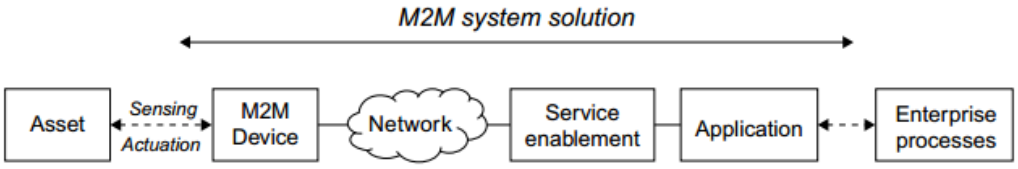
Lecture 1

Machine to Machine Communication

M2M refers to those solutions that allow communication between devices of the same type and a specific application, all via wired or wireless communication networks. M2M solutions allow end-users to capture data about events from assets, such as temperature or inventory levels.



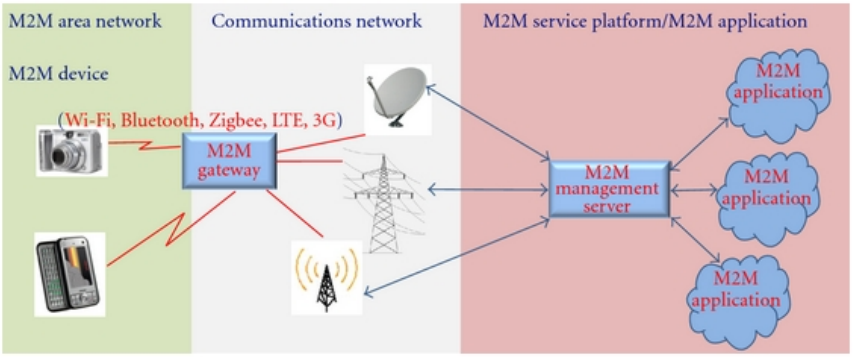
Main features of M2M communications are:

- It is a general concept involving an autonomous device communicating directly to another autonomous device. Autonomous refers to the ability of the node to instantiate and communicate information with another node without human intervention.
- The form of communication is left open to the application. It may very well be the case that an M2M device uses no inherent services or topologies for communication.
- An M2M system may communicate over non-IP based channels as well, such as a serial port or custom protocol.

M2M System Overview

The M2M system solution is used to remotely monitor and control enterprise assets of various kinds, and to integrate those assets into the business processes of the

enterprise in question. The asset can be of a wide range of types (e.g. vehicle, freight container, building, or smart electricity meter), all depending on the enterprise.



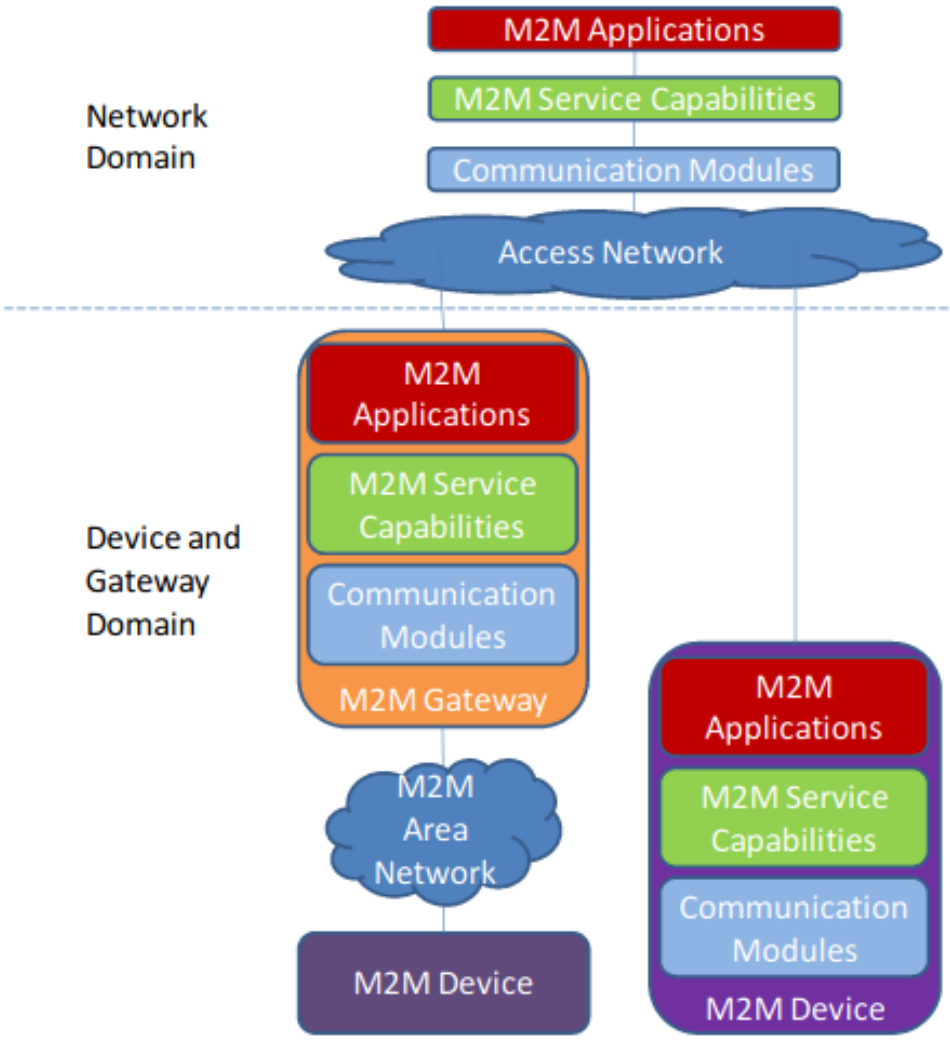
The system components of an M2M solution are as follows:

- **M2M Device**. This is the M2M device attached to the asset of interest, and provides sensing and actuation capabilities. The M2M device is here generalized, as there are a number of different realizations of these devices, ranging from low-end sensor nodes to high-end complex devices with multimodal sensing capabilities.

- **Network**. The purpose of the network is to provide remote connectivity between the M2M device and the application-side servers. Many different network types can be used, and include both Wide Area Networks (WANs) and Local Area Networks (LANs), sometimes also referred to as Capillary Networks or M2M Area Networks

- **M2M Service Enablement**. This component provides generic functionality that is common across a number of different applications. Its primary purpose is to reduce cost for implementation and ease of application development.

- **M2M Application**. The application component of the solution is a realization of the highly specific monitor and control process. The application is



[h]

further integrated into the overall business process system of the enterprise.

M2M Architecture

Machine-to-Machine (M2M) refers to networking of machines (or devices) for the purpose of remote monitoring and control and data exchange. Figure shows the end-to-end architecture for M2M system comprising of M2M area networks, communication network and application domain.

- An M2M area network comprises of machines (or M2M nodes) which have embedded hardware modules for sensing, actuation and communication.
- Various communication protocols can be used for M2M local area networks such as ZigBee, Bluetooh, ModBus, M-Bus, Wirless M-Bus, Power Line Communication (PLC), 6LoWPAN, IEEE 802.15.4, etc.
- These communication protocols provide connectivity between M2M nodes within an M2M area network. The communication network provides connectivity to remote M2M area networks.
- Since non-IP based protocols are used within M2M area networks, the M2M nodes within one network cannot communicate with nodes in an external network. To enable the communication between remote M2M area networks, M2M gateways are used.
- The communication between the M2M nodes and the M2M gateway is based on the communication protocols which are native to the M2M area network.
- M2M gateway performs protocol translations to enable IP-connectivity for M2M area networks. M2M gateway acts as a proxy performing translations from/to native protocols to/from Internet Protocol (IP).
- The M2M data is gathered into point solutions such as enterprise applications, service management applications, or remote monitoring applications.
- M2M has various application domains such as smart metering, home automation, industrial automation, smart grids, etc. M2M solution designs (such as data collection and storage architectures and applications) are specific to the M2M application domain.

