Machine-to-Machine (M2M)

- **M2M**: It is a network of machines (or devices) for the purpose of monitoring, control and data exchange.
- M2M happens when machines automatically exchange information between them, without relying on human intervention or middleware devices.
- According to the strictest definitions, there should be absolutely zero human intervention or other devices in between for communication between two devices. However, there might be some human intervention or servers in the middle of data transfer between two devices in practical use cases.
- **Examples:**
  - A vending machine relaying its stock levels to the supply chain software,
  - An ATM asking for authorization to dispense cash from the bank servers,
  - A credit card reader authorizing a transaction, etc.

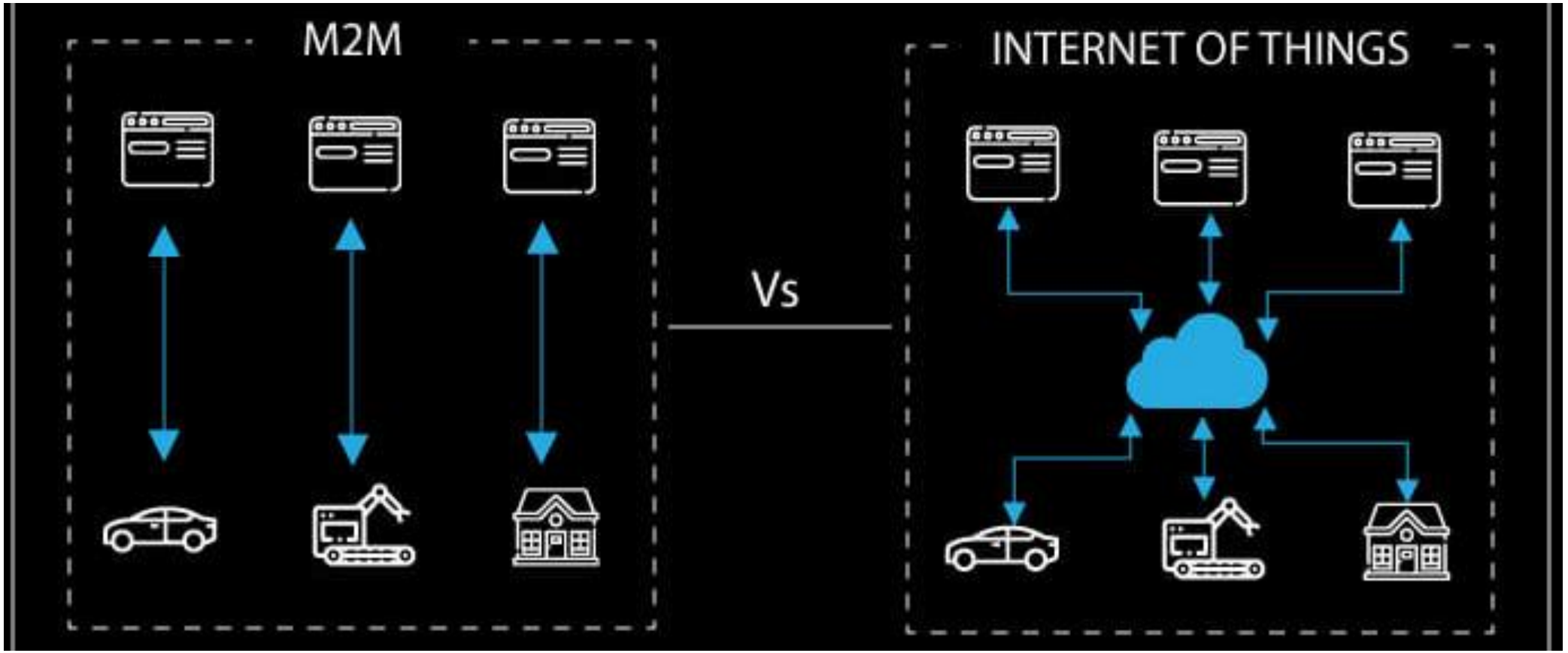
Machine-to-Machine (M2M)

- **Examples:**
  - A washing machine/dryer connected with M2M communication capabilities can be used to send a signal at the end of the washing/drying cycle to the users. The machine will have smart capabilities to enable communication to send the signal to the user.
  - HVAC systems with M2M communication capabilities can be turned on in advance using a smartphone. The house/room will have the perfect temperature by the time you arrive home.
  - Smart refrigerators can analyze what is left in them and automatically create a shopping list according to routine needs. It can even have the capability to order the products in time to arrive fresh in time for consumption.

Working of Machine-to-Machine (M2M)

- Machine to machine (M2M) is a comprehensive concept that can be used to describe any technology that allows networked devices to exchange information and inform actions automatically without human interference.
- M2M communication devices system is standalone network equipment that uses point-to-point communication between machines, sensors and hardware.
- Today, M2M is among the fastest-growing technologies because it can connect millions of devices within a single network.
- M2M technology’s purpose is to collect the data from sensors and transmit them to the cloud. It uses a mobile network for transmission purposes, which makes it a cost-effective technology. The working of M2M is that the data is collected from the device sensors, sent to the cloud via mobile towers.

Machine-to-Machine (M2M)



Wired M2M Communication

Wired and Wireless M2M Communication

The two main types of M2M communication are wired communication and wireless communication. Each of these functions a bit differently.

Wired M2M Communication

- In wired M2M communication, the data transfer between the devices occurs over a wired transfer medium. It can be fiber optic cables, EtherCAT, or even coaxial cables. The communication between devices in the same LAN network connected with only wired connections is also part of wired M2M communication.
- Wired communication networks are becoming rarer now. Most use cases are moving to wireless networks.
- Wired networks are only used in cases where wireless signals are susceptible to interference.
- It is also used in legacy systems that can be difficult to upgrade communication networks to a wireless system.

Wireless M2M Communication

When machines use wireless communication methodology to communicate between devices, it is wireless M2M communication. M2M communication that majorly uses the wireless network for communication is termed as Internet of Things or IoT. The wireless communication methodologies used have a wide range from radio waves to the latest 5G technology.

- **RFID** – RFID or Radio Frequency Identification is quite an old technology that has stood the test of time. RFID technology communicates over radio wave frequency to enable wireless transfer of data. Now it is widely used in the logistics industry to track a package from start to finish. It is also used for provenance technology block chains.
- **NFC** – Near-Field Communication or NFC is also similar to RFID but can only be used for short-range transfer of data. It is widely used for access control and payment systems. Payment applications like Apple pay and contactless credit cards make use of NFC technology to transfer data. It can also be used as an efficient mode of communication to transfer information between smart phones within a short range.

Wireless M2M Communication

- **WiFi** – WiFi or wireless fidelity is widely used in homes and offices to access the internet wirelessly. The same network can also transfer data between devices connected to the same WiFi network. There have been many newer iterations of WiFi technology that increased the bandwidth and reduced communication latency. The latest WiFi technology that FCC has approved is WiFi 6e and can bring massive improvement for IIoT communication.
- **Cellular Network** – Cellular networks can also be made use of for wireless M2M communication. The very first wireless credit card readers used GSM technology or other 2G technologies for communication. Later communication was possible over HSPA (3G) and LTE (4G) networks. Now wireless communication is at the precipice of disruption with 5G communication technologies. 5G dramatically improves over 4G technology in terms of bandwidth and latency. Thousands of devices would be able to ‘talk to each other’ very fast and with very minimal lag over a 5G network. Implementation of 5G in the industrial setting is said to be akin to a new industrial revolution in the works.

