or indications of having been instantiated.
   f. Localization determination and enforcement, in order to both limit the applicability of LI orders to the communications within appropriate jurisdiction, to instantiate LI service within correct
jurisdictions, and to limit spread of target communications (i.e. VM instantiation) to the defined location and jurisdiction.

21.11 SDN Security

Software Defined Network is comprised of three layers: forwarding plane, control plane and application plane. Each of these layers needs to be secured, as well as the interconnect points between them. The following sections address each of the applicable security points. It is assumed that all platforms for SDN and NFV will implement the platform security features discussed in previous sections above.

![SDN Security: Three Plane Architecture](image)

**21.11.1 Control Plane Security**

SDNControllers make up the control plane and are the most crucial components of the SDN network. As such, controllers need to be completely secured, both physically and logically. SDN Controllers can be deployed in a variety of configurations, i.e. as standalone hardware or virtualized appliances, in high availability clusters, or as a federated controller. Security needs to be assured in each of these deployment scenarios.

First-level of security of the controller itself would be provided through securing the server hosting the SDN controller by following the best practices. Next, the Operating System of the controller needs to be hardened and protected. Some of the applicable security measures are addressed in detail in section 18.2.2 covering installation and maintenance. To summarize, SDN Controllers should be installed, updated and upgraded using only trusted and verified packages. No unnecessary applications should be allowed to run in the same environment as the controllers. Administrative interfaces should be available only on an isolated, private network, ideally behind demilitarized zones. Strong authentication and authorization mechanisms should be used. All transactions performed by the controllers as well as all access events should be logged and best practices for log management should be followed.

SDN controllers deployed in HA clusters or federated fashion need to have a mechanism to identify a trusted peer, through certificates or key exchanges. In a federated controller scenario additional trust levels might have to be identified and enforced, i.e. for intra- and inter-domain. The messaging layer for the clustering should implement security measures appropriate to the messaging mechanism.
example is an OpenDaylight recommendation to enable Infinispan’s Jgroups AUTH and ENCRYPT support.

Next, communications between SDN Controllers and the other two SDN layers need to be secured. Southbound interfaces (names of the interfaces) from the SDN Controller to the Forwarding Plane use various protocols, some potentially with their own security measures. Protocol examples include but are not limited to OpenFlow, OVSDB, Path Computation Element Communication Protocol (PCEP), BGP-LS, OpenStack Neutron API, NETCONF, Simple Network Management Protocol (SNMP), and CLI. Available security measures for the corresponding protocol should always be utilized. Communication should occur over a secure transport, such as TLS, DTLS or IPsec.

Northbound APIs from the SDN controller plane to the Application Plane also need to be hardened. Possible threats that need to be addressed are DoS attacks (which will have more serious consequences because of centralized control) as well as the attacks that aim at gaining control over the control plane in order to make modifications to the forwarding plane. Northbound applications, including various Orchestrators, have to use TLS for secure communication with the controllers; their access has to be logged and the level of access regulated via RBAC. These applications should not be allowed administration-level access to the controllers and they need to be able to use certificates to confirm their identity.

In cases when there is any external access allowed to the SDN controller, that access needs to be closely regulated by authentication and access management and protected by firewalls and IDS/IPS. Administrative-level privileges should be disabled for external access.

21.11.2 Forwarding Plane Security

Forwarding elements need to be prevented from attaching to incorrect controllers, whether malicious or not. Some recommendations on automatic bootstrapping and the use of device registries are outlined in (reference?) and should be applied for the SDN forwarding plane setup. Additionally, securing the Northbound interface to the Control Plane, as described in the previous section will prevent spoofing and man-in-the-middle attacks affecting flows or routing tables.

In cases where the forwarding plane has the option of configuration using legacy routing protocols, i.e. in a hybrid environment, existing security measures should be implemented as well as the ability to log the source of changes.

21.11.3 Application Plane Security

Requirements for the Southbound Application APIs (interface names) have been described in the section that addressed Control Plane Northbound interface (interface name). In addition, where the application plane element possesses its own Northbound interface, the interface needs to have all the same security protections, e.g. TLS, RBAC, certificate support, etc. as discussed earlier.


21.11.4 Security at the SDN/NFV enabled gateways

The most common mobile (3G/4G) vulnerabilities will still exist and will increase with 5G and SDN-NFV architectures that decompose and drive toward separate user and control planes across the data center.
Some of the most common of these vulnerabilities include: User plane exploitation, Cascading impacts – user to control plane, Control (signaling) plane exploitation, Network Management Layer exploitation and Gateway Device Compromise.

A gateway in this context is an element reflecting the last hop in the network before the endpoint, or the boundary between two domains of control. For instance, an (v)eNB, SmallCell, Femtocell, or Home Gateway would be considered gateways. All such elements are virtualizing and adopting SDN/NFV technologies.

With SDN and NFV, it becomes viable to deploy protections against user-plane threats and threats that Cascade to signaling (L2/L3) at the gateway, through security VNFs deployed on the gateway; for instance, IPS or FW capabilities compatible with KVM and using hardware-based accelerations.

Security protections against user-plane for the vEPC/RAN deployed through security VNFs deployed on the same platforms and “chained” to the PGW or IMS assets like VOLTE.

Gateway security is most effective against mobile-based attacks like DDOS and reconnaissance from mobiles. The assumption is that legacy security controls are already in place on the PGW, stopping attacks from the Internet into the EPC. Such capabilities are routinely deployed in 3G/4G networks, however, they are far more efficient and viable in SDN/NFV environments. This also means that security threats on L3 traffic might be expanded, where they currently are most often only applied on inbound traffic from the Internet, not outbound traffic from malicious or infected mobile devices.

In such a design, IPS or FW technologies are deployed in “monitoring mode” on a network tap on the inside of the PGW, performing GTP de-encapsulation, or on the outside of the PGW before NAT has occurred. Alternately, security services like IPS and FW might be deployed in-line in active mode – according to the benefits in CAPEX and OPEX achieved by cleansing outbound traffic.

In additional to operational security benefits, gateways also offer value-added service potential for advanced services.

21.11.5 SDN Security Controller

The security vulnerabilities and value-added services which had previously been silo’d across edge, mobile core and cloud data center are now required or offered as a service via any location in a cloud model. As a result, a comprehensive Security Management architecture in alignment with the SDN/NFV architecture should be considered to mitigate and address the vulnerabilities in a hybrid environment where services cover both physical and virtual infrastructure.
The SDN-NFV architecture may benefit with the inclusion of a security controller. The inclusion of a security controller provides the following benefits:

- Orchestration for N Security Services across M Virtualized Datacenters
- Abstracts the virtualization platform from the point solutions
- Management and orchestration of distributed security solution
- Requires no network or workload-VM changes
- Non-disruptive delivery of security to workload VMs
- Seamless integration with Virtualized Platforms
- Zero service administration (automatic synchronization)

Comprehensive security policy and visibility lends itself to achieve the desired automation benefits resulting from the SDN/NFV architecture. A centralized security controller framework provides the ability to abstract security infrastructure, inject services based on policy in workflow, and be extensible to add orchestrators and VNFs, with protection and remediation that is scalable across distributed data centers and distributed NFV/SDN architectures.

### 21.12 Additional Considerations

To address SDN/NFV security end-to-end, the following aspects need to be considered as well:

- Management and Orchestration Domain Security and VNFM Security
  A large percentage of what’s new about the threat surface introduced to networks and services that run SDN-NFV has to do with an aspect of management. Management and Orchestration components are defined in the ETSI MANO architecture and are discussed in this document in detail. Figure 21-1 shows the strategic placement of the Orchestration component in the security architecture. There are a number of key concerns, each of which merits further consideration. They are summarized here.
  - Protection of PII and confidential data stored in service and device configuration files which includes an identity token mechanism to preserve and protect customer data from service portals through to the BSS where the PII is stored (just one example of many).
  - Protection of the API from the orchestration to the VM Lifecycle manager using an “API Gateway.”
  - Providing a “trust and attestation” layer to the provisioning process to insure device integrity.
- Identity “manager of managers” across the solution domains providing consistent policy control and RBAC (role based access control).
- Extend the existing DCN (management network) into the SDN-NFV space providing protected in-band or out of band connectivity preserving the integrity of the management network (or management service, MANO) itself.

While introducing additional threat surfaces, SDN/NFV also creates new security opportunities. In particular, it becomes possible to automate responses to attacks (which may be predicted through security analytics). Some examples are as follows:

- Deploying Security Monitoring Agents across SDN and NFV nodes that enable visibility of dynamic signaling and data networks, across Service Function Chains (SFCs). This enables network anomaly detection including malware and misbehaving VNFs.
- Virtual firewalls are instantiated and inserted into a service chain as needed by applications. Moreover, firewall rules are modified in real-time to adapt to suspected or detected attacks.
- In case of denial-of-service attacks, virtual anti-DDoS scrubbing functions are dynamically instantiated near the sources. Traffic is automatically routed through these functions. In addition, shaping and prioritization rules are dynamically tuned to give priority to legitimate traffic and minimize the resources consumed by nefarious or suspicious traffic.
- Virtual honeypots and honeynets are dynamically provisioned to serve as sinks to malicious traffic (such as for probing and vulnerability scanning), transparently to attackers.
- Upon detection or suspicion of a VM being compromised, the VM is automatically quarantined or migrated to particular hosts for assessment and remedial action, while a healthy VM is created to seamlessly take over the work.
- API Gateway/Security and compensating controls (Web Application Firewalls and other controls to be considered). VNF security is listed above but the security risk of losing control of the API between the Orchestrator and the VNFM (just one example) could be very disruptive to network operations.
- Remote attestation of the “service chain” itself and not just VM or VNFs
- The role and implementation scenarios for a virtual TPM
- The many touch points for PKI/Cryptography should be considered – both for VNF’s (services like SSL VPN) and for securing the SDN controller and related components in the infrastructure (especially as you get toward Internet of Things models).
- For the DDoS use case above, the agility of having a virtual DDoS scrubbing asset is emphasized. For example, BGP-LS could be used at the MD-SAL layer of an SDN controller to gather network topology information. That information could then be used to analyze the impact of mitigation actions taken, giving the operator a better view of what the mitigation might do to the critical traffic classes before the mitigation action is taken.
- Cross data center workload placement algorithm impact on the delivery of “network aware” distributed security outcomes (chains of services).

As indicated in previous sections, NFV/SDN also simplifies deployment of security updates. An upgraded instance of a VNF can be launched and tested while the previous instance remains active. Services and customers can be migrated to the upgraded instance over a period of time (shorter or longer as dictated by operational needs). Once migration is complete, the older instance is retired.
22 Devices and Applications

As Verizon moves towards a dynamic network architecture using NFV and SDN, the goal of this section of the architecture document is to address the impact on devices and related applications composed of services. Services discussed in this section of the document include:

- Video services.
- Internet of Things (IoT) services.
- Future 5G services.
- Consumer Wireline Services
- Enterprise Wireline Services

For each of the above services, this document discusses the placement of services and the impact on devices and applications, including APIs.

Video Services

Video traffic makes up the largest volume of the data traffic in the LTE and FTTH networks today. Youtube video streaming dominates the majority of the traffic along with video rentals (e.g. Netflix and Hulu) and Facebook video. Other video services can include massive events such as the Super Bowl or World Cup, creating surges in traffic.

A video stream requires a sustained high bandwidth session lasting several minutes. Video quality is impacted by packet throughput as well as loss and jitter, all of which adversely affect video delivery. The ability to deploy and dynamically allocate VNFs in the network must consider the impact of video services.

Creating caching VNFs at the edge of the network (edge caching) can reduce the impact on backhaul and improve the response time. Caching of non-encrypted content is available today but as more content providers use encrypted technologies (e.g. SSL in the case of YouTube, Facebook and soon Netflix), collaboration with content and over the top providers is required to support caching and other optimizations for encrypted content. Preloading on the handset is another option which eliminates network related overhead.