Lecture 2

Comparision of IoT and M2M

Difference between IoT and M2M

In many respects, it can initially look the same as M2M communication connecting sensors and other devices to Information and Communication Technology (ICT) systems via wired or wireless networks. IoT systems may incorporate some M2M nodes (such as a Bluetooth mesh using non-IP communication), but aggregates data at an edge router or gateway. An edge appliance like a gateway or router serves as the entry point onto the internet. In the longer term, it is envisaged that an IoT ecosystem will emerge not dissimilar to today?s Internet, allowing things and real world objects to connect, communicate, and interact with one another in the same way humans do via the web today.

Therefore, IoT is significantly different from M2M. Few important difference are:

- **Communication Protocols:** M2M and IoT can differ in how the communication between the machines or devices happens. M2M uses either proprietary or non-IP based communication protocols for communication within the M2M area networks. Commonly uses M2M protocols include ZigBee, Bluetooh, ModBus, M-Bus, Wirless M-Bus, Power Line Communication (PLC), 6LoWPAN, IEEE 802.15.4, Z-Wave, etc. The focus of communication in M2M is usually on the protocols below the network layer. The focus of communication in IoT is usually on the protocols above the network layer such as HTTP, CoAP, WebSockets, MQTT, XMPP etc.

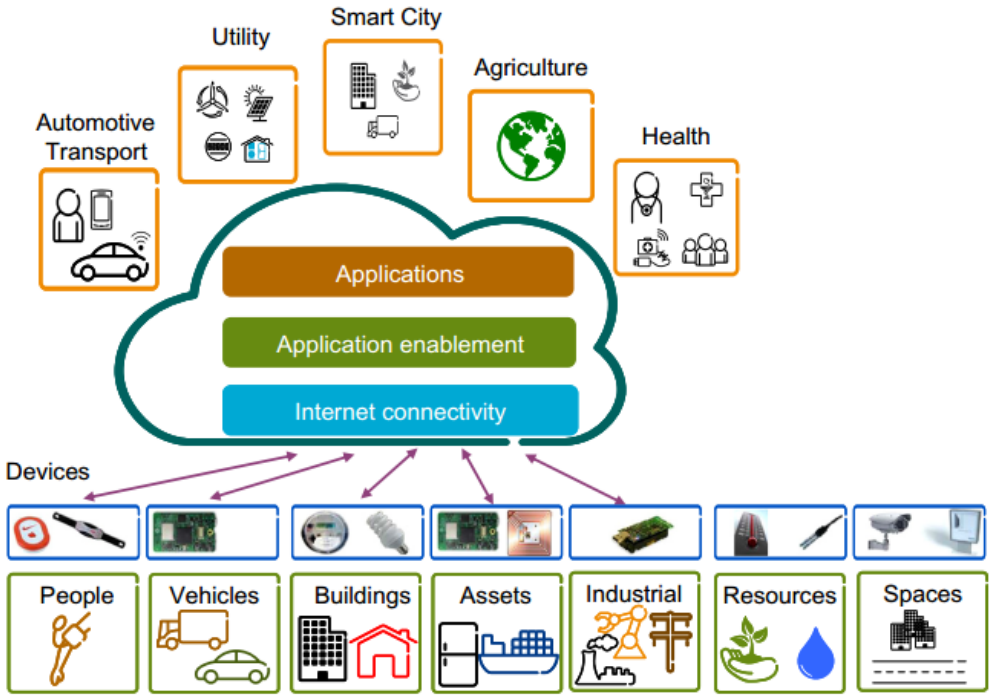
- **Machines in M2M vs Things in IoT** The “Things” in IoT refers to physical objects that have unique identifiers and can sense and communicate with their

external environment (and user applications) or their internal physical states. The unique identifiers for the things in IoT are the IP addresses (or MAC addresses). Things have software components for accessing, processing, and storing sensor information, or controlling actuators connected. IoT systems can have heterogeneous things, M2M systems, in contrast to IoT, typically have homogeneous machine types within an M2M area network.

- **Hardware vs Software Emphasis** While the emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software. loT devices run specialized software for sensor data collection, data analysis and interfacing with the cloud through IP-based communication.

- **Data Collection & Analysis** M2M data is collected in point solutions and often in on-premises storage infrastructure. In contrast to M2M, the data in IoT is collected in the cloud (can be public, private or hybrid cloud). The analytics component analyzes the data and stores the results in the cloud database. The IoT data and analysis results are visualized with the cloud-based applications. The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes. Observer nodes can process information and use it for various applications, however, observer nodes do not perform any control functions.

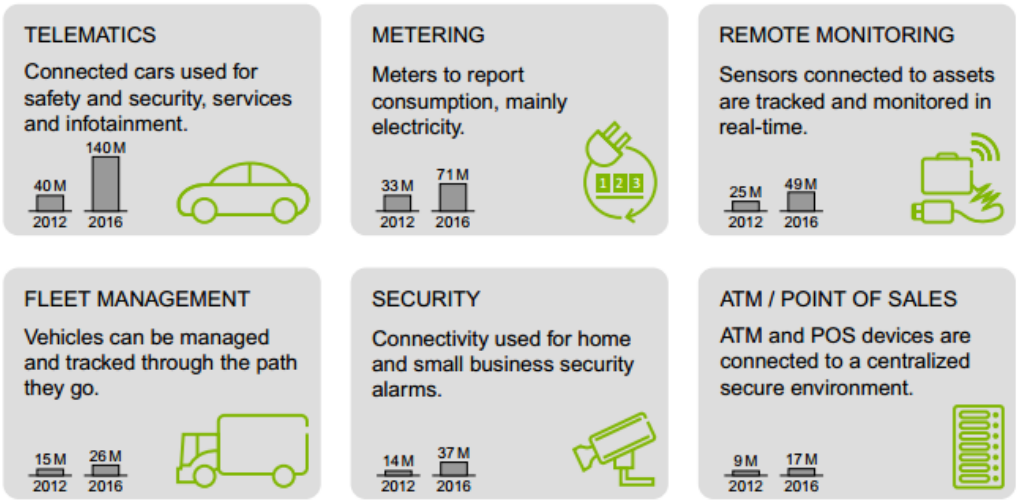
- **Applications:** M2M data is collected in point solutions and can be accessed by on-premises applications such as diagnosis applications, service management applications, and on-premisis enterprise applications. IoT data is collected in the cloud and can be accessed by cloud applications such as analytics applications, enterprise applications, remote diagnosis and management applications, etc. Since the scale of data collected in IoT is so massive, cloud-based real-time and batch data analysis frameworks are used for data analys



[h]

Application of M2M

- *Telematics*: Typical applications include navigation, remote vehicle diagnostics,pay-as-you-drive insurance schemes, road charging, and stolen vehicle recovery.
- *Metering applications*, meanwhile, include primarily remote meter management and data collection for energy consumption in the electricity utility sector, but also for gas and water consumption.
- *Remote monitoring* is more generalized monitoring of assets, and includes remote patient monitoring as one prime example.
- *Fleet management* includes a number of different applications, like data logging, goods and vehicle positioning, and security of valuable or hazardous goods.
- *Security applications* are mainly those related to home alarms and small busi-



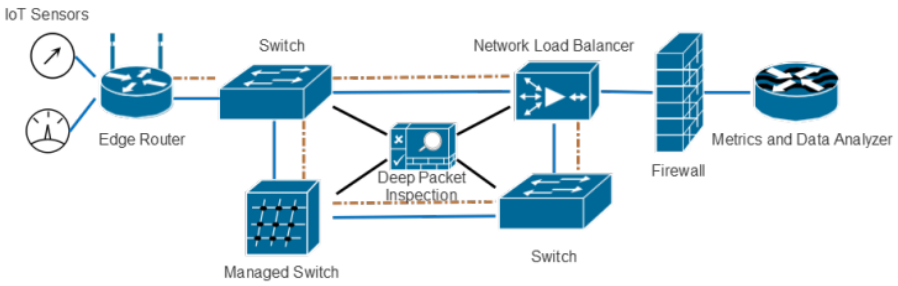
[h]

ness surveillance solutions. The final market segment is **Automated Teller Machines** (ATM) and **Point of Sales** (POS) terminals.

