CG 311 Lesson 08: Virtualization and Network Function Virtualization

Virtualization Techniques

Virtualization is essentially a mapping and emulating process. It maps the virtual resource to a native hardware resource and then uses the latter for computation. According to where the mapping occurs, we will first explain the widely used hardware-level virtualization and then cover other layers' virtualization as well.

Hardware-Level Virtualization

Figure 2.2 shows two widely used virtualization architectures. The most important component in this architecture is called the virtual machine monitor (VMM). VMM uses software to emulate the virtualized processor, I/O, memory, storage, and so on. It can support multiple VM instances simultaneously. Each VM has its application and OS layer, just as a normal compute model. The OS passes the instructions to the VMM for execution. On the left side, the VMM is a software layer sitting on the bare metal hardware. So this architecture is also called a stand-alone VM implementation. On the right side of Figure 2.2, the VMM runs as an application on the host OS. It can utilize the host OS's functionality, such as memory management, process scheduling, and hardware drivers. Both architectures have been widely used in many virtualization products.

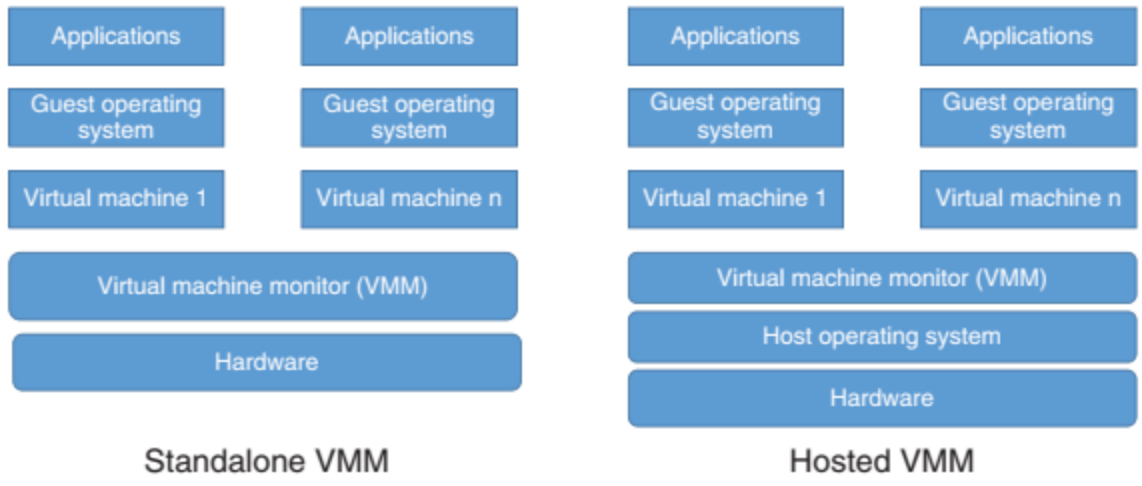


Figure 2.2 Virtual machine architectures.

The possibility of virtualization needs the support from the computer architecture level: the privileged instructions must trap so that the guest VM's privileged execution will be sent to VMM. The VMM will coordinate the scheduling of execution across VMs; either the instruction is executed on the native hardware, or an emulated result is returned. VMM is in complete control of the entire system resource.

Other Virtualization Techniques

While hardware virtualization is the fundamental building block of cloud computing, there are other types of virtualization. One of the well-known ones is the Java virtual machine (JVM), which provides virtualization abstraction at the programming language level. As a platform-independent language, Java has a backend interpreter that performs platform-dependent translations. Programs run on the JVM platform are compiled to a standardized portable binary format in the form of class files. The binary is executed by the JVM runtime, which emulates the JVM instructions.

Containers

Containers are a game-changer in the virtualization space. Starting a new virtualization cannot get easier than this. Containers are easy to use, fast, and ensure real software portability. Each container has all its required runtime dependencies and configurations. Containers that run on the same engine share the same Linux kernel of the host server and are executed by the Docker engine rather than a hypervisor.

Containers are much smaller in terms of size than virtual machines and have shorter start-up times. Therefore, Docker containers are considered the agile and lightweight solution. While the lightweight OS solutions have existed for a few years, Docket container [21] is the one that leads to the mass adoption and the hype around containerization. Docker container was initially designed as a Go language building runtime. Today, the Open Container Initiative builds the ecosystem around Docker containers. Recently, a container standardization was discussed, and a runtime implementation was produced, called runC.

Figure 2.3 shows the architecture of container. Compared to the traditional VM architecture in Figure 2.2, the container does not have the per VM guest OS, which is usually tens of gigabytes. Each VM instance has a full guest OS image in addition to the binaries and libraries needed for the applications. It usually leads to high memory and disk utilization and thus a slow start-up time.

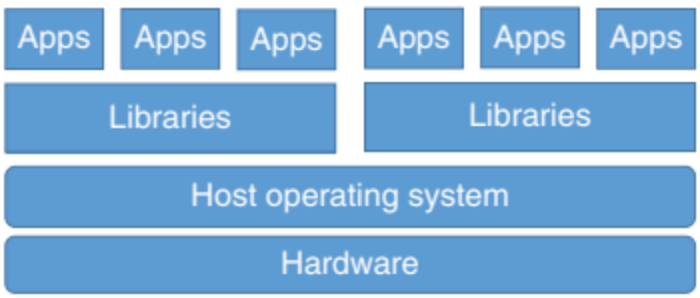


Figure 2.3 Docker container architectures.

In contrast, in a container, each application runs as an isolated process in user space on the host operating system, sharing the kernel with other containers. Each application is usually on the order of megabytes, which is of much lighter weight. Therefore, a container is a much more portable and efficient solution than VM. The resource isolation benefits are still provided by the host OS. Thus, a container is a much lighter virtualization concept.

The main technologies behind containers are namespace, control group, and union file system (UFS). Each container runs a set of namespaces, which provides the isolation. Each container can only access its own namespaces. The control group is used to set up access control to hardware resources for each container. The UFS provides lightweight and fast layering, which is

the building block of a container. Docker container builds on top of these techniques in the container format in the form of libcontainer. In addition, Docker also supports traditional Linux container LXC.

The entire Docker stack in a clustered Dockerized environment contains five major layers: Cluster Manager layer, Node (or host) layer in the cluster, DockerDemon layer that runs on each node, Containers layer created by the Docker Demon, and Applications layer that contains applications run on each container. Different Docker cluster managers are structured in different forms. For example, on Swarm, the containers are part of its structure, but on Kubernetes, there is only a Pod that represents a set of containers, and the user can control the Docker containers just through the cluster Pods.

Network Functions Virtualization (NFV)

In the evolution of modern networking, Network Functions Virtualization (NFV) and Software Defined Networking (SDN) are often conflated due to their closely related objectives. While distinct in their operational layers, the symbiosis between these technologies is a critical driver for the transformation of carrier and enterprise architectures. This lesson defines the architectural framework of NFV and examines its structural relationship with SDN.