3GPP defines the following requirements for QCI values and packet delay budget for video services that would need to be maintained in NFV/SDN architecture. The packet delay budget is provided for Guaranteed Bit Rate (GBR) and Non-Guaranteed Bit Rates.
SDN-NFV Reference Architecture v1.0

<table>
<thead>
<tr>
<th>QCI</th>
<th>Resource Type</th>
<th>Priority Level</th>
<th>Packet Delay Budget (one way packet delay from the UE to the LTE PGW)</th>
<th>Packet Error Loss Rate</th>
<th>Example Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>GBR</td>
<td>4</td>
<td>150 ms</td>
<td>$10^3$</td>
<td>Conversational Video (Live Streaming)</td>
</tr>
<tr>
<td>3</td>
<td>GBR</td>
<td>3</td>
<td>50 ms</td>
<td>$10^3$</td>
<td>Real Time Gaming</td>
</tr>
<tr>
<td>4</td>
<td>Non-GBR</td>
<td>5</td>
<td>300 ms</td>
<td>$10^6$</td>
<td>Non-Conversational Video (Buffered Streaming)</td>
</tr>
<tr>
<td>6</td>
<td>Non-GBR</td>
<td>6</td>
<td>300 ms</td>
<td>$10^6$</td>
<td>Video (Buffered Streaming) TCP-based (e.g., www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)</td>
</tr>
<tr>
<td>7</td>
<td>Non-GBR</td>
<td>7</td>
<td>100 ms</td>
<td>$10^3$</td>
<td>Voice, Video (Live Streaming) Interactive Gaming</td>
</tr>
<tr>
<td>8</td>
<td>Non-GBR</td>
<td>8</td>
<td>300 ms</td>
<td>$10^6$</td>
<td>Video (Buffered Streaming) TCP-based (e.g., www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)</td>
</tr>
<tr>
<td>9</td>
<td>Non-GBR</td>
<td>9</td>
<td>300 ms</td>
<td>$10^6$</td>
<td>Video (Buffered Streaming) TCP-based (e.g., www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)</td>
</tr>
</tbody>
</table>

Table 22-1: Video QCI Values Defined by 3GPP

3GPP TS 23.203 specifications provide latency budgets. The table below shows typical packet processing latency at each node in the 4G user plane and the budget available for backhaul.

![Diagram of packet processing latency]

<table>
<thead>
<tr>
<th>Video Traffic Class</th>
<th>U-plane end-to-end latency</th>
<th>X</th>
<th>Y</th>
<th>XX</th>
<th>YY</th>
<th>a + c + d (air + backhaul budget)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversational</td>
<td>150 ms</td>
<td>1 ms</td>
<td>3 ms</td>
<td>1 ms</td>
<td>1 ms</td>
<td>&lt; 144 ms</td>
</tr>
<tr>
<td>Interactive</td>
<td>100 ms</td>
<td>1 ms</td>
<td>3 ms</td>
<td>1 ms</td>
<td>1 ms</td>
<td>&lt; 94 ms</td>
</tr>
<tr>
<td>Streaming</td>
<td>300 ms</td>
<td>1 ms</td>
<td>3 ms</td>
<td>1 ms</td>
<td>1 ms</td>
<td>&lt; 294 ms</td>
</tr>
</tbody>
</table>

Figure 22-2: Packet Processing Latency Diagram and Table

**IoT Services**

Internet of Things (IoT) is a concept and paradigm where a massive number of things/objects use wireless/wired connections and unique addressing schemes to interact with each other and enable new services. IoT devices include utility meters, automobiles, health monitoring devices, security cameras and public safety systems with new types of devices and applications being invented daily. The figure below shows the Machine Type Communication (MTC) reference architecture as defined by 3GPP with MTC-IWF and MTC-Server providing new functions with associated interfaces.
"IoT-friendly LTE chipsets are the foundation for the a new wave of LTE device development and the best solutions are flexible, efficient, and low cost. There are LTE chipsets and modules available today that have been designed for exactly this. Highly optimized for M2M and IoT devices, these new solutions provide all the features and functionality required to build robust, long-life LTE devices for numerous applications at a low cost. Features include a small footprint, ultra-low power consumption, a mature and customizable software suite, and “drop-in” simplicity for ease-of-integration, which is necessary for the many non-traditional device-makers without wireless expertise. Overall, single-mode LTE chipsets and modules offer an ideal balance of feature functionality and cost for a price to performance ratio that solidifies the business case for many types of devices” – 3GPP/FierceWireless.

3GPP Machine Type Communications (MTC) defines the following specifications for IoT:

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTC Service Requirements (TS 22.368)</td>
</tr>
<tr>
<td>MTC Study Report (TR 22.868)</td>
</tr>
<tr>
<td>System Improvements for MTC (TR 23.888)</td>
</tr>
<tr>
<td>Feasibility study on the security aspects of remote provisioning and change of subscription for M2M equipment (TR 33.812)</td>
</tr>
</tbody>
</table>

Table 22-4: MTC IoT Specifications

In addition, oneM2M defines standards for Machine-To-Machine (M2M) and IoT with a focus on architecture, use cases, security and management.

IoT services can be characterized in the following general areas of control and user plane usage. An SDN/NFV network can be “sliced” for VNF placement using the attributes below:

<table>
<thead>
<tr>
<th>Types of Services</th>
<th>Low Signaling</th>
<th>High Signaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Traffic</td>
<td>Surveillance</td>
<td>Emergency</td>
</tr>
<tr>
<td></td>
<td>CCTV</td>
<td>Disaster</td>
</tr>
<tr>
<td>Low Traffic</td>
<td>Location Service</td>
<td>Logistics</td>
</tr>
<tr>
<td></td>
<td>Nature Environment</td>
<td>Vehicle Management</td>
</tr>
</tbody>
</table>
Providing end-to-end security is fundamental to the operation of IoT services. A 3GPP TR 33.868 study provides a list of potential threats and remedies in reference to the 3GPP MTC architecture. The study includes security architecture for IoT services at the device, network and transport protocol levels. Additional studies and recommendations are also available in the oneM2M forum. ETSI GS NFV-SEC specifications provide NFV security guidance and are referenced in the Security chapter of this document. Placement of IoT services should consider use of SDN as an additional security mechanism to detect and remove security threats. SDN/NFV applications could be built closer to the network edge to provide enterprise grade security for IoT services.

**5G Services**

The primary goal of 5G mobile technology is to push the current envelope to provide higher throughput, lower latency, ultra-high reliability, high density and mobile range. SDN/NFV architecture is a major focus of 5G to support modular networks that are scalable on demand and allow quick creation of new services. A new network architecture is required that can quickly adapt to new access networks, services and user behavior changes. One vision of the future 5G network architecture is a unified approach for both control and user plane traffic. Control plane traffic is unified by moving services control into a common architecture. User plane traffic is unified by moving bearer access to a packet forwarding function only. The control plane manipulates the user plane using standard interfaces.

"Create common composable core – To support the diversity of use cases and requirements in a cost-effective manner, the system design should move away from the 4G monolithic design optimized for mobile broadband. In this regard, a rethink of models such as bearers, APNs, extensive tunnel aggregation and gateways is needed. In addition, the UE state machine and entities which store UE context should be revisited and redesigned. Mandatory functions should be stripped down to an absolute minimum, and C/U-plane functions should be clearly separated with open interfaces defined between them, so that they can be employed on demand" – 3GPP.

"To provide further simplification, legacy interworking must also be minimized, for example towards circuit switched domain in the 2G and 3G networks. A converged access-agnostic core (i.e., where identity, mobility, security, etc. are decoupled from the access technology), which integrates fixed and mobile core on an IP basis, should be the design goal" – NGMN 5G White Paper. The paper discusses support of optimized network usage accommodating a wide range of use cases, business and partnership models.

[https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0_01.pdf](https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0_01.pdf)

### 22.1 VNF Placement and VNF Grouping

Placement in this section refers to functional placement of VNF in the data center topology. VNF placement decisions would need to be determined per service and whether placement needs be centralized or distributed in various data centers. Operators typically design networks using three types of Data Centers (Figure 15-5):

- **Local Data Centers** – Aggregate point for traffic from radios.
- **Regional Data Centers** – Voice and Internet traffic breakout.
- **National Data Centers** – Large data bases (e.g. HSS) and LTE/IMS signaling functions.
Operators plan to create national data centers for NFV infrastructure. With the use of centralized data centers, most of the network functions will be more centralized compared to current three-level data center topology. With the centralized data center approach, placement of SDN/NFV functions is critical for meeting the existing and future SLAs. Operators will need to decide if the new data centers meet all the SDN/NFV requirements, or a combination of existing data centers and new data centers should be considered for SDN/NFV placement.

Operators can place VNFs where they are most effective to meet KPI and SLA requirements. Virtualization of some of the components can be straightforward. For example, the IMS control plane is already centralized in national data centers. But there are a number of network functions that have strict delay requirements and adding indirect routing via centralized data centers will cause additional delay. Placement of VNFs or VMs closer to the edge of the network using the existing local or regional data centers is an option. One example is moving virtualized PGW closer to the eNodeBs but this could degrade some of the mobility aspects and placing a virtualized PGW in a local data would be more appropriate. Other functions such as caching systems could even be placed in the eNodeBs as virtual functions.

Below are the high-level categories of consideration for VNF placement:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>VNF placement meets the KPIs and SLAs using the NFV Infrastructure.</td>
</tr>
<tr>
<td>Stability</td>
<td>VNFs placement support resiliency within the NFVI and across data centers.</td>
</tr>
<tr>
<td>Scalability</td>
<td>VNF placement allows the service to scale locally or across data centers.</td>
</tr>
<tr>
<td>Manageability</td>
<td>VNF placement allows network service to be managed and monitored (e.g. protocol taps available).</td>
</tr>
<tr>
<td>Security</td>
<td>VNF placement meets the security needs of the operator and any 3rd party service provider.</td>
</tr>
<tr>
<td>Cost</td>
<td>Costs associated with the above decisions.</td>
</tr>
</tbody>
</table>

Table 22.6: VNF Placement Criteria

One of the major advantages of NFV “software only architecture” is that multiple functional components can now be grouped into a single VNF package with standard interfaces. One example of a group is collapsing all the EPC functions into a single VNF which could be used for specific markets or new test services. Another form of grouping is using service chaining where individual functional components can be grouped and deployed together in a data center (e.g. EPC -> Firewall -> NAT functions).

22.1.1 Video Services Placement and Grouping

Functional placement of video services requires that user traffic is offloaded to the internet peering points closest to the edge as possible (distributed model). Control plane systems can be located in the regional or national data centers but user plane traffic is offloaded to the internet either in the local or regional data centers and thereby offloading the backhaul network.

Network functions needed to support video services mainly include the LTE user plane functions with possible use of caching, transcoding and 3rd party content servers. Several topology models could be considered depending on the agreement with the content provider for the placement of functions. Caching and transcoding functions could be placed closer to the user in regional centers, and content delivery functions remain in the central locations.
Aggregating functions such as MME, SGW, PGW, and PCRF into a single EPC VNF will reduce overall management of virtual functions, and these aggregated functions can be distributed in various data centers. Another option is to combine the S/PGWs into a virtual Local Gateway (LGW) function and place it close to the edge while keeping MME centralized. Distribution of functional components across data centers does increase backend integration requirements and impact the management of services (e.g. performance and tracing capabilities are needed at distributed functions).