Lecture 3

Software Defined Network

In traditional network configuration, the data plane and control plane are unified. When the system needs to add or remove another node or set up a new data path, many of the dedicated systems need to be updated with new VLAN settings, QoS parameters, access control lists, static routes, and firewall pass-throughs.



Limitations of the conventional network architectures can be given in following points:

- **Complex Network Devices**: Conventional networks are getting increasingly complex with more and more protocols being implemented to improve link speeds and reliability. Interoperability is limited due to the lack of standard and open interfaces. The conventional networks were well suited for static traffic patterns and had a large number of protocols designed for specific applications. For IoT applications which are deployed in cloud computing environments, the traffic patterns are more dynamic.

- **Management Overhead**: Network managers find it increasingly difficult to manage multiple network devices and interfaces from multiple vendors. Upgradation of network requires configuration changes in multiple devices (switches, routers, firewalls, etc.)

- **Limited Scalability**: The virtualization technologies used in cloud computing environments has increased the number of virtual hosts requiring network access. The analytics components of IoT applications run distributed algorithms on a large number of virtual machines that require huge amounts of data exchange between virtual machines. Such computing environments require highly scalable and easy to manage network architectures with minimal manual configurations, which is becoming increasingly difficult with conventional networks.

SDN Defintion

Software-Defined Networking (SDN) is an emerging paradigm that promises to make complex IP network more manageable and flexible by breaking vertical integration(the control and data planes are bundled together), separating the network?s control logic from the underlying routers and switches, promoting (logical) centralization of network control, and introducing the ability to program the network.

The separation of concerns introduced between the definition of network policies, their implementation in switching hardware, and the forwarding of traffic, is key to the desired flexibility. By breaking the network control problem into tractable pieces, SDN makes it easier to create and introduce new abstractions in networking, simplifying network management and facilitating network evolution.

The important characteristics of SDN network are:

- The control plane is decoupled from the data plane. Data plane hardware becomes simple packet-forwarding devices.

- All forwarding decisions are flow-based rather than destination-based. A flow is a set of packets that match a criterion or filter. All packets in a flow are treated with the same forwarding and service policies. Flow programming

allows for easy scaling and flexibility with virtual switches, firewalls, and middleware.

- The control logic is also known as the SDN Controller. This software version of legacy hardware is capable of running on commodity hardware and cloud-based instances. Its purpose is to command and govern the simplified switching nodes.

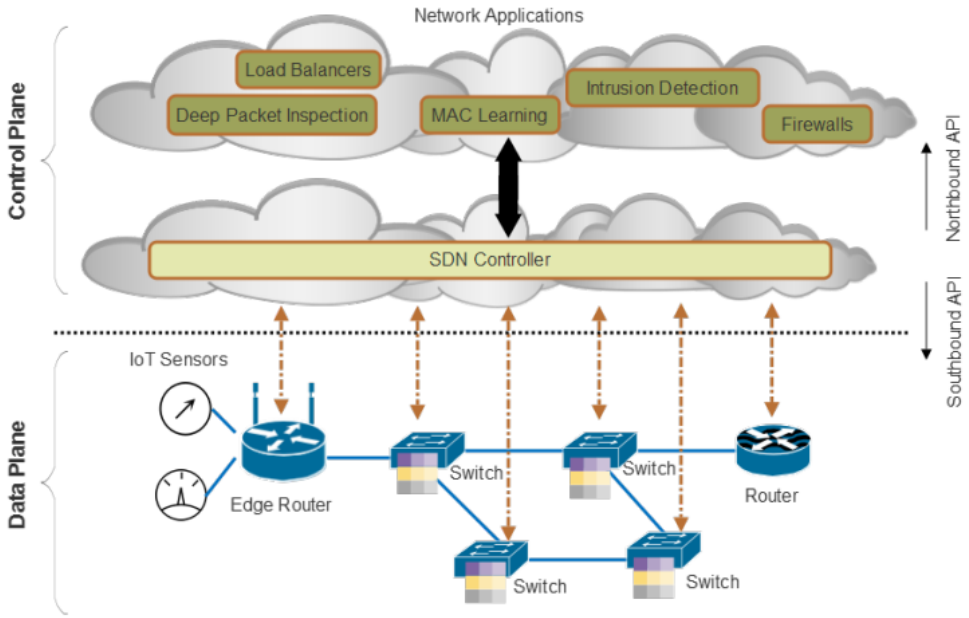
- The control logic is also known as the SDN Controller. This software version of legacy hardware is capable of running on commodity hardware and cloud-based instances. Its purpose is to command and govern the simplified switching nodes.

- Network application software can reside over the SDN controller through a northbound interface. This software can interact and manipulate the data plane with services such as deep packet inspection, firewalls, and load balancers.

The key elements of SDN network are:

- **Centralized Network Controller**: With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network, SDN applications can be deployed through programmable open APIs. This speeds up innovation as the network administrators no longer need to wait for the device vendors to embed new features in their proprietary hardware.

- **Programmable Open APIs**: SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface). With these open APIs various network services can be implemented, such as routing, quality of service (QoS), access control, etc.



- **Standard Communication Interface (OpenFlow)**: SDN architecture uses a standard communication interface between the control and infrastructure layers (Southbound interface). OpenFlow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocol for the Southbound interface. With OpenFlow, the forwarding plane of the network devices can be directly accessed and manipulated.

Advantages of SDN

SDN has many important benefits over traditional network when it comes to traffic nature of IoT.

- **Service chaining**: This allows a customer or provider to sell services a la carte. Cloud network services such as firewalls, deep packet inspection, VPNs, authentication services, and policy brokers can be linked and used on a subscription basis. Some customers may want a full set of features, others may not choose any or may change their configuration routinely. Service chaining

allows for significant flexibility in deployments.

- **Dynamic load management**: An SDN enjoys the flexibility of cloud architecture, and by design it can scale resources dynamically depending on load. This type of flexibility is crucial for the IoT as architects need to plan for capacity and scale as the number of things grows exponentially. Only virtual networking in the cloud provides the ability to scale capacity when needed.

- **Bandwidth calendaring**: This allows an operator to partition data bandwidth and usage to specified times and days. This is pertinent to IoT as many edge sensors only report data periodically or at a certain time of day. Sophisticated bandwidth sharing algorithms can be constructed to time slice capacity.