10.1 Definition and Architectural Vision

NFV was formally introduced in 2012 at the SDN and OpenFlow World Congress, with the European Telecommunications Standards Institute (ETSI) subsequently assuming leadership in developing its standards.

- **ETSI (European Telecommunications Standards Institute):** A non-profit, regional standards organization for the information and communications technology (ICT) community, responsible for the standardized architectural framework of NFV.

NFV aims to transform network architecture by evolving standard IT virtualization technology to consolidate various network equipment types onto industry-standard high-volume servers, switches, and storage. This transition involves migrating from **Physical Network Functions (PNF)** to **Virtual Network Functions (VNF)**.

- **Physical Network Function (PNF):** A network function implemented as a tightly coupled software-hardware appliance, typically utilizing proprietary application-specific integrated circuits (ASICs) or specialized hardware.

- **Virtual Network Function (VNF):** The software implementation of a network function that runs on a virtualized infrastructure (e.g., a Virtual Machine or Container) decoupled

from the underlying hardware.

Definition: Network Functions Virtualization (NFV) is a network architecture concept that utilizes virtualization technologies to manage core networking functions through software running on commodity hardware, rather than specialized hardware appliances.

Purpose and Motivation: Traditionally, network services such as firewalls, routers, or load balancers are delivered using dedicated, purpose-built hardware. NFV decouples network functions from this hardware, allowing them to run on standard x86/ARM/COTS servers, improving scalability, flexibility, and cost-efficiency.

Key Components of NFV Architecture (as defined by ETSI)

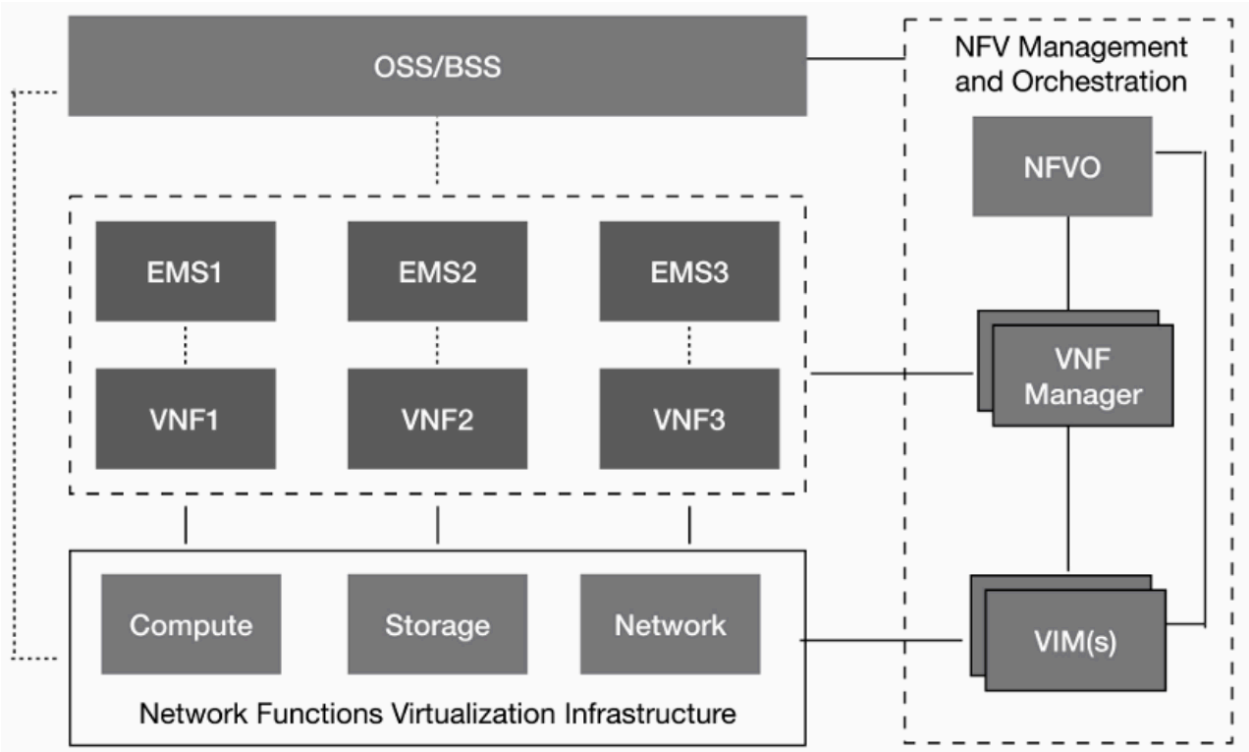
VNF (Virtual Network Function): Software-based implementation of a network function (e.g., virtual firewall, virtual router). Can be chained with others to form service chains.

NFVI (Network Functions Virtualization Infrastructure): The hardware and software resources where VNFs are deployed. Includes compute, storage, and networking resources, managed by a virtualization layer (e.g., hypervisors).

MANO (Management and Orchestration): Framework for managing VNFs, lifecycle, and infrastructure. Includes:

- **NFV Orchestrator (NFVO)** – oversees overall service orchestration.
- **VNF Manager (VNFM)** – manages VNF lifecycles.
- **Virtualized Infrastructure Manager (VIM)**-manages NFVI such as OpenStack, Kubernetes, etc.

4. NFV Reference Architecture (ETSI NFV-MANO Framework)



Component-wise breakdown of the NFV architecture shown in the Figure above, organized by its functional blocks:

Virtualized Network Functions (VNF) Block

- **VNF (Virtual Network Function):** The software implementation of a network function (like a Firewall, Router, or IMS component) that runs on virtualized infrastructure.
- **EMS (Element Management System):** Performs traditional management tasks for a VNF, such as fault, configuration, accounting, performance, and security (FCAPS) functions.

NFV Infrastructure (NFVI) Block

- **Compute/Storage/Network:** The physical hardware resources (servers, disk arrays, and switches) that provide the foundation for the virtualized environment.
- **Virtualization Layer:** The software (typically a hypervisor or container engine) that abstracts the physical hardware into virtual resources for the VNFs to use.

NFV Management and Orchestration (MANO) Block

- **NFVO (NFV Orchestrator):** Orchestrates the overall resource allocation across multiple data centers and manages the lifecycle of network services.
- **VNFM (VNF Manager):** Manages the specific lifecycle of VNF instances, including tasks like instantiation, scaling, updating, and termination.

- **VIM (Virtualized Infrastructure Manager):** Controls and manages the NFVI resources (compute, storage, and network) within a specific domain or data center.

