Test of Network Virtualization

Cap 3 —- SDN Background and Motivation

1. **Why traditional network architectures are inadequate for modern networking needs?**

   Even with the greater capacity of transmission schemes and the greater performance of network devices, traditional network architectures are increasingly inadequate in the face of the growing complexity, variability, and high volume of the imposed load. In addition, as quality of service (QoS) and quality of experience (QoE) requirements imposed on the network are expanded as a result of the variety of applications, the traffic load must be handled in an increasingly sophisticated and agile fashion.

2. **What are the standards organizations involved on normalization of SDN and NFV?**

   The Internet Society (IETF and IRTF), ITU-T, and ETSI are all making key contributions to the standardization of SDN and NFV.

3. **Why networks requirements are evolving?**

   A number of trends are driving network providers and users to reevaluate traditional approaches to network architecture. These trends can be grouped under the categories of demand, supply, and traffic patterns.

   Demand Is Increasing
   A number of trends are increasing the load on enterprise networks, the Internet, and other internets. Of particular note are the following:
   - Cloud computing: There has been a dramatic shift by enterprises to both public and private cloud services.
   - Big data: The processing of huge data sets requires massive parallel processing on thousands of servers, all of which require a degree of interconnection to each other. Therefore, there is a large and constantly growing demand for network capacity within the data canter.
   - Mobile traffic: Employees are increasingly accessing enterprise network resources via mobile personal devices, such as smartphones, tablets, and notebooks. These devices support sophisticated apps that can consume and generate image and video traffic, placing new burdens on the enterprise network.
   - The Internet of Things (IoT): Most “things” in the IoT generate modest traffic, although there are exceptions, such as surveillance video cameras. But the sheer number of such devices for some enterprises results in a significant load on the enterprise network.

   Supply Is Increasing
   As the demand on networks is rising, so is the capacity of network technologies to absorb rising loads. In terms of transmission technology established that the key enterprise wired and wireless network technologies, Ethernet and Wi-Fi respectively, are well into the gigabits per second (Gbps) range. Similarly, 4G and 5G cellular networks provide greater capacity for mobile devices from

remote employees who access the enterprise network via cellular networks rather than Wi-Fi.
The increase in the capacity of the network transmission technologies has been matched by an increase in the performance of network devices, such as LAN switches, routers, firewalls, intrusion detection system/intrusion prevention systems (IDS/IPS), and network monitoring and management systems. Year by year, these devices have larger, faster memories, enabling greater buffer capacity and faster buffer access, as well as faster processor speeds.

Traffic Patterns Are More Complex
If it were simply a matter of supply and demand, it would appear that today's networks should be able to cope with today's data traffic. But as traffic patterns have changed and become more complex, traditional enterprise network architectures are increasingly ill suited to the demand.
Until recently, and still common today, the typical enterprise network architecture consisted of a local or campus-wide tree structure of Ethernet switches with routers connecting large Ethernet LANs and connecting to the Internet and WAN facilities. This architecture is well suited to the client/server computing model that was at one time dominant in the enterprise environment. With this model, interaction, and therefore traffic, was mostly between one client and one server. In such an environment, networks could be laid out and configured with relatively static client and server locations and relatively predictable traffic volumes between clients and servers.

4. Why traditional networks are inadequate? Which are the main limitations?

- Static, complex architecture: To respond for demands such as differing levels of QoS, high and fluctuating traffic volumes, and security requirements, networking technology has grown more complex and difficult to manage. This has resulted in a number of independently defined protocols each of which addresses a portion of networking requirements. An example of the difficulty this presents is when devices are added or moved. The network management staff must use device-level management tools to make changes to configuration parameters in multiple switches, routers, firewalls, web authentication portals, and so on. The updates include changes to access control lists (ACLs), virtual LAN settings, QoS settings in numerous devices, and other protocol-related adjustments. Another example is the adjustment of QoS parameters to meet changing user requirements and traffic patterns. Manual procedures must be used to configure each vendor's equipment on a per-application and even per-session basis.
- Inconsistent policies: To implement a network-wide security policy, staff may have to make configuration changes to thousands of devices and mechanisms. In a large network, when a new virtual machine is activated, it can take hours or even days to reconfigure ACLs across the entire network.
- Inability to scale: Demands on networks are growing rapidly, both in volume and variety. Adding more switches and transmission capacity, involving multiple vendor equipment, is difficult because of the complex, static nature of the network. One strategy enterprises have used is to oversubscribe network links based on predicted traffic patterns. But with the increased use of virtualization and the increasing variety of multimedia applications, traffic patterns are unpredictable.

- Vendor dependence: Given the nature of today's traffic demands on networks, enterprises and carriers need to deploy new capabilities andservices rapidly in response to changing business needs and user demands. A lack of open interfaces for network functions leaves the enterprises limited by the relatively slow product cycles of vendor equipment.

5. Which are the main requirements for SDN approach?

- Adaptability: Networks must adjust and respond dynamically, based on application needs, business policy, and network conditions.
- Automation: Policy changes must be automatically propagated so that manual work and errors can be reduced.
- Maintainability. Introduction of new features and capabilities (software upgrades, patches) must be seamless with minimal disruption of operations.
- Model management: Network management software must allow management of the network at a model level, rather than implementing conceptual changes by reconfiguring individual network elements.
- Mobility: Control functionality must accommodate mobility, including mobile user devices and virtual servers.
- Integrated security: Network applications must integrate seamless security as a core service instead of as an add-on solution.
- On-demand scaling: Implementations must have the ability to scale up or scale down the network and its services to support on-demand requests.

6. Which are the modern approach for computing and networking?

Today, the computing environment is characterized by extreme openness and great customer flexibility. The bulk of computing hardware consists of x86 and x86-compatible processors for standalone systems and ARM processors for embedded systems. This makes it easy to port operating systems implemented in C, C++, Java, and the like. Even proprietary hardware architectures, such as IBM's zEnterprise line, provide standardized compilers and programming environments and so can easily run open sources operating systems such as Linux. Therefore, applications written for Linux or other open operating systems can easily be moved from one vendor platform to another. Even proprietary systems such as Windows and Mac OS provide programming environments to make porting of applications an easy matter. It also enables the development of virtual machines that can be moved from one server to another across hardware platforms and operating systems.

The networking environment today faces some of the same limitations faced in the pre-open era of computing. Here the issue is not developing applications that can run on multiple platforms. Rather, the difficulty is the lack of integration between applications and network infrastructure. As demonstrated in the preceding section, traditional network architectures are inadequate to meet the demands of the growing volume and variety of traffic.

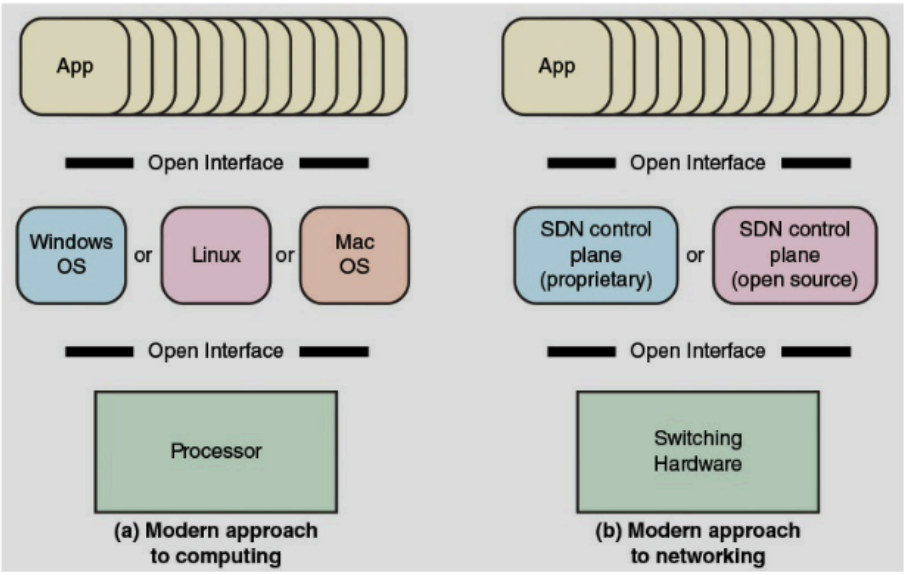

FIGURE 3.1 The Modern Approach to Computing and Networking

7. Which are the main characteristics of Software Defined Networks?

- The control plane is separated from the data plane. Data plane devicesbecome simple packet-forwarding devices.
- The control plane is implemented in a centralized controller or set of coordinated centralized controllers. The SDN controller has a centralized view of the network or networks under its control. The controller is portable software that can run on commodity servers and is capable of programming the forwarding devices based on a centralized view of the network.
- Open interfaces are defined between the devices in the control plane (controllers) and those in the data plane.
- The network is programmable by applications running on top of the SDN controllers. The SDN controllers present an abstract view of network resources to the applications.

8. How is implemented the control plane?

The control plane provides the “intelligence” in designing routes, setting priority and routing policy parameters to meet QoS and QoE requirements and to cope with the shifting traffic patterns.

The control plane is implemented in a centralized controller or set of coordinated centralized controllers. The SDN controller has a centralized view of the network or networks under its control. The controller is portable software that can run on commodity servers and is capable of programming the forwarding devices based on a centralized view of the network.