Lecture 4

SDN Architecture

Elements of SDN Architecture

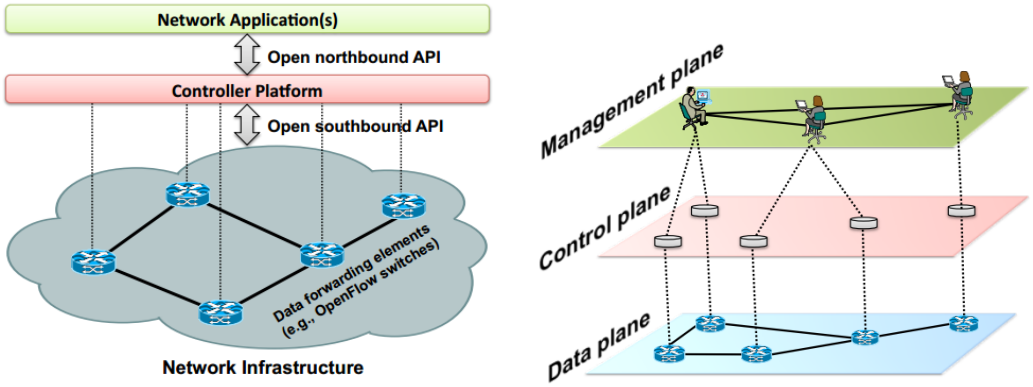
1. **Forwarding Devices (FD):** Hardware- or software-based data plane devices that perform a set of elementary operations. The forwarding devices have well-defined instruction sets (e.g., flow rules) used to take actions on the incoming packets (e.g., forward to specific ports, drop, forward to the controller, rewrite some header). These instructions are defined by southbound interfaces (e.g., OpenFlow, ForCES) and are installed in the forwarding devices by the SDN controllers implementing the southbound protocols.

2. **Data Plane (DP)** Forwarding devices are interconnected through wireless radio channels or wired cables. The network infrastructure comprises the interconnected forwarding devices, which represent the data plane.

3. **Southbound Interface (SI):** The instruction set of the forwarding devices is defined by the southbound API, which is part of the southbound interface. Furthermore, the SI also defines the communication protocol between forwarding devices and control plane elements. This protocol formalizes the way the control and data plane elements interact.

4. **Control Plane (CP):** Forwarding devices are programmed by control plane elements through well-defined SI embodiments. The control plane can therefore be seen as the ?network brain?. All control logic rests in the applications and controllers, which form the control plane.

5. **Northbound Interface (NI):** The network operating system can offer an API to application developers. This API represents a northbound interface,

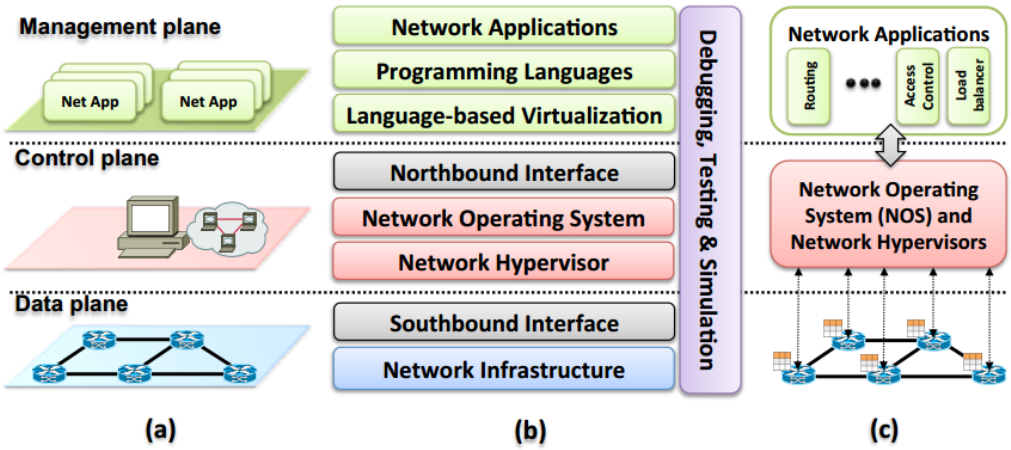


i.e., a common interface for developing applications. Typically, a northbound interface abstracts the low level instruction sets used by southbound interfaces to program forwarding devices

6. **Management Plane (MP):** The management plane is the set of applications that leverage the functions offered by the NI to implement network control and operation logic. This includes applications such as routing, firewalls, load balancers, monitoring, and so forth. Essentially, a management application defines the policies, which are ultimately translated to southbound-specific instructions that program the behavior of the forwarding devices

SDN Layered Architecture

An SDN architecture can be depicted as a composition of different layers. While some of them are always present in an SDN deployment, such as the southbound API, network operating systems, northbound API and network applications, others may be present only in particular deployments, such as hypervisor- or language-based virtualization.



Layer I: Infrastructure

An SDN infrastructure, similarly to a traditional network, is composed of a set of networking equipment (switches, routers and middlebox appliances). The difference is that those traditional physical devices are now simple forwarding elements without embedded control or software to take autonomous decisions.Configuration and communication compatibility and interoperability among different data and control plane devices are ensured by standard open interfece like OpenFlow.

Layer II: Southbound Interfaces

Southbound interfaces (or southbound APIs) are the connecting bridges between control and forwarding elements, thus being the crucial instrument for clearly separating control and data plane functionality. As a central component of its design the southbound APIs represent one of the major barriers for the introduction and acceptance of any new networking technology therefore standard SDN southbound API like OpenFlow Emerges .

The *OpenFlow* protocol provides three information sources for network operating systems. First, event-based messages are sent by forwarding devices to the controller when a link or port change is triggered. Second, flow statistics are generated by the

forwarding devices and collected by the controller. Third, packet-in messages are sent by forwarding devices to the controller when they do not known what to do with a new incoming flow or because there is an explicit "send to controller? action in the matched entry of the flow table.

Layer III: Network Hypervisors

Hypervisors enable distinct virtual machines to share the same hardware resources. Virtual machines can be easily migrated from one physical server to another and can be created and/or destroyed on-demand, enabling the provisioning of elastic services with flexible and easy management. Item Unfortunately, virtualization has been only partially realized in practice.

Layer IV: Network Operating Systems / Controllers

Networks have so far been managed and configured using lower level, device-specific instruction sets and mostly closed proprietary network operating systems (e.g., Cisco IOS and Juniper JunOS) SDN is promised to facilitate network management and ease the burden of solving networking problems by means of the logically-centralized control offered by a network operating system (NOS) The crucial value of a NOS is to provide abstractions, essential services, and common application programming interfaces (APIs) to developers. Generic functionality as network state and network topology information, device discovery, and distribution of network configuration can be provided as services of the NOS. With NOSs, to define network policies a developer no longer needs to care about the low-level details of data distribution among routing elements, for instance

Layer V: Northbound Interfaces

The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs. Open and standard northbound interfaces are crucial to promote application portability and interoperability among the different the control platforms. A northbound API can be compared to the POSIX standard in operating systems, representing an abstraction that guarantees programming language and controller independence.

Layer VI: Language-based Virtualization

Two essential characteristics of virtualization solutions are the capability of expressing modularity and of allowing different levels of abstractions while still guaranteeing desired properties such as protection Pyretic is example of such language that offers this type of high-level abstraction of network topology. It incorporates this concept of abstraction by introducing network objects. These objects consist of an abstract network topology and the sets of policies applied to it. Network objects simultaneously hide information and offer the required services.

Layer VII: Programming languages

In SDNs, high-level programming languages can be designed and used to create higher level abstractions for simplifying the task of programming forwarding devices; It enable more productive and problem-focused environments for network software programmers, speeding up development and innovation; It also promote software modularization and code reusability in the network control plane and foster the development of network virtualization.