Applications of Machine-to-Machine (M2M)

1. **Security :** Surveillances, Alarm systems, Access control, Car/driver security
2. **Tracking & Tracing :** Fleet Management, Order Management, Pay as you drive, Asset Tracking, Navigation, Traffic information, Road tolling, Traffic optimization/steering
3. **Payment :** Point of sales, Vending machines, Gaming machines
4. **Health :** Monitoring vital signs, Supporting the aged or handicapped, Web Access Telemedicine points, Remote diagnostics
5. **Remote Maintenance/Control :** Sensors, Lighting, Pumps, Valves, Elevator control, Vending machine control, Vehicle diagnostics
6. **Metering :** Power, Gas, Water, Heating, Grid control, Industrial metering
7. **Manufacturing :** Production chain monitoring and automation
8. **Facility Management :** Home / building / campus automation

Key Features of Machine-to-Machine (M2M)

Some of the key features of M2M communication system are given below:

1. **Low Mobility :** M2M Devices do not move, move infrequently, or move only within a certain region
2. **Time Controlled :** Send or receive data only at certain pre-defined periods
3. **Time Tolerant :** Data transfer can be delayed
4. **Packet Switched :** Network operator to provide packet switched service with or without an MSISDN
5. **Online small Data Transmissions:** MTC Devices frequently send or receive small amounts of data.
6. **Monitoring:** Not intend to prevent theft or vandalism but provide functionality to detect the events
7. **Low Power Consumption :** To improve the ability of the system to efficiently service M2M applications
8. **Location Specific Trigger :** Intending to trigger M2M device in a particular area e.g. wake up the device

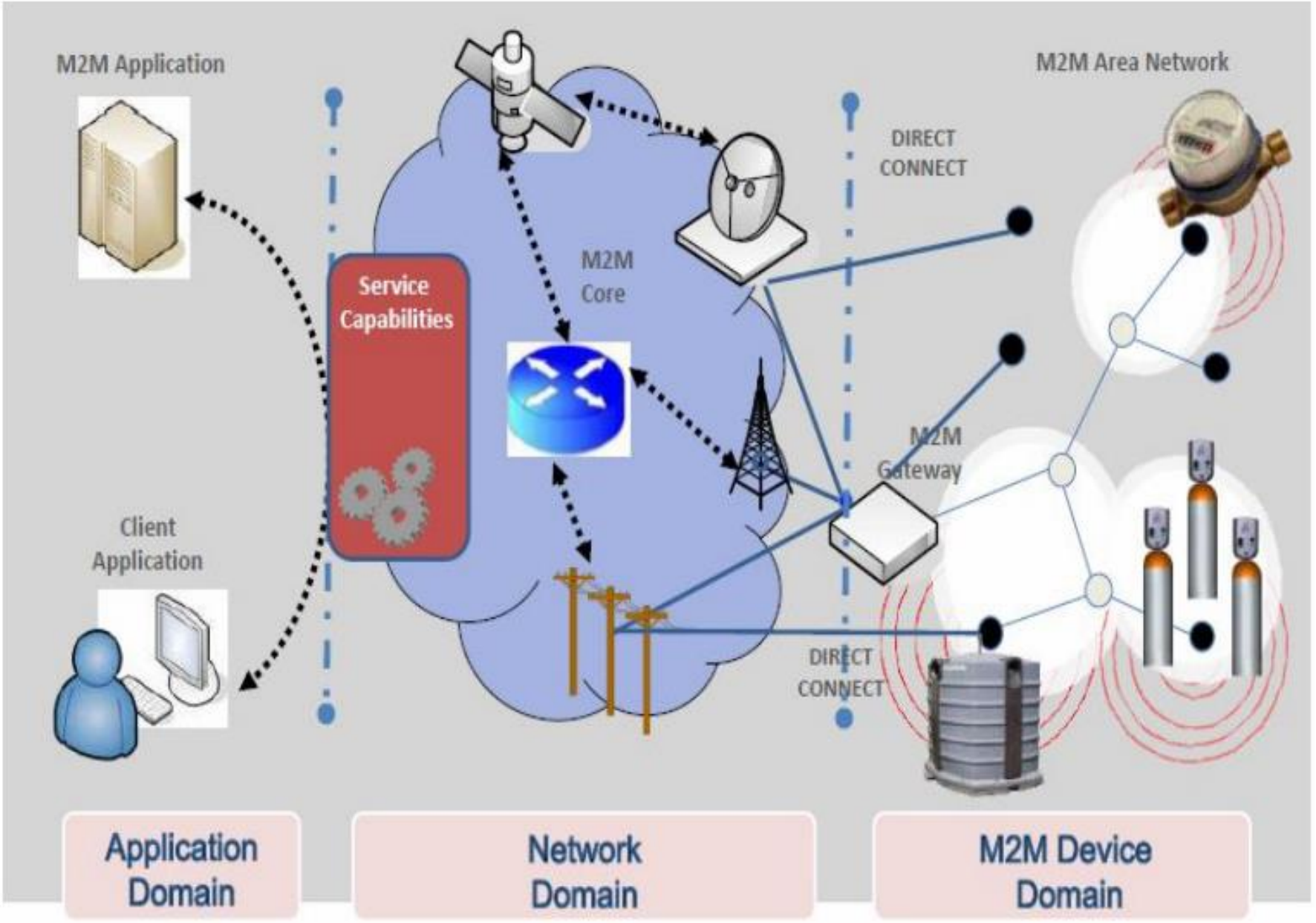
Architecture & Components of Machine-to-Machine (M2M)

1. **M2M Device:**

- Device capable of replying to request for data contained within those devices or capable of transmitting data autonomously.
- Sensors and communication devices are the endpoints of M2M applications. Generally, devices can connect directly to an operator's network, or they will probably interconnect using WPAN technologies such as ZigBee or Bluetooth.
- Backhaul to an operator's network is than achieved via gateways that encapsulate and manage all devices. Consequently, addressing and identifying, e.g., routing, of the devices relies heavily on the gateways. Devices that connect via gateways are normally outside the operator's responsibility but belong to M2M applications that are provided by service or application providers.

2. **M2M Area Network (Device Domain):** Provide connectivity between M2M Devices and M2M Gateways, e.g. personal area network.

Architecture & Components of Machine-to-Machine (M2M)



Architecture & Components of Machine-to-Machine (M2M)

2. M2M Gateway:

- Equipment that uses M2M capabilities to ensure M2M Devices inter-working and interconnection to the communication network.
- Gateways and routers are the endpoints of the operator’s network in scenarios where sensors and M2M devices do not connect directly to the network.
- An **M2M Gateway (Machine-to-Machine Gateway)** is an important component in **IoT and MTC (Machine-Type Communication)** systems.
- It acts as a **bridge between devices (sensors, machines, controllers) and external networks (cloud, internet, or enterprise systems).**

Architecture & Components of Machine-to-Machine (M2M)

Functions of an M2M Gateway

1. **Protocol Translation**

   Devices use different communication protocols (Zigbee, Bluetooth, Modbus, CAN, etc.).

   The gateway **translates these into standard IP-based protocols** (like MQTT, CoAP, HTTP).

   Example: A Zigbee-based smart bulb communicating with a cloud server over MQTT via the gateway.

2. **Data Aggregation & Filtering**

   Collects data from multiple devices.

   Filters or aggregates before sending to reduce bandwidth usage.

   Example: Instead of sending every single temperature reading, it may send average, min, and max values every 5 minutes.