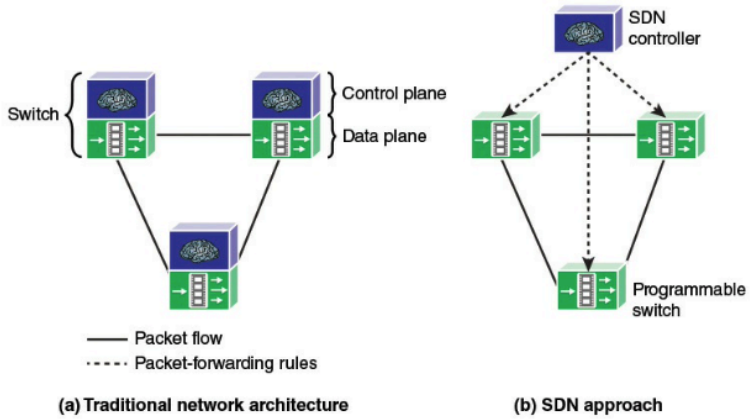

FIGURE 3.2 Control and Data Planes

9. Unlike WiFi, there is no single but several organizations for creating Standards for SDN / NFV. Give some examples.

Organization	Mission	SDN- and NFV-Related Effort
Open Networking Foundation (ONF)	An industry consortium dedicated to the promotion and adoption of SDN through open standard development.	OpenFlow
Internet Engineering Task Force (IETF)	The Internet's technical standards body. Produces RFCs and Internet standards.	Interface to routing systems (I2RS) Service function chaining
European Telecommunications Standards Institute (ETSI)	An EU-sponsored standards organization that produces globally applicable standards for information and communications technologies.	NFV architecture
OpenDaylight	A collaborative project under the auspices of the Linux Foundation.	OpenDaylight
International Telecommunication Union—Telecommunication Standardization Sector (ITU-T)	United Nations agency that produces Recommendations with a view to standardizing telecommunications on a worldwide basis.	SDN functional requirements and architecture
Internet Research Task Force (IRTF) Software Defined Networking Research Group (SDNRG)	Research group within IRTF. Produces SDN-related RFCs.	SDN architecture
Broadband Forum (BBF)	Industry consortium developing broadband packet networking specifications.	Requirements and framework for SDN in telecommunications broadband networks
Metro Ethernet Forum (MEF)	Industry consortium that promotes the use of Ethernet for metropolitan and wide-area applications.	Defining APIs for service orchestration over SDN and NFV
IEEE 802	An IEEE committee responsible for developing standards for LANs.	Standardize SDN capabilities on access networks.
Optical Internetworking Forum (OIF)	Industry consortium promoting development and deployment of interoperable networking solutions and services for optical networking products.	Requirements on transport networks in SDN architectures
Open Data Center Alliance (ODCA)	Consortium of leading IT organizations developing interoperable solutions and services for cloud computing.	SDN usage model
Alliance for Telecommunications Industry Solutions (ATIS)	A standards organization that develops standards for the unified communications (UC) industry.	Operational opportunities and challenges of SDN/NFV programmable infrastructure
Open Platform for NFV (OPNFV)	An open source project focused on accelerating the evolution of NFV.	NFV infrastructure

TABLE 3.1 SDN and NFV Open Standards Activities

10. To speed up the existence of software and accelerate its use, there are also several open software development initiatives for NFV / SDN. Comment and present some examples.

OpenDaylight

OpenDaylight is an open source software activity under the auspices of the Linux foundation. Its member companies provide resources to develop an SDN

controller for a wide range of applications. Although the core membership consists of companies, individual developers and users can also participate, so OpenDaylight is more in the nature of an open development initiative than a consortium. ODL also supports network programmability via southbound protocols, a bunch of programmable network services, a collection of northbound APIs, and a set of applications.
OpenDaylight is composed of about 30 projects, and releases their outputs in simultaneous manner. After its first release, Hydrogen, in February 2014, it successfully delivered the second one, Helium, at the end of September 2014.

Open Platform for NFV
Open Platform for NFV is an open source project dedicated to acceleration the adoption of standardized NFV elements. OPNFV will establish a carrier- grade, integrated, open source reference platform that industry peers will build together to advance the evolution of NFV and to ensure consistency, performance, and interoperability among multiple open source components. Because multiple open source NFV building blocks already exist, OPNFV will work with upstream projects to coordinate continuous integration and testing while filling development gaps.

OpenStack
OpenStack is an open source software project that aims to produce an open source cloud operating system. It provides multitenant Infrastructure as a Service (IaaS), and aims to meets the needs of public and private clouds regardless of size, by being simple to implement and massively scalable. SDN technology is expected to contribute to its networking part, and to make the cloud operating system more efficient, flexible, and reliable.
OpenStack is composed of a number of projects. One of them, Neutron, is dedicated for networking. It provides Network as a Service (NaaS) to other OpenStack services. Almost all SDN controllers have provided plug-ins for Neutron, and through them services on OpenStack and other OpenStack services can build rich networking topologies and can configure advanced network policies in the cloud.

Cap 4 —- SDN Data Plane

1. What is the fundamental difference between SDN data layer equipment and conventional network switches?

   The important characteristic of the network devices in an SDN network is that these devices perform a simple forwarding function, without embedded software to make autonomous decisions.

   The data plane consists of physical switches and virtual switches. In both cases, the switches are responsible for forwarding packets. The internal implementation of buffers, priority parameters, and other data structures related to forwarding can be vendor dependent. However, each switch must implement a model, or abstraction, of packet forwarding that is uniform and open to the SDN controllers.

2. **The principal functions of the network device are the following:**

   Control support function: Interacts with the SDN control layer to support programmability via resource-control interfaces. The switch communicates with the controller and the controller manages the switch via the OpenFlow switch protocol.

   Data forwarding function: Accepts incoming data flows from other network devices and end systems and forwards them along the data forwarding paths that have been computed and established according to the rules defined by the SDN applications.

3. **The OpenFlow protocol fulfills two requirements that are indispensable to make the SDN concept a practical implementation. Describe them briefly.**

   - There must be a common logical architecture in all switches, routers, and other network devices to be managed by an SDN controller. This logical architecture may be implemented in different ways on different vendor equipment and in different types of network devices, as long as the SDN controller sees a uniform logical switch functionality.
   - A standard, secure protocol is needed between the SDN controller and the network device.

4. **What are the main components of an OpenFlow switch? It communicates with the SDN controller using the OpenFlow protocol supported by TLS over TCP / IP. Communicates with the other switches using TCP / IP. It has flow tables and group tables. Try to give a more detailed description.**

   An SDN controller communicates with OpenFlow-compatible switches using the OpenFlow protocol running over Transport Layer Security (TLS). Each switch connects to other OpenFlow switches and, possibly, to end-user devices that are the sources and destinations of packet flows. On the switch side, the interface is known as an OpenFlow channel. These connections are via OpenFlow ports. An OpenFlow port also connects the switch to the SDN controller. OpenFlow defines three types of ports:

   - Physical port: Corresponds to a hardware interface of the switch. For example, on an Ethernet switch, physical ports map one to one to the Ethernet interfaces.
   - Logical port: Does not correspond directly to a hardware interface of the switch. Logical ports are higher-level abstractions that may be defined in the switch using non-OpenFlow methods (for example, link aggregation groups, tunnels, loopback interfaces). Logical ports may include packet encapsulation and may map to various physical ports. The processing done by the logical port is implementation dependent and must be transparent to OpenFlow processing, and those ports must interact with OpenFlow processing like OpenFlow physical ports.

   - Reserved port: Defined by the OpenFlow specification. It specifies generic forwarding actions such as sending to and receiving from the controller,

flooding, or forwarding using non-OpenFlow methods, such as “normal” switch processing.

Within each switch, a series of tables is used to manage the flows of packets through the switch.
The OpenFlow specification defines three types of tables in the logical switch architecture. A flow table matches incoming packets to a particular flow and specifies what functions are to be performed on the packets. There may be multiple flow tables that operate in a pipeline fashion, as explained subsequently. A flow table may direct a flow to a group table, which may trigger a variety of actions that affect one or more flows. A meter table can trigger a variety of performance-related actions on a flow. Using the OpenFlow switch protocol, the controller can add, update, and delete flow entries in tables, both reactively (in response to packets) and proactively.

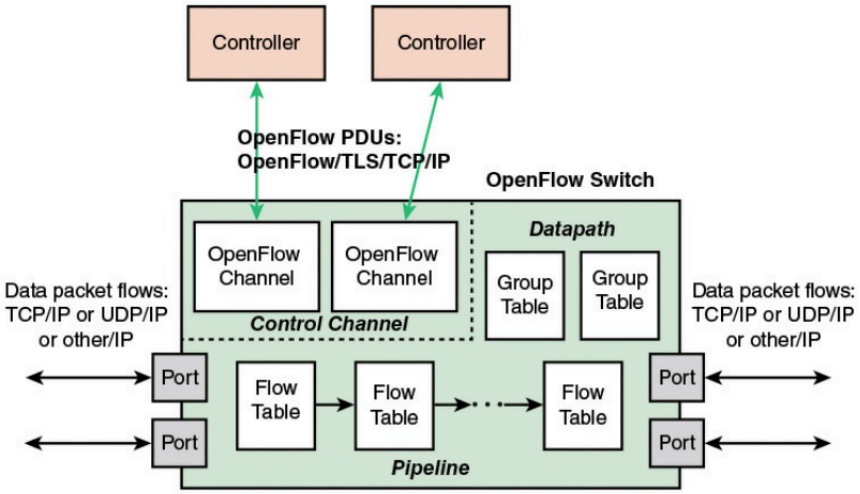

FIGURE 4.4 OpenFlow Switch

5. **What are fields like macth, priority, counters, instructions, timeout, cookie, flags for? What information is there in the match?**

Match: Used to select packets that match yhe values in the fields.

Priority: Relative priority of table entries. This is a 16-bit field with 0 corresponding to the lowest priority. In principle, there could be 216 = 64k priority levels.

Counters: Updated for matching packets. The OpenFlow specification defines a variety of counters.

Intructions: Instructions to be performed if a match occurs.