Business and Operations Block

- **OSS (Operations Support System):** Handles the management of the network infrastructure, including service provisioning, network configuration, and fault management.
- **BSS (Business Support System):** Manages customer-facing activities such as billing, order management, and customer relationship management (CRM).

Network Functions Virtualisation aims to transform the way that network operators architect networks by evolving standard IT virtualisation 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 end-user premises. It involves the implementation of network functions in software that can run on a range of industry-standard server hardware and that can be moved to or instantiated in various locations in the network as required, without the need for installation of new equipment. This definition is made with reference to Fig. 10.1.

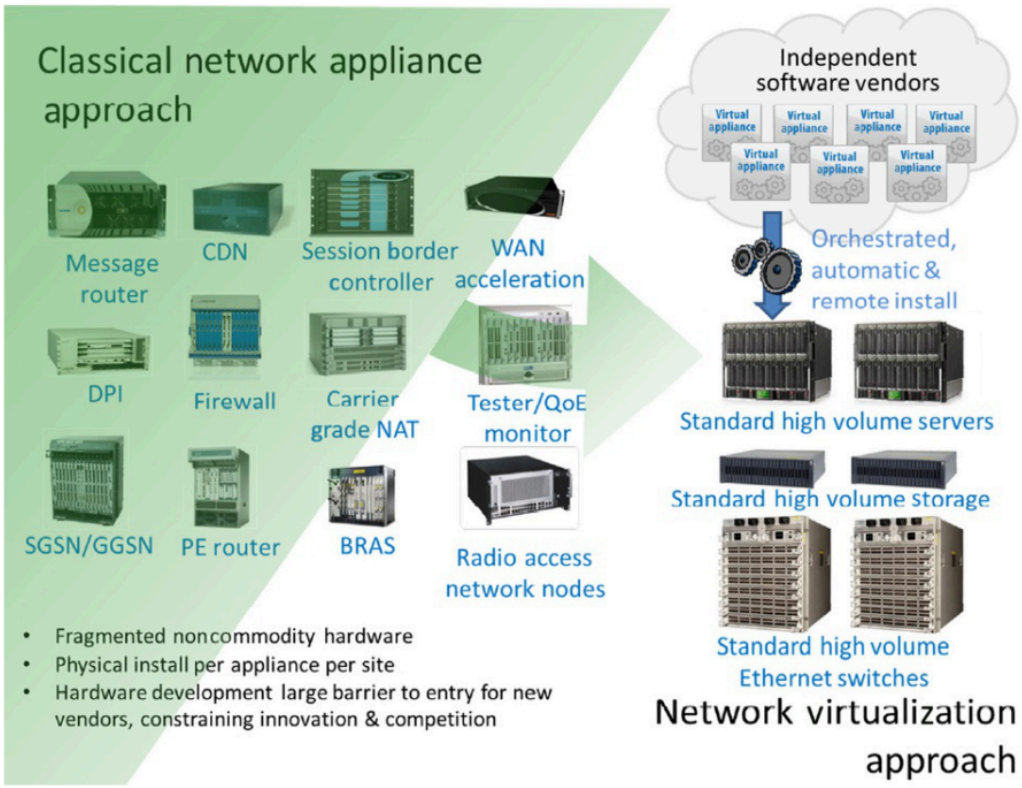


FIG. 10.1

Vision for NFV.

FIGURE 10.1: Vision for NFV – Transition from Classical to Virtualized Approaches: Fig. 10.1 illustrates a paradigm shift in network deployment. On the left, the Classical Network Appliance Approach shows a fragmented ecosystem of non-commodity hardware where each function—such as firewalls, Deep Packet Inspection (DPI), or Message Routers—requires a physical installation per site. This creates a high barrier to entry and constrains innovation due to proprietary hardware dependencies.

On the right, the **Network Virtualization Approach** depicts the consolidation of these functions. Independent software vendors provide virtual appliances that undergo orchestrated, automatic, and remote installation onto a unified pool of **Standard High-Volume Servers**, **Storage**, and **Ethernet Switches**. The logic flow demonstrates the decoupling of the software layer from the hardware layer, allowing for dynamic instantiation of services without the need for manual hardware deployment.

The ETSI NFV Framework

The ETSI framework defines the structural components necessary to create and manage virtualized network services. Central to this is the **NFV Infrastructure (NFVI)** and the

management entities.

- **NFV Infrastructure (NFVI):** The totality of hardware and software components (compute, storage, and networking) that build up the environment in which VNFs are deployed.

- **NFV Orchestrator (NVFO):** A functional block responsible for the orchestration of NFVI resources across multiple Virtualized Infrastructure Managers (VIMs) and the lifecycle management of Network Services.

- **VNF Manager:** A functional entity responsible for the lifecycle management of VNF instances, including instantiation, scaling, and health monitoring.

The orchestrator serves a role conceptually similar to a hypervisor in cloud computing, responsible for instantiating the compute, memory, and storage resources required by the virtual machines hosting the VNFs. Furthermore, the orchestrator manages **Service Chaining**.

- **Service Chaining:** The process of connecting a suite of virtualized network services (VNFs) in a specific order over a single network connection to provide a comprehensive end-to-end service.

Table 1: Comparison of Hardware Approaches

The following table summarizes the transition from proprietary hardware to the NFV model.

Feature	Classical Appliance Approach	NFV / Virtualized Approach
Hardware Type	Proprietary, non-commodity	Industry-standard high-volume servers
Deployment	Physical install per site/appliance	Remote, orchestrated instantiation
Scaling	Vertical (Requires new hardware)	Horizontal (Software-driven resource allocation)

Innovation	Low (High barrier to entry)	High (Driven by independent software vendors)

The data in Table 1 indicates that the classical approach is limited by the physical constraints of "fragmented" hardware, requiring significant manual labor and capital expenditure for every new service. In contrast, the NFV approach leverages commodity hardware to lower costs and increase agility. For a network engineer, this means a shift from managing physical hardware assets to managing software lifecycles and resource orchestration.

> **Modern Insight (2026): CNFs and Microservices**
> While the early ETSI standards focused heavily on Virtual Machines (VMs), current 2026 industry standards have shifted toward Cloud-native Network Functions (CNFs). CNFs utilize containerization (e.g., Kubernetes) rather than full VMs, allowing for even faster instantiation times and more granular scaling. This evolution further optimizes the resource utilization described in the original NFV framework.

10.2 MANO: Management and Orchestration

The ETSI model for NFV defines a specific management and orchestration architecture, nicknamed **MANO**. This framework addresses the immense complexity of deploying and managing thousands of virtual machines (VMs) across large-scale carrier networks.

- **MANO (Management and Orchestration):** The functional block in the ETSI NFV architecture that manages the lifecycles of VNFs and the infrastructure resources they require.