Architecture & Components of Machine-to-Machine (M2M)

Functions of an M2M Gateway

3. **Security & Authentication**

   Provides device authentication, data encryption, and secure communication with cloud servers.

   Example: Encrypts sensor data before uploading to AWS IoT or Azure IoT Hub.

4. **Local Processing (Edge Computing)**

   Performs basic analytics and decision-making locally before sending data to the cloud.

   Reduces latency and dependency on internet connectivity.

   Example: In smart homes, the gateway can turn on a fan immediately when temperature > 30°C, without waiting for cloud commands.

5. **Network Bridging**

   Connects local non-IP devices to IP networks (like 4G/5G, Wi-Fi, Ethernet).

   Example: Connecting LoRaWAN sensor nodes to the internet via cellular.

Architecture & Components of Machine-to-Machine (M2M)

6. Device & Resource Management

Manages device configuration, firmware updates (OTA), and monitoring.

Example: Updating all smart meters in a city with new software via the gateway.

Example of M2M Gateway in Real Life

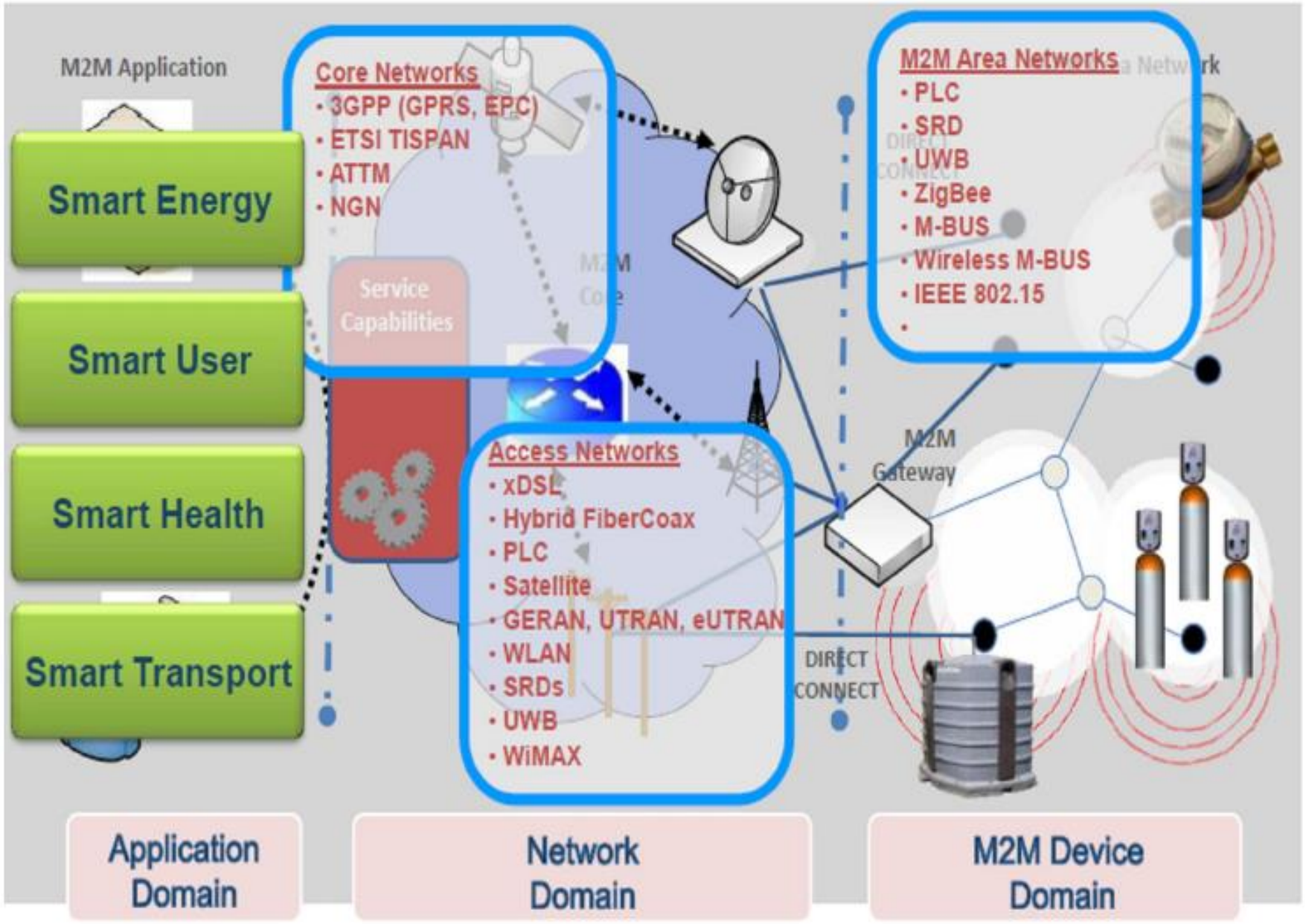
- **Smart City**: Streetlights with Zigbee controllers → Gateway translates to MQTT → Sends data to cloud for monitoring.
- **Industrial IoT**: Machines use Modbus → Gateway converts to OPC-UA/MQTT → Cloud for predictive maintenance.
- **Smart Home Hub (like Amazon Echo, Google Nest Hub)** → Works as an M2M Gateway between IoT devices and the internet.

Architecture & Components of Machine-to-Machine (M2M)

3. **M2M Communication Networks (Network Domain):** It covers the communications between the M2M Gateway(s) and M2M application(s), e.g. xDSL, LTE, WiMAX, and WLAN.

4. **M2M Applications:** It contains the middleware layer where data goes through various application services and is used by the specific business-processing engines. M2M applications will be based on the infrastructural assets (e.g., access enablers) that are provided by the operator. Applications may either target at end users, such as user of a specific M2M solution, or at other application providers to offer more refined building blocks by which they can build more sophisticated M2M solutions and services. e.g. customer care functionality, elaborate billing functions, etc. Those services, or service enablers, may be designed and offered by an application provider, but they might be offered by the operator via the operator platform itself.