Timeout: Maximum amount of idle time before a flow is expired by the switch. Each flow entry has an idle_timeout and a hard_timeout associated with it. A nonzero hard_timeout field causes the flow entry to be removed after the given number of seconds, regardless of how many packets it has matched. A nonzero idle_timeout field causes the flow entry to be removed when it has matched no packets in the given number of seconds.

Cookie: 64-bitopaquedatavaluechosenbythecontroller.Maybeusedby the controller to filter flow statistics, flow modification and flow deletion; not used when processing packets.

Flags: Flags alter the way flow entries are managed; for example, the flag OFPFF_SEND_FLOW_REM triggers flow removed messages for that flow entry.

The match fields component of a table entry consists of the following required fields:

- Ingress port: The identifier of the port on this switch on which the packet arrived. This may be a physical port or a switch-defined virtual port. Required in ingress tables.
- Egress port: The identifier of the egress port from action set. Required in egress tables.
- Ethernet source and destination addresses: Each entry can be an exact address, a bitmasked value for which only some of the address bits are checked, or a wildcard value (match any value).
- Ethernet type field: Indicates type of the Ethernet packet payload.
- IP: Version 4 or 6.
- IPv4 or IPv6 source address, and destination address: Each entry can be an exact address, a bitmasked value, a subnet mask value, or a wildcard value.
- TCP source and destination ports: Exact match or wildcard value.
- UDP source and destination ports: Exact match or wildcard value.

6. **What counters are there, and what statistics are of interest to obtain?**

Counter	Usage	Bit Length
Reference count (active entries)	Per flow table	32
Duration (seconds)	Per flow entry	32
Received packets	Per port	64
Transmitted packets	Per port	64
Duration (seconds)	Per port	32
Transmit packets	Per queue	64
Duration (seconds)	Per queue	32
Duration (seconds)	Per group	32
Duration (seconds)	Per meter	32

- Cookie: 64-bit opaque data value chosen by the controller. May be used by the controller to filter flow statistics, flow modification and flow deletion; not used when processing packets.

7. **What kind of structuring is done in the flow tables, namely in Pipelines, Ingress, Egress?**

A switch includes one or more flow tables. If there is more than one flow table, they are organized as a pipeline, with the tables labeled with increasing numbers starting with zero. The use of multiple tables in a pipeline, rather than a single flow table, provides the SDN controller with considerable flexibility. The OpenFlow specification defines two stages of processing:

- Ingress processing: Ingress processing always happens, beginning with Table 0, and uses the identity of the input port. Table 0 may be the only table, in which case the ingress processing is simplified to the processing performed on that single table, and there is no egress processing.

- Egress processing: Egress processing is the processing that happens after the determination of the output port. It happens in the context of the output port. This stage is optional. If it occurs, it may involve one or more tables. The separation of the two stages is indicated by the numerical identifier of the first egress table. All tables with a number lower than the first egress table must be used as ingress tables, and no table with a number higher than or equal to the first egress table can be used as an ingress table.

8. **What is the idea behind nested flows?**

Nesting of flows, or put another way, the breaking down of a single flow into a number of parallel subflows.

An entry in Table 0 defines a flow consisting of packets traversing the network from a specific source IP address to a specific destination IP address. Once a least-cost route between these two endpoints is established, it might make sense for all traffic between these two endpoints to follow that route, and the next hop on that route from this switch can be entered in Table 0. In Table 1, separate entries for this flow can be defined for different transport layer protocols, such as TCP and UDP. For these subflows, the same output port might be retained so that the subflows all follow the same route. However, TCP includes elaborate congestion control mechanisms not normally found with UDP, so it might be reasonable to handle the TCP and UDP subflows differently in terms of quality of service (QoS)-related parameters. Any of the Table 1 entries could immediately route its respective subflow to the output port, but some or all of the entries may invoke Table 2, further dividing each subflow.

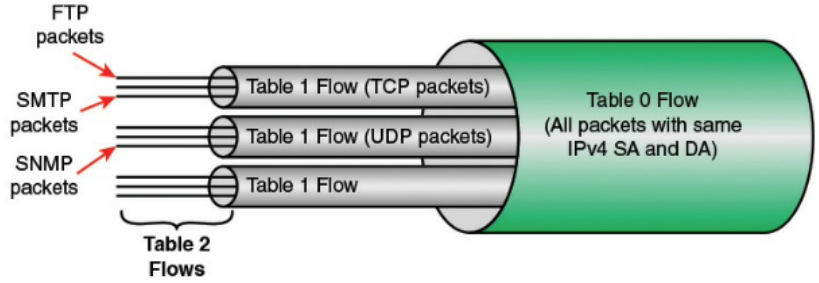

FIGURE 4.9 Example of Nested Flows

9. **What is the meaning of group table, group type, counters and action buckets?**

Group table: group table and group actions enable OpenFlow to represent a set of ports as a single entity for forwarding packets. Different types of groups are provided to represent different forwarding abstractions, such as multicasting and broadcasting.

Each group table consists of a number of rows, called group entries, consisting of four components: group identifier, group type, counters and action buckets.

Group type: to determinate group semantics.

Counters: Updated when packets are processed by a group.

Action buckets: An ordered list of action buckets, where each action bucket contains a set of actions to execute and associated parameters.

10. **What type of messages can occur between the controller and the switch?**

Message	Description
Controller-to-Switch	
Features	Request the capabilities of a switch. Switch responds with a features reply that specifies its capabilities.
Configuration	Set and query configuration parameters. Switch responds with parameter settings
Modify-State	Add, delete, and modify flow/group entries and set switch port properties.
Read-State	Collect information from switch, such as current configuration, statistics, and capabilities.
Packet-out	Direct packet to a specified port on the switch.
Barrier	Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.
Role-Request	Set or query role of the OpenFlow channel. Useful when switch connects to multiple controllers.
Asynchronous-Configuration	Set filter on asynchronous messages or query that filter. Useful when switch connects to multiple controllers.

11. Defines the following acronyms and definitions:

Action bucket: An ordered list of action buckets, where each action bucket contains a set of actions to execute and associated parameters.

Action list: the action list in a bucket is executed when a packet reaches a bucket. The action list is executed in sequence and generally ends with the Output action, which forwards the packet to a specified port. The action list may also end with the Group action, which sends the packet to another group. This enables the chaining of groups for more complex processing.

Action set: list of actions associated with a packet that are accumulated while the packet is processed by each table and that are executed when the packet exits the processing pipeline.

Egress table: **Egress processing:** Egress processing is the processing that happens after the determination of the output port. It happens in the context of the output port. This stage is optional. If it occurs, it may involve one or more tables. The separation of the two stages is indicated by the numerical identifier of the first egress table. All tables with a number lower than the first egress table must be used as ingress tables, and no table with a number higher than or equal to the first egress table can be used as an ingress table.

Flow: A sequence of packets between a source and destination that are recognized by the network as related and are treated in a uniform fashion.

Flow table: matches incoming packets to a particular flow and specifies what functions are to be performed on the packets. There may be multiple flow tables that operate in a pipeline fashion.

Group table: The group table and group actions enable OpenFlow to represent a set of ports as a single entity for forwarding packets. Different types of groups are provided to represent different forwarding abstractions, such as multicasting and broadcasting. Consists of a number of rows, called group entries, consisting of four components: group identifier; group type; counters and action buckets.

Ingress table: Ingress processing: Ingress processing always happens, beginning with Table 0, and uses the identity of the input port. Table 0 may be the only table, in which case the ingress processing is simplified to the processing performed on that single table, and there is no egress processing.

Match fields: Used to select packets that match the values in the fields.

OpenFlow action: **Actions** are used in **OpenFlow** flows to describe what to do when the flow matches a packet, and in a few other places in **OpenFlow**. Each version of the **OpenFlow** specification defines standard **actions**, and beyond that many **OpenFlow** switches, including Open vSwitch, implement extensions to the standard.

OpenFlow instruction: The instructions component of a table entry consists of a set of instructions that are executed if the packet matches the entry.

> The types of instructions can be grouped into four categories:
>
> **Direct packet through pipeline:** The Goto-Table instruction directs the packet to a table farther along in the pipeline. The Meter instruction directs the packet to a specified meter.
>
> **Perform action on packet:** Actions may be performed on the packet when it is matched to a table entry. The Apply-Actions instruction applies the specified actions immediately, without any change to the

action set associated with this packet. This instruction may be used to modify the packet between two tables in the pipeline.

Update action set: The Write-Actions instruction merges specified actions into the current action set for this packet. The Clear-Actions instruction clears all the actions in the action set.

Update metadata: A metadata value can be associated with a packet. It is used to carry information from one table to the next. The Write-Metadata instruction updates an existing metadata value or creates a new value.

OpenFlow message: **OpenFlow messages** are sent and received between the controller and the datapaths (**OpenFlow** instances or devices) it manages. These **messages** are byte streams, the structure of which is documented in the **OpenFlow** Protocol Specification documents published by the Open Networking Foundation (ONF).

OpenFlow port: **OpenFlow ports** are the network interfaces for passing packets between **OpenFlow** processing and the rest of the network. **OpenFlow** switches connect logically to each other via their **OpenFlow ports**. An **OpenFlow** switch makes a number of **OpenFlow ports** available for **OpenFlow** processing.

OpenFlow switch: **OpenFlow switch** is a network **switch** based on the **OpenFlow** protocol that employs software-**defined** network (**SDN**) techniques to forward packets in a network.

SDN data plane: The SDN data plane, referred to as the resource layer in ITU-T Y.3300 and also often referred to as the infrastructure layer, is where network forwarding devices perform the transport and processing of data according to decisions made by the SDN control plane. The important characteristic of the network devices in an SDN network is that these devices perform a simple forwarding function, without embedded software to make autonomous decisions.

