# Part III

# Traffic Transport and Network Management

# 10

# Mobile Network Function and Service Delivery Virtualization and Orchestration

Peter Bosch,[1] Alessandro Duminuco,[1] Jeff Napper,[1] Louis (Sam) Samuel,[2] and Paul Polakos[3]
[1] *Cisco Systems, Aalsmeer, The Netherlands*
[2] *Cisco Systems, Middlesex, UK*
[3] *Cisco Systems, San Jose, CA, USA*

## 10.1 Introduction

The concept of virtualization in telecommunications is not new. Ever since the need for the testing of telecommunication functions appeared, there had to be some means of testing a function either in a simulated environment or simulating the function itself to see if modifications and evolutions of the functions were viable prior to deployment. In either case, virtualization has gained ground as the potency of computation and storage has increased with every evolution of processors and storage. This has led to the point in telecommunications where network functions that were once considered only suitable to be run on bespoke hardware because of various limitations are now potential applications that can be run on common off-the-shelf processing. The idea of orchestration in telecoms is also not a new one. Orchestration in essence is the intelligent automation of repetitive processes whether for business or engineering.

Modern telecommunication networks have evolved over many decades. The evolution has usually meant that newer portions of the network have to coexist with older portions of the network. The resulting heterogeneity naturally led to increased network complexity. What this also meant is that the mechanisms by which networks and their elements are provisioned

*Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*, First Edition.
Edited by Madhusanka Liyanage, Andrei Gurtov, and Mika Ylianttila.
© 2015 John Wiley & Sons, Ltd. Published 2015 by John Wiley & Sons, Ltd.

have become tied to vendor-specific systems. The implication of both of these things is that to deploy new services and simply manage the existing network has become a relatively expensive task.

The confluence of improving off-the-shelf processing, leading to the enablement of virtualization, and of the application of more engineering processes can lead to an advantageous situation where not only large portions of the network can be virtualized onto inexpensive commoditized hardware but also the ensemble can be orchestrated into new solutions. The telecommunication industry has realized this, and there are many initiatives underway that seek to exploit this confluence. One such initiative is the ETSI network function virtualization (NFV). This initiative seeks to standardize the interfaces between the virtualized functions and the overall management of these functions. NFV relies heavily on an underlying programmable networking substrate, commonly referred to as software defined networking (SDN), to allow dynamic deployment, isolation, and control of a multiplicity of network services (NS) in a multitenant data center environment. In this chapter, we describe in detail and in the mobile network context the ETSI NFV architecture and underlying support provided by SDN.

## 10.2 NFV

The ETSI NFV architecture is set apart from other cloud management approaches through its aim to manage and operate virtualized critical network functions in a private data center (such as packet cores, IMS, etc.). In this architecture, guarantees for high availability and reliability (greater or equal to the current “5–9 s”[1]) are to be met while simultaneously achieving lower capital and operational expenses for the same services provided via traditional means. At the time of writing, work is not complete and there are still debates on how such data centers will be operated. However, there is enough information to report on their architecture and likely modes of operation. The basic premise of NFV is that network functions have their software implementation decoupled from the computation, storage, and network resources that they use. This means that for the telecom service provider (SP), new ways of operating, administering, maintaining, and provisioning such network functions will emerge.

### *10.2.1 The Functionality of the Architecture*

Figure 10.1 shows the current NFV Management Architecture, and each of its functional blocks is described in more detail in the following text (for greater detail, see Ref. [1]).

#### 10.2.1.1 Network Function Virtualization Orchestrator

The top-layer orchestrator of the NFV system is the Network Function Virtualization Orchestrator (NFVO). It instantiates and manages services provided through virtual network functions (VNFs) of a VNF 3GPP virtual Evolved Packet Core (vEPC) or 3GPP IMS. The NFVO endeavors to accomplish three objectives: (i) instantiate VNFs, (ii) manage the

[1] In short, they have service outages of less than 315 seconds per year.

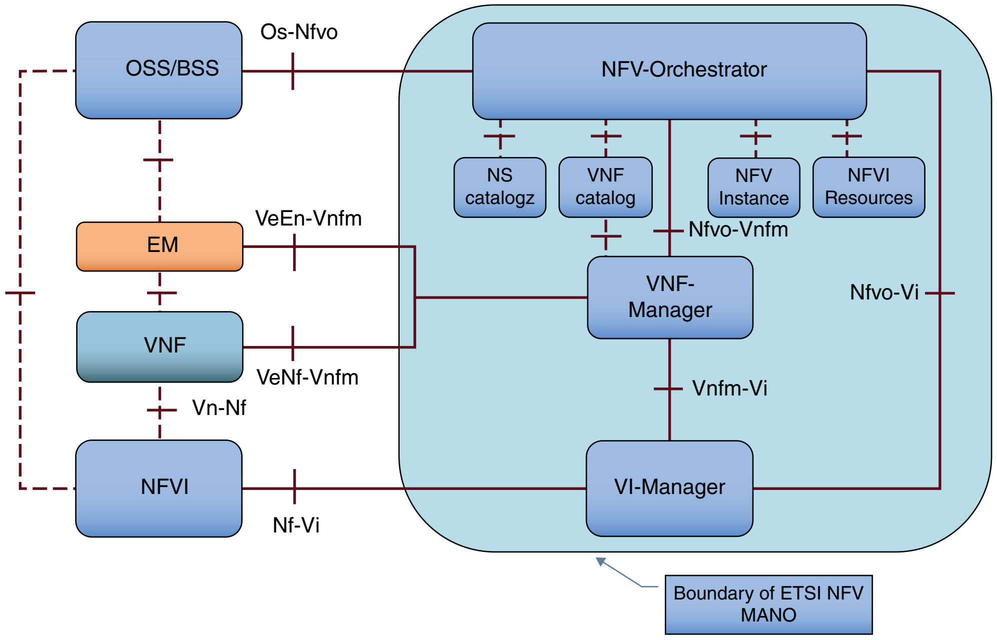

**Figure 10.1** ETSI NFV Management and Orchestration (MANO) architecture.

service-level agreements (SLAs) of the VNFs, and (iii) perform NS orchestration. In order to achieve these objectives, the NFVO uses a set of descriptor files (NS, network function, virtual link, and Virtual Network Function Forwarding Graph Descriptors (VNFFGD)) to drive the instantiation of the VNFs by orchestration system. The descriptor files are templates or models that describe a VNF or an NS in terms of its required resources, configurations, and associated SLA descriptions. The descriptor files are covered in more detail in a later section.