The MANO framework consists of three primary functional blocks:

1. **NFV Orchestrator (NFVO):** Responsible for "onboarding" new Network Services (NS) and VNF Packages. It manages the global resource pool, validates resource requests, and handles policy management for NS instances.
2. **VNF Manager:** Provides lifecycle management for VNF instances. It coordinates configuration and event reporting between the infrastructure and higher-level management systems like the **EMS** and **NMS**.
    - **EMS (Element Management System):** A system responsible for the functional management (configuration, fault, performance) of one or more network elements.

    - **NMS (Network Management System):** A platform used for monitoring and controlling a network at a systemic level.
3. **Virtualized Infrastructure Manager (VIM):** Controls and manages the NFVI resources (compute, storage, and network) within a specific operator's subdomain. It is responsible for collecting and forwarding performance measurements and events.

10.3 Virtualization Scope and Use Cases

A common point of confusion is the distinction between **Network Virtualization (NV)** and **Network Functions Virtualization (NFV)**.

- **Network Virtualization (NV):** The process of creating an overlay of the physical network, typically using tunnels to provide connectivity between virtual machines.
- **NFV:** Specifically, virtualizes Layer 4 through Layer 7 functions, such as firewalls and load balancers.

What Can We Virtualize?

Virtually any Physical Network Function (PNF) can be migrated to a VNF. ETSI identifies several high-priority use cases:

Edge and Routing Components:

- **AR (Access Router / Enterprise CPE):**
  The Access Router or Customer Premises Equipment (CPE) serves as the gateway for an enterprise branch or office. It connects the local area network (LAN) to the wide area network (WAN) or the Internet, handling basic routing, NAT, and sometimes VPN termination.
- **PE (Provider Edge Router):**
  Located at the edge of a Service Provider's network, the PE router interfaces directly with the customer's CPE. In MPLS (Multi-Protocol Label Switching) networks, it is responsible for adding or removing labels to route traffic across the provider's core to the correct destination.

Security and Inspection:

- **FW (Enterprise Firewall):**
  A traditional Firewall acts as a barrier between a trusted internal network and untrusted external networks. It filters traffic based on established security rules, typically looking at Layer 3 and Layer 4 information (IP addresses and ports).
- **NG-FW (Next-Generation Firewall):**
  An NG-FW goes beyond traditional port/protocol inspection. It includes

application-level inspection (Layer 7), integrated intrusion prevention (IPS), and advanced malware detection to identify and block sophisticated threats that traditional firewalls might miss.

- **DPI (Deep Packet Inspection):**
  DPI is a method of packet filtering that examines the data part (and the header) of a packet as it passes an inspection point. It is used to identify the specific application or service generating the traffic, allowing for granular security filtering or traffic shaping.
- **IPS (Intrusion Prevention System):**
  An IPS is a security appliance or function that monitors network traffic for malicious activity. Unlike an IDS (Detection System) which only alerts, an IPS is "in-line" and can actively block or drop suspicious traffic in real-time.

Optimization and Visibility:

- **WOC (WAN Optimization Controller):**
  A WOC is used to increase the efficiency of data transfer across WAN links. It employs techniques like data deduplication, compression, and protocol optimization to reduce latency and save bandwidth for distributed enterprises.
- **Network Performance Monitoring (NPM):**
  NPM refers to the tools and processes used to track the health and quality of a network. It measures key metrics such as latency, jitter, packet loss, and throughput to ensure the infrastructure meets business requirements and to assist in troubleshooting.

Service and Control Layer:

IP Multimedia Subsystem (IMS):

The **IP Multimedia Subsystem (IMS)** is a standardized architectural framework used to deliver IP-based multimedia services such as **VoLTE** (Voice over LTE), **VoWiFi** (Voice over Wi-Fi), and **RCS** (Rich Communication Services) over mobile and fixed-line networks. Originally designed to bridge the gap between traditional cellular services and Internet technologies, IMS utilizes the **Session Initiation Protocol (SIP)** to manage communication. In modern environments, IMS operates over the **Evolved Packet Core (EPC)** and is typically deployed using **Network Functions Virtualization (NFV)**, allowing these components to run as virtual machines or containers rather than dedicated proprietary hardware.

Key Virtualized Components

The IMS framework relies on several core functions to manage user sessions, security, and data:

- **P-CSCF (Proxy Call Session Control Function):** The initial point of contact for a user's device (UE) within the IMS network. It functions as a security gateway, handling

subscriber authentication, signaling encryption, and SIP compression to ensure efficient communication.

- **S-CSCF (Serving Call Session Control Function):** The central node of the signaling plane. It manages user registration, maintains the "state" of each session, and coordinates with application servers to trigger specific services, such as call forwarding or multiparty conferencing.
- **HSS (Home Subscriber Server):** The master database for the network. It stores subscriber profiles, performs authentication and authorization of the user, and maintains real-time information regarding the user's physical location and registration status.
- **PCRF (Policy and Charging Rules Function):** The policy engine that operates in real-time to manage **Quality of Service (QoS)** and billing. It ensures that multimedia sessions (like a video call) receive the necessary bandwidth and priority while simultaneously communicating with the charging system to track usage.

Component Summary Table

Component	Category	Primary Responsibility
P-CSCF	Signaling/Security	Entry point, SIP proxy, and encryption.
S-CSCF	Session Control	Registration, session routing, and service logic.
HSS	Data Management	Subscriber profiles, authentication, and location.
PCRF	Policy/Charging	Bandwidth allocation and QoS rule enforcement.

Core Network Layer: Evolved Packet Core (EPC)

The **Evolved Packet Core (EPC)** is the flat, all-IP backbone of the **4G LTE** network architecture. It is designed to converge voice and data into a single packet-switched system,

replacing the dual circuit-switched (voice) and packet-switched (data) domains found in 2G and 3G networks.

By using a simplified architecture with fewer nodes, the EPC reduces latency, increases throughput, and allows for seamless mobility between different radio access technologies (like moving from LTE to Wi-Fi or 3G). In modern deployments, EPC functions are often virtualized (**vEPC**) to run on standard cloud infrastructure.

Key EPC Components

While you've already noted the signaling and subscriber database roles (HSS, PCRF, CSCFs), the EPC includes these critical "traffic-handling" nodes:

- **MME (Mobility Management Entity):** The "control room" of the EPC. It does not handle actual user data (video, web traffic); instead, it handles signaling. It manages your device's attachment to the network, tracks your location as you move between cell towers, and chooses which gateway should handle your traffic.
- **SGW (Serving Gateway):** The local anchor point for your data. The SGW routes and forwards user data packets between the cell tower (**eNodeB**) and the rest of the core network. It acts as a stationary point while you move between different towers, ensuring your connection doesn't drop during a handover.
- **PGW (Packet Data Network Gateway):** The "exit and entry" point of the mobile network. The PGW connects the EPC to external IP networks, such as the Internet or a corporate VPN. It is also responsible for allocating your device's IP address and enforcing the quality and security rules defined by the PCRF.