Layer VIII: Network Applications

Network applications can be seen as the ?network brains?. They implement the control-logic that will be translated into commands to be installed in the data plane, dictating the behavior of the forwarding devices. Software-defined networks can be deployed on any traditional network environment, from home and enterprise networks to data centers and Internet exchange points. Such variety of environments has led to a wide array of network applications. Existing network applications perform traditional functionality such as routing, load balancing, and security policy enforcement, but also explore novel approaches, such as reducing power consumption. Other examples include fail-over and reliability functionalities to the data plane, end-to-end QoS enforcement, network virtualization, mobility management in wireless networks, among many others. The variety of network applications, combined with real use case deployments, is expected to be one of the major forces on fostering a broad adoption of SDN

Lecture 5

Network Function Virtualization

Motivation for NFV

NFV originated from discussions among major network operators and carriers about how to improve network operations in the high-volume multimedia era. These discussions resulted in the publication of the original NFV white paper, Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action. In this white paper, the group listed as 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.

The white paper highlights that the source of the need for this new approach is that networks include a large and growing variety of proprietary hardware appliances, leading to the following negative consequences:

- New network services may require additional different types of hardware appliances, and finding the space and power to accommodate these boxes is becoming increasingly difficult.
- New hardware means additional capital expenditures.
- Once new types of hardware appliances are acquired, operators are faced with the rarity of skills necessary to design, integrate, and operate increasingly complex hardware-based appliances.
- Hardware-based appliances rapidly reach end of life, requiring much of the procure-design-integrate-deploy cycle to be repeated with little or no revenue benefit.

- As technology and services innovation accelerates to meet the demands of an increasingly network-centric IT environment, the need for an increasing variety of hardware platforms inhibits the introduction of new revenue-earning network services.

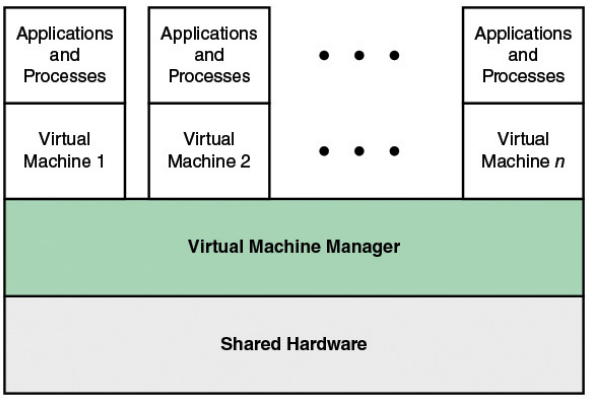
The NFV approach moves away from dependence on a variety of hardware platforms to the use of a small number of standardized platform types, with virtualization techniques used to provide the needed network functionality.

Virtualization

Virtualization encompasses a variety of technologies for managing computing resources, providing a software translation layer, known as an abstraction layer, between the software and the physical hardware. Virtualization turns physical resources into logical, or virtual, resources. Virtualization enables users, applications, and management software operating above the abstraction layer to manage and use resources without needing to be aware of the physical details of the underlying resources.

Virtualization technology enables a single PC or server to simultaneously run multiple operating systems or multiple sessions of a single OS. A machine running virtualization software can host numerous applications, including those that run on different operating systems, on a single hardware platform. The solution that enables virtualization is a virtual machine monitor (VMM), or commonly known today as a hypervisor. This software sits between the hardware and the VMs acting as a resource broker. Simply put, the hypervisor allows multiple VMs to safely coexist on a single physical server host and share that host?s resources.

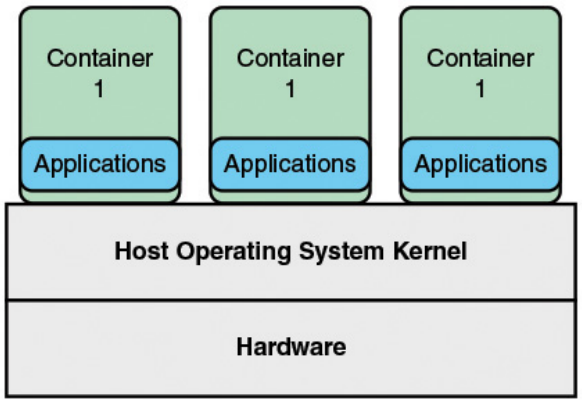
A key aspect of server virtualization is that, in addition to the capability of running multiple VMs on one machine, VMs can be viewed as network resources. Server virtualization masks server resources, including the number and identity of



individual physical servers, processors, and operating systems, from server users. This makes it possible to partition a single host into multiple independent servers, conserving hardware resources. It also makes it possible to quickly migrate a server from one machine to another for load balancing or for dynamic switchover in the case of machine failure. Server virtualization has become a central element in dealing with big data applications and in implementing cloud computing infrastructures.

Container Virtualization

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.



NFV Concept

Network Function Virtualization is defined as the virtualization of network functions by implementing these functions in software and running them on VMs. NFV decouples network functions, such as Network Address Translation (NAT), firewalling, intrusion detection, Domain Name Service (DNS), and caching, from proprietary hardware appliances so that they can run in software on VMs.NFV builds on standard VM technologies, extending their use into the networking domain.

The network-based devices, include the following:

- **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.

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.

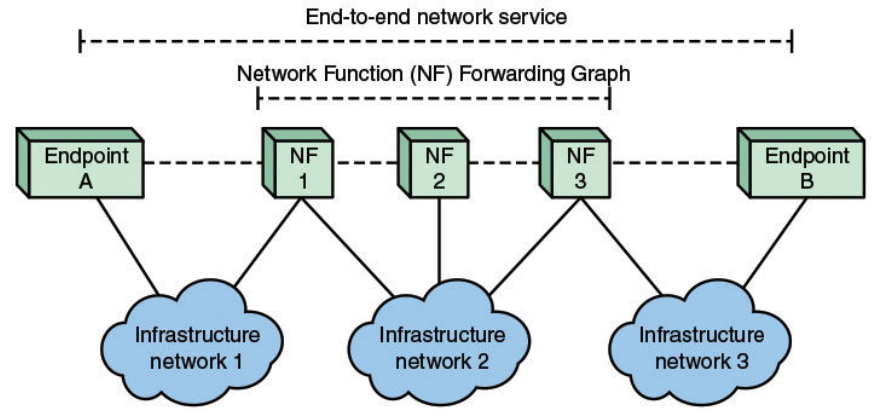
Figure 1: Vision for Network Functions Visualization

By broad consensus, the Network Functions Virtualization Industry Standards Group (ISG NFV), created as part of the European Telecommunications Standards Institute (ETSI), has the lead and indeed almost the sole role in creating NFV standards.ISG NFV was established in 2012 by seven major telecommunications network operators.

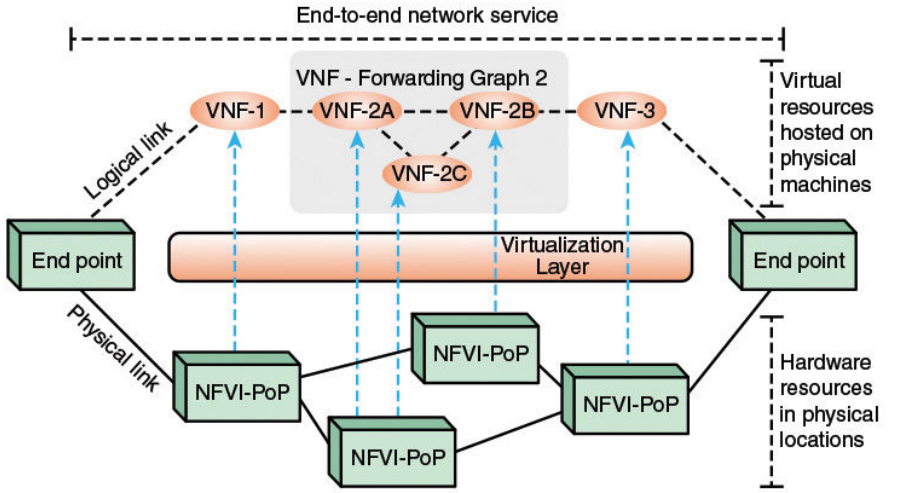
Simple Example of the Use of NFV

This section considers a simple example from the NFV Architectural Framework document. Part a of Figure 7.6 shows a physical realization of a network service. At a top level, the network service consists of endpoints connected by a forwarding graph of network functional blocks, called network functions (NFs). Examples of NFs are firewalls, load balancers, and wireless network access points. In the Architectural Framework, NFs are viewed as distinct physical nodes. The endpoints are beyond the scope of the NFV specifications and include all customer-owned devices. So, in the figure, endpoint A could be a smartphone and endpoint B a content delivery network (CDN) server.