In order to manage the SLA of the NS and VNFs under its control, the NFVO has the end-to-end view of the resources being allocated to the NS and VNFs. To accomplish this, the NFVO possesses the current snapshot of the distribution of in-use and available NFV data center resources retrieved from the Virtualized Infrastructure Manager (VIM) via the NFVO-Vi interface. The NFVO uses this information along with application-specific information contained in the NFV descriptors (such as scaling rules for a given VNF) and the overall projected resource consumption to make placement decisions where the NFV should be instantiated into the NFV data center. To assist in this activity, the NFVO notionally uses a number of databases. There are essentially two categories of databases:

- Databases that hold information (model definitions and descriptions) regarding NS and their components—the NS Catalog and VNF Catalog (see Fig. 10.1)
- Databases that describe what NS are currently deployed and state and availability of infrastructural resources—the NFV Instance and Network Function Virtual Infrastructure (NFVI) Resource databases (see Fig. 10.1)

***NS Catalog***

The NS Catalog is a repository of the onboarded NS definitions. An NS is described by its Network Service Descriptor (NSD). The NSD is a template or model that describes the deployment of an NS. The description includes the service topology (the VNFs used in the service and the relationships between them, including VNF Forwarding Graph) as well as NS characteristics such as SLAs and any other relevant information necessary for the NS onboarding and life cycle management of its instances. Currently, ETSI NFV proposes the use of TOSCA [2] templates or YANG [3] models for this purpose.

***VNF Catalog***

This is a database of all the VNFs that can be started by the NFV orchestration system. The onboarded VNFs are described by their configuration files known as Virtual Network Function Descriptor (VNFD). The VNFD contains information and parameters sufficient to describe the VNF's operational and life cycle behavior. Potentially, the VNFD is also a TOSCA template or a YANG model.

***NFV Instance***

This is a database of all the currently active (running) VNF applications and their mapping to virtual NS as well as additional run-time instance-specific information and constraints.

***NFVI Resource***

This is effectively a database that an inventory of all the available, reserved, and allocated resources across the entirety of SP domains. This database is kept consistent with the state of the NFV system so that it can be interrogated to reserve, allocate, or monitor the state of resources in the system. This is the entry point to the process by which SLAs in the system are maintained.

Although the NFVO in principle possesses broad information on a VNF, it in fact does not and should not need to understand the functional role or operation of the VNF itself. The information it does have is there to manage a service and enforce its SLA.

#### 10.2.1.2 VNF Manager

The middle layer of the orchestration system is provided by the VNF Manager (VNFM). The VNFM is responsible for the life cycle management of VNF instances. Life cycle management in this context means:

- Handling VNF instantiation, that is, the allocation and configuration of virtual machines (VMs)[2] for the VNF.
- Monitoring the VNF—This can mean one of two things:
  - Either direct monitoring of the VMs in terms of CPU load, memory consumption, etc.
  - Or the presentation of application-specific data from the VNF VMs.

  In both cases, the VNFM collects the VNF application-specific data and NFVI performance measurements and events. This data is passed up to the assurance function of the NFVO for analysis or to the VNF Element Management System (EMS).

[2] In ETSI NFV terminology, an NFV VM is termed as Virtual Deployment Unit (VDU).

- Elastic control of the VNF VMs, that is, if the VNF application allows it, given a set of triggering events spin up new VMs or remove existing VMs.
- Assisted or automatic healing of the VNF, that is, restarted, stalled, or stopped VMs (assuming the VNF is capable of such management).
- Terminating the VNF VMs, that is, withdraw the VNF from service when requested by the OSS via the NFVO.

A VNFM can potentially manage single or multiple VNF instances of the same or different types. Moreover, as NFV matures, the possibility of a generic VNFM becomes likely. However, for reasons of pragmatism (ETSI NFV may mature slowly), the NFV Management and Orchestration (MANO) architecture also supports cases where VNF instances need specific functionality for their life cycle management, and this may be delivered as part of the VNF package. In this context, a VNF package is deemed to be the collection of the VNFD and its software image(s) and any additional information used to prove the validity of the package. This means that the VNFM can come bundled with the application. The advantages and disadvantages of this approach are covered in a later section.

#### 10.2.1.3 VIM and NFVI

The lower layer of the orchestration system is the VIM. It is responsible for controlling and managing the compute, storage, and network resources within one operator's subdomain (i.e., a physical data center). In ETSI NFV, the physical resources (the compute, storage, and network) and software resources (such as the hypervisor) are collectively termed the NFVI. The VIM may be capable of handling multiple types of NFVI resources (via, e.g., Openstack cloud management system), or it may only be capable of handling a specific type of NFVI resources (an example of this would be VMware's vSphere).

In either case, the VIM is responsible for:

- The management of allocation/upgrade/deallocation and reclamation of NFVI resources
- The association of the virtualized resources to the compute, storage, and networking resources
- The management of hardware (compute, storage, and networking) and software (hypervisor) resources
- The collection and forwarding of performance measurements and events from the hardware and software to either the VNFM or the NFVO

At the time of writing, there is some debate as to whether network resources are explicitly managed by the VIM, for example, through open-source SDN controllers as provided by OpenDaylight [4], in which Openstack Neutron can interface with OpenDaylight [5].

#### 10.2.1.4 Traditional Elements: Element Manager and Operation and Business Subsystems

Figure 10.1 shows some traditional elements included within. These are the Element Management (EM) and Operation and Business Subsystems (OSS/BSS) responsible for the conventional means of managing and assuring network functions. There are a couple of implications associated with the inclusion of these elements. Firstly, the inclusion of both elements means that a VNF can be managed via conventional means, and secondly, their inclusion provides a starting point for operational migration. We note that at the time of writing, ETSI NFV has not resolved how functions such as the EM would evolve given that the combination of NFVO and descriptor files has the nucleus of similar functionality.

Figure 10.1 shows the current NFV Management Architecture, and each of its functional blocks is described in more detail in the following text (for greater detail, see Ref. [1]). The NFV MANO architecture is a three-layer orchestration system. The top layer (the NFVO) orchestrates the NS, the middle layer (the VNFM) orchestrates the life cycle of virtual functions, and the bottom layer orchestrates data center resources and infrastructure. There is cardinality associated between these layers. There is a single NFVO that controls one or more NFV data centers. Therefore, one NFVO controls one or more VIMs. There is a one-to-one relationship between the VIM and a data center or data center partition. An NFVO can be associated with multiple VNFM, and a VNFM can have an association with one or more VNFs.

### *10.2.2 Operation of the ETSI NFV System*

The entire ETSI NFV system can work in one of two main modes:

(i) Traditional, that is, driven directly via the OSS through an existing NMS/EMS that manages the VNF. In this mode, essentially all that has changed is that the network function has been virtualized.
(ii) Orchestrated, that is, driven via the ETSI NFV MANO system. In this mode, the entire system is driven through a collection of descriptor files that are held in the NFVO catalogs.