**22.1.2 Impacts of encryption/header compression on caching mechanisms**

As over the top (OTT) providers such as Google and Facebook are moving towards using encrypted transport for video and other content, it is becoming increasing difficult for service providers to detect, optimize or cache any information locally in the provider network. Google proposed the SPDY protocol early in 2009 to improve loading of web pages, and functions of SPDY have been implemented in HTTP/2.0 specifications. One of the features of HTTP 2.0/SPDY is the use of TLS to support end to end encryption. Use of end-to-end HTTPS tunnels requires cooperation with the content provider to facilitate caching.

![HTTP/2.0 Stack with TLS](image)

HTTPS tunnels makes it difficult for intermediaries to be used to allow caching, to provide anonymity to a User-Agent, or to provide security by using an application-layer firewall to inspect the HTTP traffic on behalf of the User-Agent (e.g. to protect it against cross-site scripting attacks). HTTPS tunnels also remove the possibility to enhance delivery performance based on the knowledge of the network status, which becomes an important limitation especially with HTTP 2.0 when multiple streams are multiplexed on top of the same TCP connection. One possibility to have a trusted intermediary (and still providing confidentiality towards not trusted elements in the network) is to have separate TLS sessions between the User-Agent and the proxy, on one side, and the proxy and the server on the other side. Operators could coordinate with the content providers to support use of intermediary proxy VNFs either managed by the operator or a 3rd party.

**22.1.3 Device Caching & Device to Device (D2D) Communication**

One of the options for video content caching is for devices to locally cache the content, which is either received over the LTE network or when a user is connected to a Wi-Fi network.

3GPP added the D2D function in Release 12 to support device-to-device discovery and communication and in Release 13, a device relay function. Public safety services are one of the main drivers of D2D features.
Content could be cached in the devices for communicating with other devices. For a device to provide content caching for itself on behalf of another device (D2D), the device needs to have enough memory and the processing power to accomplish the delivery, storage, encryption and decryption of the content to the user. Cache memories in UEs for the CPU are on-die memories and are the most expensive memory types. For over the top applications, off die RAM is used. As the content requirements increase (e.g. 4K content), the size of these memories will also have to expand to accommodate the required processing and content storage. There will be impact on die size and power consumption, and therefore the cost of the device. As these architectures develop, cost must be taken into consideration.

If Digital Rights Management (DRM) content is used for caching, the issue becomes more complicated as the DRM protected content is stored and encoded within a secure zone in the device’s SoC and needs to provide a high speed interface for viewing. The secure element memory has to accommodate the size required by the type of application/content and must have the processing power to deliver the required user experience. UE delivering DRM content to another UE is an issue that will require the content owner’s approval.

Both scenarios are not necessarily impossible for current technology to provide, within the context of a single UE, however both scenarios cause higher battery consumption, increase the device cost and involve business issues that must be taken into consideration by industry consortiums working to standardize an architecture.

In the case of D2D, the work (as of now) is in the research stage where transmission technologies are being studied to evaluate their performance. Regardless of the type of transmission between UEs (LTE-direct, microwave, mm wave), broadcasting from the UE will impact power consumption and physical size, and may require additional RF modules and higher CPU speeds, which in turn increases the cost of the device.

D2D also brings challenges involving the streaming of cached content on a device to the requesting device. To give this problem context, assume current HTTP Adaptive Video Streaming is used. This method detects the device’s bandwidth and CPU capacity and adjusts the quality of streaming accordingly. In a D2D network this would generate messaging either between devices and/or between the control mechanism of the network and the device that is allocated to stream the video content to the requesting UE. Since each UE can be running different applications, their CPU capacity and bandwidth usage will change, which will involve content delivery issues.

Additional research needs to be completed, especially for pay-per-view content where the user’s expectations of quality of service must be met.

In the final analysis, the architecture for device caching of content and D2D delivery must take into consideration specific use cases, which, in turn, will impact the architecture and implementation in the network and user clients.

22.1.4 IoT Services Placement and Grouping

IoT services require that networks be able to support multiple variations of user traffic, and provide application support. The SDN/NFV architecture provides a major benefit to IoT services as network resources can now be “sliced”, deployed and scaled as software packages. Each individual service requires vertical integration between the operator’s network and service providers.
The following figure shows one of the existing topologies for deploying IoT services where internet traffic remains on the existing network and separate virtualized EPCs are used for deploying IoT services. The virtual M2M vEPC is further divided into individual PGWs to support various service characteristics.

Figure 22-8: Example of Virtualized MTC Network

A similar approach for IoT service placement needs to be considered for Operator networks. The main characteristics that will determine the placement of services are the following (in addition to typical performance, stability, scalability, and manageability requirements):

| Call Model | IoT services in various traffic model flavors where some services use high control plane/low data plane while others have high data plane usage. Placement of higher control plane services can be centralized, and higher data plane users distributed to local or regional data centers. |
| Vertical Integration | Each IoT service requires vertical integration with the 3rd party provider. Vertical integrators need access to IoT services, optimally using breakout points at the data centers. Placement of the “MTC-Server” (as defined in 3GPP) involves restrictions on where vertical integration can occur. One option could be a virtual “MTC-Server” close to the integration points. |

Table 22-9: Example of Virtualized MTC Network

Similar to video services, operators might want to aggregate functions such MME, SGW, PGW into a single EPC VNF or combine MTC-specific functions such as the MTC-IWF/MTC Server/Charging into a single VNF package. The scale of the aggregated functions can be based on the type of call model required by the IoT service.
The current MTC architecture uses APNs and PLMN-ID to redirect traffic to various services. The future evolution should consider use of 3GPP Release 13 “DÉCOR” specifications for redirecting traffic based on the type of service and specific VNFs.

22.1.5 5G Services Placement and Grouping

The NGMN’s alliance paper focuses on “slicing” networks into multiple layers with each layer designed to support a set of services. This has been described in Section 15.3.

Placement and grouping of 5G network slices described by the alliance is composed of a collection of 5G network functions and specific RAT settings that are combined together for the specific use case or business model. Thus, a 5G slice can span all domains of the network: software modules running on cloud nodes, specific configurations of the transport network supporting flexible location of functions, a dedicated radio configuration or even a specific RAT, as well as configuration of 5G devices. Not all slices contain the same functions, and some functions that today seem essential for a mobile network might even be missing in some of the slices. The intention of a 5G slice is to provide only the traffic treatment that is necessary for the use case, and avoid all other unnecessary functionality. The flexibility behind the slice concept is a key enabler to both expand existing businesses and create new businesses. Third-party entities can be given permission to control certain aspects of slicing via a suitable API, in order to provide tailored services.

For such a 5G slice, all the necessary (and potentially dedicated) functions can be instantiated at the cloud edge node due to latency constraints, including the necessary vertical application. To allow on-boarding of such a vertical application on a cloud node, sufficient open interfaces should be defined. For a 5G slice supporting massive machine type devices (e.g., sensors), some basic C-plane functions can be configured, omitting e.g. any mobility functions, with contention-based resources for access. There could be other dedicated slices operating in parallel, as well as a generic slice providing basic best-effort connectivity, to cope with unknown use cases and traffic. Irrespective of the slices to be supported by the

Figure 22-10: NGMN View of a 5G Network
network, the 5G network should contain functionality that ensures controlled and secure operation of the network end-to-end in any circumstance. The decomposition in the 5G slice needs a balance between granularity and flexibility. The flexibility adds complexities in supporting a smaller grouping of functions.

The NGMN alliance provides three options for placing and integrating 5G network functions. Option 3 in the figure below is identified as preferred by NGMN with new interfaces needed between the legacy and 5G functions.

<table>
<thead>
<tr>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPC</td>
<td>EPC</td>
<td>EPC</td>
</tr>
<tr>
<td>Functions</td>
<td>5G NW</td>
<td>Functions</td>
</tr>
<tr>
<td>Fixed NW</td>
<td>Functions</td>
<td>Fixed NW</td>
</tr>
<tr>
<td>4G Evolution</td>
<td>New RAT</td>
<td>Fixed NW</td>
</tr>
<tr>
<td></td>
<td>4G Evolution</td>
<td>4G Evolution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pros</td>
<td>Pros</td>
<td>Pros</td>
</tr>
<tr>
<td>• No changes to 4G RAN</td>
<td>• No changes to 4G RAN</td>
<td>• 5G NW functions/new RAT design can be optimized to fully benefit from new technologies (e.g., virtualization)</td>
</tr>
<tr>
<td>• No Need for revolutionary 5G NW functions design</td>
<td>• 5G NW functions/new RAT design can be optimized to fully benefit from new technologies (like virtualization)</td>
<td>• New design could only be utilized where there is new RAT coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Potential signaling burden due to mobility if the new RAT does not provide seamless coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Potential impact on legacy RAN to operate concurrently with legacy CN functions and 5G NW functions</td>
</tr>
<tr>
<td>Cons</td>
<td>Cons</td>
<td>Cons</td>
</tr>
<tr>
<td>• Tied to the legacy paradigm for all the use cases (which may be expensive)</td>
<td>• New design could only be utilized where there is new RAT coverage</td>
<td>• Tied to the legacy paradigm for all the use cases (which may be expensive)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 22-11: NGMN Options for 5G Network Integration

### 22.2 VNF Placement Impacts on Devices

Virtualization in this document refers to virtualization of network functions. It is assumed that initial phases of network virtualization will include migrating existing IMS/LTE core and related management services. Initial migration could be further divided into sub-phases and this will include migrating the control plane and the data plane in phases.

Migrating existing network services to SDN/NFV-based systems do not change the 3GPP interface specifications defined for LTE or IMS services. For example, in the diagram below, all the IMS functional components can be virtualized but the interfaces defined between the components do not change with SDN/NFV. Other user plane requirements such as throughput and latency must still be met.

Network virtualization provides a more dynamic and optimized way to deliver services to devices without impacting any of the existing 3GPP interfaces. Some examples of network optimization include separating control/user plane, moving the user plane closer to the edge of the network or deploying a group of VNFs as a single service. Network virtualization will allow operators to create and deploy new network and device related services faster.

No change to device APIs is expected with support of EPC/IMS VNFs and grouping of VNFs. Device APIs will continue to use 3GPP specifications for existing services. 3GPP LTE and IMS interfaces are provided in earlier sections of this document.
22.3 Evolution of Device APIs

3GPP introduced new functions in Release 13 for support of device-to-device and proximity-based services. Additional changes are being considered to support IoT, including service-specific functions such as improved battery life, device triggering and mobility.

22.4 Trends in applications and placement/virtualization

The majority of the of the 3rd party applications running on IOS, Android or Windows run in the cloud space with Amazon AWS being the largest in addition to Google and Windows Azure, which provide support for mobile devices. All large cloud operators provide some of form of development tools and support to integrate the applications in their systems and support backend integration.

Integration and placement of IoT-related services is specified by 3GPP specifications (TR 23.888). The MTC server provides an entry point into the Operator’s network for 3rd party application developers.

22.5 IOT aspects: flexibility in connectivity

The NFV framework lends itself to the IoT paradigm, allowing for isolated separation of applications, such that each application can be served by a core that is tailored to the service itself. Today’s LTE architecture facilitates “always-on” user and control planes, which is critical for applications such as mobility alongside web browsing, streaming, and voice. However, M2M-type applications make use of multiple types of traffic, from massive data exchanges, to small data exchanges occurring periodically (e.g. in metering), to large always-on services such as video surveillance. The always-on services can be “sliced” to support existing signaling and bearer mechanisms. But small data exchanges that occur periodically, and involve bursty traffic, can impact MME and Bearer traffic. 3GPP defines multiple options to reduce small data transmission by using SMS (limited to 140 bytes) via the MME as an option, and the Small Data Transmission (SDT) protocol (TR 23.887). Stateless IPv6 address auto configuration is used with SDT when T5 messaging is used to carry IP datagrams, thereby removing the use of TCP/UDP-based bearer traffic.