The interconnections among the NFs and endpoints are depicted by dashed lines, representing logical links. These logical links are supported by physical paths through infrastructure networks (wired or wireless).



(a) Graph representation of an end-to-end network service



(b) Example of an end-to-end network service with VNFs and nested forwarding graphs

Part b of Figure illustrates a virtualized network service configuration that could be implemented on the physical configuration of part a of Figure VNF-1 provides network access for endpoint A, and VNF-2 provides network access for B. The figure also depicts the case of a nested VNF forwarding graph (VNF-FG-2) constructed from other VNFs (that is, VNF-2A, VNF-2B and VNF-2C). All of these VNFs run as VMs on physical machines, called points of presence (PoPs). This configuration illustrates several important points. First, VNF-FG-2 consists of three VNFs even though ultimately all the traffic transiting VNF-FG-2 is between VNF-1 and VNF-3.

The reason for this is that three separate and distinct network functions are being performed. For example, it may be that some traffic flows need to be subjected to a traffic policing or shaping function, which could be performed by VNF-2C. So, some flows would be routed through VNF-2C, while others would bypass this network function.

A second observation is that two of the VMs in VNF-FG-2 are hosted on the same physical machine. Because these two VMs perform different functions, they need to be distinct at the virtual resource level but can be supported by the same physical machine. But this is not required, and a network management function may at some point decide to migrate one of the VMs to another physical machine, for reasons of performance. This movement is transparent at the virtual resource level.

NFV Principle

The VNFs are the building blocks used to create end-to-end network services. Three key NFV principles are involved in creating practical network services:

- **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 func-

tionality. Each VNFC may be deployed in one or multiple instances. These instances may be deployed on separate, distributed hosts to provide scalability and redundancy.

NFV Advantages

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.
- 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.
- The ability to innovate and roll out services quickly and 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.

NFV Requirements

NFV must be designed and implemented to meet a number of requirements and technical challenges

- **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 Inter-

faces (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.

Lecture 6

NFV Architecture

NFV Terminology

- **Network Function:** 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 Services:** A composition of network functions that is defined by its functional and behavioral specification.
- **Network Function Virtualization (NFV):** The principle of separating network functions from the hard- ware they run on by using virtual hardware abstraction.
- **Network Function 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 execu- tion environment for VNFs.
- **Physical Network Function (PNF):** An implementation of a NF via a tightly coupled software and hardware system. This is typically a proprietary system.

- **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.
- **Virtual Network Function (VNF):** An implementation of an NF that can be deployed on an NFVI.
- **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.
- **VNF Forwarding Path:** Graph of logical links connecting VNF nodes for the pur- pose of describing traffic flow between these network func- tions.

High-Level NFV Framework

This framework supports the implementation of network functions as software-only VNFs.

The NFV framework consists of three domains of operation:

- **Virtualized network functions**: The collection of VNFs, implemented in software, that run over the NFVI.
- **NFV infrastructure (NFVI)**: The NFVI performs a virtualization function on the three main categories of devices in the network service environment: computer devices, storage devices, and network devices.

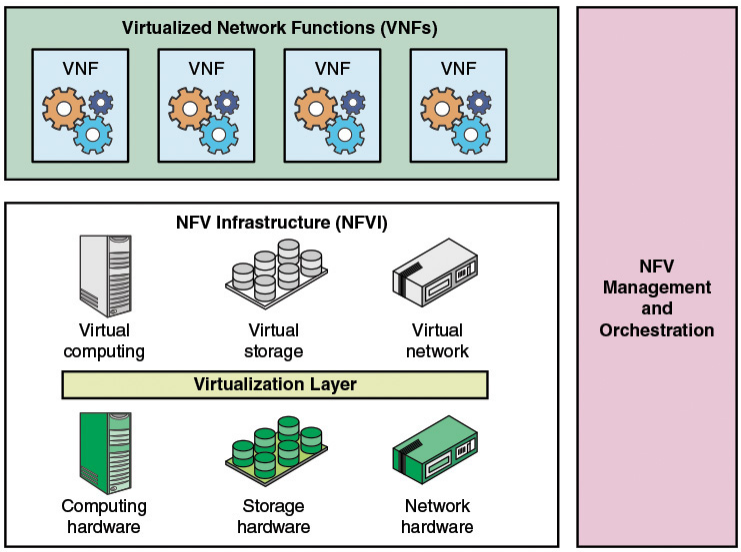


Figure 2: High-Level NFV Framework

- **NFV management and orchestration**: Encompasses the orchestration and lifecycle management of physical/software resources that support the infrastructure virtualization, and the lifecycle management of VNFs. NFV management and orchestration focuses on all virtualization-specific management tasks necessary in the NFV framework.

The ISG NFV Architectural Framework document specifies that in the deployment, operation, management and orchestration of VNFs, two types of relations between VNFs are supported:

- **VNF forwarding graph (VNF FG)**: Covers the case where network connectivity between VNFs is specified, such as a chain of VNFs on the path to a web server tier (for example, firewall, network address translator, load balancer).

- **VNF set**: Covers the case where the connectivity between VNFs is not specified, such as a web server pool.

NFV Reference Architecture

We have discussed high-level framework in above section. Figure given below shows a more detailed look at the ISG NFV reference architectural framework. You can view this architecture as consisting of four major blocks:

- **NFV infrastructure (NFVI)**: Comprises the hardware and software resources that create the environment in which VNFs are deployed. NFVI virtualizes physical computing, storage, and networking and places them into resource pools.
- **VNF/EMS**: The collection of VNFs implemented in software to run on virtual computing, storage, and networking resources, together with a collection of element management systems (EMS) that manage the VNFs.
- **NFV management and orchestration (NFV-MANO)**: Framework for the management and orchestration of all resources in the NFV environment. This includes computing, networking, storage, and VM resources.
- **OSS/BSS**: Operational and business support systems implemented by the VNF service provider.

It is also useful to view the architecture as consisting of three layers. The NFVI together with the virtualized infrastructure manager provide and manage the virtual resource environment and its underlying physical resources. The VNF layer provides the software implementation of network functions, together with element management systems and one or more VNF managers. Finally, there is a management, orchestration, and control layer consisting of OSS/BSS and the NFV orchestrator.