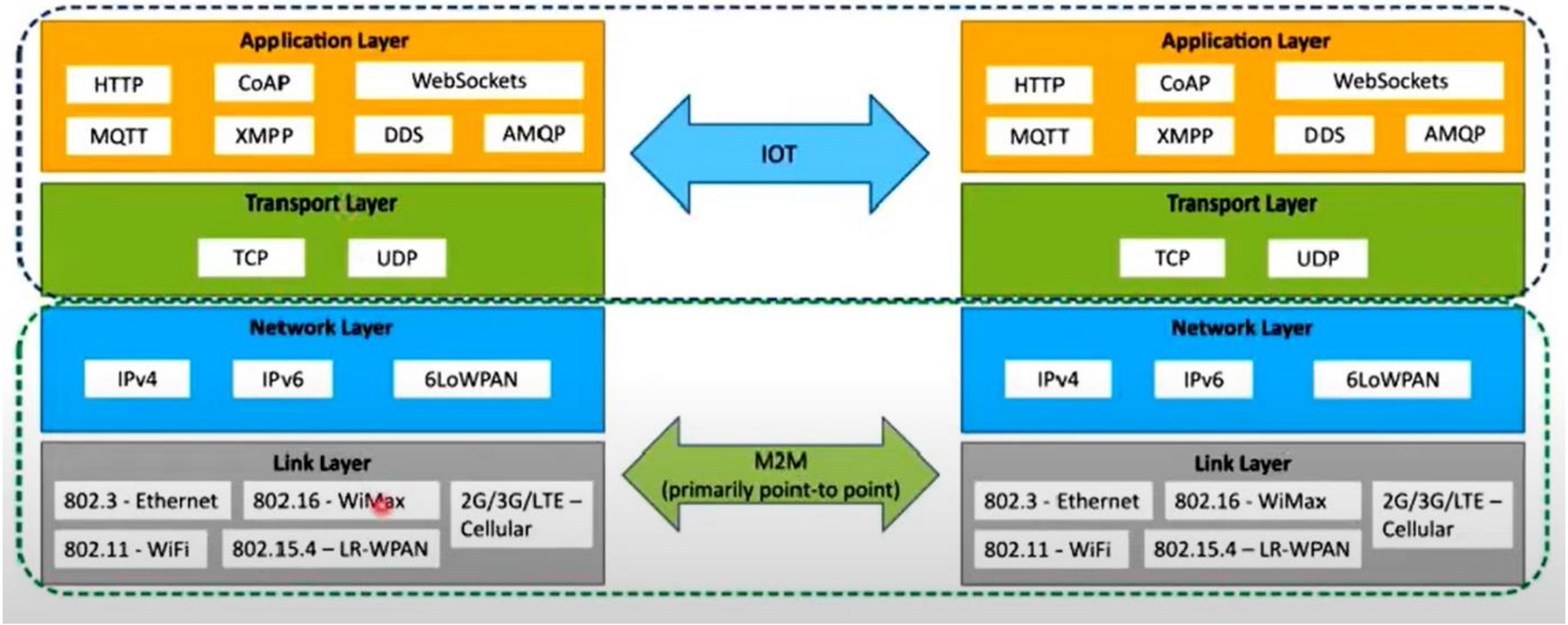
Architecture & Components of Machine-to-Machine (M2M)



IoT Vs M2M

Topic	M2M	IoT
Communication Protocols	Proprietary or Non-IP Based	IP Based
Machines in M2M vs Things in IoT	Homogeneous Machine	Physical Objects that have Unique Identifiers
Hardware vs Software Emphasis	More On Hardware	More On Software
Data Collection & Analysis	Collected In Point Solutions And Often In On-premises Storage	Collected In The Cloud (Can Be Public, Private Or Hybrid Cloud)
Applications	Diagnosis Applications, Service Management Applications, And On-premises Enterprise Applications	Analytics Applications, Enterprise Applications, Remote Diagnosis And Management Applications, etc.

IoT Vs M2M


Application Layer
HTTP
CoAP
WebSockets
MQTT
XMPP
DDS
AMQP
Transport Layer
TCP
UDP
Network Layer
IPv4
IPv6
6LoWPAN
Link Layer
802.3 - Ethernet
802.16 - WiMax
2G/3G/LTE – Cellular
802.11 - WiFi
802.15.4 – LR-WPAN
IOT
M2M
(primarily point-to point)
Application Layer
HTTP
CoAP
WebSockets
MQTT
XMPP
DDS
AMQP
Transport Layer
TCP
UDP
Network Layer
IPv4
IPv6
6LoWPAN
Link Layer
802.3 - Ethernet
802.16 - WiMax
2G/3G/LTE – Cellular
802.11 - WiFi
802.15.4 – LR-WPAN

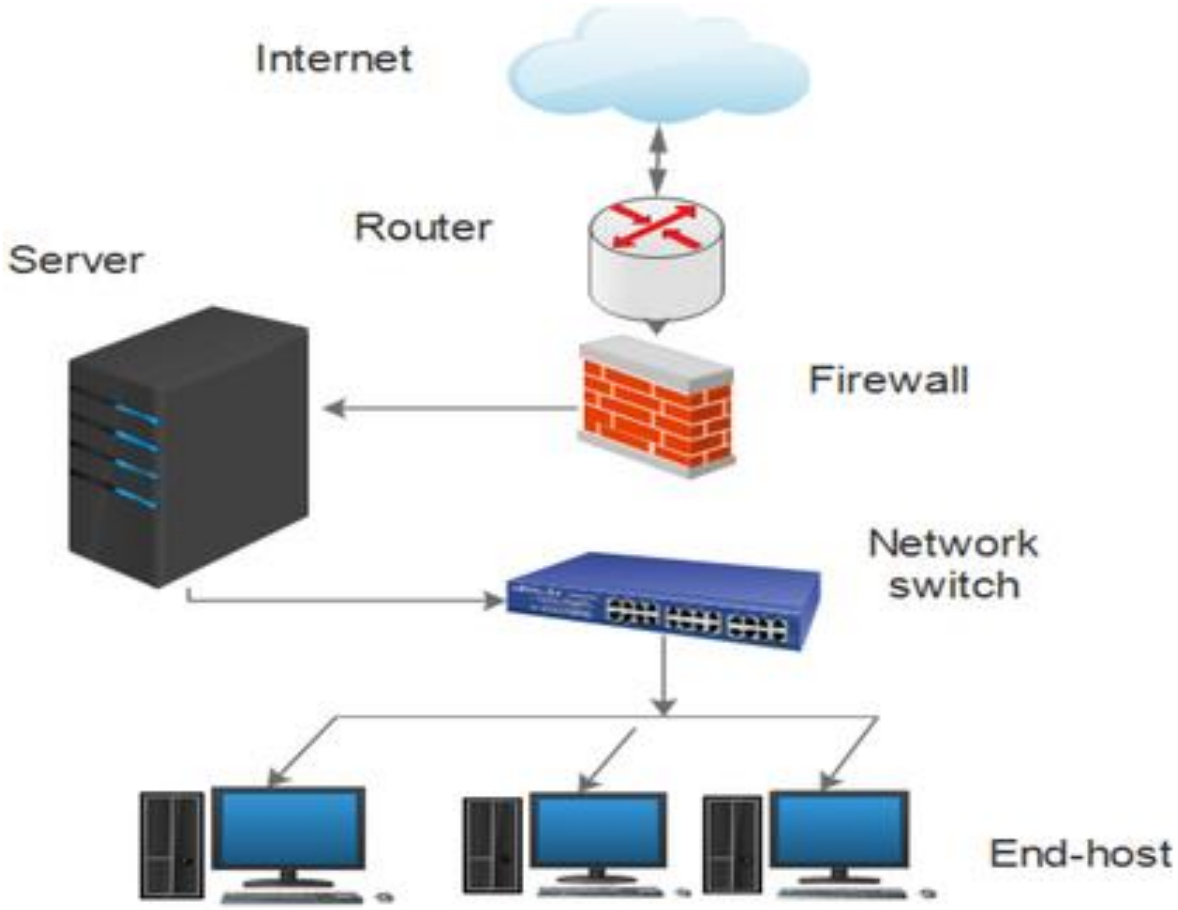
SDN (Software Defined Network)

Software-Defined Networking (SDN) is a network architecture approach that enables the network to be intelligently and centrally controlled, or ‘programmed,’ using software applications. This helps operators manage the entire network consistently and holistically, regardless of the underlying network technology.