With SDN/NFV, specific IoT services could be grouped with MME, PGW/SGW and IWF functions to isolate traffic and scale the service as needed. An “IoT” device that may need to be woken-up every so
often would require the call context to be maintained by the core. However, a device which originates an occasional push of data could attach to the network only as needed, reducing the impact on core resources. Further, utilizing the control plane (such as SMS/SDT) to transmit this data may completely eliminate the need for a user plane connection. Each application’s VNF(s) and by extension, the devices themselves, will match the signaling and user requirements of the application.

IoT services will require a variety of latency, bandwidth and packet loss requirements based on type service and service level agreements. Operators could define few “cookie cutter” network “slices” each with specific requirements. Below are few of the services operators have considered for deployment.

<table>
<thead>
<tr>
<th>Asset Tracking</th>
<th>Telematics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Monitoring</td>
<td>Automated Retail</td>
</tr>
<tr>
<td>Fleet Management</td>
<td>Digital Signage</td>
</tr>
<tr>
<td>Smart Energy</td>
<td>Wireless Kiosks</td>
</tr>
<tr>
<td></td>
<td>Security</td>
</tr>
</tbody>
</table>

IoT services can be deployed in various deployment models. Operators would need to consider the VNF packaging for IoT from the management perspective. Reduced number of VNFCs in a VNF could reduce the amount of management/automation required by the operators. An example below shows consolidated and separated control/user plane options for deployment and scaling consideration. The figure below shows multiple control planes in different virtual machines managing a single user plane. VNFs can be deployed to meet specific engineering requirements of a variety of IoT applications.

**22.6 Multi (dual) access connectivity aspects**

Dual access here is referring to users and devices accessing the same set of services from multiple access types. The dual access described in the figure below is with LTE and Wi-Fi. Here the PGW acts as the tunnel anchor between the LTE and WiFi networks:
Non-3GPP access, such as Wi-Fi, can provide relief where coverage gaps exist and allow a user to seamlessly transition between Wi-Fi and LTE. Applications such as VoWiFi/LTE and video calling can benefit greatly with this approach. A flexible NFV core environment can adjust resources to accommodate the user patterns and behavior; for example, as users head home during the evening hours, fewer resources might be needed for mobility/signaling while more could be allocated for an ePDG VNF to support potentially higher non-3GPP access data. For VoLTE subscribers who might have both a WiFi and an LTE connection, the operator could impose a preferred method for registering on the IMS network, and allocate the VNF’s respectively.

### 22.7 Wireline Services

The wireline network provides services to both consumers and enterprises. Consumer services are typically Internet and Video, whereas enterprise services are virtual private networks, virtual wireline services and Internet. Today the vast majority of these services are delivered from specialized, purposed built hardware. NFV and SDN allow for wireline services to be delivered from the Data Center. This section of the document explores the potential impact of NFV and SDN on:

- Consumer Video and Broadband Services
- The Connected Home
- Enterprise Services
- Converged Wireline and Wireless Services

#### 22.7.1 Consumer Broadband and Video Services

Wireline consumer broadband and video services are typically delivered over some type of high-speed access technology such as FTTP.
The Broadband Network Gateway (BNG) aggregates subscriber sessions to provide network access. The BNG is the first hop that has session layer awareness. It utilizes RADIUS/Diameter with PCRF and DHCP for subscriber identification, authorization, address allocation and subscriber policies. A backend database is typical. A subscriber is a Broadband Home Router (BHR) that receives access services from the BNG. The BNG is located between the subscribers and an upstream network that the subscribers access (e.g. Internet access and video network services). The BNG builds a context for each subscriber. This allows subscribers to be uniquely identified based on an identifier such as DHCP option-82 information. Per subscriber policies can be applied based on locally configured rules or via an external policy server (e.g., PCRF).

Video services are delivered to subscribers from a video based content delivery network that provides access to broadcast and on demand content. Bandwidth is limited for subscribers and requires mechanisms to regulate and manage access to resources at the BNG and in the access layer. Several levels of queuing, scheduling, and Quality of Service (QoS) are required to effectively manage available bandwidth between subscribers as well as Internet access, video broadcast, and video on demand.

IP based television (IPTV) service is implemented via Multicast for requesting TV broadcast services.

Video on Demand (VoD) is a video service allows subscribers to stream video content at any time. Content is transferred from the video content delivery network via the BHR to the set top box (STB) in the home. Content may also be downloaded, then decoded and played on customer devices.

VoIP is supported on a VLAN separate from the VLAN used to support data/video on the interface between the OLT and the BNG.

IPTV and VoD are subject to admission control decisions based upon available capacity at various points in the wireline FTTP access network.

The BNG, OLT and BHR implement queueing, scheduling and QoS on various levels using certain packet fields. These elements also use various fields to differentiate and admit/block traffic.
The video headend functions are responsible for the video services, including:

- Content acquisition and conversion from analog to digital. Sources of content are varied and include satellite, off-air and fiber.
- Video signal processing, including ad insertion
- Video encoding
- Content personalization and distribution

Video headend functions can be delivered from national or regional data centers as shown in Figure 15-5. Many of the functions may also be virtualized, although some functions may be better suited for bare metal application or hardware appliances.

### 22.7.2 Connected Home

VNFs and virtualization in general are causing the traditional model of home services to evolve. Today the Broadband Home Router (BHR) represents the subscriber identity in a wireline service as shown below; however, since it performs routing the MAC address is removed and it also performs NAT so that the actual identity of the users of the service is not known to the upstream network functions (unlike the situation in mobile networks).

Instead of a BHR, a simpler Network Interface Device (NID) could implement L2 connectivity (either directly or via a tunnel) over a simpler L2 access network to a future (potentially virtual) vCPE in which at least MAC level knowledge of the device in the home could be known. This "Logical per NID L2 network" would still need to implement some of the scheduling, queuing and QoS functions currently implemented by the BNG and OLT. However, a Next Generation OLT (NGOLT) using International FTTH standards could implement some of this functionality more efficiently than that currently performed by custom ASICs in the BNG. These changes would enable services similar to that offered to mobile devices described previously in this section, such as IOT, home monitoring, parental controls, etc. The vCPE would provide (access to) the DHCP, Diameter/policy information done by the BNG today.
NFV and virtualization can potentially unlock additional value in this environment for providers. The potential exists to virtualize the home network and value added consumer services in the cloud. This could lead to a simplification of the BHR function to that of a NID and partition the transport requirements between a NGOLT and a logical per NID L2 access network. It can also increase visibility into subscriber flows at the vCPE, and other functions similar to those previously described for mobile in service function chains. New service creation can be simplified, reducing the number of systems and touch points involved.

Figure 22-17: Connected Home Services - vCPE

The home network can be centrally administered from the cloud. New services can be deployed, without requiring physical CPE upgrades. This allows for more granular service control and analytics for a given home and for users within the home. The potential also exists for the customer self provision and self-support.

22.7.3 Enterprise Services

Today most enterprise services consist of two distinct offerings built on a common shared infrastructure:

1. Layer-3 MPLS VPN Service (L3VPN)
2. Layer-2 Service (L2VPN)

NFV and virtualization promise cost savings in many aspects of the overall enterprise service offering. A primary area for cost reduction is the virtualization of the physical Provider Edge (PE) equipment for existing L3VPN and L2VPN services. In addition, the creation of a new service can be simplified by reducing the number of systems and touch points involved. Multiple VNFs may be chained together using orchestration to create new services, which may be specifically tailored to a given enterprise.

An L3VPN service is a network based IP VPN that typically provides any-to-any connectivity and Quality of Service (QoS). It is normally based on MPLS as the underlying technology. Other technologies such as IPSec can be used to provide an IP VPN service, however MPLS L3VPN based on the IETF’s RFC 4364 is the most common.

Figure 22-18 below depicts the key physical and logical components that comprise a classic RFC 4364 based network.
The primary components for delivering enterprise services are the Provider Edge (PE) and Customer Edge (CE) platforms. Both the PE and CE routers are implemented in physical network functions today. These functions may be virtualized in order to:

3. Orchestrate and automate the end to end VPN service
4. Lower the cost of the PE and CE functions
5. Leverage orchestration and additional VNFs to insert value-added services
6. Provide a single tenancy PE option

Virtualization of the PE router should take into consideration the networking requirements that are of concern to enterprise customers. Many enterprise customers are interested in service level agreements (SLA) for their VPN service. To support these enterprise SLAs, the virtualized PE must implement DiffServ compliant QoS mechanisms such as:

- Ingress policing/metering using Committed Access Rate (CAR)
- Marking
- Priority/Low-Latency Queuing (PQ)
- Class-Based Weighted Fair Queuing (CBWFQ)
- Weighted Random Early Discard (WRED)

These mechanisms will drive additional requirements for networking in the NFVI. Single Root IO Virtualization (SR-IOV) may be necessary for data plane resource management and higher performance. Other options like Intel’s Data Plane Development Kit or vendor specific virtual forwarders may also be utilized to address the SLA requirements of enterprise customers.

In addition to basic any-to-any access, enterprises may benefit from additional services as an add-on to their L3VPN. Add-on services, such as provider hosted web proxy or security services may be used to supplement a basic L3VPN service.
Whether a L3VPN service is delivered via a PNF, VNF or a hybrid of both, value added services may be introduced by instantiating and chaining together a series of VNFs with enterprise VPN membership.

The classic CE is an IP router, located at the customer site, which connects to the PE router using some access technology (physical or tunneled). CE devices peer with the PE as part of a unique routing/forwarding instance for a given enterprise. However, as enterprises shift application workloads to public and hybrid cloud environments, the requirements of CE is evolving. The CE device itself may take the form of a ‘classic’ L3 router, L3 VNF + x86 or a simply L2 NID. But, while there will be some physical CE device at the enterprise to terminate an access circuit, the provisioning and management of that device is quickly evolving to a cloud based model.

As enterprise services migrate from the premise to the cloud, orchestration will act as the glue, binding together customers, services and resources. A modeling language may be used to describe the intent or end goal of the service. The service model should be independent of the underlying infrastructure. The Orchestrator should have knowledge of both the service intent and the physical and virtual devices required to host the service. The Orchestrator is then able to instantiate the enterprise service across the infrastructure.

Service models, mapped to the capabilities of devices and finally instantiated on the infrastructure topology is a powerful tool, enabling the rapid deployment of new services. Service models may be generic or highly customized; allowing operators to offer unique services beyond the traditional L2VPN and L3VPN.
22.8 Converged Wireline and Wireless Services

As mobile and home services move to the cloud, consideration should be given to a composite subscriber identity and data management. This composite architecture enables:

- Cross access subscriber identification and billing
- Backend Infrastructure maps billable Identity to multiple IDs/credentials from:
  - Application Layer
  - Session layer
  - Network Layer
- Enables common policy by tracking multi-Identity (wireless and wireline)

Converged access, policy and charge enables flexible options for shared and common experience across wireline and wireless entities.
Annex A: Intent-Based Networking

Intent-Based Networking (IBN) is a new network operating model and interface being developed in the ONF as a controller and infrastructure agnostic, common interface across multiple diverse infrastructure controllers. The benefits of adopting IBN include:

- Elimination of vendor lock as a barrier to choice, agility, and innovation
- Ability to “write once” for integrating workloads and applications with infrastructure.
- Ability to mix and match best-of-breed network service implementations from a diverse ecosystem of independent software vendors.
- Ability to “bake off” differing implementations of desired features and choose vendors, protocols, interfaces, etc. based on empirical data.

This work was initiated and is proceeding based on the assumption that there is no operator-friendly justification for continuing to build an industry where choice and competition are stifled by proprietary and non-interoperable infrastructures interfaces. Providers should consider carefully whether the benefits listed above are important and valuable for the next generation network plan.