#### 10.2.2.1 ETSI NFV Descriptor Hierarchy

The ETSI NFV descriptor files have a hierarchy to them (Fig. 10.2). At the top level are the NSDs: the NSD and VNFFGD. The NSD and the VNFFGD describe the service in terms of the VNFs that compose the service, its required network topology, and its KPIs. The middle level of the hierarchy includes the Virtual Link Descriptor (VLD) and VNFD. The VLD describes the inter-VNF link requirements (bandwidth, QoS, Hypervisor, vSwitch, NIC, etc.), while the VNFD describes the VNF. The VNF is further broken down to form the lower level of the hierarchy and is composed of three subcomponents; these are the VNFD_element, VNF_element, and VDU_element. The VNFD_element essentially names the VNFD. The VNF_element describes the composition of the VNF in terms of the number of separate VMs and their overall management, whereas the VDU_element describes the individual VMs that make up the VNF. The NSD and VNFD are described in more detail in the following subsections.

***The NSD***

The NSD describes the service in terms of a list of VNFs that go to compose the service, a list of VNF Forwarding Graphs that essentially describe how the various NVFs are connected together to effect the service, and a list of the associated dependencies for each of the NFV. The dependencies can be as simple as describing the order in which the NFV are brought up to more complicated relationships, for example, predication by other NS. The NSD also

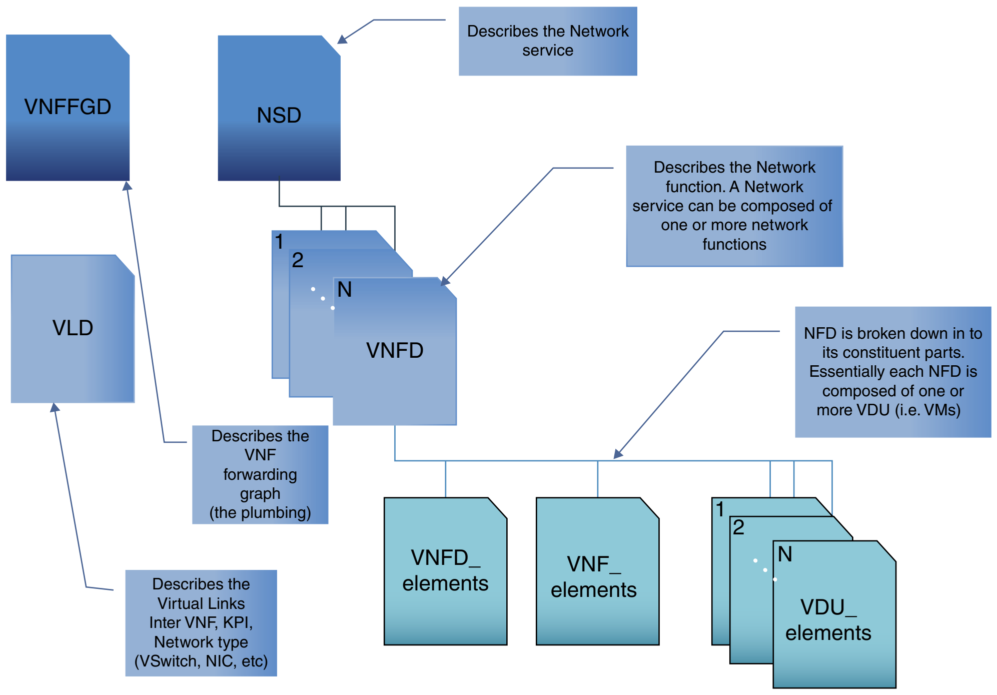
Describes the Network service
VNFFGD
NSD
Describes the Network function. A Network service can be composed of one or more network functions
1
2
N
VNFD
VLD
NFD is broken down in to its constituent parts. Essentially each NFD is composed of one or more VDU (i.e. VMs)
Describes the VNF forwarding graph (the plumbing)
VNFD_ elements
VNF_ elements
VDU_ elements
Describes the Virtual Links Inter VNF, KPI, Network type (VSwitch, NIC, etc)

**Figure 10.2** Hierarchy of NFV descriptors.

includes information that describes the SLA for the service. This is achieved by disclosing a list of parameters to monitor associated with a given configuration of the service and a list of autoscaling policies described in terms of metadata for that service.

With reference to TMF SID (see Ref. [4] and references contained therein), the NSD loosely approximates a Customer Facing Service (CFS), that is, something that can be ordered by a customer of the system. In this respect, examples of an NSD would be voice services, Internet access, video optimization, or a content delivery network—depending on who or what the end user is (e.g., a subscriber or an enterprise).

### *The VNFD*

The VNFD describes the VNF components down to the VM level. The VNFD is composed of three elements:

- VNFD_element—Basically the name of the VNF and who the vendor of the VNF is.
- VNF_element—This is a list of elements that make up the VNF. Examples of information contained in this element are the number of VMs that make up the VNF; the set of workflows that describe initiation, termination, graceful shutdown, and other life cycle events; a list of the type of network connectivity required by the NFV components; a list of the external interfaces exposed by the VMs enabling connection to other virtual and physical network functions; a list of the internal connectivity/interfaces between the VNF VMs; a list of dependencies of the VNF components; a list of monitoring parameters (e.g., CPU utilization,

interface bandwidth consumption, and NFV-specific parameters); a list of deployment configurations (flavors) and an associated list of assurance parameters and requirements for each deployment flavor; a list of autoscale policies; and a manifest file containing a list of all the files included in the package, the modeling language version, and encoding format.
- VDU_element—This is essentially a description of an individual VM that makes up the collection of distinct VMs that compose the VNF. Naturally, there is one VDU_element per VM. Examples of information that this element contains are the number of instances of **this** type of VM; a reference to the VM image (i.e., the image location); the VM storage characteristics in terms of size of storage and key quality indicators (such as performance, reliability, and availability); the VM processing characteristics in terms of processing power and key quality indicators; the VM memory requirement; the VM I/O virtual bandwidth; the VM initiation, termination, and graceful shutdown workflows; the redundancy model (e.g., active–active, active–passive); and finally the scaling parameters, for example, the minimum and maximum number of instances that can be created to support scale-out.

With reference to TMF SID [4], the VNFD is the equivalent of a Resource Facing Service (RFS), that is, something that describes a function in terms of the resources it consumes.

#### 10.2.2.2 Initiating NS and VNFs

Once the descriptors are onboarded, then the NFV MANO system can finally begin to do useful things with them, for example, bringing up a VNF or start an NS.

***NS Instantiation***

Exactly how the OSS gets the stimulus for initiating the NFV NS is out of scope of this document; suffice it to say that an initiation command is passed between the OSS and NFVO. Currently, this is the only way that an NFV service can be brought into existence. However, there are proposals for a more dynamic operation of NS instantiation, where the stimulus can arrive from more than just the OSS, for example, a self-service portal that the SP may expose to its business partners.