Cap 5 —— SDN Control Plane

1. List and explain the main functions of the SDN control plane.

- Shortest path forwarding: Uses routing information collected from switches to establish preferred routes.
- Notification manager: Receives, processes, and forwards to an application events, such as alarm notifications, security alarms, and state changes.
- Security mechanisms: Provides isolation and security enforcement between applications and services.
- Topology manager: Builds and maintains switch interconnection topology information.
- Statistics manager: Collects data on traffic through the switches.
- Device manager: Configures switch parameters and attributes and manages flow tables.

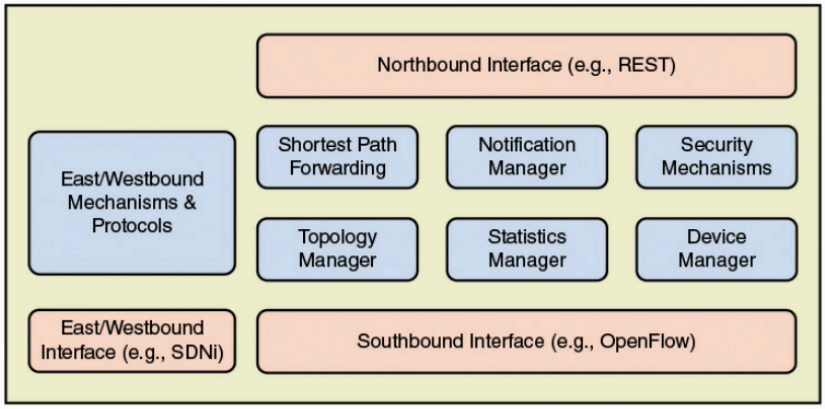

FIGURE 5.2 SDN Control Plane Functions and Interfaces

2. **Why the functionalities offered by control plane can be viewed as a Network Operating System (NOS)?**

   As with a conventional OS, an NOS provides essential services, common application programming interfaces (APIs), and an abstraction of lower-layer elements to developers. The functions of an SDN NOS, such as those in the preceding list, enable developers to define network policies and manage networks without concern for the details of the network device characteristics, which may be heterogeneous and dynamic. The northbound interface, provides a uniform means for application developers and network managers to access SDN service and perform network management tasks. Further, well-defined northbound interfaces enable developers to create software that is independent not only of data plane details but to a great extent usable with a variety of SDN controller servers.

3. **What are the main interfaces and protocols used by the different controllers?**

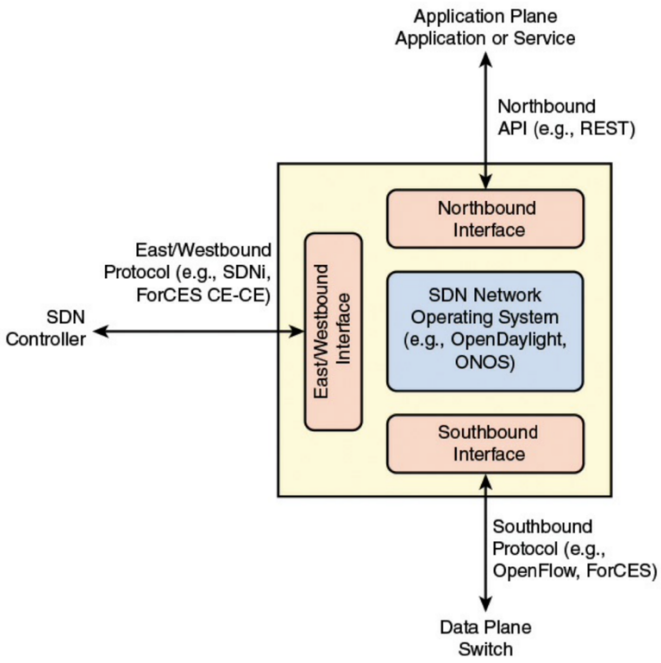

FIGURE 5.3 SDN Controller Interfaces

The southbound interface provides the logical connection between the SDN controller and the data plane switches. Some controller products and configurations support only a single southbound protocol. A more flexible approach is the use of a southbound abstraction layer that provides a common interface for the control plane functions while supporting multiple southbound APIs.
The northbound interface enables applications to access control plane functions and services without needing to know the details of the underlying network switches. The northbound interface is more typically viewed as a software API rather than a protocol.
Unlike the southbound and eastbound/westbound interfaces, where a number of heterogeneous interfaces have been defined, there is no widely accepted standard for the northbound interface. The result has been that a number of unique APIs have been developed for various controllers, complicating the effort to develop SDN applications.

4. O que são o OVS, o POF e o ForCES?

Open vSwitch (OVS): Open vSwitch (OVS) an open source software project which implements virtual switching that is interoperable with almost all popular hypervisors.

Protocol Oblivious Forwarding (POF): This is advertised as an enhancement to OpenFlow that simplifies the logic in the data plane to a very generic forwarding element that need not understand the protocol data unit (PDU) format in terms of fields at various protocol levels. Rather, matching is done by means of (offset, length) blocks within a packet. Intelligence about packet format resides at the control plane level.

Forwarding and Control Element Separation (ForCES): An IETF effort that standardizes the interface between the control plane and the data plane for IP routers.

5. What is the Latitude of Northbound Interfaces?

The northbound interface enables applications to access control plane functions and services without needing to know the details of the underlying network switches. The northbound interface is more typically viewed as a software API rather than a protocol.

A useful insight of the NBI-WG is that even in an individual SDN controller instance, APIs are needed at different “latitudes.” That is, some APIs may be “further north” than others, and access to one, several, or all of these different APIs could be a requirement for a given application.

From the NBI-WG charter document (October 2013), illustrates the concept of multiple API latitudes. For example, an application may need one or more APIs that directly expose the functionality of the controller, to manage a network domain, and use APIs that invoke analytic or reporting services residing on the controller.

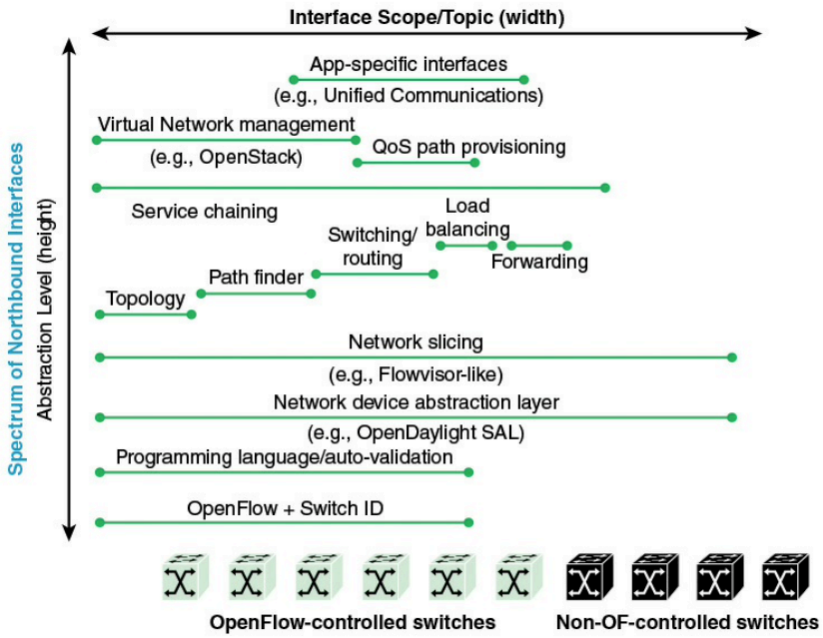

FIGURE 5.4 Latitude of Northbound Interfaces

6. **What comprises the routing function?**

   The routing function comprises a protocol for collecting information about the topology and traffic conditions of the network, and an algorithm for designing routes through the network.

7. **What are the Exterior Routing Protocols (ERPs) and the Interior Routing Protocols (IRPs)? Give examples of such protocols.**

   IRP is concerned with discovering the topology of routers within an AS and then determining the best route to each destination based on different metrics. Two widely used IRPs are Open Shortest Path First (OSPF) Protocol and Enhanced Interior Gateway Routing Protocol (EIGRP). An ERP need not collect as much detailed traffic information. Rather, the primary concern with an ERP is to determine reachability of networks and end systems outside of the AS. Therefore, the ERP is typically executed only in edge nodes that connect one AS to another. Border Gateway Protocol (BGP) is commonly used for the ERP.

8. **Which is the role of ASN (Autonomous System Number)?**

   Autonomous System Number (**ASN**) is a globally unique identifier that defines a group of one or more IP prefixes run by one or more network operators that maintain a single, clearly-defined routing policy. The **ASN** allows the **autonomous systems** to exchange routing information with other **autonomous systems.**

9. What are the main differences between routing in SDN and traditional legacy networks?

As with any network or internet, an SDN network requires a routing function. In general terms, the routing function comprises a protocol for collecting information about the topology and traffic conditions of the network, and an algorithm for designing routes through the network. There are two categories of routing protocols: interior router protocols (IRPs) that operate within an autonomous system (AS), and exterior router protocols (ERPs) that operate between autonomous systems.
Traditionally, the routing function is distributed among the routers in a network. Each router is responsible for building up an image of the topology of the network. For interior routing, each router as well must collect information about connectivity and delays and then calculate the preferred route for each IP destination address.

10. What are the functions done by a centralized routing application?

The centralized routing application performs two distinct functions: link discovery and topology manager.
For link discovery , the routing function needs to be aware of links between data plane switches. Note that in the case of an internetwork, the links between routers are networks, whereas for Layer 2 switches, such as Ethernet switches, the links are direct
physical links. In addition, link discovery must be performed between a router and a host system and between a router in the domain of this controller and a router in a neighboring domain. Discovery is triggered by unknown traffic entering the controller's
network domain either from an attached host or from a neighboring router.
The topology manager maintains the topology information for the network and calculates routes in the network. Route calculation involves determining the shortest path between two data plane nodes or between a data plane node and a host.