IBN is based on the idea that we should describe the application’s network requirements in application-domain terms, rather than in network expert terms, which is the dominant operating model today; “Don’t model the network, model the application communications needs, and let a piece of smart software translate that into protocols, interface, media types, vendor extensions and other concepts from the domain of the networking expert”. Today, a single, common NorthBound Interface (NBI) is being defined in ONF and implemented in multiple popular open source projects including Open DayLight, ONOS, and OpenStack. Vendors are working to build commercial, supported, easily deployed distributions of these open source projects. The success of IBN is expected based on a network effect in which multiple infrastructure controllers implement the NBI, causing more people to use the NBI, causing more infrastructure projects to implement the NBI, etc.

IBN solutions are not available to deploy today, but will certainly be available within the lifespan of the network architecture currently being developed. Providers should carefully consider whether IBN solutions need to be included in a plan for deployment in the next several years. Today a protocol like OpenFlow can allow a relatively small number of very specialized experts to “program the network”. IBN allows millions of people who know little or nothing about networking to “program the network”. Providers should consider whether there is commercial opportunity and ROI in a platform that can enable mass-customization by non-expert subscribers.

The ONF Information Model for IBN is provided as a model based development artifact in the form of UML, YANG, and other modeling languages and language bindings. The IBN NBI becomes the interface to the SDN controller. There are no additional controller devices required or additional infrastructure complexity compared to a non-IBN SDN solution.

Reduction in complexity, better resource sharing

The diagram below compares the major components and the nature of development work between a system where the thing we call an SDN application or SDN service directly generates low level device programming such as using openflow, and one where the app simply pushes intent into the engine that provides an intent-to-device-rules service. In this example two different forms of media streaming originally have two different SDN apps pushing openflow rules. One is for interactive audio and video communications, the other is for streaming movies. There is great overlap between the switching rules...
needed for the interactive flows and the streaming flows. This can easily be generalized so that a single set of flow logic can support both requirements. However, because these are two different applications, each has a set of similar, yet completely different logic for directly generating device rules. In addition, both of these applications believe that they exclusively own the flow tables in the switches and as a result they make conflicting changes causing system failure. In the intent based model the common logic is pushed into the intent engine. Now the developers of the two applications each write substantially less code, and don’t deal with any of the complexity of low level device programming. In addition, by using a common system for rendering the low-level instructions, they completely avoid the multiple writers problem and have a single manager of a coherent flow table.

Additional Background Material on IBN:

https://www.sdxcentral.com/articles/contributed/network-intent-summit-perspective-david-leEE0w/2015/02/
Annex B: Federated Inter-Domain Controller

Orchestration Challenges and The Problem With Federated Control

On a broader view, there are currently unsolved issues when the network is constructed of multiple administrative domains. These issues are even further exacerbated in the realization of XaaS offerings, where for most telco environments the vision is “NFVaaS or VNFAAS” (as articulated in the original ETSI NFV use cases) - whether applied to Enterprise customer or peer telco/SP.

The fundamental problem is the federation of controllers and orchestrators. Federated control generally devolves into a data-sharing problem (notably, sharing with policy controls) that has implications at the application development level.

One of the strengths of the existing paradigms of orchestration systems and network controllers is that the application developer should not need to be aware of data location, and is instead afforded an API that makes this transparent. Another strength is that a rich set of appropriate data is available for programmatic decision-making and end-to-end management – and that would certainly apply to the internal-only use of systems that (for example) segregate WAN control from DC control (there would be an expectation that an application making end-to-end decisions would have access to the data in both systems). It would be a goal of such federation to preserve these constructs.

The possibility to federate depends on the type of access desired between peers. A client-server solution is often demonstrated through enterprise kiosks into service provider systems for controlled read/write. This type of sharing doesn’t make the data from the resulting transaction permanently available to a separate system, but provides a controlled “view” into a system – thus it really doesn’t satisfy any of the data sharing requirements for application development that spans domains.

Existing thinking on the topic of federation tends to create linkages between controllers for sharing a particular data set for read access: BGP for reachability (but not flow, until you include flow spec) and BGP-LS for topology (assuming your controller provides a similar topology model). While these mechanisms satisfy the need for data sharing with policy, they are limited to subsets of data and often incur a translation penalty. Ultimately, service assurance and analytics breaks any such model – as these two components entail pure data sharing (logs, stats, events) with associated policy considerations. Write access will leverage different pathways, APIs, and interfaces that have to be synchronized with read access to allow for basic concurrence operations in programming (e.g. test and set). These pathways need to be simultaneously aware of partitions (or at least tolerant of partitions). The write pathway may be driven via orchestration, which may also have to be federated (with conflict adjudication as well) in a manner beyond the aforementioned kiosk example (federated orchestration). This is more expected in the inter-domain set of use-case, for example, enterprise partnerships.

Both problems are more easily solved if the federation is internal, using a master-of-masters approach (hierarchical controllers) with a shared master orchestrator and shared master controller. If this master-of-masters approach is used in internal federation to obviate the need for more sessions/protocols to share individual repositories (read problem), it would likely force a move to single vendor for the solution at all levels of the hierarchy, until federation protocols are developed and standardized. Even then, policy controls on sharing may not be very granular and this is an under-explored area. But, this approach would (at least) align schemas, actors and other database behaviors. Such a mechanism has never had great longevity at a business level for external “partnerships” (e.g. control of the shared resource almost always causes ownership contention), so federated external partnerships are a higher hurdle to cross.
requiring a truly transparent and controllable data-sharing mechanism (assuming that neither partner can dictate the solution of the other).

Keeping in mind the goal of federation (as stated above) and the numerous current impediments in sharing, there are a number of avenues to explore. For example, ODL has some potential through the AQMP plugin to provide pub/sub subscription with filters to other ODL admin domains (while single-source, this might abridge both internal and external partnerships). Named Data Networking also presents a potential future solution to the entire data-sharing problem as far as transparent access (but translation could still be an issue).

In closing, while the typical NFV architecture today shows multiple levels of controller, the consequences of operating these (particularly if they are multi-vendor or multi-source) are something to consider. Even for internal-use-only environments (ignoring peering and enterprise customer partnerships), the convenience of avoiding hard decisions about consolidating the network control ownership (political) and vendor selection (risk) has a fairly steep price in complexity. More study is needed on this topic.
Annex C: Segment Routing

Segment Routing works by encoding a path across a network as an ordered list of abstract instructions, or segments, which may be routing instructions, locators, autonomous systems, service functions, and more. SR uses common data plane technologies, such as MPLS and IPv6, with little (IPv6) to no (MPLS) modification, and requires only very modest changes to existing routing protocols. SR is also fully documented in IETF drafts with both multi-vendor and multi-operator contribution. Finally, it can be realized in an SDN environment.

Overview of Segment Routing Technology

Segment Routing is a fundamentally simple technology. The basic premise is centered on the notion of source routing, where the source, or ingress node, directs the path that a packet will take by including the path in the packet itself. Indeed, we can easily describe how it works by taking an example.

Consider a network comprised of some number N nodes (routers, for example), and some number A adjacencies (Ethernet links, for example) between them, organized in some arbitrary partial mesh topology. Let us assume that a link state IGP is running on the network, such as OSPF, and that the protocol is operating in a single area. We can also assume that there are IPv4 addresses on each link and each router has a loopback address (IPv6 can be substituted here). Typically, the IGP will discover the topology and then distribute each router’s information (in the form of LSA/Ps) to each other router in the network. The network will eventually converge around a stable topology and each node in the network will have a complete view of the network. Such a network is shown in the figure below:

Now that each node has a complete view of the network, each node will compute the shortest path to reach each other node in the network using Dijkstra’s algorithm and install routing state to each prefix in the network along the shortest path as computed prior.

This sequence of operations has been the fundamental underpinning of the Internet for almost 20 years. However, there are a few specific drawbacks that, to date, require additional technology to solve:

1. There is no way to connect the leaf nodes of the network to each other in a way that alleviates the core nodes from awareness of this connectivity, unless we tunnel the traffic and route it abstractly across the shortest path (Service tunneling, ie, VPN services, is the canonical example).

2. There is no way to deviate from the shortest path or to request and reserve bandwidth (traffic engineering).
3. There are certain topologies that cannot guarantee loop freedom and outage resilience when link or node failures occur. (IP Fast Reroute with Loop Free Alternates).

In order to ameliorate these considerations, MPLS technology was developed and has been widely used since. It offers opaque service transport across the core of the network, supports rich traffic engineering capabilities using RSVP-TE, and can handle protection requirements when appropriate presignaled paths are in place. However, MPLS, as it has been used from its inception, suffers from its own set of drawbacks:

1) It requires new signaling protocols that must interact with the existing routing protocols with great precision and some degree of delicacy.

2) The Traffic Engineering capability borne through RSVP-TE has real scaling challenges that have limited its deployment outside of WAN cores. This restriction of use has also had the effect of creating a complexity mystique around RSVP that limits its exposure outside of large service providers.

3) LDP, the more simple of the two common MPLS protocols, has no inherent TE capability and only really provides a substrate (transport or service) that relies on other protocols in order to work. This reliance limits LDPs true utility beyond providing for simple connections.

4) The traffic engineering capabilities suffer from a strange, but harmful dimorphism – online TE using Constrained SPF has scaling and performance limitations (bin packing, deadlocking, scheduling, etc) that significantly degrade its overall performance. Offline TE is able to avoid these problems, but still requires significant signaling overhead and well-developed (yet computationally hard) mathematical algorithms to accurately compute potential paths for use.

5) Transit node state accumulation has real, practical limitations. In large networks with full meshes of RSVP-TE tunnels, it is not uncommon for transit nodes to have tens of thousands of transit tunnels at any given time, commensurate with the attendant state management burden at each node required of such state. As a result, certain types of failure events on these nodes can cause considerable reconvergence protraction – in some cases resulting in a failure to completely converge – which may require more aggressive intervention in order to re-establish a steady state.

Network operators and equipment vendors have known of these limits for some time. Recently, the IETF and has begun addressing these limits with a new technology called Segment Routing.

Returning to our example, consider a requirement to deliver a certain amount of traffic, say 20 Gbps, from node A to node Z. Let us also assume that the shortest path from A to Z traverses A-B-C-D-Z. But let us also assume that there is a link on the path A-B-C-D-Z that is congested and cannot accommodate 20Gbps of additional demand. Using RSVP-TE, we could signal a path, say A-N-O-P-Z, that may have enough capacity to meet this demand.

At this time it is important to state what may not be obvious at this point in the example. How does ‘A’, or ‘we’ know that the path A-B-C-D-Z cannot accommodate 20Gbps? In the past, the answer may have been, “from the network management system”, or perhaps, “from the capacity management system”, where the data was gathered by polling SNMP MIBs or Netflow collectors. And this would have been the right answer. But today, we can do more. With the advent of Software Defined Networking (SDN), the ascendancy of cloud networking and scaled out compute, and the introduction of powerful, programmatic interfaces and protocols into the network, it is now possible to have a network controller dynamically and
adaptively perform application-level network admission control, optimized explicit routing, and real-time performance analytics, the results of which can be fed back into the optimization engine to re-route demands as network conditions change. Such a paradigm shift may allow applications themselves to adapt their traffic parameters, including routing directives, in ways not before considered possible. Examples of systems that leverage these parameters and directives is a WAN Orchestration System (such as Cisco’s Network WAN Automation Engine) or a WAN SDN controller like ONOS or OpenDaylight. We will use the simple moniker ‘WAN Controller’ to identify the offline system that helps identify these paths through the network throughout this white paper.

Back to our example, let us assume that the WAN Controller is aware of the congestion along A-B-C-D-Z and informs the ingress node via a programmatic interface that a better path would be A-N-O-P-Z. This is shown in the next figure.