This instantiation command carries the NSID information. This information includes:

- NSD reference that is the NS identifier of an already onboarded NS
- Optionally, a list of already running VNFs from which the service can be constructed
- The method of scaling for the network work service (manual or automatic)
- The flavor (the particular configuration) for the service
- A list of threshold descriptors for the service
- A list of autoscale policy descriptors for the service if autoscaling is permitted

The logical implication of the last three items in the NSID is that the:

- VNFs for this particular service must be tagged in a way that the VNFM can pass monitoring data relevant to the service to an assurance system capable of grouping the data for monitoring how the service is performing.

- Thresholds and autoscale descriptors override the existing thresholds and autoscale descriptors contained in the NSD if they are valid (consistent and within accepted permitted range of values).

Service instantiation can proceed along one of several alternative paths. The NFVO determines the appropriate path by comparing the service VNF dependencies (as described in the NSD) against the list of active VNFs maintained in the NFV instance database. The paths are categorized as follows:

- *None of the service components are running.* If the VNFs that compose the NS already exist in the VNF Catalog and none of the VNFs are already instantiated, then the NFVO instantiates the creation of the necessary VNFs using the information already stored to in the VNF Catalog, and the service is stitched together using the information contained in the VNFFG references of the NSD.
- *All of the service's VNFs are already running.* If this is the case and the necessary VNFs have no restriction on them being used for multiple services, then all that may be required is for the service to be plumed using the VNFFG references of the NSD.
- *Some of the service's VNFs are running.* In this case, the NFVO then initiates only those that are missing and once again stitches together the VNFs (new and existing) using the information contained in the VNFFG references of the NSD.

The flow captured in Figure 10.3 shows this activity.

***VNF Instantiation***

VNF instantiation refers to the process of identifying and reserving the virtualized resources required for a VNF, instantiating the VNF, and starting the VMs associated with each VNF, in short the initial phases of VNF life cycle management.

Unlike the NS instantiation request, the VNF instantiation request can come from a number of sources. It can be as part of an NS instantiation (see previous section); it can be as part of commissioning a new VNF as a direct request from the OSS; it can be as a result of a scaling request from a VNFM; or it can be as a request from the EMS of the VNF. In any event, the NFVO receives a Virtual Network Function Initiation Descriptor (VNFID) that includes (i) a reference to the VNFD to instantiate the VNF, (ii) a list of network attachment points for the VNF, (iii) the scaling methodology and threshold descriptors (i.e., the parameters on which to trigger scaling if automatic scaling has been selected), and (iv) the flavor (configuration) to use.

We note that much of the information presented here is still under discussion by ETSI NFV and that at the time of writing, the operational flows and procedures have yet to be defined normatively and may therefore change.

### *10.2.3 Potential Migration and Deployment Paths*

Given the architectural blocks described in a previous section, it can be seen that the VNF, EMS, and VNFM can be packaged in a number of configurations that can give rise to a number of potential deployment migration paths from current to future modes of operation. These configurations range from having individual VNFs mostly orchestrating themselves to fully

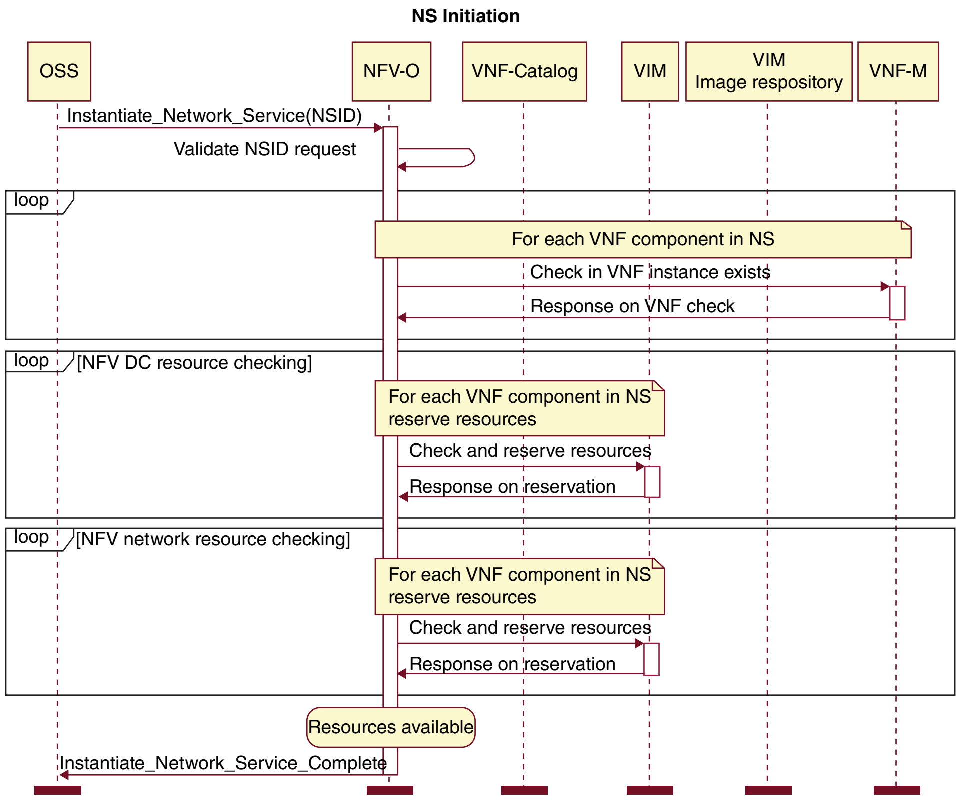
NS Initiation
OSS
NFV-O
VNF-Catalog
VIM
VIM Image respository
VNF-M
Instantiate_Network_Service(NSID)
Validate NSID request
loop
For each VNF component in NS
Check in VNF instance exists
Response on VNF check
loop [NFV DC resource checking]
For each VNF component in NS reserve resources
Check and reserve resources
Response on reservation
loop [NFV network resource checking]
For each VNF component in NS reserve resources
Check and reserve resources
Response on reservation
Resources available
Instantiate_Network_Service_Complete

**Figure 10.3** Initiation of network service.

automated operation of the SP network based on ETSI NFV MANO. The next four subsections indicate those migration configurations for a VNF, from the simplest and closest to current mode of network operation to the configurations closest to the future mode of operation end state.

### 10.2.3.1 VNFM Packaged with EMS and VNF

The simplest configuration is where the VNFM is packaged with the application (the VNF) and its EMS/NMS (see Fig. 10.4). The advantages of this configuration are that it:

- Hides the complexity of running the application from the NFVO:
  - The VNF is self-contained. In effect, the NFVO system brings up the very first VM of the application—the VNFM. The VNFM then brings up the rest of the application.
  - The assurance for the application is provided by traditional means. The VNFM may provide an aggregation point for collecting monitoring and performance information before passing it to the application EMS.