EPC Component Summary

Component	Function Type	Primary Responsibility
MME	Control Plane	Signaling, mobility tracking, and security setup.
SGW	User Plane	Data routing and local mobility anchoring.

PGW	User Plane	Internet gateway, IP allocation, and policy enforcement.

A list of some possible NFV elements [6]isshowninFig. 10.2.1 NFV elements are the discrete hardware and software requirements that are managed in an NFV installation to provide new communication and application services that have traditionally been performed by application-specific appliances.

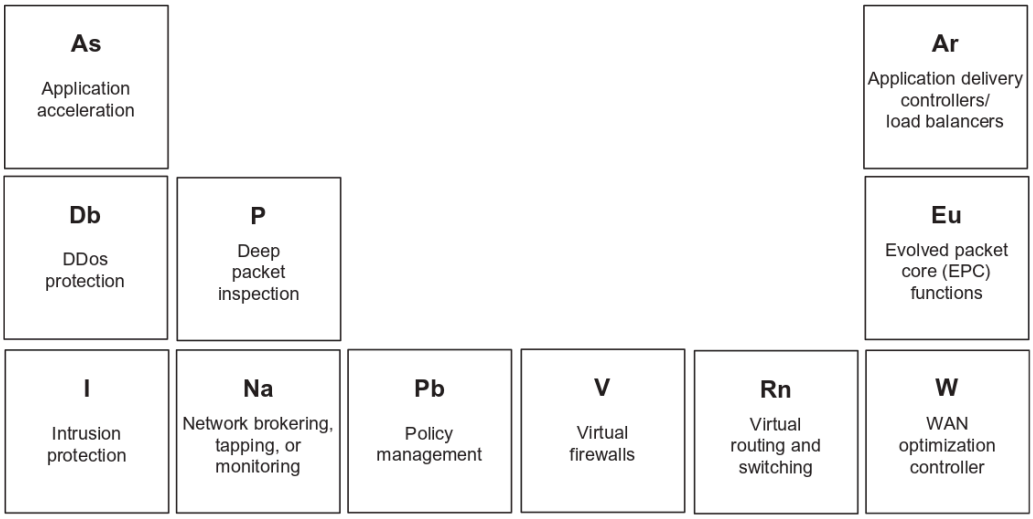


FIG. 10.2

Elements of NFV.

FIGURE 10.2: Elements of NFV – A Periodic Table Representation: Figure 10.2 represents the various discrete network functions that can be virtualized, organized in a grid format. Each block represents a specific service:

- **Security:** DDoS protection (**Db**), Intrusion Protection (**I**), and Virtual Firewalls (**V**).
- **Traffic Management:** Application acceleration (**As**), Deep Packet Inspection (**P**), and Load Balancers (**Ar**).
- **Core Infrastructure:** Evolved Packet Core (**Eu**), Virtual routing/switching (**Rn**), and WAN optimization (**W**).

The logic flow demonstrates that these functions, once requiring application-specific hardware, are now discrete software requirements managed within an NFV installation to provide communication and application services.

Table 2: Comparison of MANO Components

Component	Primary Responsibility	Scope of Control
NFVO	Network Service orchestration	Global resource/service lifecycle
VNF Manager	VNF health and lifecycle	Individual VNF instances
VIM	Infrastructure resource control	Compute, Storage, Network hardware

Table 2 delineates the hierarchical nature of the MANO framework. The NFVO acts as the high-level brain for service logic, the VNF Manager acts as the dedicated administrator for specific software instances, and the VIM acts as the resource controller for the physical and virtualized hardware. For a network engineer, understanding these boundaries is essential for troubleshooting whether a service failure is due to a software logic error (NFVO/VNF Manager) or a resource exhaustion issue (VIM).

Modern Insight (2026): P4 and Programmable Data Planes

In modern architectures, the Rn (Virtual Routing and Switching) block in Figure 10.2 is increasingly implemented using P4 (Programming Protocol-independent Packet Processors). This allows for deep customization of how the virtualized infrastructure handles packets at wire speed, moving beyond the fixed-function limitations of early NFV implementations.

10.4 OPNFV: Open Platform for NFV

While ETSI serves as the de facto standards body, **OPNFV** was established as an open-source project to accelerate the practical evolution of NFV through an integrated, collaborative platform. It utilizes the ETSI architectural framework as its baseline to foster an ecosystem of interoperable open-source products.

- **OPNFV (Open Platform for NFV):** An open-source project under the Linux Foundation focused on creating a carrier-grade, integrated platform for NFV that bridges the gap between standards and implementation.
- **Virtualized Infrastructure Manager (VIM):** A functional block responsible for controlling and managing the NFVI compute, storage, and network resources, often implemented using platforms like OpenStack.

OPNFV is highly complementary to ETSI's efforts. While ETSI standardizes the "what," OPNFV focuses on the "how" by integrating existing open-source components into a reference platform. OPNFV includes an initial build of the NFVI and Virtual Infrastructure Manager (VIM) components. The diagram of their second release Brahmaputra is shown in Fig. 10.3.