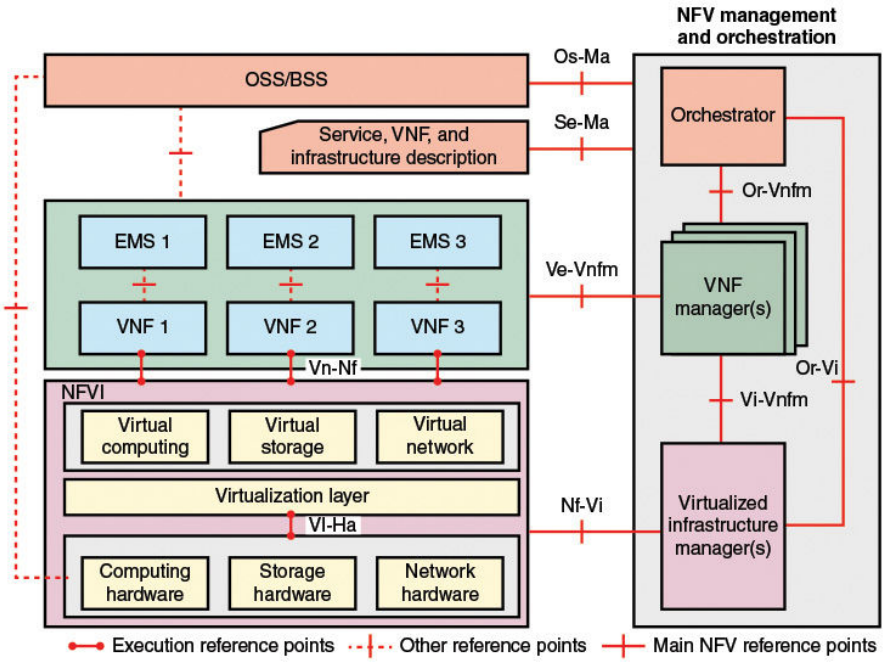


Figure 3: High-Level NFV Framework

NFV Management and Orchestration

The NFV management and orchestration facility includes the following functional blocks:

- **NFV orchestrator**: Responsible for installing and configuring new network services (NS) and virtual network function (VNF) packages, NS lifecycle management, global resource management, and validation and authorization of NFVI resource requests.
- **VNF manager**: Oversees lifecycle management of VNF instances.
- **Virtualized infrastructure manager**: Controls and manages the interaction of a VNF with computing, storage, and network resources under its authority, in addition to their virtualization.

Reference Points

Architecture also defines a number of reference points that constitute interfaces between functional blocks. The main (named) reference points and execution reference points are shown by solid lines and are in the scope of NFV. These are potential targets for standardization.

The main reference points include the following considerations:

- **Vi-Ha**: Marks interfaces to the physical hardware. A well-defined interface specification will facilitate for operators sharing physical resources for different purposes, reassigning resources for different purposes, evolving software and hardware independently, and obtaining software and hardware component from different vendors.

- **Vn-Nf**: These interfaces are APIs used by VNFs to execute on the virtual infrastructure. Application developers, whether migrating existing network functions or developing new VNFs, require a consistent interface the provides functionality and the ability to specify performance, reliability, and scalability requirements.

- **Nf-Vi**: Marks interfaces between the NFVI and the virtualized infrastructure manager (VIM). This interface can facilitate specification of the capabilities that the NFVI provides for the VIM. The VIM must be able to manage all the NFVI virtual resources, including allocation, monitoring of system utilization, and fault management.

- **Or-Vnfm**: This reference point is used for sending configuration information to the VNF manager and collecting state information of the VNFs necessary for network service lifecycle management.

- **Vi-Vnfm**: Used for resource allocation requests by the VNF manager and the exchange of resource configuration and state information.
- **Or-Vi**: Used for resource allocation requests by the NFV orchestrator and the exchange of resource configuration and state information.
- **Os-Ma**: Used for interaction between the orchestrator and the OSS/BSS systems.
- **Ve-Vnfm**: Used for requests for VNF lifecycle management and exchange of configuration and state information.
- **Se-Ma**: Interface between the orchestrator and a data set that provides information regarding the VNF deployment template, VNF forwarding graph, service-related information, and NFV infrastructure information models.

Lecture 7

SDN and NFV Comparison

The core similarity between software-defined networking (SDN) and network functions virtualization (NFV) is that they both use network abstraction. Both depend heavily on virtualization to enable network design and infrastructure to be abstracted in software and then implemented by underlying software across hardware platforms and devices. SDN seeks to separate network control functions from network forwarding functions, while NFV seeks to abstract network forwarding and other networking functions from the hardware on which it runs. SDN abstracts physical networking resources ?switches, routers and so on ? and moves decision making to a virtual network control plane. In this approach, the control plane decides where to send traffic, while the hardware continues to direct and handle the traffic. NFV aims to virtualize all physical network resources beneath a hypervisor, which allows the network to grow without the addition of more devices. When SDN executes on an NFV infrastructure, SDN forwards data packets from one network device to another. At the same time, SDN's networking control functions for routing, policy definition and applications run in a virtual machine somewhere on the network. Thus, NFV provides basic networking functions, while SDN controls and orchestrates them for specific uses. SDN further allows configuration and behavior to be programmatically defined and modified.

The concern of a network service provider is about the set of network devices (such as routers) and the control and management of the functions they perform (such as packet forwarding). Both SDF and NFV can be used seperately to provide this but SDN and NFV are not mutually exclusive. If both SDN and NFV are implemented for a network, the following relationships hold:

- The relationship between SDN and NFV is perhaps viewed as SDN functioning

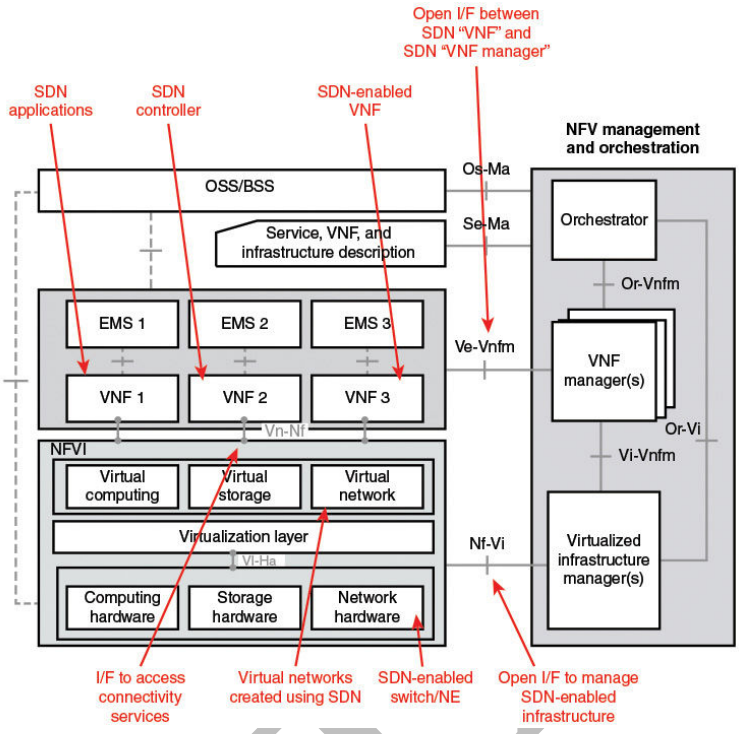
as an enabler of NFV.

- Network data plane functionality is implemented on VMs.
- The control plane functionality may be implemented on a dedicated SDN platform or on an SDN VM.
- A major challenge with NFV is to best enable the user to configure a network so that VNFs running on servers are connected to the network at the appropriate place, with the appropriate connectivity to other VNFs, and with desired QoS
- With SDN, users and orchestration software can dynamically configure the network and the distribution and connectivity of VNFs.
- Without SDN, NFV requires much more manual intervention, especially when resources beyond the scope of NFVI are part of the environment.

NFV and SDN combined architecture

Some of the ways that ETSI believes that NFV and SDN complement each other include the following:

- The SDN controller fits well into the broader concept of a network controller in an NFVI network domain.
- SDN can play a significant role in the orchestration of the NFVI resources, both physical and virtual, enabling functionality such as provisioning, configuration of network connectivity, bandwidth allocation, automation of operations, monitoring, security, and policy control.
- SDN can provide the network virtualization required to support multitenant NFVIs.



- Forwarding graphs can be implemented using the SDN controller to provide automated provisioning of service chains, while ensuring strong and consistent implementation of security and other policies.