By informing the ingress node that the FEC for Z should be installed in the routing table using an alternate path, we can avoid the congestion on the shortest path. Typically, this is done with MPLS RSVP-TE signalled LSPs. But, in the absence of explicit signaling, how can we enforce a non-shortest path?

How Segment Routing Works

Segment Routing works by encoding, in the network topology itself, the set of all possible nodes and links that a path across the network may visit. Given that the IGP discovers and constructs the entire topology in the link state database, adding values, or Identifiers, that can be used as transit points in the network takes only a minor attribute extension to the link state IGPs such as OSPF and IS-IS. We define this new IGP attribute, called a Segment Identifier or SID, which indicates an “instruction” that nodes that process this particular SID must execute. Each instruction may be as simple as an IGP forwarding construct, such as “forward along the shortest path”, or more complex, such as a locator, service context, or even as a representation of an opaque path across an Autonomous System. The key here is that each node will have identified one or more segment identifiers, that have some explicitly defined function, and should they receive a packet with such an identifier encoded in it, for example, as an MPLS label, then they should lookup the instruction in the forwarding table and execute it.

Global and Local SIDs – Prefixes and Adjacencies

We define two types of SIDs: Global and Local. A Local SID has local scope, is installed independently by each node, is advertised in the IGP, but is only installed in the SR FIB of the node that originates and advertises the SID. An example of such a SID is the Adjacency SID, which is used to identify (and forward out of) a particular adjacency between two nodes. An example is shown in this figure:
In the figure, we can see that each node allocates a locally unique identifier for each of its links to its IGP neighbors. In the figure, for brevity, we only include one Adjacency SID per link, but in fact, there are two – one in each direction, allocated by the node upon which the specific adjacency resides. For example, between A and B, A has an adjacency SID of 9004 that defines it’s adjacency to B, and B has an adjacency of 9004 for it’s adjacency to A. This is ONLY shown this way for convenience. No method is employed to synchronize SID values. The SIDs are allocated independently on each node.

A Global SID is one that has network-wide use and scope. Consider a node, such as Z, and consider the reverse shortest path tree rooted at Z that defines every other node’s shortest path to Z. Should Z allocate a specific SID for itself (its loopback address) and distribute it throughout the network via the IGP, and should it do so with the specification that each node should install this SID into its forwarding table along the shortest path back to Z, then, using one SID for each node, we can identify each node’s reverse tree and therefore we can enumerate the shortest path to every node using one SID per node (as represented by its tree). Such a SID is called a prefix SID.

If, for example, node Z allocates a prefix SID for it’s loopback address, and distributes it throughout the network via OSPF or IS-IS, then each node can install SID Z along the shortest path to Z. If the data plane is MPLS, then Z can be encoded as a label, and each node can install in its LFIB an entry that takes label Z, swaps it for label Z, and forwards it along the shortest path to Z. This is shown in the figure below. When a prefix SID is used in this fashion, we call it a Node SID.

3 In truth, there are at least two per adjacency. It is possible to encode additional SIDs per adjacency, thus providing additional capabilities, such as defining affinities or colors that can be used by online CSPF. An example is shown in the appendix.
Note that the ingress need only push one label on the stack, as each node recognizes that this particular SID is a prefix SID and thus is associated with the instruction to forward the packet along the shortest path the Z.

Explicit Source Routing

In the previous example, we leveraged Segment Routing in order to forward packets along the shortest path to Z. However, it is also possible – indeed, desirable in many instances – to forward packets along an explicit, non-shortest path toward Z, using several techniques. One way would be to identify one or more anchor points or transit nodes in the network and then send packets along the network with a stack of segment identifiers representing these anchor points in sequence. Each of these anchor points terminates (and initiates) a segment, which is identified by its segment identifier. Each of these SIDs may be an Adjacency SID, a Node SID, or a combination of Adjacency and Node SIDs.4

For example, we can send the packet along the path ANBCOPZ by first sending it to N, then sending it to C, then sending it to O, then sending it to P, and then sending it to Z. The way we can encode this in MPLS is by building a label stack at ingress A that includes each node SID that must be visited in order to establish this explicit path. This is shown in the figure below:

---

4 Recall that a Node SID is a specific type of Prefix SID that identifies a node itself. There could be more than one Node SID (for example, if there are multiple loopbacks used for service differentiation).
Another option would be to leverage the shortest path between each node SID to minimize the amount of labels on the stack and to take advantage of ECMP load balancing. We can do this by choosing a node SID that is equally distant across more than one path from the previous node SID in the sequence (or from the ingress). This is shown in the figure below:

When the node SID is used in this fashion, it resembles a loose source route, in that we don’t explicitly define the exact path between node SIDs. Rather, each node that inspects the current, or active SID, forwards the packet along its view of the shortest path to the node that originated the active SID, which then continues the sequence until there are no more SIDs on the stack. This is shown in the figure below:

Segment routing also works with BGP, where it can be used to both distributed labels across a BGP-based topology (such as a datacenter) and to encode egress explicit paths when more than one link can be used to reach a specific inter-AS location.

**Segment Routing Benefits**

One of the key benefits of Segment Routing is that it is a rapid enabler of application-aware, network-based SDN. An application can inject a packet onto the network with a Segment List, thereby directing the network to deliver traffic in a certain way. Of course, the network can perform ingress admission control to these applications (for example, by only allowing traffic matching a certain label stack, or by throttling traffic, or by a variety of other mechanisms). The application can request the label stack from the SDN controller. And, because there is no accumulating state in the network (only at the edges and in the controller), the network can absorb an immense amount of custom crafted flows. This enables real service differentiation and service creation potential. Finally, segment routing can be implemented in classical, hybrid, and pure SDN control environments, with easy migration and coexistence between all three.
Segment Routing allows operators to program new service topologies without the traditional concerns around network state explosion. Using SDN procedures, an ingress node can encode a packet with an ordered list of segment identifiers (MPLS labels or SIDs in IPv6 extension headers) that enable explicit source routing. This can allow for service level differentiation, potentially providing new enhanced service offering opportunities for operators. SR can be introduced slowly into a network without any flag day events, thereby minimizing disruption. Finally, because the network state is stored in the cloud and only programmed at the ingress node, the state is stored in the packet header itself, with no per-flow state in the core, thereby allowing for massive scale.

For example, consider a data center network with 10,000 servers hosting 1 million virtual machines all interconnected by 1000 switches. Traditionally, in order to interconnect these VMs with explicit paths, you would need to create 1 Trillion LSPs (O(10E12)). With SR, you need to distribute the state associated with the switching topology, which is O(10,000). Any additional LSPs do not accumulate any new state. This means that any VM or any host or app or handset can send traffic across a protected, explicit path, without any signaling state. There are many potential applications of this technology.
Annex D: SDN Controllers

SDN Controller Requirements

The SDN controller should be both a platform for deploying SDN applications and provide (or be associated with) an SDN application development environment.

An SDN controller platform should be built to meet the following key requirements:

- **Flexibility**: The controller must be able to accommodate a variety of diverse applications; at the same time, controller applications should use a common framework and programming model and provide consistent APIs to their clients. This is important for troubleshooting, system integration, and for combining applications into higher-level orchestrated workflows.

- **Scale the development process**: There should be no common controller infrastructure subsystem where a plugin would have to add code. The architecture must allow for plugins to be developed independently of each other and of the controller infrastructure, and it must support relatively short system integration times. As an industry example, there are more than 18 active OpenDaylight projects. Fifteen more projects are in the proposal stage. OpenDaylight projects are largely autonomous, and are being developed by independent teams, with little coordination between them.

- **Run-time Extensibility**: The controller must be able to load new protocol and service/application plugins at run-time. The controller’s infrastructure should adapt itself to data schemas (models) that are either ingested from dynamically loaded plugins or discovered from devices. Run-time extensibility allows the controller to adapt to network changes (new devices and/or new features) and avoids the lengthy release cycle typical for legacy EMS/NMS systems, where each new feature in a network device results in a manual change of the device model in the NMS/EMS.

- **Performance & scale**: A controller must be able to perform well for a variety of different loads/applications in a diverse set of environments; however, performance should not be achieved at the expense of modularity. The controller architecture should allow for horizontal scaling in clustered/cloud environments.

SDN Controllers as Application Development Environments

To support development of SDN applications, an SDN controller should also provide (or be associated with) an application development environment that should meet the following key requirements:

- **Use a domain-specific modeling language to describe internal and external system behavior; this fosters co-operation between developers and network domain experts and facilitates system integration.**

- **Code generation from models should be used to enforce standard API contracts and to generate boilerplate code performing repetitive and error-prone tasks, such as parameter range checking.**

- **A domain-specific modeling language and code generation tools should enable rapid evolution APIs and protocols (Agility).**

- **Code generation should produce functionally equivalent APIs for different language bindings.**

- **Modeling tools for the controller should be aligned with modeling tools for devices. Then, a**
common tool chain can be used for both, and device models can be re-used in the controller, creating a zero-touch path between the device and a controller application/plugin that uses its models.

- Domain-specific language/technologies/tools used in the controller must be usable for modeling of generic network constructs, such as services, service chains, subscriber managements and policies.
- The tool chain should support code generation for model-to-model adaptations for services and devices.

The SDN Controller should leverage Model Driven Software Engineering (MDSE) defined by the Object Management Group (OMG). MDSE describes a framework based on consistent relationships between (different) models, standardized mappings and patterns that enable model generation and, by extension, code/API generation from models. This generalization can overlay any specific modeling language. Although OMG focus their MDA solution on UML, YANG has emerged as the data modeling language for the networking domain.

The models created are portable (cross platform) and can be leveraged to describe business policies, platform operation or component specific capability (vertically) as well as (in the case of SDN) manifest at the service interfaces provided by a network control entity (horizontally and vertically within an application service framework).

The outlined solution allows both customer and Service Provider applications to interact with the network to improve the end user's experience, ease the management burden and provide a cost-optimized infrastructure. Initially, it includes the following components:

- Orchestration: tools, technologies, and protocols for centralized intelligence to provide network abstraction and to program network infrastructure such that the end user's experience can be orchestrated and controlled, the network can be automated and optimized on a multilayer basis, and service.

- Application Programming Interfaces: APIs reside at multiple levels of the SDN infrastructure. At the lowest level, programmatic interfaces need to be available at the device level such that software applications can control and access data directly from network equipment. Additional APIs are used higher in the SDN architecture for end user applications to communicate with the controller layers.

- Infrastructure: Over the last decade, the number of network layers and associated technologies required to build modern packet networks has been dramatically reduced. In addition to the physical infrastructure there is also often a virtual network and services infrastructure that needs to be orchestrated. As a consequence the SDN infrastructure needs to be multilayer-aware and able to control physical and virtual network infrastructures.
Annex E: IMS VNF Management Example

This Annex provides an overview of IMS VNF Service Orchestration, Network Service Descriptor, instantiation & scaling and IMS VNF on-boarding. The following figure shows an NFV Orchestration overview including different layers in a deployment:

The ETSI NFV Reference Architecture framework defines a modular and layered architecture with clear roles and responsibilities for each component. This enables openness where an ecosystem of vendors can participate together to deliver a best-of-breed Telco cloud management and orchestration solution to meet an operator’s needs.

As a new element the ETSI NFV Reference Architecture framework introduces a VNF Manager. The VNF Manager is responsible for IMS VNF lifecycle management which includes automating the deployment of VNFs within minutes, the termination of VNFs, automated VNF healing and automated elasticity management for VNFs, including scaling out new VMs during peak conditions and scaling them in when the peak has passed.

The NFV Orchestrator is responsible for automating the lifecycle management of network services. A network service can be composed of one or more VNFs with specific interconnections between the VNFs and connections to existing networks. The VNFs of a network service can be located in one datacenter or the VNFs can be distributed across several datacenters. Depending on the NS structure, the NFVO will access SDN controllers inside datacenters and/or WAN SDN controllers to create and update the networking between VNFs to support network services as shown in the figure above.