- Scaling events (for instance, to satisfy SLA requirements) can be handled either as manual interventions into the EMS via the OSS or as events triggered by scaling rules in the VNFM that subsequently lead to automatic action or manual intervention from the OSS.

  Note that all requests for additional resources (VMs and network bandwidth) still have to be made via the NFVO and authorized before the resources can be used.
- Allows the application to be run via more traditional means (OSS/NMS/EMS/VNF). In other words, operation of the VNF is possible in the absence of an NFVO.

There are possible long-term disadvantages to this configuration. Future SP operation may require composite end-to-end solutions requiring tighter control over the way the NFV data center operates as an NFV forms part of a larger solution. Therefore, assurance, control, and apportionment of resources may be better satisfied via actions taken at the NFVO and generic VNFM level rather than as depicted in Figure 10.4.

To support such cases, the VNFM-specific functionality needs to be integrated with the VNFM generic common functionality or, more broadly, integrated into the NFVO. How such integration is achieved is out of scope for NFV MANO but is a much relevant operational problem for any data center's NFV orchestration system.

Pragmatically, in the early stages of NFV, it is likely that many virtualized applications (VNFs) will be capable of running independent of the ETSI NFV system.

### 10.2.3.2 VNFM Packaged with VNF (Separate or Existing EMS/NMS)

This configuration (see Fig. 10.5) is much the same as the previous one, with the exception that the VNF and VNFM come as a single package and may integrate with an existing EMS and NMS. The advantages and disadvantages are much the same as stated previously.

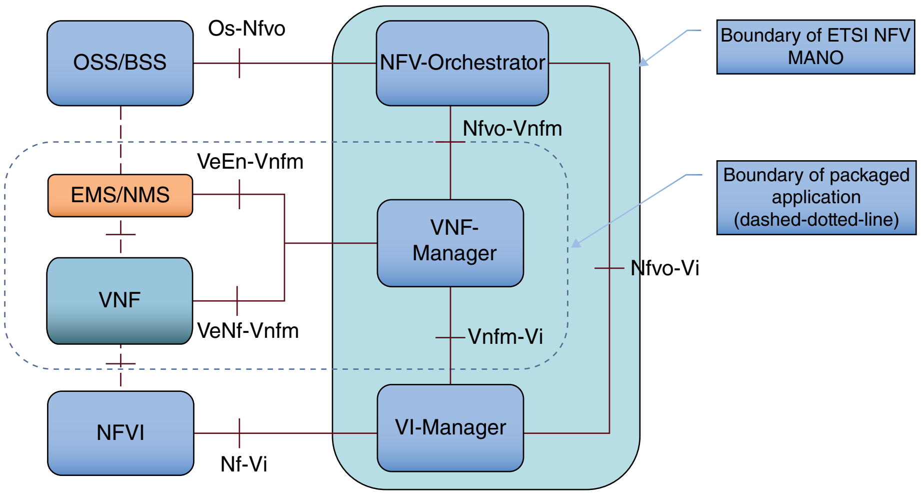

**Figure 10.4** NFV Management Architecture where the VNFM is supplied with the application and its EMS/NMS.

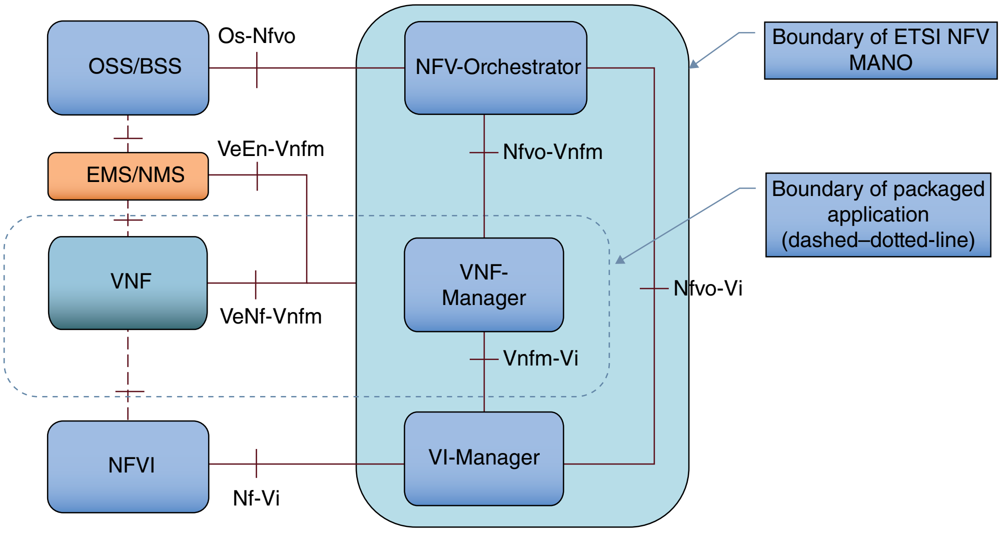

**Figure 10.5** NFV Management Architecture where the VNFM is supplied with application.

This configuration is the next step along the operational migration path. In this configuration, the assurance of the VNF may not necessarily be presented to the existing EMS/NMS system, and assurance information in the future may be reported back through the NFVO via the VNFM. This situation may occur in the future when composite end-to-end solutions may be composed out of the NFV Catalog.

#### 10.2.3.3 Generic VNFM EMS/NMS Packaged with VNF

In this configuration (see Fig. 10.6), the VNF comes packaged with its EMS and NMS but uses the services of the NFV MANO system's generic VNFM.

The future implication of this configuration is that the VNF now needs the NFV MANO system to operate. In this respect, the NFD contains sufficient information to parameterize the VNFM, and the VNF is capable of presenting the monitoring data in suitable format for consumption by the VNFM. The assurance is still performed by the VNF EMS and optionally via the NFV MANO system.

#### 10.2.3.4 Generic VNFM, Separate VNF Package, and Preexisting EMS/NMS

In this configuration (see Fig. 10.7), only VNF and its descriptor files are supplied. This is the near final evolution of the NFV system.

With this configuration, the VNF is predominantly driven by the NFV MANO system. In this regard, the VNF uses the services of the ETSI MANO system to provide a generic VNFM that is parameterized via the information contained in the NFV descriptors that are onboarded to the system (see later section for more information). The VNF has a monitoring agent within it (e.g., Ganglia [5]) and provides all the necessary monitoring data to the generic VNFM.