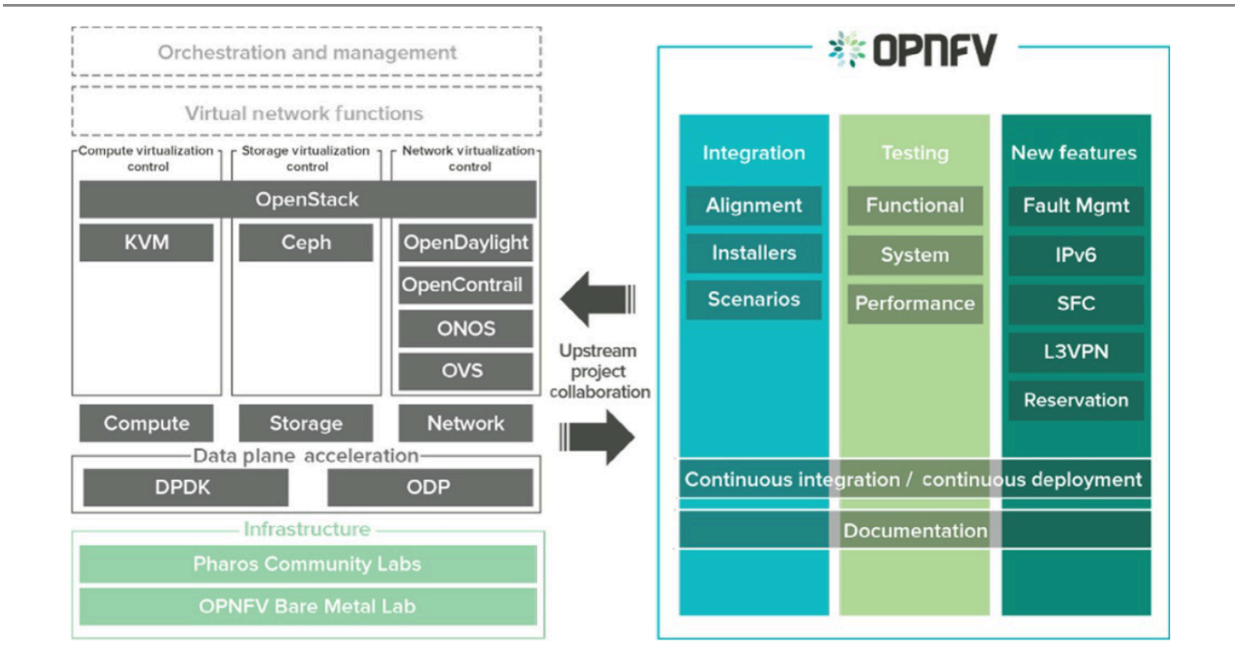


FIG. 10.3

OPNFV second release Brahmaputra.

FIGURE 10.3: OPNFV Second Release – Brahmaputra Architecture: The OPNFV architecture represents a layered stack designed for Continuous Integration/Continuous Deployment (CI/CD). The infrastructure layer consists of hardware resources managed through Pharos Community Labs and OPNFV Bare Metal Labs. Above this sits the Data Plane Acceleration layer, utilizing DPDK (Data Plane Development Kit) and ODP (Open Data Plane) to enhance packet processing speeds.

The control layer is dominated by **OpenStack**, which manages compute (via **KVM**), storage (via **Ceph**), and networking (via SDN controllers like **OpenDaylight**, **ONOS**, and **OVS**). The right side of Fig. 10.3 illustrates OPNFV's operational pillars: **Integration** (alignment and

installers), **Testing** (functional, system, and performance), and **New Features** (IPv6, Service Function Chaining (SFC), and L3VPN) .

10.5 Leading NFV Vendors

The NFV marketplace is diverse, with established vendors and specialized players offering solutions across different network sectors.

Table 3: Leading NFV Vendors

Company	Carrier NFV	Wireless Carriers	Enterprise and Data Centers	Products
Arista	Yes	Yes	Yes	Distribution, core, and access switches
Blue Coat	Yes	-	Yes	Forward and reverse proxy (ProxySG)
Brocade	Yes	Yes	Yes	Switches, firewalls, load balancers, WAN edge, application accelerators
Checkpoint	-	-	Yes	Edge/perimeter firewalls

Cisco	Yes	Yes	Yes	UCS fabric interconnect, firewalls, routers, load balancers, wireless LAN controllers, WAN optimization (VWAAS)
Citrix	-	-	Yes	Load balancer
Ericsson	Yes	Yes	-	Carrier aggregation
F5	Yes	-	-	Application accelerators
Fortinet	-	-	Yes	Edge/perimeter firewall
HP	Yes	-	Yes	[General NFV solutions]
Hitachi	Yes	-	Yes	Access switches
Huawei	Yes	Yes	-	Universal service routers
Juniper	Yes	-	Yes	Device role, edge/perimeter firewalls

Nokia	Yes	Yes	Yes	Edge gateway, virtual access switches, vCenter
Palo Alto Networks	Yes	-	Yes	Edge/perimeter firewall, compute firewall VGW
Radware	-	-	Yes	Load balancer/ADC
Riverbed	-	-	Yes	Load balancer/ADC, WAN optimization
VMware	Yes	Yes	Yes	Edge gateway, virtual access switches, vCenter

Table 3 illustrates that vendors often specialize in specific market segments. Large-scale infrastructure providers like Cisco, Ericsson, Huawei, and Nokia maintain a presence in the high-complexity Carrier NFV and Wireless sectors. In contrast, security-focused vendors like Checkpoint and Fortinet are primarily positioned in the Enterprise and Data Center segments. For a network architect, this data confirms that NFV is not a monolithic product but a modular ecosystem where different vendors provide specific VNFs (e.g., firewalls vs. carrier aggregation) that must be interoperable via the MANO framework.

Modern Insight (2026): The Anuket Project

As of 2026, the industry has largely transitioned from the standalone OPNFV project to the Anuket Project. Anuket merged OPNFV with the Cloud Infrastructure Telco Taskforce (CNTT) to provide a unified reference infrastructure and compliance framework for both Virtual Machines and Containers. This move further standardizes the "VIM" layer, reducing deployment friction across different

hardware vendors.

10.6 SDN vs. NFV: Synergies and Distinctions

While often discussed together, Software Defined Networking (SDN) and Network Functions Virtualization (NFV) target different operational challenges. Open SDN focuses on the separation of the **Control Plane** from the **Data Plane** to make the network programmable via standardized protocols like OpenFlow. In contrast, NFV is designed to reduce costs and time-to-market by deploying network functions as software on general-purpose hardware.

Functional Comparisons

- **Control and Management**: SDN provides the mechanism to control and manage any resource contributing to a service, including the instantiation of VNFs.
- **Forwarding Behavior**: SDN changes the forwarding behavior of the physical network, whereas NFV adds virtualized tunnels and functional layers (L4–L7) to the network.
- **Resource Perspective**: From the view of an SDN controller, a VNF is simply another programmable resource within the network topology.

Collaboration and Lifecycle Management

For effective synergy, both technologies must be aware of each other's state. This is achieved through a common repository or data store that maintains the state of services.

- **Orchestration Interaction**: The SDN controller communicates with the NFV MANO to "spin up" (instantiate) or "tear down" (terminate) VNFs based on service requirements.
- **State Awareness**: If a VNF is part of an active service chain, the MANO framework must not terminate it; this requires real-time state synchronization via an inventory database.

Collaboration in Practice:

- The SDN controller views a VNF as just another resource. ETSI's NFV architecture framework includes components that turn up VNFs and manage their lifecycle.
- Since SDN and NFV are complementary, network engineers need not choose between them, but rather understand how they might work together to address their needs.
- For this synergy to be effective, SDN and NFV need to know what the other is doing.
    1. The ETSI framework stipulates that there is a VNF manager that can turn up or turn down a VNF.
    2. If that VNF is part of an active service being provided to a customer, then the framework should not turn down the VNF.
- Therefore, to bring up or down VNFs, SDN should be able to communicate with the

NFV framework to bring up VNFs to support a provisioned service.

- The state of this service should be maintained so that the NFV framework and its MANO capabilities understand what is in use and when it can turn down a VNF.
- This mandates a common repository that SDN and NFV may access to understand the state of a service. Ideally, some type of repository holding state for an individual or set of VNFs should exist.