It is the task of the NFV Orchestrator to integrate different products from different vendors and provide integration with the Network Infrastructure:

- Integrates the Network Infrastructure: Routers, Switches, etc. to enable an automated service deployment.
- SDN based Network Infrastructure may or may not be in place for each router, firewall, etc. but control is needed to support an automatic deployment of the services.
- The Network Service Descriptor describes the resource and interworking requirements of a set of VNFs, which together realize a service.
OSS/BSS layer has functionalities such as umbrella monitoring, umbrella performance management, unified service quality management, service fulfillment/orchestration applications such as workflow management, business service catalog, inventory and billing systems, for example.

**F.1 Network Service instantiation use cases**

The figure below gives an overview about the instantiation of a network service being defined by a NSD for VoLTE/IMS. The implementation must consider virtualization to achieve isolation and resource control for cloud applications, orchestration to achieve E2E automation, Control and User Plane split, and SDN. Also, common tooling to describe, deploy, start, stop, control, maintain the network.

The next figure shows a more detailed message sequence chart about the single steps of instantiating a network service and the interworking of the single components:
1. NFVO will be triggered by a service orchestration or service fulfillment application to instantiate a network service defined by the assigned network service descriptor (NSD).

2. NFVO will analyze the NSD regarding the included VNF types, the VNF vendor, the flavors of the VNFs, the resource requirements of the VNFs and the connections between the VNFs. Based on this analysis the NFVO creates a workflow and decides on which VIM each VNF should be placed, which VNF external vLANs needs to be created and to which existing network the NS will be connected.

3. NFVO creates the VNF external vLANs with which the VNFs will be connected to each other.

4. NFVO triggers the VNF Manager being assigned to the VNF to instantiate the VNF on the selected VIM and passes required metadata to the VNF Manager. VNF Manager will fetch all VNF details from the referred VNFD and will send a ‘grant request’ message to the NFVO to get approval about the required virtual resources.

5. NFVO checks the availability of the requested virtual resources and approves the ‘grant request’ if the virtual resources are available. If not the ‘grant request’ will be rejected.

6. VNF Manager receives ‘grant request’ approval the VNFM:
   - Creates the VNF internal vLANs by accessing the networking APIs provided by the VIM (native VIM or SDN controller)
   - Allocates the required virtual resources for each VNFC
   - Starts the VMs for all VNFCs
   - Allocates the VM storage
   - Assigns the vNICs to the VNF internal and external virtual LANs

7. After the VNF and all VNFCs are up and running some basic configuration data will be sent to the VNF to finalize the VNF startup and in case of IMS the VNF fetches required configuration data from the CM repository.

8. After all VNFs are successfully instantiated the NFVO will update its available virtual resource data.

9. Connects the new network service instance to the already existing network to go into operation.
F.2 VNF scale out use case

The Figure below shows the message sequence chart for the automated scale out of a VNF and the interworking of the different components:

**VNF Scale Out**

1. The VNFM collects the Telco elasticity management data directly from VNF by using the TEM interface of VNF, and computes a scaling decision.
2. If NFV Orchestrator is part of the operability solution and in case of a scaling-out decision VNFM sends 'Grant request' message to NFVO to approve scale-out of VNF.
3. NFVO sends 'Grant request' approval to VNFM in case of required virtual resources being available.
4. The VNFM instructs VIM to allocate all required virtual resources and to power on all VMs, which are required for the scale out of the VNF.
   Note: If required virtual resources for scale-out are not available, VNFM will stop the scale-out process and will send an error message to NFVO.
5. The VIM starts all VMs being required for the scale-out of the VNF.
6. The VIM signals back to VNFM about the successful instantiation of additional VNFCs.
7. VNF recognizes additional VNFCs and rebalances the traffic load.
8. VNFM informs NFVO about successful scale-out of VNF.

F.3 IMS-VNF on-boarding via NFV Orchestrator

The IMS VNF SW package consists of:

- VNF template package, which includes a standard VNF descriptor and vendor specific enhancements of the VNFD
- One basic SW image valid for all VNFCs
- Single SW packages for each VNFC
The figure above shows a high-level view of the VNF-on-boarding process controlled by the NFVO as it is defined by ESTI NFV MANO:

1. NFVO imports the VNF SW package from SW delivery platform.
2. NFVO uploads the basic SW image to the image store of the virtual infrastructure manager (VIM).
3. NFVO uploads VNF template package to the VNF catalog common to NFVO and VNFM.
4. VNFM fetches standard VNFD and vendor specific enhancements from the VNF catalog. NFVO uploads VNFC SW packages to the IMS SW repository.
References

There are numerous efforts underway across the industry related to SDN/NFV. Some of these efforts are open source and others driven by standard organization activities. The below identifies some of the more recognized and participated efforts in the industry that were referenced in this document.

<table>
<thead>
<tr>
<th>Standards Development</th>
<th>European Telecommunications Standards Institute (ETSI)</th>
<th><a href="http://www.etsi.org">www.etsi.org</a></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Internet Engineering Task Force (IETF)</td>
<td><a href="http://www.ietf.org">www.ietf.org</a></td>
</tr>
<tr>
<td></td>
<td>3GPP</td>
<td><a href="http://www.3gpp.org">www.3gpp.org</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vendor/user community projects</th>
<th>Open Network Foundation (ONF)</th>
<th><a href="http://www.opennetworking.org">www.opennetworking.org</a></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Open Stack Foundation</td>
<td><a href="http://www.openstack.org">www.openstack.org</a></td>
</tr>
<tr>
<td></td>
<td>Alliance for Telecom Industry Solutions (ATIS)</td>
<td><a href="http://www.atis.org">www.atis.org</a></td>
</tr>
<tr>
<td></td>
<td>Open vSwitch</td>
<td><a href="http://www.openvswitch.org">www.openvswitch.org</a></td>
</tr>
<tr>
<td></td>
<td>Data Plane Development Kit (DPDK)</td>
<td>dpdk.org</td>
</tr>
<tr>
<td></td>
<td>ONOS Project</td>
<td><a href="http://www.onosproject.org">www.onosproject.org</a></td>
</tr>
<tr>
<td></td>
<td>TM Forum</td>
<td><a href="http://www.tmforum.org">www.tmforum.org</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linux Foundation</th>
<th>Open Daylight Project (ODL)</th>
<th><a href="http://www.opendaylight.org">www.opendaylight.org</a></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Platform for NFV (OPNFV)</td>
<td><a href="http://www.opnfv.org">www.opnfv.org</a></td>
</tr>
</tbody>
</table>

The below references provide links to efforts underway on particular topics covered in this document.

**NFV Architecture Specifications (ETSI):**

1. Network Functions Virtualization (NFV); Terminology for Main Concepts in NFV (ETSI GS NFV 003 V1.2.1 (2014-12))
2. Network Functions Virtualization (NFV); Architectural Framework (ETSI GS NFV 002 V1.2.1 (2014-12))
3. Network Functions Virtualization (NFV); Infrastructure Overview (ETSI GS NFV-INF 001 V1.1.1 (2015-01))
4. Network Functions Virtualization (NFV); Infrastructure; Compute Domain (ETSI GS NFV-INF 003 V1.1.1 (2014-12))
5. Network Functions Virtualization (NFV); Infrastructure; Hypervisor Domain (ETSI GS NFV-INF 004 V1.1.1 (2015-01))
6. Network Functions Virtualization (NFV); Infrastructure; Network Domain (ETSI GS NFV-INF 005 V1.1.1 (2014-12))
7. Network Functions Virtualization (NFV); Service Quality Metrics (ETSI GS NFV-INF 010 V1.1.1 (2014-12))
8. Network Functions Virtualization (NFV); Management and Orchestration (ETSI GS NFV-MAN 001 V1.1.1 (2014-12))
9. Network Functions Virtualization (NFV); Resiliency Requirements (ETSI GS NFV-REL 001 V1.1.1 (2015-01))
10. Network Functions Virtualization (NFV); NFV Security; Security and Trust Guidance (ETSI GS NFV-SEC 003 V1.1.1 (2014-12))
11. Network Functions Virtualization (NFV); Virtual Network Functions Architecture (ETSI GS NFV-SWA 001 V1.1.1 (2014-12))

IMS, EPC, Policy and Charging Control

1. 3GPP TS 23.002 - Network architecture
2. 3GPP TS 23.228 - IP Multimedia (IM) Subsystem; Stage 2
3. GSMA FCM.01 - VoLTE Service Description and Implementation Guidelines
4. 3GPP TS 24.229 - IP Multimedia Call Control based on SIP and SDP; Stage 3
5. ETSI GS NFV 002 - Network Functions Virtualization (NFV); Architectural Framework
7. 3rd Generation Partnership Project (3GPP), "TS 23.060, General Packet Radio Service (GPRS); Service description," ed.
8. 3rd Generation Partnership Project (3GPP), "TS 29.212, Policy and Charging Control (PCC); Reference points," ed.