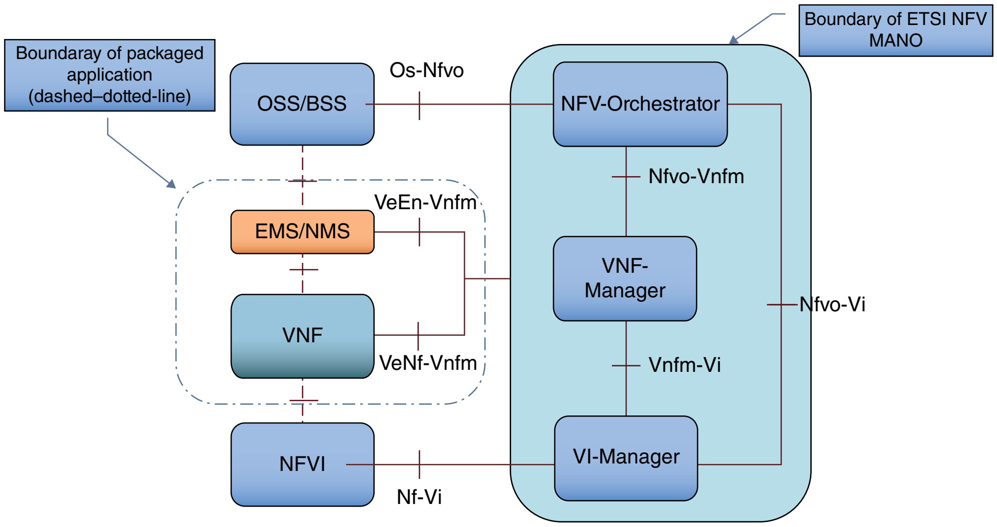

**Figure 10.6** NFV Management Architecture where the VNF is supplied with its EMS/NMS. The VNFM is generic to the ETSI NFV MANO architecture.

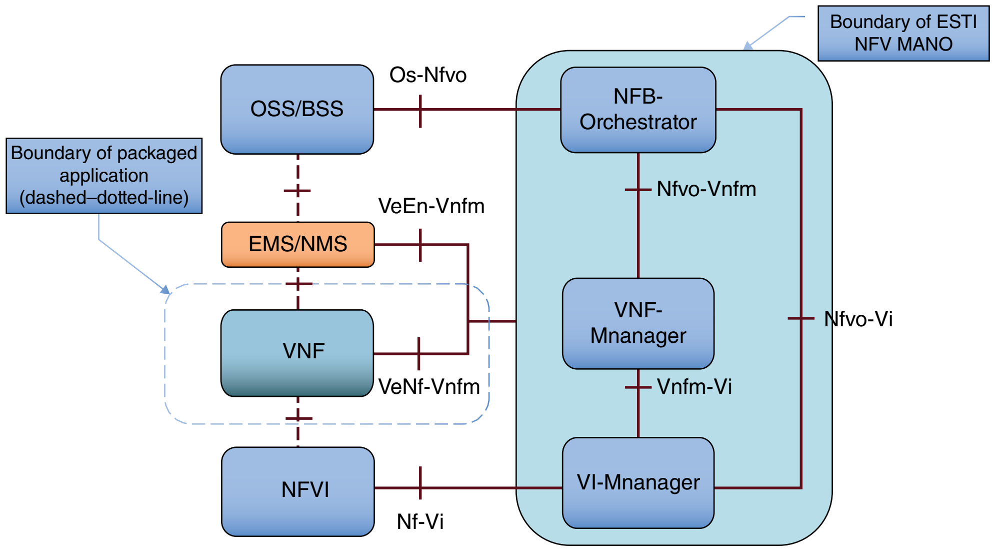

**Figure 10.7** NFV Management Architecture where the VNF is supplied as an executable.

The assurance of the VNF may also be provided by the NFV MANO system. In theory, the traditional OSS system can drive the NFV, but then if this continues to happen, there is little point to the NFV MANO system. In effect, in this final configuration, there is also an implied OSS transformation from an OSS that manages the network via direct stimulation of the NMS and EMS for given network functions to one where the network is managed through configuration scripts (automatically) driven by the NFV MANO system.

### *10.2.4 NFV Summary*

This section described the ETSI NFV orchestration system, its main components, method of operation, and the likely groupings of functions for deployment. As previously noted, ETSI NFV is still evolving and is not yet a normative standard. Nevertheless, the orchestration system described previously provides a framework to enable automatic operation of virtualized SP network functions and services. The story is completed when we provide an automation framework for the *underlying network* that is responsible for integrating these functions and services into a fully connected end-to-end system. SDN provides such a framework. Through logical separation of control and forwarding planes, SDN introduces a flexible programming environment for dynamic network control. The next section examine SDN from a number of perspectives that reflect the multiple ways that SDN can be used

## 10.3 SDN

SDN [6–8] is a framework for separating control of data networking from the actual act of forwarding packets. The original idea of SDN is that the separation of control and data planes allow optimization, evolution, and innovation to proceed independently for these two domains. With SDN, the control of network forwarding engines can be aggregated by common control nodes, and moreover, data plane functions are programmed with well-defined interfaces between the control nodes and data plane functions. SDN thus enables easy adoption of networking to specific (application or use case) needs.

While initial SDN solutions primarily focus on deployments of physical forwarding engines, it is a natural extension to apply these ideas to virtualized infrastructures, including those inside data centers and/or cloud infrastructures. By implementing data plane functionality with virtualized networking platforms, the data networking and virtual compute instances can be more tightly coupled and, if need be, allow the application some degree of control over the data networking itself.

We illustrate this concept in the following section by presenting, as examples, how to virtualize traditional mobile telecommunication networking functions ("virtual telco"). Generally, these applications may include a 3GPP mobile packet core ("Evolved Packet Core" (EPC)) [9–11], a 3GPP-based Gi-LAN service area [12], IMS and VoLTE [13], or other typical 3GPP-specific solutions. When hosted in data centers or through cloud infrastructures, these mobile telecommunication networking functions must continue to provide high-performance, low-latency, and dependable networking, similar to those described earlier in this chapter, to meet the availability requirements of such "telco" applications. Generally, individual virtual telco applications need to operate with availability requirements of 99.999% or better, have stringent requirements in terms of failover and fault recovery, and have stringent requirements for packet loss and packet delivery times. FCAPS lists a series of requirements specifically for such applications. In other words, these are critical network applications that require a high level of quality assurance.

When deploying "virtual telco" services in data centers, some of these applications require a "service chain" of functions, that is, a construct where one or more packet flows are steered through one or more virtualized or service functions that adapt the packet flows with service functionality as these packets traverse between ingress and egress. An example of this is a Gi-LAN service area that is comprised of, for example, a packet classification function, an

HTTP application function, a DPI application function, and an NA(P)T application function. To support such functions in data centers and/or cloud infrastructures, each of the application function's output (virtual) Ethernet needs to be connected to the next service function's input (virtual) Ethernet to form a service chain of services. This procedure is generally called service chaining [14]. Steering and classification principles can then be used to selectively assign packet flows to service chains of varying types.