Fig. 10.4 depicts a scenario where SDN and NFV would collaborate. The controller would contact the NFV MANO to spin up or tear down VNFs and remain aware of the state of the VNF through a common data store.

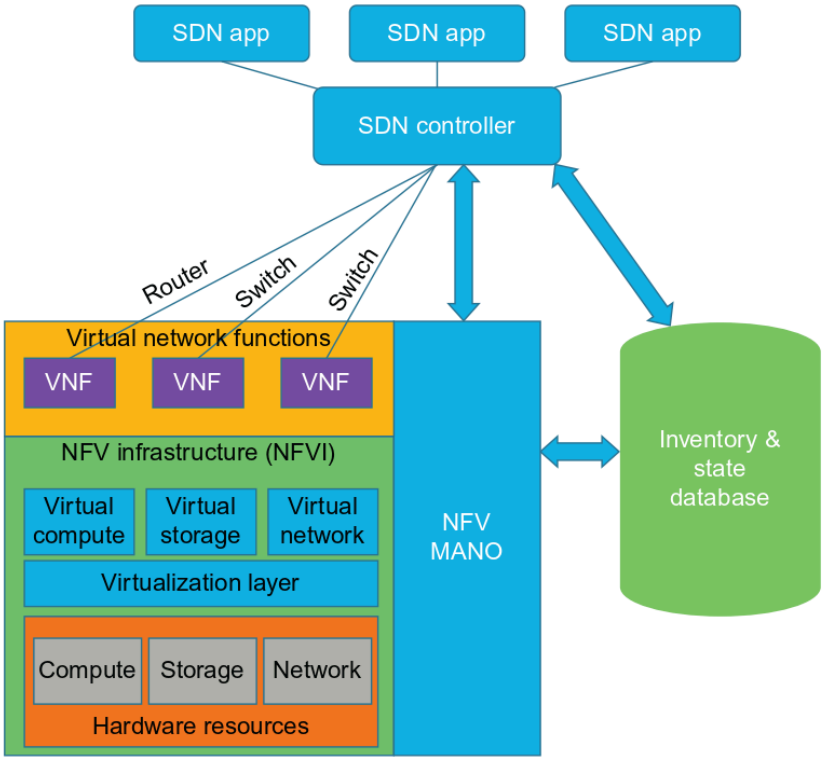


FIG. 10.4

SDN, NFV collaborating with VNFs.

FIGURE 10.4: SDN and NFV Collaboration Framework: Figure 10.4 illustrates the architectural integration of SDN and NFV. At the top, SDN Applications interface with the SDN Controller, which manages physical infrastructure (Routers and Switches) and interacts directly with the NFV MANO. The logic flow shows a bidirectional exchange between the SDN Controller and the Inventory & State Database, ensuring the controller is aware of the VNF statuses. Within the NFVI, the virtualization layer abstracts the Hardware Resources (Compute, Storage, Network) into Virtual Resources, which support the VNFs. This diagram demonstrates that SDN acts as the overarching control logic, while NFV provides the flexible, virtualized

functional components.

10.7 In-Line Network Functions

In-Line Network Functions are services that must inspect packets as they traverse the data plane. Traditionally, these are implemented as "bumps-in-the-wire"—physical appliances inserted directly into the traffic path. They include functionality such as load balancers, firewalls, IDS, and IPS. These services must be able to inspect the packets passing through them, and are often delivered in specialized appliances, which can be very expensive.

Types of In-Line Functions

- **Firewalls**: These functions admit or deny packets based on inspection criteria such as source/destination IP, TCP/UDP ports, or payload URLs.
    - **Intrusion Detection/Prevention Systems (IDS/IPS)**: Security systems that analyze packet content for malicious threats. An **IDS** monitors mirrored traffic in parallel, while an **IPS** sits in-line to actively block detected threats.
- **Load Balancers**: Devices that distribute incoming traffic across a set of servers to ensure even resource utilization.

The SDN/NFV Advantage

Deploying these as VNFs rather than physical appliances offers significant cost savings and flexibility. In an SDN paradigm, a hybrid system can shunt only specific packets requiring deep inspection to an out-of-band VNF, reducing latency for "clean" traffic.

For a network engineer, the move to SDN/NFV means that security and traffic management are no longer limited by physical cabling. Instead, traffic can be dynamically routed to inspection engines only when necessary, optimizing both performance and hardware utilization.

> **Modern Insight (2026): Service Function Chaining (SFC)**
>
> By 2026, the collaboration depicted in Figure 10.4 has matured into standardized Service Function Chaining (SFC) protocols (like NSH - Network Service Header). This allows the SDN controller to tag packets so they automatically traverse a specific sequence of VNFs (Firewall → IDS → Load Balancer) without needing complex re-routing at every hop.

10.7.1 SDN Applied to Server Load-Balancing

In a traditional networking environment, load balancers are specialized and costly hardware units. By utilizing SDN and the OpenFlow protocol, standard networking hardware can be programmed to perform these functions at a fraction of the cost.

- **Operational Logic**: The SDN controller has a global view of the network, allowing it to understand the load on each link and the status of various servers behind a firewall.
- **Matching Criteria**: OpenFlow 1.0 supports matching against a 12-tuple of header fields. This allows the controller to make forwarding decisions based on the source IP address, ensuring that a particular endpoint communicates with the same server for the duration of a transaction.
    - **Server Continuity**: A requirement where a user's transaction must be handled by the same server for its duration to maintain state across multiple packet exchanges.

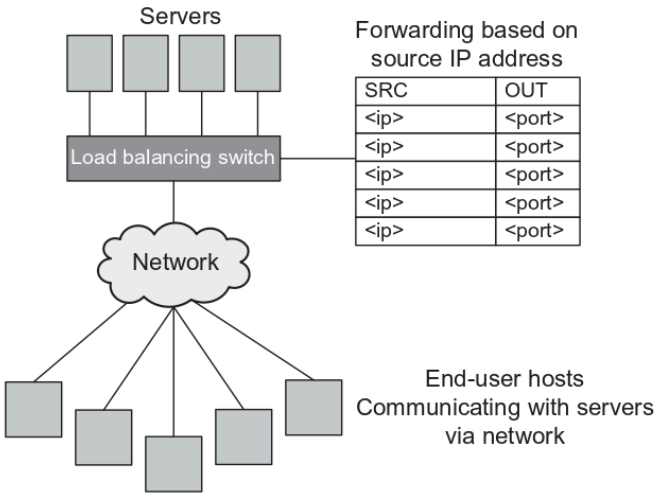


FIG. 10.5

Load balancers using OpenFlow.

FIGURE 10.5: Load Balancers using OpenFlow: It depicts a Load Balancing Switch acting as the intermediary between End-user hosts and a cluster of Servers. The logic flow starts with an incoming packet where the switch matches the source IP address (SRC) against its flow table. Based on the programmed rules (matching <ip> and <port>), the switch directs traffic to a specific output port (OUT) to reach the appropriate server. This illustrates how the controller's centralized knowledge enables load-balancing decisions based on broader criteria than a single appliance could observe.