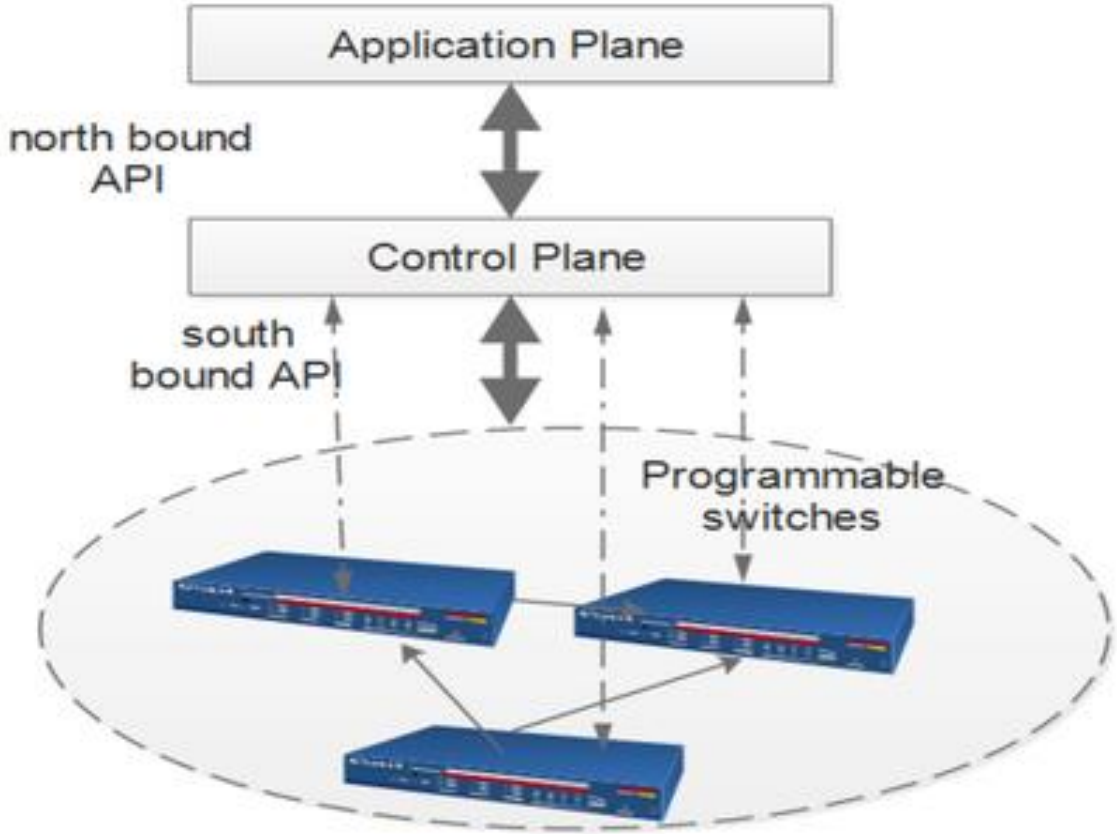
This model differs from that of traditional networks, which use dedicated hardware devices (i.e., routers and switches) to control network traffic. SDN can create and control a virtual network – or control a traditional hardware – via software.

While network virtualization allows organizations to segment different virtual networks within a single physical network, or to connect devices on different physical networks to create a single virtual network, software-defined networking enables a new way of controlling the routing of data packets through a centralized server.

SDN (Software Defined Network)

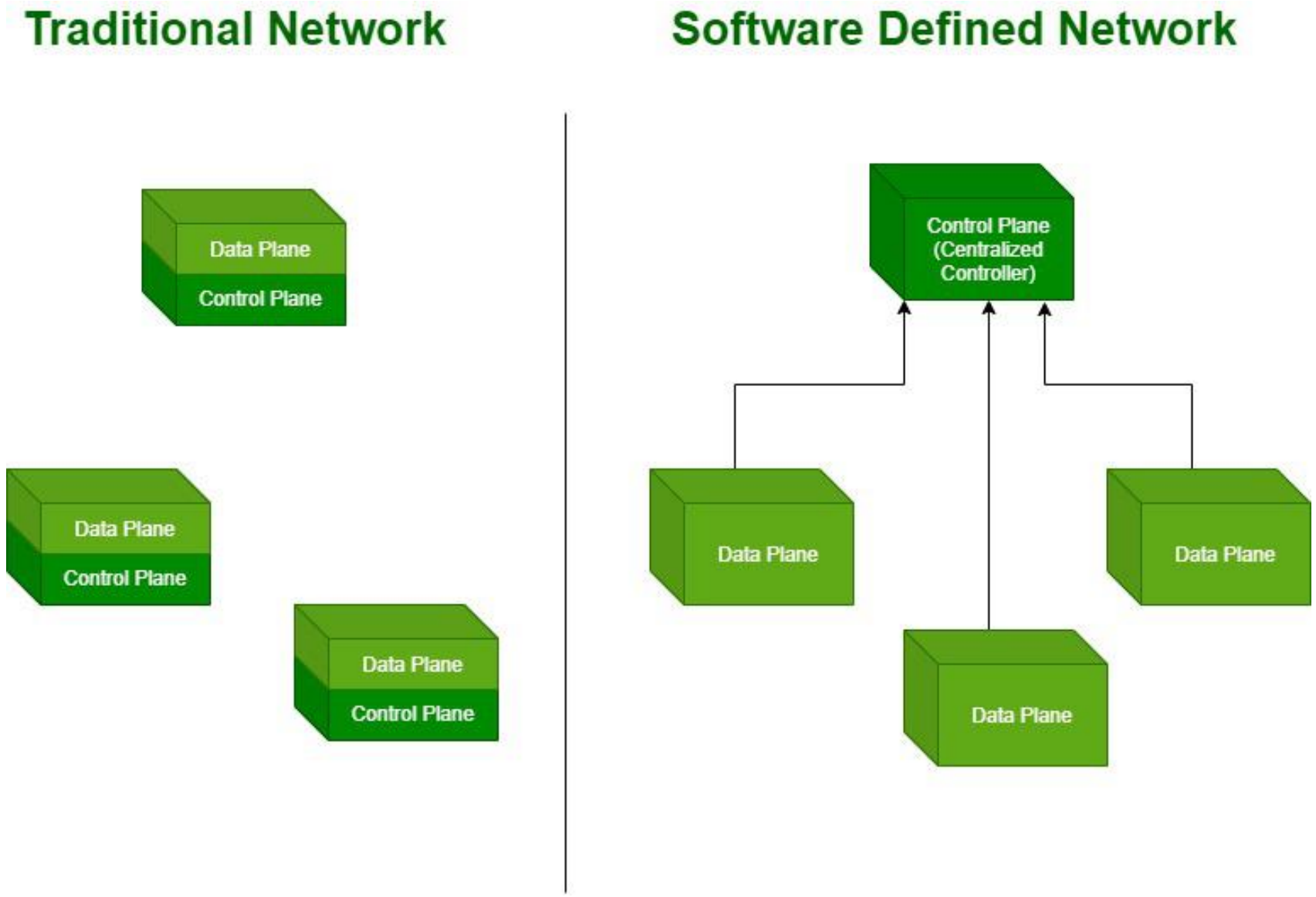


a Traditional Network architecture



b SDN architecture

SDN (Software Defined Network)



SDN (Software Defined Network)

Software-Defined Networking	Traditional Networking
It uses a virtualized approach to manage networks.	It depends on dedicated hardware devices to control network traffic.
It has centralized control using a software-based controller.	It has distributed control, with each device managing its own operations.
It is programmable. So it is highly flexible.	It is non-programmable. So it is less adaptable to changes.
It supports open interfaces for interoperability.	It depends on proprietary systems from single vendors.
Data and control planes are separated.	Data and control planes are combined in each device.
You can automate configuration. So it can save time.	It requires manual configuration. It takes more time.
It can prioritize specific network packets based on needs.	It handles all network traffic in the same way, without prioritization.