11. What are the structure of SDN Control Plane in terms of SDN high-level architecture defined in ITU-T Y.3300

As defined in Y.3300, the application layer is where SDN applications specify network services or business applications by defining a service-aware behavior of network resources. The applications interact with the SDN control layer via APIs that form an application-control interface. The applications make use of an abstracted view of the network resources provided by the SDN control layer by means of information and data models exposed via the APIs.
The control layer provides a means to dynamically control the behavior of network resources, as instructed by the application layer.
The control layer can be viewed as having the following sublayers:

- Application support;

- Orchestration;
- Abstraction;
- Control Support;
- Data transport and processing.

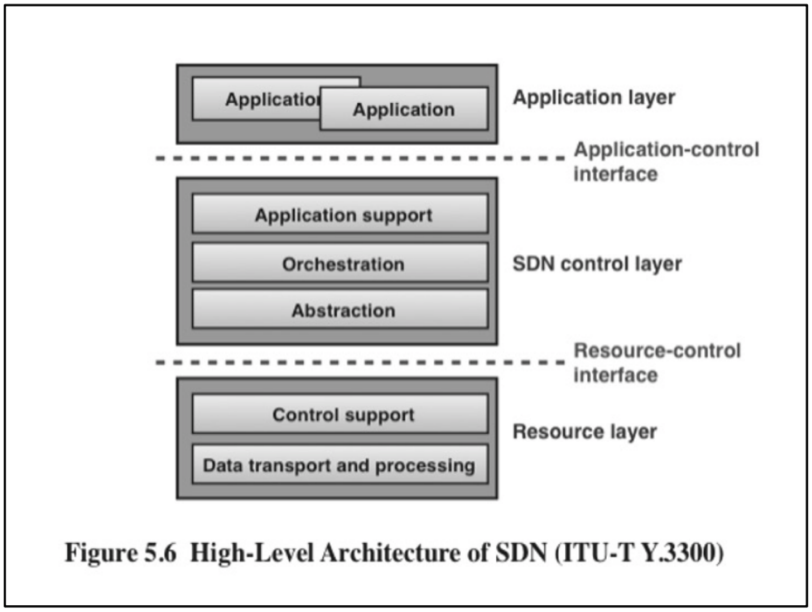

Figure 5.6 High-Level Architecture of SDN (ITU-T Y.3300)

12. What is the mean of REST interface?

The **uniform interface** constraint defines the **interface** between clients and servers. It simplifies and decouples the architecture, which enables each part to evolve independently.

13. What are the six constrains needed for an interface to be RESTful?

- Client-server;
- Stateless;
- Cache;
- Uniform interface;
- Layered system;
- Code on demand.

14. What means the Client-Server Constraint?

This simple constraint dictates that interaction between application and server is in the client-server request/response style. The principle defined for this constraint is the separation of user interface concerns from data storage concerns. This separation allows client and server components to evolve independently and supports the portability of server-side functions to multiple platforms.

15. What means the Stateless Constrain?

The stateless constraint dictates that each request from a client to a server must contain all the information necessary to understand the request and cannot take advantage of any stored context on the server. Similarly, each response from the server must contain all the desired information for that request. One consequence is that any “memory” of a transaction is maintained in a session state kept entirely on the client. Because the server does not retain any record of the client state, the result is a more efficient SDN controller. Another consequence is that if the client and server reside on different machines, and therefore communicate via a protocol, that protocol need not be connection oriented.
REST typically runs over Hypertext Transfer Protocol (HTTP), which is a stateless protocol.

16. What means the Cache Constraint?

The cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cacheable or noncacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests. That is, the client is given permission to remember this data because the data is not likely to change on the server side. Therefore, subsequent requests for the same data can be handled locally at the client, reducing communication overhead between client and server, and reducing the server’s processing burden.

17. What means the Uniform Interface Constraint?

REST emphasizes a uniform interface between components, regardless of the specific client-server application API implemented using REST. This enables controller services to evolve independently and provides the ability for an SDN controller provider to use software components from various vendors to implement the controller.
To obtain a uniform interface, REST defines four interface constraints:

- Identification of resources: Individual resources are identified using a resource identifier (for example, a URI).
- Manipulation of resources through representations: Resources are represented in a format like JSON, XML, or HTML.
- Self-descriptive messages: Each message has enough information to describe how the message is to be processed.
- Hypermedia as the engine of the application state: A client needs no prior knowledge of how to interact with a server, because the API is not fixed but dynamically provided by the server.

The REST style emphasizes that interactions between clients and services is enhanced by having a limited number of operations (verbs). Flexibility is provided by assigning resources (nouns) their own unique Uniform Resource Identifier (URI) . Because each verb has a specific meaning (GET, POST, PUT, and DELETE), REST avoids ambiguity.
The benefit of this constraint, for an SDN environment is that different applications, perhaps written in different languages, can invoke the same controller service via a REST API.

18. What means the Layered System Constraint?

    The layered system constraint simply means that a given function is organized in layers, with each layer only having direct interaction with the layers immediately above and below. This is a fairly standard architecture approach for protocol architectures, OS design, and system services design.

19. What means the Code on demand Constraint?

    REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented. Allowing features to be downloaded after deployment improves system extensibility.

20. What kind of benefits can be achieved using several controllers for the same or a cluster of controllers?

    - Scalability: The number of devices an SDN controller can feasibly manage is limited. Therefore, a reasonably large network may need to deploy multiple SDN controllers.
    - Reliability: The use of multiple controllers avoids the risk of a single point of failure.
    - Privacy: A carrier may choose to implement different privacy policies in different SDN domains. For example, an SDN domain may be dedicated to a set of customers who implement their own highly customized privacy policies, requiring that some networking information in this domain (for example, network topology) should not be disclosed to an external entity.
    - Incremental deployment: A carrier's network may consist of portions of legacy and nonlegacy infrastructure. Dividing the network into multiple individually manageable SDN domains allows for flexible incremental deployment.

21. A Federation of controllers can be used for access, distribution, core and datacenter networks? Explain this kind of collaboration.

    Subscribers are connected to the service network through a hierarchy of access, distribution, and core networks. These intermediate networks may all be operated by the data center network, or they may involve other organizations. In the latter case, if all the networks implement SDN, they need to share common conventions for share control plane parameters, such as quality of service (QoS), policy information, and routing information.

22. Explain the interaction between BGP and OSPF for routing information exchange between in context of SDN.

    Within the non-SDN AS, OSPF is used for interior routing. OSPF is not needed in an SDN domain; rather, the necessary routing information is reported from each data plane switch to the centralized controller using a southbound protocol

(in this case, OpenFlow). Between each SDN domain and the AS, BGP is used to exchange information, such as the following:

- Reachability update: Exchange of reachability information facilitates inter-SDN domain routing. This allows a single flow to traverse multiple SDNs and each controller can select the most appropriate path in the network.
- Flow setup, tear-down, and update requests: Controllers coordinate flow setup requests, which contain information such as path requirements, QoS, and so on, across multiple SDN domains.
- Capability Update: Controllers exchange information on network- related capabilities such as bandwidth, QoS and so on, in addition to system and software capabilities available inside the domain.

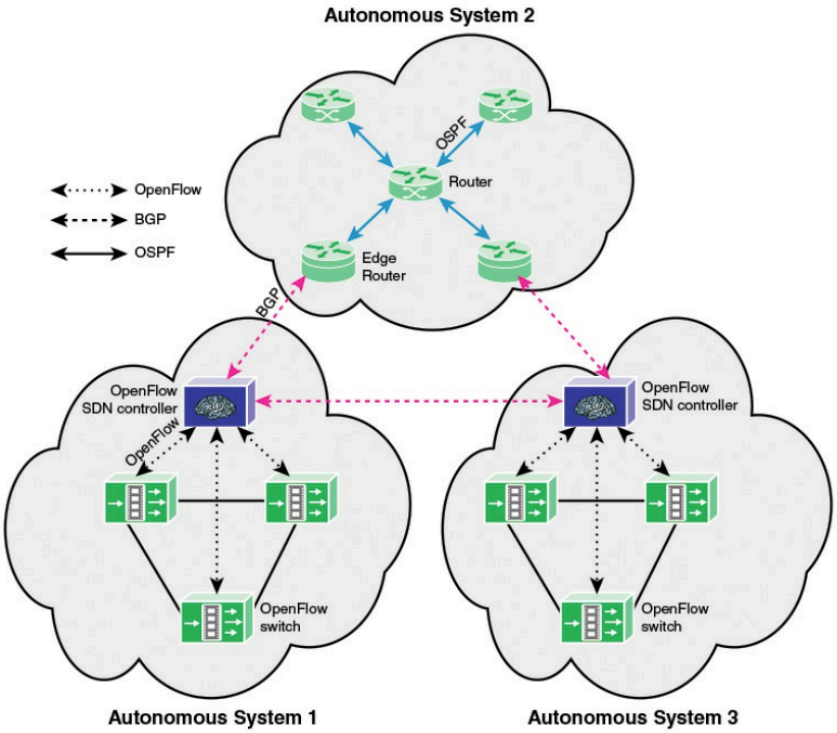

FIGURE 5.12 Heterogeneous Autonomous Systems with OpenFlow and Non-OpenFlow Domains