Given that mobile solutions oftentimes produce relatively modest amounts of bandwidth (typically up to a maximum of a few hundred gigabits/second), managing all of the data flows for mobile service delivery is well suited for data center and/or cloud infrastructures. The prime benefits of hosting such functions in the data center and/or cloud infrastructures then involve the speed of service delivery and speed with which capacity can be provided. In the following text, we explore how such virtual service delivery can be provided for, specifically for a combined vEPC and Gi-LAN service area solution.

## 10.4 The Mobility Use Case

As an example of SDN applied to mobile networks, we explore the use case of a combined vEPC and Gi-LAN service gateway as depicted in Figure 10.8. In this example, the vEPC provides mobile subscriber line termination and charging functionality, while a Gi-LAN service gateway provides an infrastructure to host value-added services toward mobile service delivery. Here, we only describe the high-level functionality of a vEPC, and we refer to external material for more in-depth descriptions of the functions of an EPC [10, 11].

An EPC function is composed of a control and data plane. The control plane ("Mobility Management Entity" (MME)) addresses mobility, authentication, and paging operations, while the data plane is best characterized as a series of GPRS Tunneling Protocol (GTP) routers: the serving gateway (router) ("S-GW") providing for local mobility operations, while

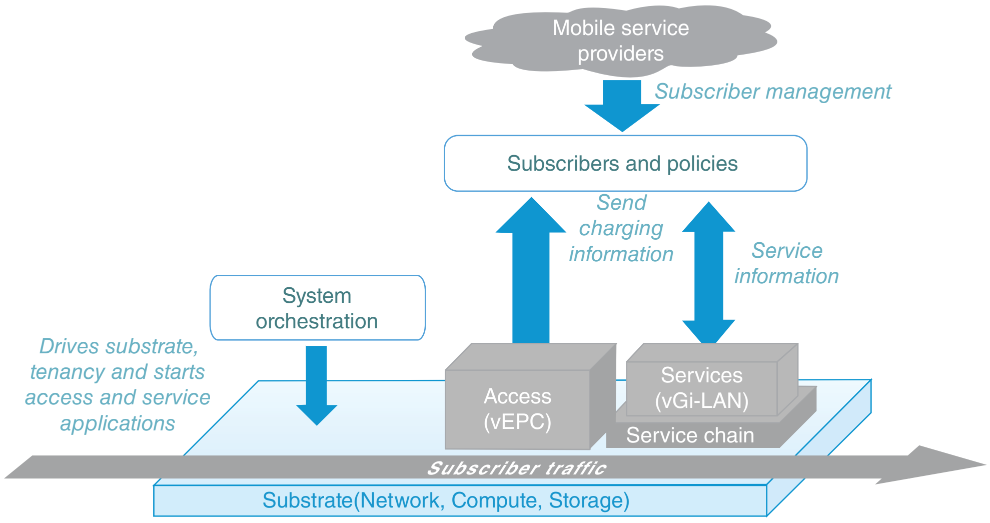

**Figure 10.8** High-level view of mobility use cases.

the Packet Data Network (PDN) Gateway ("P-GW") providing for GTP line termination, roaming, and charging functions. Typically, if a subscriber is "at home," the S-GW and P-GW are collapsed into a single entity and termed as "SAE-GW."

An S-GW connects to base stations (eNBs) southbound, while a P-GW connects to the Internet or Gi-LAN area northbound. Southbound connections from MME/S-GW are 3GPP specific (S1-U/S1-MME reference points) and carry data packets in an IP-in-GTP tunnel or SCTP [15] sessions between base station and S-GW and/or MME. Northbound traffic is routed by way of standard routing protocols and may involve MPLS, virtual local area network (VLANs), or other network connectivity. BGP is oftentimes used for signaling connectivity, and ECMP may be used for load distribution.

A typical EPC is internally composed of two main components—GTP-based load balancers and GTP session engines. The load balancers terminate external sessions, for example, the GTP session originated by the base stations and the PDN sessions toward the Internet, while the session engines address the actual 3GPP-specific functions. Communication between the EPC load balancers and session engines is oftentimes a proprietary function and uses 3GPP session state for appropriate distribution of load across various servers. The reason for such 3GPP-specific solutions is primarily legacy—when 3GPP systems were first designed, "SDN" as a concept simply did not exist, virtual operation of said systems was not considered or deemed inappropriate, and application-specific "SDN"-like functionality was devised for load distribution of 3GPP calls in fixed network elements.

While "SDN" techniques may be applicable to operating EPCs in the cloud, immediately transforming today's EPCs to be more "data center- and/or cloud-like" takes time and investment. A fine balance needs to be found between efficiently using existing EPC assets while transitioning to new "cloud-like" environments. One approach to this transformation is to host the 3GPP load balancers and session functions as VMs in a data center while leveraging existing 3GPP assets to implement the business end of the service. Over time, these legacy solutions can be replaced by virtualized implementations. In this sense, the networking data path between GTP load balancer and GTP session function can be considered a service chain in its own right, and in the future, releases of (virtual) packet core SDN-provided service chains may prove useful.

When virtualizing EPCs, EPC load balancers may announce their service availability by way of announcing the loop-back address of the VM through BGP. If there are multiple parallel load balancers, it goes that ECMP techniques may be used to route packet to those load balancers and ECMP thus becomes an effective load balancer in the data center. It goes that ECMP may be implemented on standard routing solutions or can be implemented in a (distributed) virtual form in the data centers.

The Gi-LAN service area hosts one or more services. The area can host multiple service functions on which it applies "3GPP" policies. This "3GPP" policy infrastructure is based on the Policy and Charging Rules Function (PCRF) [16] and is augmented to carry additional information toward the Gi-LAN area to help it make "service chain" steering decisions. These steering decisions are made combining a first-sign-of-life packet with the PCRF subscriber record. Based on the combination of the two, a "3GPP classifier" decides which service chain is carrying the data packets for that specific flow.

The Gi-LAN service area needs to cater for bandwidths that are in excess of today's top of rack to host Ethernet link. To address this networking bottleneck, the gateway may use multiple classifiers operating in parallel to spread packets across multiple classifiers. Signaling via

BGP may provide upstream nodes the necessary information on how to distribute load across the classifiers, for example, by way of ECMP [17]. When BGP is used to announce service availability, loop-back addresses signaled are equivalent to the IP addresses of the VMs implementing the classification function.