SGi-LAN

SDN


Security

10. Telephone Industry Association (TIA) specification J-STD-025
Acronyms

3GPP – 3rd Generation Partnership Project
5G – Fifth Generation (mobile network)
AAA – Authentication, Authorization and Accounting
AES – Advanced Encryption Standard
AES-NI – AES New Instructions
AF – Application Function
AGW – Application GateWay
AN – Access Node
API – Application Programming Interface
APN – Access Point Name
ARM – Advanced RISC Machines
AS – Application Server
ASIC – Application-Specific Integrated Circuit
ATCF – Access Transfer Control Function
ATGW – Access Transfer Gateway
AVP – Attribute Value Pair
AVS – Adaptive Video Streaming
BBERF – Bearer Binding and Event Reporting Function
BGCF – Breakout Gateway Control Function
BGF – Border Gateway Function
BGP – Border Gateway Protocol
BGP-LS – BGP Link State
BIOS – Basic Input/Output System
BMSC – Broadcast-Multicast Service Center
BNG – Broadband Network Gateway
BSS – Business Support Systems
BW – BandWidth
C-BGF – Core Border Gateway Function
CALEA – Communications Assistance for Law Enforcement Act
CAT – Cache Allocation Technology
CC – Content of Communication
CCF – Charging Collection Function
CDMA – Code Division Multiple Access
CDN – Content Delivery Network
CDR – Charging Data Records
CGF – Charging Gateway Function
CI/CD – Continuous Integration/Continuous Deployment
CLI – Command Line Interface
CM – Configuration Management
CMP – Cloud Management Platform
CMT – Cache Monitoring Technology
CMTS – Cable Modem Terminal System
CoS – Class of Service
CP – Control Plane
CPE – Customer Premise Equipment
CRF – Charging Rules Function
CRUD – Create Read Update Delete
CS – Circuit Switched
CSCF - Call Session Control Function
CSP – Communication Service Provider
CUPS - Control and User Plane Separation
D2D – Device to Device
DC – Data Center
DCI – Data Center Interconnect
DDIO – Data Direct I/O
DÉCOR – Dedicated Core Network
DevOps – Development and Operations
DHCP – Dynamic Host Configuration Protocol
DNS – Domain Name System
DOA – Dead On Arrival (applies to VMs)
DoS – Denial of Service
DP – Data Plane
DPDK – Data Plane Development Kit
DPDK-AE - Data Plane Development Kit – Acceleration Enhancements
DPU – Drop Point Units
DRA – Diameter Routing Agent
DRM – Digital Rights Management
DSCP - Differentiated Services Code Point
DSL – Digital Subscriber Line
DSLAM - Digital Subscriber Line Access Multiplexer
DSP – Digital Signal Processor
DWDM – Dense Wavelength Division Multiplexing
E2E – End-to-End
E-CSCF – Emergency CSCF
E-LAN – Ethernet LAN
E-LINE- Ethernet Line
E-TREE – Ethernet Tree
E-UTRAN - Evolved Universal Terrestrial Access Network
ECMP - Equal-Cost Multi-Path routing
EENSD - End-to-End Network Service Descriptor
eIMS-AGW - IMS Access GateWay enhanced for WebRTC
EM – Element Manager or Elasticity Manager
eMBMS – LTE version of Multimedia Broadcast Multicast Services
EMS – Element Management System
ENUM – Electronic Numbering
EPC – Evolved Packet Core
ePDG – Evolved Packet Data Gateway
ETSI – European Telecommunications Standards Institute
EV B - Edge Virtual Bridging
EVPN – Ethernet VPN
FCoE - Fibre Channel over Ethernet
FE – Forwarding Entity
FIB – Forwarding Information Base
FMSS – Flexible Mobile Services Steering
FSOL – First Sign Of Life
FW – FireWall
Gi-LAN – G Interface LAN
GBR – Guaranteed Bit Rate
GGSN - Gateway GPRS Support Node
GID – Group ID
GNU – GNU’s Not Unix\(^5\)
GPE – Generic Protocol Extension
GPG - GNU Privacy Guard
GRE - Generic Routing Encapsulation
GTP – GPRS Tunneling Protocol
G-VNF – Generic VNF
GW – GateWay
HA – High Availability
HSS – Home Subscriber Server
HSTS – HTTP Strict Transport Security
HTTP – HyperText Transport Protocol
HTTPS – HTTP Secure
HW – HardWare
I2RS – Interface to the Routing System
I-CSCF – Interrogating CSCF
IaaS – Infrastructure as a Service
IAB – Internet Architecture Board
IBCF – Interconnection Border Control Function
IBN – Intent-Based Networking
IDS – Intrusion Detection System
IETF – Internet Engineering Task Force
IGP – Interior Gateway Protocol
IMEI - International Mobile Station Equipment Identity
IMPI – IP Multimedia Private Identity
IMPU – IP Multimedia Public Identity
IMS – IP Multimedia Subsystem
IMS-MGW - IMS Media Gateway
IMSI - International Mobile Subscriber Identity
IoT – Internet of Things
IP – Internet Protocol
IP-CAN – IP Connectivity Access Network
IPMI – Intelligent Platform Management Interface
IPS – Intrusion Prevention System
IPv4 – IP version 4
IPv6 – IP version 6
IPSE – IP Switching Element
IPSec – IP Security
IRI – Intercept Related Information
ISIM - IP Multimedia Services Identity Module
IWF – InterWorking Function.

\(^5\) GNU – GNU is a recursive acronym
KPI – Key Performance Indicator
KVM – Kernel Virtual Machine
L0 – Layer 0 in the protocol stack (optical/DWDM layer)
L1 - Layer 1 in the protocol stack (Physical layer)
L2 – Layer 2 in the protocol stack (Data Link layer)
L3 – Layer 3 in the protocol stack (Network layer)
L4 – Layer 4 in the protocol stack (Transport layer)
LAG – Linkg Aggregation
LAN – Local Area Network
LDAP - Lightweight Directory Access Protocol
LEA – Law Enforcement Agency
LGW – Local GateWay
LI – Lawful Intercept
libvirt – Library virtualization
LIMS – Lawful Interception Management System
LIPA - Local IP Access
LLC – Logical Link Control
LRF – Location Retrieval Function
LRO – Large Receive Offload
LSA – Link State Advertisements
LSO – Large Segment Offload
LSP – Label Switched Path
LTE – Long Term Evolution
M2M – Machine-to-Machine
MAC – Media Access Control
MACD - Moving Average Convergence Divergence
MANO – MANagement and Orchestration
MBB – Mobile BroadBand
MBM – Memory Bandwidth Monitoring
MBMS - Multimedia Broadcast Multicast Services
MCC – Mobile Country Code
MDU – Multi-Dwelling Units
MGCF – Media Gateway Control Function
MGW – Media Gateway
MM/mm – Millimeter Wave
MME – Mobility Management Entity
MP-BGP - Multi-Protocol BGP
MPLS – MultiProtocol Labeling Switching
MPS – Multimedia Priority Service
MRB – Media Resource Broker
MRF – Media Resource Function
MRFC - Multimedia Resource Function Controller
MRFP - Multimedia Resource Function Processor
MSCI - Media Specific Compute Instructions
MSDC - Massively Scalable Data Center
MSISDN - Mobile Station International Subscriber Directory Number
MTBF – Mean Time Between Failure

---

6 libvirt – An open source API, daemon and management tool
MTC – Machine Type Communications
MTSO – Mobile Telephone Switching Office
MTTD – Mean Time To Diagnosis
MTTR – Mean Time To Repair
MVNO – Mobile Virtual Network Operator
NAPT – Network Address and Port Translation
NAT - Network Address Translation
NB – NorthBound
NBI – NorthBound Interface
NENA – National Emergency Number Association
NETCONF - NETwork CONfiguration Protocol
NetVirt – Network Virtualization
NF – Network Function
NFV – Network Function Virtualization
NFVI – Network Function Virtualization Infrastructure (sometimes written as NFVI)
NFVaaS – NFV Infrastructure as a Service
NFVO – Network Function Virtualization Orchestrator
NG-OSS – Next Generation OSS
NGMN - Next Generation Mobile Networks (Alliance)
NIC – Network Interface Card
NLRI - Network Layer Reachability Information
NS – Network Service
NSD – Network Service Descriptor
NSH – Network Service Header
NUMA – Non-Uniform Memory Access
NVGRE - Network Virtualization using Generic Routing Encapsulation
O&M – Operations and Maintenance
OCP – Open Compute Project
OCS – Online Charging System
ODAM - Operations, Orchestration, Data Analysis & Monetization
ODP – Open Data Plane
OF – OpenFlow
OFCS - Offline Charging System
OLT – Optical Line Terminal
OAM&P - Operations, Administration, Maintenance,and Provisioning
ONF – Open Networking Foundation
ONOS – Open Network Operating System
ONT – Optical Network Terminal
ONU – Optical Network Unit
OPNFV – Open Platform for NFV
OS – Operating System
OSPF - Open Shortest Path First
OSS – Operations Support System
OTN – Optical Transport Network
OTT – Over The Top
OVF – Open Virtual Format
OVS – Open Virtual Switch
OVSDB - Open vSwitch DataBase management protocol
P router – Provider router
P-CSCF – Proxy CSCF
PaaS – Platform as a Service
PCC – Policy Charging and Control
PCE – Path Computation Element
PCEF – Policy and Charging Enforcement Function
PCEP – Path Computation Element Protocol
PCI – Peripheral Component Interconnect
PCI-SIG - Peripheral Component Interconnect Special Interest Group
PCIe - Peripheral Component Interconnect Express
PCRF – Policy and Charging Rules Function
PDN - Packet Data Network
PNF – Physical Network Function
PE – Provider Edge
PGW - Packet Data Network Gateway
PIM – Physical Infrastructure Management/Manager
PLMN - Public Land Mobile Network
PMD - Poll Mode Drivers
PNF – Physical Network Function
PNFD – PNF Descriptor
PNMTM - Private Network Traffic Management
Pol – Point of Interception
PoP – Point of Presence
PPV – Pay Per View
QAT – Quick Assist Technology
QCI – QoS Class Indicator
QEMU – Quick Emulator
QoE – Quality of Experience
QoS – Quality of Service
QSFP - Quad Small Form-factor Pluggable
RAA - Re-Auth-Answer (Diameter protocol)
RAN – Radio Access Network
RAS – Reliability, Availability, Servicability
RBAC – Role-Based Access Control
RCAF – RAN Congestion Awareness Function
RCS – Rich Communication Services
RdRand – Random Number (Intel instruction)
ReST – Representational State Transfer
ReSTCONF – ReST CONFiguration protocol
RGW – Residential Gateway
RO – Resource Orchestrator
ROADM - Reconfigurable Optical Add-Drop Multiplexer
RSS – Receive Side Scaling
RTT – Round Trip Time
S-CSCF – Serving CSCF
S-VNFM – Specific VNFM
SaaS – Software as a Service
SAEGW - System Architecture Evolution GateWay
SAML - Security Assertion Markup Language
SB – SouthBound
SBI – SouthBound Interface
SDF – Service Data Flow
SDM – Subscriber Data Management
SDN – Software Defined Networking
SDT – Small Data Transmission
Seccomp – Secure computing mode
SEG – Security Gateway
SELinux – Security Enhanced Linux
SFC – Service Function Chaining
SFTP – Secure File Transfer Protocol
SGi-LAN – SG Interface LAN
SGW – Serving Gateway
SIM – Subscriber Identification Module
SIP – Session Initiation Protocol
SIPTO – Selected Internet IP Traffic Offload
SLA – Service Level Agreement
SLF – Subscriber Location Function
SMS – Short Message Service
SMSC – Short Message Service Center
SMT – Simultaneous Multi-Threading
SNMP – Simple Network Management Protocol
SoC – System on a Chip
SPI – Service Path Identifier
SPR – Subscriber Policy Repository or Subscription Profile Repository
SPT – Shortest Path Tree
SR-IOV – Single Root - I/O Virtualization
SR OAM – Segment Routing - Operations And Management
SRVCC - Single Radio Voice Call Continuity
SSD – Solid-State Disk
SSL – Secure Sockets Layer
sVirt – Secure Virtualization
SW – SoftWare
TAS – Telephony Application Server
TCAM - Ternary Content-Addressable Memory
TCP – Transmission Control Protocol
TDF – Traffic Detection Function
TLS – Transport Layer Security
ToR – Top of Rack
TOSCA – Topology and Orchestration Specification for Cloud Applications
TrGW – Transition Gateway
TA – Traffic Analysis
TS – Technical Specification
TSSF – Traffic Steering Support Function
TTM – Time To Market
TXT – Trusted Execution Technology
UDC – User Data Convergence
UDP – User Datagram Protocol
UDR – User Data Repository
UE – User Equipment
UID – User ID
UPCON – User Plane CONgestion
USB – Universal Serial Bus
USIM - Universal Subscriber Identity Module
vCPE – virtualized CPE
VEB – Virtual Ethernet Bridge
VEPA - Virtual Ethernet Port Aggregator
vEPC – virtualized EPC
VIM – Virtual Infrastructure Management
vIMS – virtualized IMS
VL – Virtual Link
VLAN – Virtual LAN
VLD – Virtual Link Descriptor
VLR – Virtual Link Record
VM – Virtual Machine
VMDQ - Virtual Machine Device Queues
VMNIC – Virtual Machine Network Interface Card
VNDq - Virtual Machine Device Queues
VNF – Virtual Network Function
VNFaaS – VNF as a Service
VNFC – VNF Component
VNFD – VNF Descriptor
VNFFG – VNF Forwarding Graph
VNF-FG - VNF Forwarding Graph
VNFFGD – VNF Forwarding Graph Descriptor
VNFFGR - VNFFG Record
VNFM – Virtual Network Function Manager
VNFR – VNF Record
vNIC – Virtualized NIC
VNPPaaS – Virtual Network Platform as a Service
VoLTE – Voice over LTE
VoWiFi – Voice over WiFi
vPE – virtual Provider Edge (router)
VPLS – Virtual Private LAN Service
VRF – Virtual Routing and Forwarding
VPN – Virtual Private Network
vSwitch – Virtual Switch
VTEP – VXLAN Tunnel EndPoint
VXLAN - Virtual Extensible LAN
WAN – Wide Area Network
WebRTC – Web Real Time Communications
YANG – Yet Another Next Generation

---

7 YANG – YANG is yet-another-next-generation modeling language