Cap 6 —— SDN Application Plane

1. **What are the main SDN application plane functions and interfaces?**

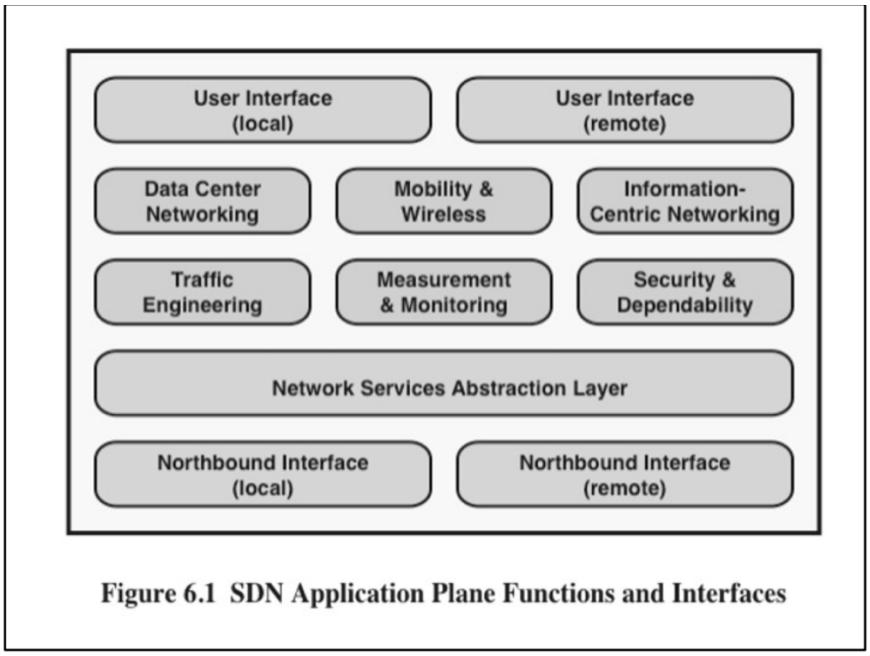

Figure 6.1 SDN Application Plane Functions and Interfaces

The northbound interface enables applications to access control plane functions and services without needing to know the details of the underlying network switches. Typically, the northbound interface provides an abstract view of network resources controlled by the software in the SDN control plane.

Network services abstraction layer between the control and application planes and describes it as a layer that provides service abstractions that can be used by applications and services.

There are many network applications that could be implemented for an SDN.

The user interface enables a user to configure parameters in SDN applications and to interact with applications that support user interaction. Again, there are two possible interfaces. A user that is collocated with the SDN application server (which may or may not include the control plane) can use the server's keyboard/display. More typically, the user would log on to the application server over a network or communications facility.

2. **Does the SDN application plane have to run on the same server as the applications or the SDN control plane?**

Typically, the northbound interface provides an abstract view of network resources controlled by the software in the SDN control plane.
The northbound interface can be a local or remote interface. For a local interface, the SDN applications are running on the same server as the control plane software (controller network operating system). Alternatively, the applications could be run on remote systems and the northbound interface is a protocol or application programming interface (API) that connects the applications to the controller network operating system (NOS) running on

central server. An example of a northbound interface is the REST API for the Ryu SDN network operating system.

3. **List and describe six major application areas of interest for SDN.**

   - Traffic engineering;
   - Measurement and monitoring applications;
   - Security.
   - Data Center Networking
   - Mobility and wireless
   - Information-centric networking

4. **Discuss the following affirmation: "While the SDN data and control planes are well defined, there is much less agreement on the nature and scope of the application plane. At minimum, the application plane includes a number of network applications—that is, applications that specifically deal with network management and control."**

   There is no agreed-upon set of such applications or even categories of such applications. Further, the application layer may include general-purpose network abstraction tools and services that might also be viewed as part of the functionality of the control plane.

5. **What is the role of Network Abstraction Layer?**

   RFC 7426 defines a network services abstraction layer between the control and application planes and describes it as a layer that provides service abstractions that can be used by applications and services.

6. **List and explain the three forms of abstraction in SDN namely forwarding abstraction, distribution abstraction and specification abstraction.**

   The forwarding abstraction allows a control program to specify data plane forwarding behavior while hiding details of the underlying switching hardware. This abstraction supports the data plane forwarding function. By abstracting away from the forwarding hardware, it provides flexibility and vender neutrality.

   The distribution abstraction provides a global view of the network as if there is a single central controller, even if multiple cooperating controllers are used.

   The specification abstraction then provides an abstract view of the global network. This view provides just enough detail for the application to specify goals, such as routing or security policy, without providing the information needed to implement the goals.

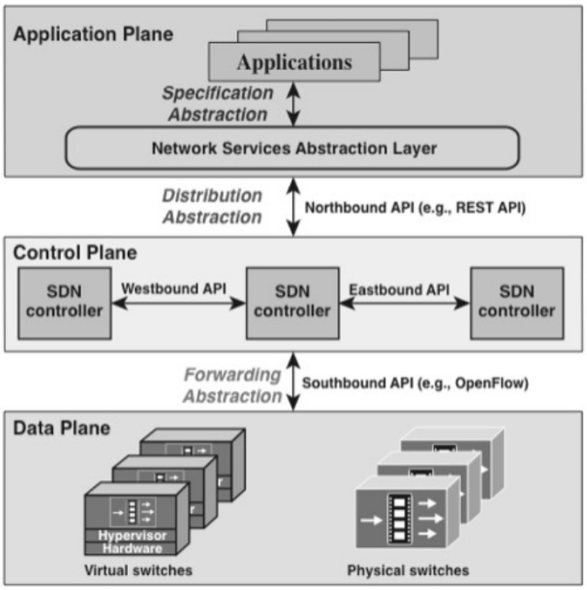

Figure 6.2 SDN Architecture and Abstractions

7. **Why the Frenetic Language can be seen as an example of network services abstraction layer?**

   An example of a network services abstraction layer is the programming language Frenetic. Frenetic enables networks operators to program the network as a whole instead of manually configuring individual network elements. Frenetic was designed to solve challenges with the use of OpenFlow-based models by working with an abstraction at the network level as opposed to OpenFlow, which directly goes down to the network element level.

8. **What is Traffic Engineering?**

   Traffic engineering is a method for dynamically analyzing, regulating, and predicting the behavior of data flowing in networks with the aim of performance optimization to meet service level agreements (SLAs). Traffic engineering involves establishing routing and forwarding policies based on QoS requirements. With SDN, the task of traffic engineering should be considerably simplified compared with a non-SDN network. SDN offers a uniform global view of heterogeneous equipment and powerful tools for configuring and managing network switches.

9. **Present four examples of Traffic Engineering Functions that have been implemented as SDN Applications.**

   - On-demand virtual private networks
   - Load balancing
   - Energy-aware routing
   - Quality of service (QoS) for broadband access networks
   - Scheduling/optimization
   - Traffic engineering with minimal overhead
   - Dynamic QoS routing for multimedia apps
   - Fast recovery through fast-failover groups

- QoS policy management framework
- QoS enforcement
- QoS over heterogeneous networks
- Multiple packet schedulers
- Queue management for QoS enforcement
- Divide and spread forwarding tables

10. What is the Service Layer Agreement?

You can define QoS as the measurable end-to-end performance properties of a network service, which can be guaranteed in advance by a service level agreement (SLA) between a user and a service provider, so as to satisfy specific customer application requirements.

11. Describe PolicyCop, that is an instructive example of traffic engineering SDN application.

PolicyCop consists of eleven software modules and two databases, installed in both the application plane and the control plane. PolicyCop uses the control plane of SDNs to monitor the compliance with QoS policies and can automatically adjust the control plane rules and flow tables in the data plane based on the dynamic network traffic statistics.

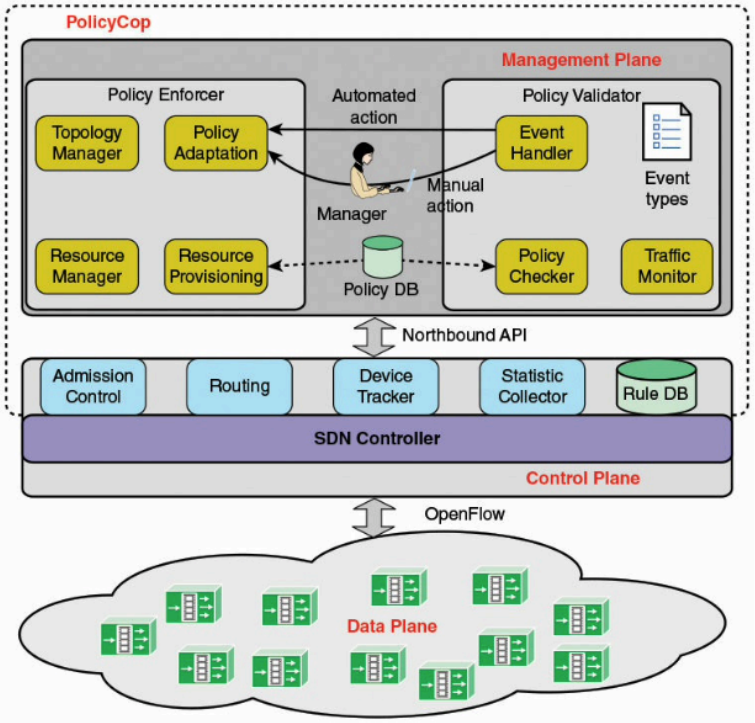

FIGURE 6.5 PolicyCop Architecture

12. In context of problems of SLA parameters (packet loss, throughput, latency, jitter, device failure), what kind of Policy Adaptation Actions can be used?

SLA Parameter	PAA Functionality
Packet loss	Modify queue configuration or reroute to a better path
Throughput	Modify rate limiters to throttle misbehaving flows
Latency	Schedule flow through a new path with less congestion and suitable delay
Jitter	Reroute flow through a less congested path
Device failure	Reroute flows through a different path to bypass the failure

TABLE 6.1 Functionality of Some Example Policy Adaptation Actions (PAAs)

13. What security concerns are introduced by SDN?

- Address security concerns related to the use of SDN: SDN involves a three-layer architecture (application, control, data) and new approaches to distributed control and encapsulating data. All of this introduces the potential for new vectors for attack. Threats can occur at any of the three layers or in the communication between layers. SDN applications are needed to provide for the secure use of SDN itself.