Network vs. Application Server Load Balancing

It is important to distinguish between simple network load balancing and complex application-level balancing:

- **Network Load Balancing**: Relies on simple parameters like IP address and TCP port.
- **Application Server Load Balancing**: Requires more variables, such as server load and processing time. SDN applications can gather these external variables to assist the controller in routing transactions more effectively.

10.7.2 SDN Applied to Firewalls

SDN-based firewalls utilize OpenFlow match fields to make forwarding or drop decisions for incoming packets. Like load balancers, firewalls using SDN and OpenFlow have the ability to use any of the standard 12 match fields for making these forward or drop decisions. Packets matching certain protocol types (e.g., HTTPS, HTTP, Microsoft Exchange) are forwarded to the destination, and those that do not match are dropped. Simple firewalls that are ideally suited to an SDN-based solution include firewalls that are based on blocking or allowing specific IP addresses or specific TCP/UDP ports.

They are particularly effective for blocking or allowing specific IP addresses and TCP/UDP ports.

- **Traffic Shunting**: One SDN use case applied to firewalls is to insert an OpenFlow-enabled switch in front of a battery of firewalls. One benefit of this is that the SDN switch can load balance between the firewall devices, permitting a greater scale of firewall processing. More importantly, the SDN switch can shunt known- safe traffic around the firewalls. This is possible since OpenFlow rules can be programmed such that we offload traffic from the firewalls that are known to be problem-free. The benefit of this is that less firewall equipment and/or power is needed, and less traffic is subjected to the latency imposed by firewall processing.
- **Network Slicing**: The process of partitioning a single physical network into multiple virtual, isolated networks managed by different controllers for each network slice. It provides another example of an SDN firewall-type implementation that slices the network into different departments.
- **FlowVisor**: A specific implementation of an OpenFlow controller that enforces isolation between different networks on the same campus. This allows the FlowVisor OpenFlow controller to enforce isolation between different networks on the same campus.

Table 4: Comparison of SDN-Based In-Line Functions

Function	OpenFlow Implementation	Key Benefit
Server Load Balancer	Matches 12-tuple header fields to distribute traffic.	Cost reduction; uses commodity hardware.
Firewall	Uses flow rules to permit or drop specific IP/Port traffic.	Shunting safe traffic to reduce latency.
Network Slicing	Partitions network resources using tools like FlowVisor.	Enforces isolation on shared hardware.

Table 4 summarizes how SDN transforms traditional in-line functions. For a network engineer, the shift from proprietary hardware to OpenFlow-based logic means that security and performance can be optimized programmatically. By shunting traffic or slicing the network, engineers can scale services without a linear increase in specialized hardware costs.

> **Modern Insight (2026): Zero Trust and Micro-segmentation**
>
> In 2026, the SDN firewall concepts described in Section 10.7.2 have evolved into Zero Trust Architecture (ZTA). While early SDN firewalls focused on campus-wide isolation via FlowVisor, modern systems use Micro-segmentation to apply security policies to individual virtualized workloads, ensuring that even internal "east-west" traffic is continuously validated.

10.7.3 SDN Applied to Intrusion Detection

Intrusion Detection Systems (IDS) have traditionally functioned as passive security devices that monitor all ingress and egress traffic to identify suspicious patterns or application protocol violations. In the SDN paradigm, an application can be designed to support a VNF that acts as a virtualized network tap.

An IDS looks for application protocol violations and uses rules that analyze statistics and traffic patterns that may indicate an attack. It means just as in the case of a firewall, an SDN application can be created to support a VNF that would act as a traditional network tap and

passively route packets to the application for analysis. However, there is some concern about whether the performance of VMs in this role can compete with customized hardware optimized for this task. It is noteworthy that Cisco has commercialized virtual IPSs and IDSs.

- **Deep Packet Inspection (DPI)**: A process-intensive analysis of the packet payload—searching for signs of viruses, trojans, and malware—that must occur outside the primary data plane to avoid performance degradation.
- **Virtual Taps**: A software-based mechanism that copies and forwards specific traffic to an analysis system, replacing physical hardware taps or patch panels.

The OpenFlow Advantage for IDS

Physical taps and pre-SDN solutions (like SPAN ports) are often port-based or VLAN-based and can be prohibitively expensive in large networks. OpenFlow is naturally suited for this role because it allows for flow entries based on diverse criteria:

1. **Selective Mirroring**: A tap system using OpenFlow can be programmed to match specific criteria—such as device address, VLAN, or ingress port—and forward those packets to the IP address of an IDS.
2. **Scalable IPS Batteries**: Similar to firewalls, an SDN switch can act as a front-end for a battery of Intrusion Prevention System (IPS) appliances, shunting safe traffic and load-balancing the remainder across multiple units in parallel.
3. **FlowScale Project**: This open-source initiative demonstrates front-end load balancing by sending mirrored traffic to a battery of IDS hosts.

10.8 Conclusion

The integration of NFV and SDN represents a fundamental shift for data centers and telecommunications carriers. By virtualizing the network, organizations can achieve dramatic cost reductions through lower labor requirements and automated, policy-enforced configurations.

Under ETSI's leadership, vendor interoperability is increasing, and the embrace of open-source projects like **OPNFV** is expected to further drive down costs. Given these economic and operational advantages, the adoption of NFV is projected to accelerate significantly.

Table 5: Summary of SDN-Enhanced In-Line Functions

Function	Technical Logic for Virtualization	Practical Implication
Server Load Balancing	Matches 12-tuple headers to maintain server state/continuity.	Replaces expensive hardware with commodity switches.
Firewalling	Uses flow rules to shunt known-safe traffic or block specific ports.	Reduces latency and equipment overhead.
Intrusion Detection	Programmatically creates virtual taps to mirror specific flows.	Eliminates the need for costly physical patch-panels.

The data in Table 5 illustrates the transition from a hardware-dependent "bump-in-the-wire" model to a programmable, software-defined ecosystem. For a network architect, the logic dictates that network functions are no longer static physical nodes but dynamic, scalable services. The relationship between the SDN controller (the brain) and the VNF (the function) allows for an optimized traffic path where security and performance are managed via centralized policy rather than manual cabling.

Modern Insight (2026): Moving to Cloud-Native Functions (CNFs)

While the focus is on virtual machines (VNFs), the 2026 industry standard is shifting toward Cloud-Native Network Functions (CNFs). CNFs run in lightweight containers (e.g., Kubernetes), allowing for sub-second "spin-up" times compared to the minutes required for traditional VMs. This agility is critical for implementing Zero Trust security, where micro-segmentation rules must change in real-time.