The business end of the Gi-LAN service gateway can be defined by static service chains that carry packet flows between the P-GW function and the Internet. In this configuration, multiple logical service chains can share services, services can off-load flows to the fast path, services may participate in further classification of the services, and packet flows can be completely agnostic; these are being used as part of a Gi-LAN service area. A thorough description of such service chaining can be found separately [14].

## 10.5 Virtual Networking in Data Centers

In today's data centers, all hosts are connected to an Ethernet-based local area network (LAN) that supports VLAN tagging. In a VLAN, every Ethernet frame carries a VLAN tag that identifies tenancy. A VLAN in a data center operates as a standard Ethernet-based layer 2 network, with all associated helper functions such as the Address Resolution Protocol (ARP) [18], potentially augmented with the Dynamic Host Configuration Protocol (DHCP) [19] combined with all other IP- and Ethernet-based tools to operate such a virtualized or isolated layer 2 network.

In IaaS, the notion of a "tenant" exists. A tenant is defined as a project, group, enterprise, or any other grouping for which joint operations exist. Groups of hosts can be assigned to a tenant, and IaaS uses tenant information to allocate VMs to hosts, and networking constructs such as VLANs can be assigned specifically to tenants.

Allocation of VLANs to tenants is a function defined within the broad context of "system orchestration." Such system orchestration tool needs to maintain a VLAN allocation table, and it needs the ability to manage the various hardware-based routers and switches to connect the VLANs to the appropriate hosts. Various modes for such orchestration exist, ranging from manual configuration to fully automated orchestration. Management of such VLAN ranges enables a data center operator to segregate traffic between the various VLAN domains and thus tenants.

Many data centers are managed by way of an IaaS/Openstack [20] orchestration system. The role of Openstack is to start, manage, and stop VMs on Openstack-managed "host" computers and to establish a host networking infrastructure to connect tenant VMs to the appropriate VLANs. Moreover, the host networking infrastructure (Quantum/Neutron) for Openstack based on Open vSwitch (OVS) is elaborate and feature rich: through a series of software bridges and IP rule tables, packets are steered from a host-resident Ethernet controller card by way of the VLAN tagging to the appropriate tenant VM.

While most virtual telco applications can be considered single tenant, it is not uncommon that control over such telecommunication's applications and the "data plane" for such applications are separated in multiple segregated domains, that is, the "virtual telco" application as a whole becomes multitenant. Inherently, the control and data portion of such applications needs to communicate, and thus, bridging capabilities are needed between various tenants. This bridging may potentially involve IP address translation and firewalling and may include functions to detect intrusions. Given that all assets in a data center are to be virtualized, all these tenant bridging functions need to operate on data center resources themselves.

## 10.6 Summary

NFV promises the SP greater operational flexibility in the future leading to improved OPEX and CAPEX. The chapter explored two tools that are used by NFV to help achieve these aims, namely, orchestration and SDN. As this chapter has indicated, while the technology used for both SDN and orchestration is still maturing, the standardization of NFV is not quite at the same level of maturity. The implication is that NFV is likely to continue to evolve as the technology and the application models are refined further.

## References

[1] ETSI, "GS NFV MAN 001 V0.3.3 (2014–02) Network Function Virtualization (NFV) Management and Orchestration," ETSI, Sophia-Antipolis Cedex, March 2014.
[2] OASIS, "Topology and Orchestration Specification for Cloud Applications Version 1.0," OASIS Standard, 25 November 2013 [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html (accessed January 19, 2015).
[3] M. Bjorklund, "RFC 6020—YANG-A Data Modeling Language for the Network Configuration Protocol," October 2010 [Online]. Available: http:/tools.ietf.org/html/rfc6020 (accessed January 19, 2015).
[4] tmforum.org, "tmforum Information Framework (SID)," 2014 [Online]. Available: http://www.tmforum.org/DownloadRelease14/16168/home.html (accessed January 19, 2015).
[5] sourceforge, "Ganglia Monitoring System," 2014 [Online]. Available: http://ganglia.sourceforge.net (accessed January 19, 2015).
[6] T. Lakshman, N. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "The SoftRouter Architecture," in *HotNets-III*, San Diego, CA, 2004.
[7] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks," April 13, 2013 [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf (accessed February 18, 2015).
[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, p. 6, 2008.
[9] 3rd Generation Partnership Project, "The Evolved Packet Core," 2014 [Online]. Available: http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core (accessed January 19, 2015).
[10] 3rd Generation Partnership Project, "General Packet Radio Service (GPRS) Enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Access, 3GPP TS23.401, v12.4.0," March 2014 [Online]. Available: http://www.3gpp.org/DynaReport/23401.htm (accessed January 19, 2015).
[11] 3rd Generation Partnership Project, "Architecture Enhancements for Non-3GPP Accesses, 3GPP TS23.402, v12.4.0," March 2014 [Online]. Available: http://www.3gpp.org/DynaReport/23402.htm (accessed January 19, 2015).
[12] H. La Roche and P. Suthar, "GiLAN and Service Chaining," Cisco Live, May 14, 2014. [Online]. Available: https://www.ciscolive2014.com/connect/sessionDetail.ww?SESSION_ID=3138 (accessed January 19, 2015).
[13] 3rd Generation Partnership Project, "IP Multimedia Subsystem (IMS); Stage 2, 3GPP TS 23.228," 24 June 2014 [Online]. Available: http://www.3gpp.org/DynaReport/23228.htm (accessed January 19, 2015).
[14] W. Haeffner, J. Napper, N. Stiemerling, D. Lopez and J. Uttaro, "IETF draft—Service Function Chaining Use Cases in Mobile Networks," July 4, 2014 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sfc-use-case-mobility/ (accessed January 19, 2015).
[15] R. Stewart, "RFC 4960—Stream Control Transmission Protocol," September 2007 [Online]. Available: http://tools.ietf.org/html/rfc4960 (accessed January 19, 2015).
[16] 3rd Generation Partnership Project, "Policy and Charging Control Architecture, 3GPP TS23.203, v12.4.0," March 2014 [Online]. Available: http://www.3gpp.org/DynaReport/23203.htm (accessed January 19, 2015).
[17] C. Hopps, "RFC 2992—Analysis of an Equal-Cost Multi-Path Algorithm," November 2000 [Online]. Available: http://tools.ietf.org/html/rfc2992 (accessed January 19, 2015).

[18] D. Plummer, "RFC 826—Ethernet Address Resolution Protocol," November 1982 [Online]. Available: http://tools.ietf.org/html/rfc826 (accessed January 19, 2015).
[19] R. Droms, "RFC 2131—Dynamic Host Configuration Protocol," March 1997 [Online]. Available: http://tools.ietf.org/html/rfc2131 (accessed January 19, 2015).
[20] Openstack Foundation, "Openstack Cloud Software," [Online]. Available: http://www.openstack.org (accessed January 19, 2015).