- Use the functionality of SDN to improve network security: Although SDN presents new security challenges for network designers and managers, it also provides a platform for implementing consistent, centrally managed security policies and mechanisms for the network. SDN allows the development of SDN security controllers and SDN security applications that can provision and orchestrate security services and mechanisms.

14. What requirements are need for use of SDNs on Data Center Networking (DCN), Integrated Network Control for Big Data Applications, Cloud Networking, Mobility and Wireless and Information Centric Networking?

[KREU15] lists the following as key requirements for data centers: high and flexible cross-section bandwidth and low latency, QoS based on the application requirements, high levels of resilience, intelligent resource utilization to reduce energy consumption and improve overall efficiency, and agility in provisioning network resources (for example, by means of network virtualization and orchestration with computing and storage).

Mobile users are continuously generating demands for new services with high quality and efficient content delivery independent of location. Network providers must deal with problems related to managing the available spectrum, implementing handover mechanisms, performing efficient load balancing, responding to QoS and QoE requirements, and maintaining security.

Cap. 7 —— Network Function Virtualization

1. **What is the goal of Network Function Virtualization?**

   The overall objective of NFV is leveraging standard IT virtualization technology to consolidate many network equipment types onto industry standard high-volume servers, switches, and storage, which could be located in data centers, network nodes, and in the end-user premises.

2. **What is Virtual Machine Consolidation?**

   The number of guests that can exist on a single host is measured as a consolidation ratio. For example, a host that is supporting six VMs is said to have a consolidation ration of 6 to 1, also written as 6:1.

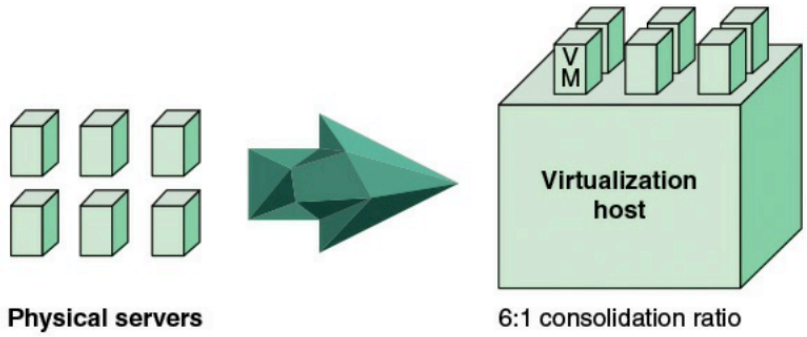

FIGURE 7.2 Virtual Machine Consolidation

3. **What are the architectural approaches of Virtual Machines?**

   Virtualization is all about abstraction. Much like an operating system abstracts the disk I/O commands from a user through the use of program layers and interfaces, virtualization abstracts the physical hardware from the VMs it supports. As noted already, virtual machine monitor, or hypervisor, is the software that provides this abstraction. It acts as a broker, or traffic cop, acting as a proxy for the guests (VMs) as they request and consume resources of the physical host.
   A VM is a software construct that mimics the characteristics of a physical server. It is configured with some number of processors, some amount of RAM, storage resources, and connectivity through the network ports. Once that VM is created, it can be powered on like a physical server, loaded with an operating system and applications, and used in the manner of a physical server. Unlike a physical server, this virtual server sees only the resources it has been configured with, not all the resources of the physical host itself. This isolation allows a host machine to run many VMs, each running the same or different copies of an operating system, sharing RAM, storage, and network bandwidth, without problems. An operating system in a VM accesses the resource that is presented to it by the hypervisor. The hypervisor facilitates the translation of I/O from the VM to the physical server devices, and back again to the correct VM. To achieve this, certain privileged instructions that a “native” operating system would be executing on its host’s hardware now trigger a hardware trap and are run by the hypervisor as a proxy for the VM. This creates some performance

degradation in the virtualization processthough over time both hardware and software improvements have minimalized this overhead.
VMs are made up of files. A typical VM can consist of just a few files. There is a configuration file that describes the attributes of the VM. It contains the server definition, how many virtual processors (vCPUs) are allocated to this VM, how much RAM is allocated, which I/O devices the VM has access to, how many network interface cards (NICs) are in the virtual server, and more. It also describes the storage that the VM can access. Often that storage is presented as virtual disks that exist as additional files in the physical file system. When a VM is powered on, or instantiated, additional files are created for logging, for memory paging, and other functions. That a VM consists of files makes certain functions in a virtual environment much simpler and quicker than in a physical environment. Since the earliest days of computers, backing up data has been a critical function. Because VMs are already files, copying them produces not only a backup of the data but also a copy of the entire server, including the operating system, applications, and the hardware configuration itself.
To create a copy of a physical server, additional hardware needs to be acquired, installed, configured, loaded with an operating system, applications, and data, and then patched to the latest revisions, before being turned over to the users. This provisioning can take weeks or even months depending on the processes in places. Because a VM consists of files, by duplicating those files, in a virtual environment there is a perfect copy of the server available in a matter of minutes. There are a few configuration changes to make (server name and IP address to name two), but administrators routinely stand up new VMs in minutes or hours, as opposed to months.
Another method to rapidly provision new VMs is through the use of templates. A template provides a standardized group of hardware and software settings that can be used to create new VMs configured with those settings. Creating a new VM from a template consists of providing unique identifiers for the new VM and having the provisioning software build a VM from the template and adding in the configuration changes as part of the deployment.
In addition to consolidation and rapid provisioning, virtual environments have become the new model for data center infrastructures for many reasons.

One of these is increased availability. VM hosts are clustered together to form pools of computer resources. Multiple VMs are hosted on each of these servers and in the case of a physical server failure, the VMs on the failed host can be quickly and automatically restarted on another host in the cluster. Compared with providing this type of availability for a physical server, virtual environments can provide higher availability at significantly lower cost and less complexity. For servers that require greater availability, fault tolerance is available in some solutions through the use of shadowed VMs in running lockstep to ensure that no transactions are lost in the event of a physical server failure, again without increased complexity. One of the most compelling features of virtual environments is the capability to move a running VM from one physical host to another, without interruption, degradation, or impacting the users of that VM. vMotion, as it is known in a VMware environment, or Live Migration, as it is known in others, is used for a number of crucial tasks. From an availability standpoint, moving VMs from one host to another without incurring downtime allows administrators to perform work on the physical hosts without impacting operations. Maintenance can be performed on a weekday morning instead of during scheduled downtime on a weekend. New servers can be added to the

environment and older servers removed without impacting the applications. In addition to these manually initiated migrations, migrations can be automated depending on resource usage. If a VM starts to consume more resources than normal, other VMs can be automatically relocated to hosts in the cluster where resources are available, ensuring adequate performance for all the VMs and better overall performance. These are simple examples that only scratch the surface of what virtual environments offer.

4. **What are the differences between Hypervisor type I and type II ? (sketch the respective representation diagrams)**

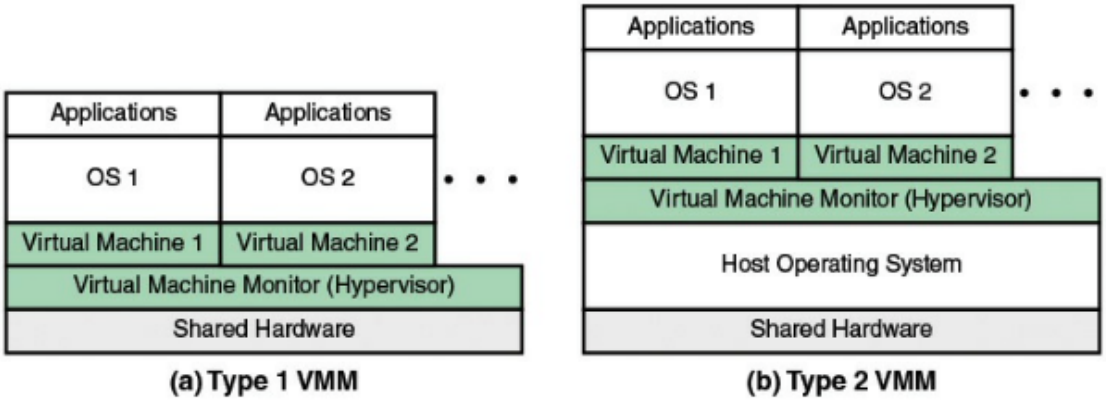

FIGURE 7.3 Type 1 and Type 2 Virtual Machine Monitors

The hypervisor sits between the hardware and the VMs. There are two types of hypervisors, distinguished by whether there is another operating system between the hypervisor and the host. A Type 1 hypervisor is loaded as a thin software layer directly into a physical server, much like an operating system is loaded. Once it is installed and configured, usually within a matter of minutes, the server can then support VMs as guests. In mature environments, where virtualization hosts are clustered together for increased availability and load balancing, a hypervisor can be staged on a new host, the new host can be joined to an existing cluster, and VMs can be moved to the new host without any interruption of service. Some examples of Type 1 hypervisors are VMware

ESXi, Microsoft Hyper-V, and the various open source Xen variants. This idea that the hypervisor is loaded onto the “bare metal” of a server is usually a difficult concept for people to understand. They are more comfortable with a solution that works as a traditional application, program code that is loaded on top of a Microsoft Windows or UNIX/Linux operating system environment. This is exactly how a Type 2 hypervisor is deployed. Some examples of Type 2 hypervisors are VMware Workstation and Oracle VM Virtual Box.