- The SDN controller can be run as a VNF, possibly as part of a service chain including other VNFs. For example, applications and services originally developed to run on the SDN controller could also be implemented as separate VNFs.

- SDN enabled switch/NEs include physical switches, hypervisor virtual switches, and embedded switches on the NICs.

- Virtual networks created using an infrastructure network SDN controller provide connectivity services between VNFC instances.

- SDN controller can be virtualized, running as a VNF with its EM and VNF manager. Note that there may be SDN controllers for the physical infrastructure, the virtual infrastructure, and the virtual and physical network functions. As such, some of these SDN controllers may reside in the NFVI or management and orchestration (MANO) functional blocks (not shown in figure).
- SDN enabled VNF includes any VNF that may be under the control of an SDN controller (for example, virtual router, virtual firewall).
- SDN applications, for example service chaining applications, can be VNF themselves.
- Nf-Vi interface allows management of the SDN enabled infrastructure.
- Ve-Vnfm interface is used between the SDN VNF (SDN controller VNF, SDN network functions VNF, SDN applications VNF) and their respective VNF Manager for lifecycle management.
- Vn-Nf allows SDN VNFs to access connectivity services between VNFC interfaces.

Lecture 8

Importance of SDN and NFV for IoT

Challanges of IoT for Communication Network.

The increasing number of physical objects connected to the Internet at a remarkable rate brings the idea of the rapid evolution of the Internet of Things (IoT).The Internet of Things is understood to be a global network infrastructure composed of diverse heterogeneous devices that rely on sensory, communication, networking and information processing technologies required to provide advanced services aimed to improve our quality of life. The application doamins stems from healthcare (medical monitoring devices) to smart grids, transportation systems, industrial and automation sectors, just to mention a few. IoT draws together various technologies such as Radio Frequency Identification (RFID), Near Field Communication (NFC), Wireless Sensor Networks (WSN), Machine-to-Machine (M2M) communications. However, the complexity, possible limitations and heterogeneity of various IoT devices connected to the internet will require even more specific tools to manage them and to improve the performance of the whole network. The critical aspects are often not only related on the characteristics of the devices but also on the adopted proprietary architectures. This is particular evident in the emerging domain of "deterministic networking" mainly for industrial applications, and on the Low Power, low bit rate Wide Area Networks (LPWAN) domain such as LoRa or Sigfox, mainly for telemetry applications.

Traditional architectures and network protocols for IoT devices are not designed to support high level of scalability, high amount of traffic and mobility together with the above mentioned requirements. They are inefficient and have limitations to satisfy these new requirements. Moreover, with the incredible numbers of connected

objects forecasted according to Gartner1, it?s difficult to manage these devices generating

an impressive amount of data as a whole, without having elasticity and flexibility inherently defined in the network. If the networks are not prepared, the flood of IoT where a lot of traffic are generated could leave the network paralysed.

To achieve such goals, emerging technologies such as software defined networking (SDN) and Network Function Virtualization (NFV) are being considered as technology enablers to provide adequate solutions. SDN has recently received a great deal of attention from researchers and has proven itself in data centers networks, where joint optimization of network and IT resources are the main goals. Just for an example, Google has revealed the use of SDN for managing its networking infrastructure interconnecting their data centres has revealed the use of SDN for managing its networking infrastructure interconnecting their data centres.

SDN is a new approach for network programmability where the network operator programs the controller to automatically manage data plane devices and optimize network resource usage. This results in improved network performance in terms of network management, control and data handling. Network Function Virtualization (NFV) is a network architecture concept of replacing dedicated network appliances such as switches, routers, firewalls, to name a few, with software running on commercial off-the-shelf servers. This brings advantages in terms of energy savings, load optimization and network scalability. NFV can serve SDN by virtualizing the SDN controller to be rendered in the cloud thus allowing dynamic migration of the controllers to the optimal locations while SDN can serve NFV by providing programmable network connectivity between NFVs to achieve optimized traffic engineering.

Advantage of SDN/NFV in IoT

The key advantages in employing SDN/NFV in IoT are:

1. *Solving Interoperability in the Internet of Things*: Interoperability challenge in IoT arises when we have heterogeneous devices exchanging data formats and diverse protocols for machine to machine (M2M) data exchange, and also with the interconnectivity of large number of different devices, there is lack of cooperation and capability mismatch between devices which can hinder the performance of the network. However, SDN approach brings flexibility which can be used to allow different objects connected to heterogeneous networks to communicate with each other. This will be able to handle simultaneous connections of various communication technologies. Network management decisions such as routing, scheduling can be done at the SDN controller and moreover, the programmability allows for any updates for new proposals or even clean state approaches.

2. *Discoverability*: Discoverability in IoT devices is one of the main factors in achieving a successful deployment of IoT applications to prevent long outages and configuration errors. The ability to self-configure and adapt to the environment without human

   intervention brings about requirements such as resource and service discovery. It is not feasible to manually configure each and every device in order to discover which objects are nearby and which functions they provide. SDN approach can be used to address this issue which allows applications to operate with devices with minimal or no configuration.

3. *Security*: Having large number of heterogeneous devices that are involved in the Internet of Things, there is high propensity

to be vulnerable to attacks, and ensuring data protection, data authentication needs to be taken seriously. Security threats can be easier to attack through the improved visibility SDN provides to the network. SDN can also provide a dynamic, intelligent, selflearning layered model of security that provides access rules to ensure authorization for people who are allowed to change the configuration of devices.

4. **Management:** The SDN controller manages and supervises the entire network. The centralized position of the SDN controller makes it suitable to have a global vision of the network topology and conditions, performing network control such as routing and QoS control. The controller can determine the best routing decisions and inserting these decisions into the flow tables. The sensor nodes do not make routing decisions but only forward and drop packets according to the rules set by the controller. The scheduling must be built over defined routes and the controller can optimize the sleep/active cycles of the sensors by choosing the most energy-efficient set in every scheduling cycle. Latency can be reduced and significant energy savings can be achieved.

5. **Scalability Issues:** The rapid growth of embedded technologies is leading to enormous deployment of miniaturized devices (sensors, actuators, etc.). As the number of devices grows, the data produced by these devices grow unboundedly SDN can improve scalability issues in IoT network where the SDN controller oversees the network domain and communicates with other SDN controllers to exchange aggregated network-wide formation. The distributed SDN model tends to share the load among several controllers. This easily helps in adapting to the users and applications. This is a distributed way and spreads functionality across several nodes. This brings many benefits such as (1) scalability (2) reducing latency from the sensor nodes to closest controller,

(3) load balancing (4) fault-tolerance among others. With multiple controllers, this can be used to offload computational tasks which brings benefits in terms of administration. Each domain has its SDN controller which controls all traffic in its domain. When one SDN controller fails, another SDN controller can take control to avoid network failures.

6. *Application Specific Requirements:* As earlier stated that some IoT applications works in realtime, so a need to support real-time applications is required in an IoT environment where it is expected to monitor different things at different time periods. SDN is able to strengthen network controlling ability and also perform dynamic adaptation of control logic by the devices in real-time. SDN and its extension to the Wireless Sensors and Actuators Domain will give the possibility to support application specific requirements with control logic that jointly act at the network and processing level enhancing the QoS/QoE of the entire system.

7. *Service Chain Simplification and Application Provisioning:* SDN and NFV can make service chain shorter and simpler by increasing the efficiency and capacity of the network without radically changing hardware making it easier to spin up IoT applications. SDN & NFV will help service providers to enhance their service delivery infrastructure which is very important in the contribution to the IoT. This addresses the issues of network ossification by utilizing the network resources in a better way and transiting to software-centric programmable networks due to the rapid evolution and dynamicity of IoT applications.