SDN (Software Defined Network)

Software-Defined Networking	Traditional Networking
It is easier to program and reprogram as needs evolve.	It is tough to modify and reprogram the network once its in place.
It is cost-effective due to simplified hardware needs.	It has higher hardware costs because of specialized devices.
It has lower structural complexity to manage.	It has higher structural complexity. So it is tough to manage.
It is easier to troubleshoot and report issues due to centralized control.	Its troubleshooting is tough because of distributed control.
Its maintenance costs are lower than traditional networks.	Its maintenance costs have higher than SDN.

SDN Architecture

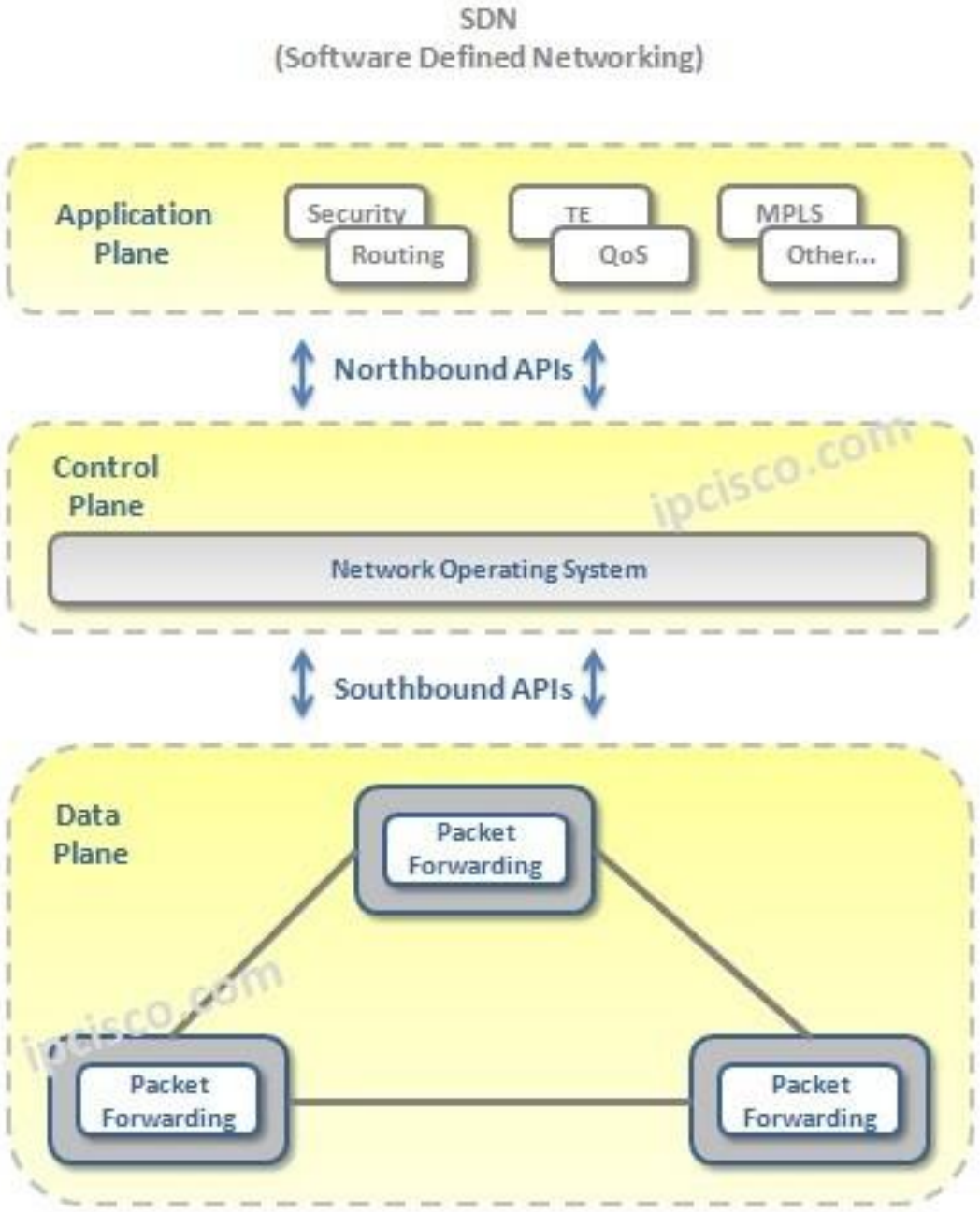
The SDN Architecture is:

- **DIRECTLY PROGRAMMABLE:** Network control is directly programmable because it is decoupled from forwarding functions.
- **AGILE:** Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- **CENTRALLY MANAGED:** Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- **PROGRAMMATICALLY CONFIGURED:** SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- **OPEN STANDARDS-BASED AND VENDOR-NEUTRAL:** When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

SDN (Software Defined Network)

SDN Architecture consist of different components. Here, we will see all these SDN Architecture Components and their duty one by one. What are these SDN Components? The basic SDN Terms, SDN Components are:

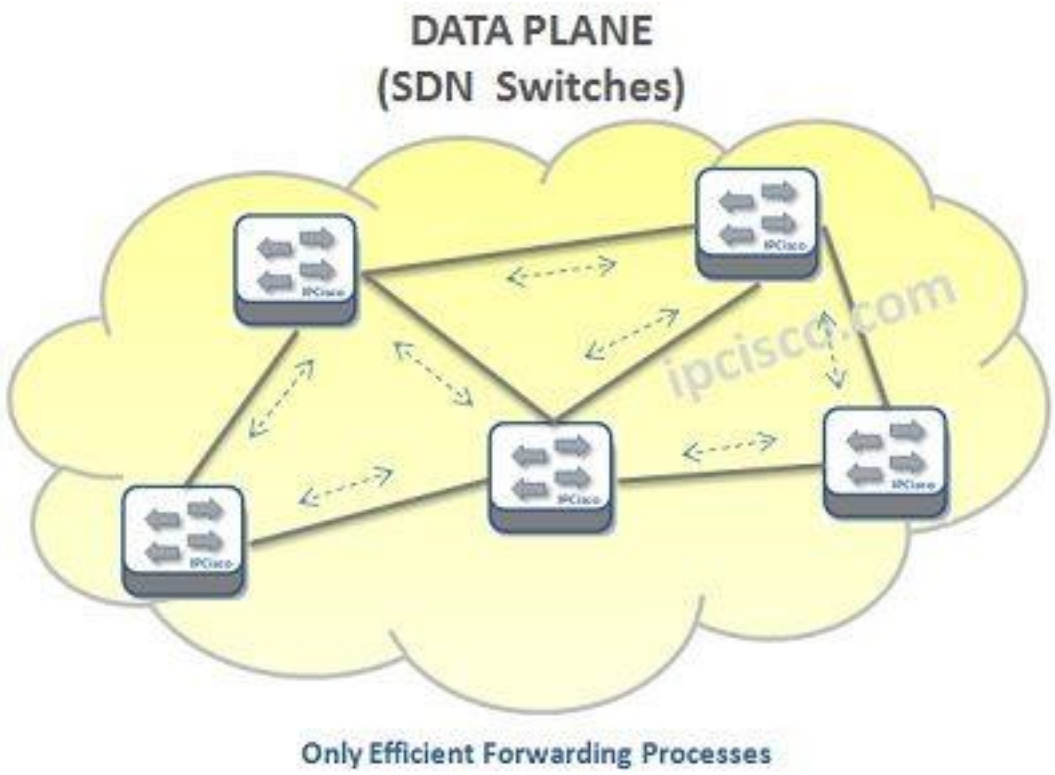
- Network Devices (Data Plane)
- SDN Controller (Control Plane)
- Southbound Interface
- Northbound Interface
- Network Operating System (NOS)
- Application and Services (Application Plane)