5. **Present the differences between Virtualization Container and the Virtual Machine, presenting advantages and disadvantages.**

A relatively recent approach to virtualization is known as container virtualization. In this approach, software, known as a virtualization container, runs on top of the host OS kernel and provides an execution environment for applications. Unlike hypervisor-based VMs, containers do not aim to emulate physical servers. Instead, all containerized applications on a host share a

common OS kernel. This eliminates the resources needed to run a separate OS for each application and can greatly reduce overhead.

Because the containers execute on the same kernel, thus sharing most of the base OS, containers are much smaller and lighter weight compared to a hypervisor/guest OS VM arrangement. Accordingly, an OS can have many containers running on top of it, compared to the limited number of hypervisors and guest operating systems that can be supported.

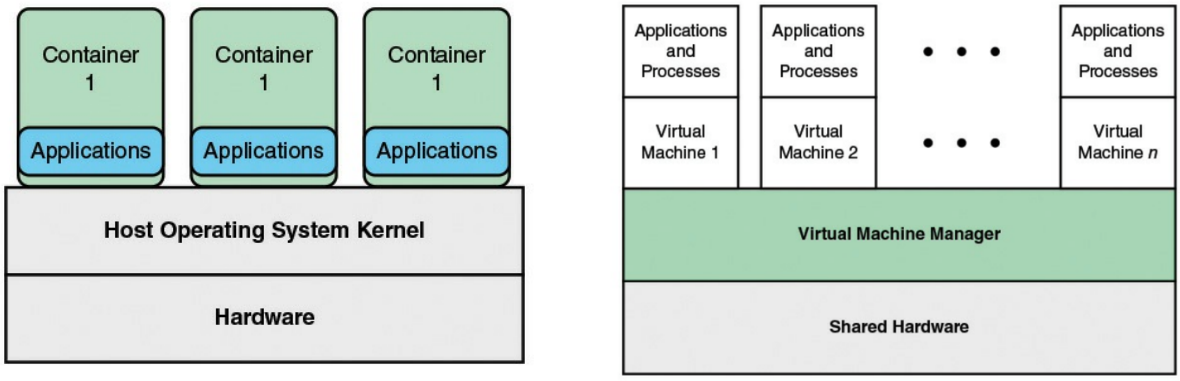

FIGURE 7.1 Virtual Machine Concept

6. How we can define NFV?

Virtualization of network functions by implementing these functions in software and running them on VMs.

7. What are the main differences between NFV and the traditional approaches to networking?

In traditional networks, all devices are deployed on proprietary/closed platforms. All network elements are enclosed boxes, and hardware cannot be shared. Each device requires additional hardware for increased capacity, but this hardware is idle when the system is running below capacity. With NFV, however, network elements are independent applications that are flexibly deployed on a unified platform comprising standard servers, storage devices, and switches. In this way, software and hardware are decoupled, and capacity for each application is increased or decreased by adding or reducing virtual resources.

8. **Please list examples of network function devices, network-related computer devices and network attached computer devices.**

- Network function devices: Such as switches, routers, network access points, customer premises equipment (CPE), and deep packet inspectors (for deep packet inspection).
- Network-related compute devices: Such as firewalls, intrusion detection systems, and network management systems.
- Network-attached storage: File and database servers attached to the network.

9. Define the following NFV terminology: compute domain, infra-structure network domain, network function, network function virtualization, network function virtualization infra-structure (NFVI), NFVI-node, NFVI-PoP, network forwarding path, network point of presence (N-PoP), network service, NFV Intra-structure, Physical Network Function (PNF), Virtual Machine (VM), Virtual Network, Virtual Network Function (VNF), VNF Forwarding Graph (VNF FG), VNF set.

Term	Definition
Compute domain	Domain within the NFVI that includes servers and storage.
Infrastructure network domain (IND)	Domain within the NFVI that includes all networking that interconnects compute/storage infrastructure.
Network function (NF)	A functional block within a network infrastructure that has well-defined external interfaces and well-defined functional behavior. Typically, this is a physical network node or other physical appliance.
Network functions virtualization (NFV)	The principle of separating network functions from the hardware they run on by using virtual hardware abstraction.
Network functions virtualization infrastructure (NFVI)	The totality of all hardware and software components that build up the environment in which virtual network functions (VNFs) are deployed. The NFVI can span across several locations (that is, multiple points of presence [N-PoPs]). The network providing connectivity between these locations is considered to be part of the NFVI.
NFVI-Node	Physical devices deployed and managed as a single entity, providing the NFVI functions required to support the execution environment for VNFs.
NFVI-PoP	An N-PoP where a network function is or could be deployed as a VNF.
Network forwarding path	Ordered list of connection points forming a chain of NFs, along with policies associated with the list.
Network point of presence (N-PoP)	A location where a network function is implemented as either a physical network function (PNF) or a VNF.
Network service	A composition of network functions that is defined by its functional and behavioral specification.
Physical network function (PNF)	An implementation of a NF via a tightly coupled software and hardware system. This is typically a proprietary system.
Virtual machine (VM)	A virtualized computation environment that behaves very much like a physical computer/server.
Virtual network	A topological component used to affect routing of specific characteristic information. The virtual network is bounded by its set of permissible network interfaces. In the NFVI architecture, a virtual network routes information among the network interfaces of VM instances and physical network interfaces, providing the necessary connectivity.
Virtualized network function (VNF)	An implementation of an NF that can be deployed on an NFVI.
VNF forwarding graph (VNF FG)	Graph of logical links connecting VNF nodes for the purpose of describing traffic flow between these network functions.
VNF set	Collection of VNFs with unspecified connectivity between them.

10. **Explain the key NFV principles used to create practical network services namely:**

    Service Chaining - VNFs are modular and each VNF provides limited functionality on its own. For a given traffic flow within a given application, the service provider steers the flow through multiple VNFs to achieve the desired network functionality. This is referred to as service chaining.

    Management and Orchestration (MANO) - This involves deploying and managing the lifecycle of VNF instances. Examples include VNF instance creation, VNF service chaining, monitoring, relocation, shutdown, and billing. MANO also manages the NFV infrastructure elements.

    Distributed Architecture - A VNF may be made up of one or more VNF components (VNFC), each of which implements a subset of the VNF's functionality. Each VNFC may be deployed in one or multiple instances. These instances may be deployed on separate, distributed hosts to provide scalability and redundancy.

11. **What are the NFV Benefits when compared with traditional network approaches**

    If NFV is implemented efficiently and effectively, it can provide a number of benefits compared to traditional networking approaches. The following are the most important potential benefits:

    - Reduced CapEx, by using commodity servers and switches, consolidating equipment, exploiting economies of scale, and supporting pay-as-you grow models to eliminate wasteful overprovisioning. This is perhaps the main driver for NFV.

    - Reduced OpEx, in terms of power consumption and space usage, by using commodity servers and switches, consolidating equipment, and exploiting economies of scale, and reduced network management and control expenses. Reduced CapeX and OpEx are perhaps the main drivers for NFV.

    - The ability to innovate and roll out services quickly, reducing the time to deploy new networking services to support changing business requirements, seize new market opportunities, and improve return on investment of new services. Also lowers the risks associated with rolling out new services, allowing providers to easily trial and evolve services to determine what best meets the needs of customers.

    -Ease of interoperability because of standardized and open interfaces.

    - Use of a single platform for different applications, users and tenants. This allows network operators to share resources across services and across different customer bases.

    - Provided agility and flexibility, by quickly scaling up or down services to address changing demands.

- Targeted service introduction based on geography or customer sets is possible. Services can be rapidly scaled up/down as required.

- A wide variety of ecosystems and encourages openness. It opens the virtual appliance market to pure software entrants, small players and academia, encouraging more innovation to bring new services and new revenue streams quickly at much lower risk.

12. What are the NFV Requirements

- Portability/interoperability: The capability to load and execute VNFs provided by different vendors on a variety of standardized hardware platforms. The challenge is to define a unified interface that clearly decouples the software instances from the underlying hardware, as represented by VMs and their hypervisors.
- Performance trade-off: Because the NFV approach is based on industry standard hardware (that is, avoiding any proprietary hardware such as acceleration engines), a probable decrease in performance has to be taken into account. The challenge is how to keep the performance degradation as small as possible by using appropriate hypervisors and modern software technologies, so that the effects on latency, throughput, and processing overhead are minimized.
- Migration and coexistence with respect to legacy equipment: The NFV architecture must support a migration path from today's proprietary physical network appliance-based solutions to more open standards-based virtual network appliance solutions. In other words, NFV must work in a hybrid network composed of classical physical network appliances and virtual network appliances. Virtual appliances must therefore use existing northbound Interfaces (for management and control) and interwork with physical appliances implementing the same functions.
- Management and orchestration: A consistent management and orchestration architecture is required. NFV presents an opportunity, through the flexibility afforded by software network appliances operating in an open and standardized infrastructure, to rapidly align management and orchestration northbound interfaces to well defined standards and abstract specifications.
- Automation: NFV will scale only if all the functions can be automated. Automation of process is paramount to success.
- Security and resilience: The security, resilience, and availability of their networks should not be impaired when VNFs are introduced.
- Network stability: Ensuring stability of the network is not impacted when managing and orchestrating a large number of virtual appliances between different hardware vendors and hypervisors. This is particularly important when, for example, virtual functions are relocated, or during reconfiguration events (for example, because of hardware and software failures) or because of cyber-attack.
- Simplicity: Ensuring that virtualized network platforms will be simpler to operate than those that exist today. A significant focus for network operators is simplification of the plethora of complex network platforms and support systems that have evolved over decades of network

technology evolution, while maintaining continuity to support important revenue generating services.

- Integration: Network operators need to be able to “mix and match” servers from different vendors, hypervisors from different vendors, and virtual appliances from different vendors without incurring significant integration costs and avoiding lock-in. The ecosystem must offer integration services and maintenance and third-party support; it must be possible to resolve integration issues between several parties. The ecosystem will require mechanisms to validate new NFV products.