SDN (Software Defined Network)

SDN Architecture : Network Devices (Data Plane)

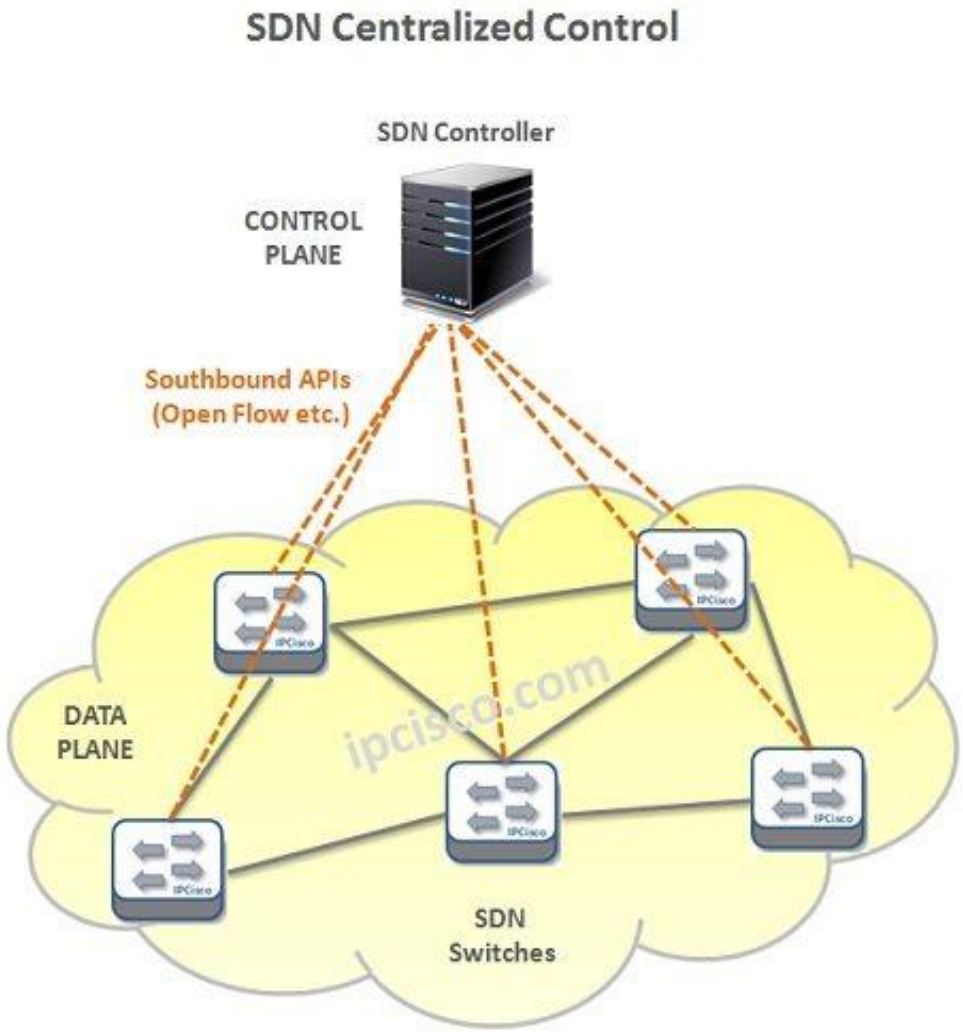
- The **data plane** (also called the forwarding plane) is made up of network devices like switches, routers, and access points that forward packets based on flow rules.
- **Functions:**
  - It **forwards data packets** between network endpoints according to the flow tables programmed by the SDN controller.
  - It does **not make routing decisions** — it just executes what the controller instructs.
- **Example:**
  - OpenFlow-enabled switches that take instructions from the SDN controller about how to handle specific traffic flows.



SDN (Software Defined Network)

SDN Architecture : SDN Controller (Control Plane)

- SDN Controller is the Center of the SDN Architecture and the most important one of SDN Architecture Components.
- The **brain of the SDN network**. It is a logically centralized software component that manages and controls all data plane devices.
- SDN Controller communicate and control these upper and lower layer with APIs through Interfaces.



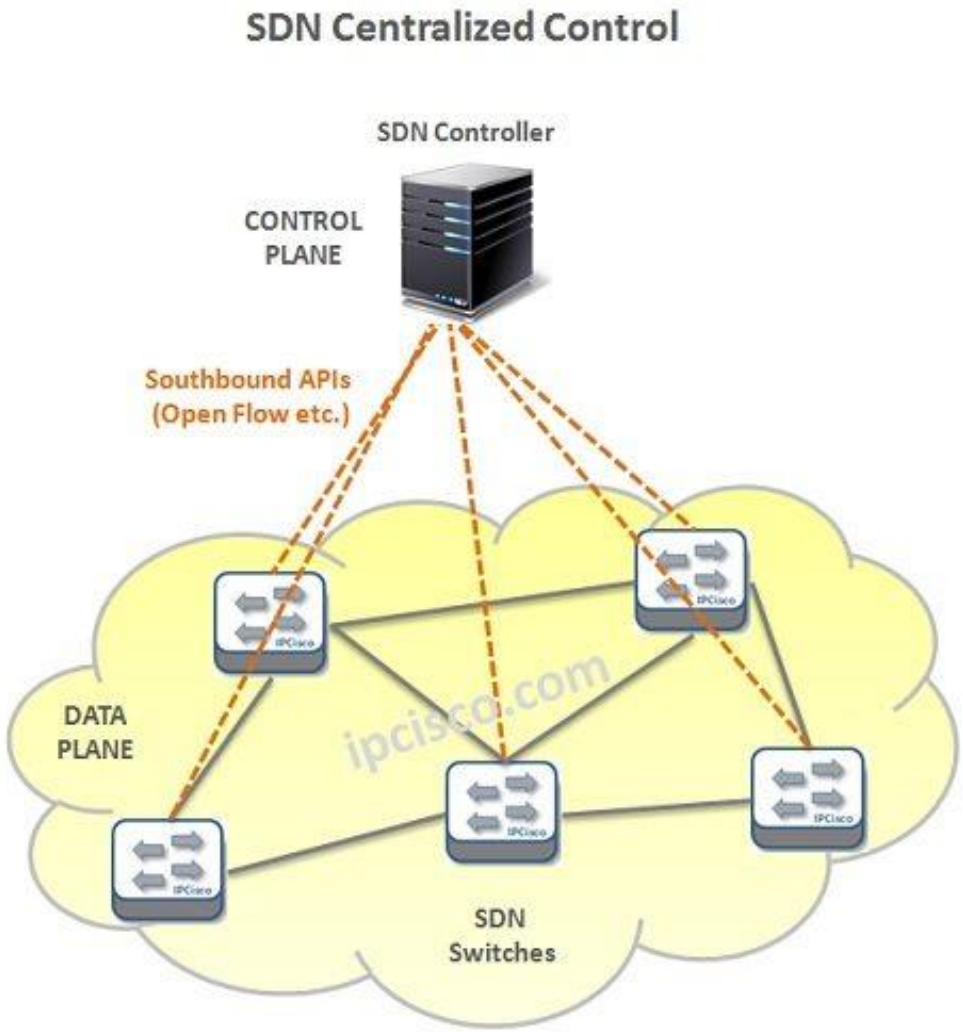
- **Functions:**
  - Maintains a **global view** of the entire network.
  - Installs forwarding rules on switches (flow entries).
  - Optimizes traffic flow, enforces security policies, and performs load balancing.
- **Examples of Controllers:**

  OpenDaylight, ONOS, Ryu, Floodlight.

SDN (Software Defined Network)

SDN Architecture : Southbound APIs

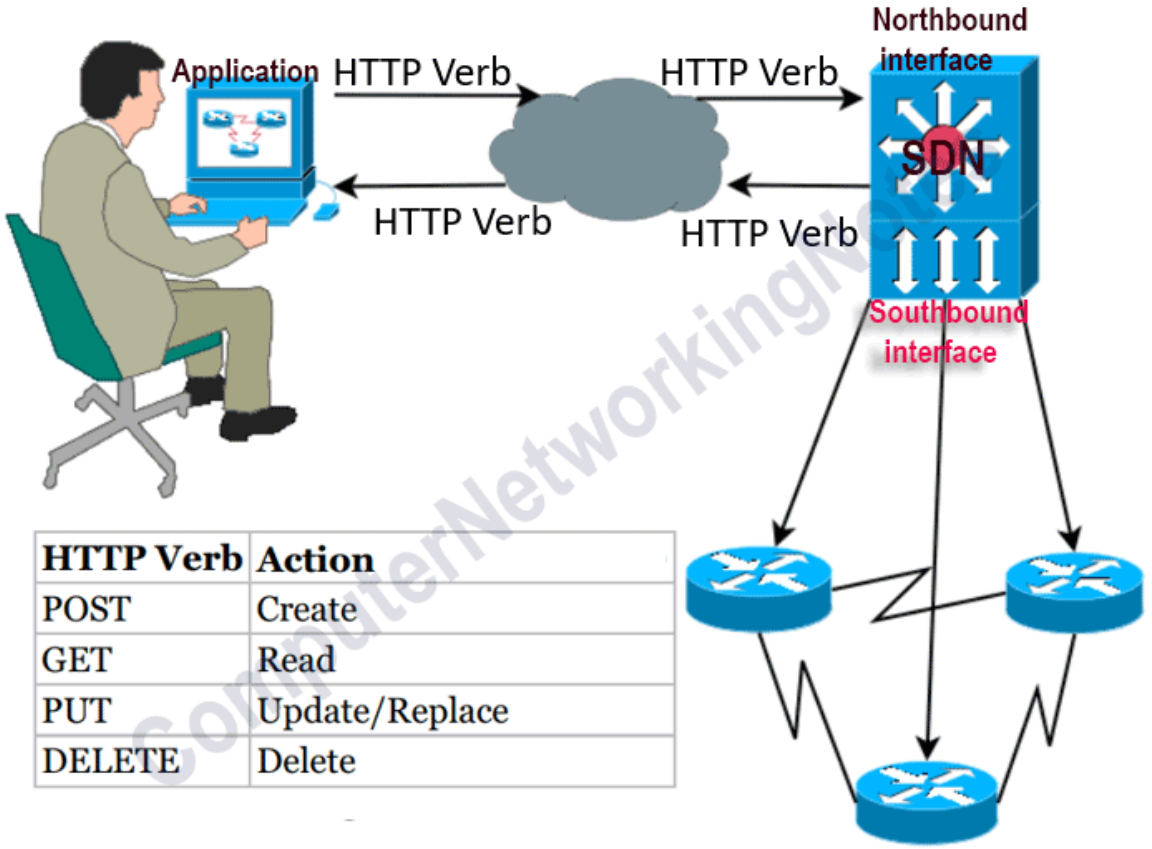
- The communication channel between the SDN controller (control plane) and network devices (data plane).
- **Functions:**
  - Allows the controller to **send instructions** (like flow rules) to switches and receive status updates back.
  - Ensures compatibility between hardware and software layers.
- **Common Protocols:**
  - **OpenFlow** (most popular)
  - NETCONF, BGP-LS, P4Runtime, gNMI



SDN (Software Defined Network)

SDN Architecture : Northbound APIs

- The communication channel between the SDN controller and applications/services (application plane).
- **Functions:**
  - Provides **APIs** (usually REST APIs) that allow applications to request network information or instruct the controller to configure the network.
  - Enables **programmability** — network admins can write custom apps for security, QoS, load balancing, etc.

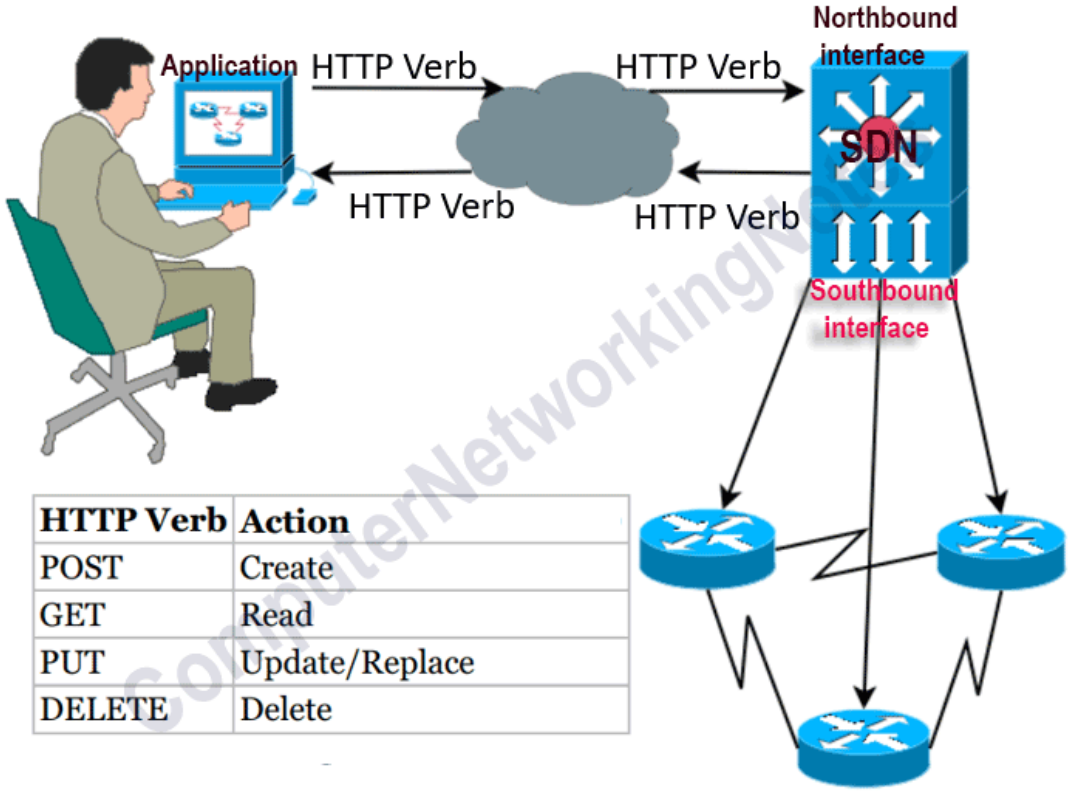


HTTP Verb	Action
POST	Create
GET	Read
PUT	Update/Replace
DELETE	Delete

SDN (Software Defined Network)

SDN Architecture : Northbound APIs

- The administrator sends HTTP GET requests to the SDN controller.
- In SDN implementation, the application uses HTTP GET requests to identify an object on the SDN controller, typically a data structure that the application needs to learn and then process.
- For example, it might identify an object that lists physical interfaces on a specific network device along with the status of each.



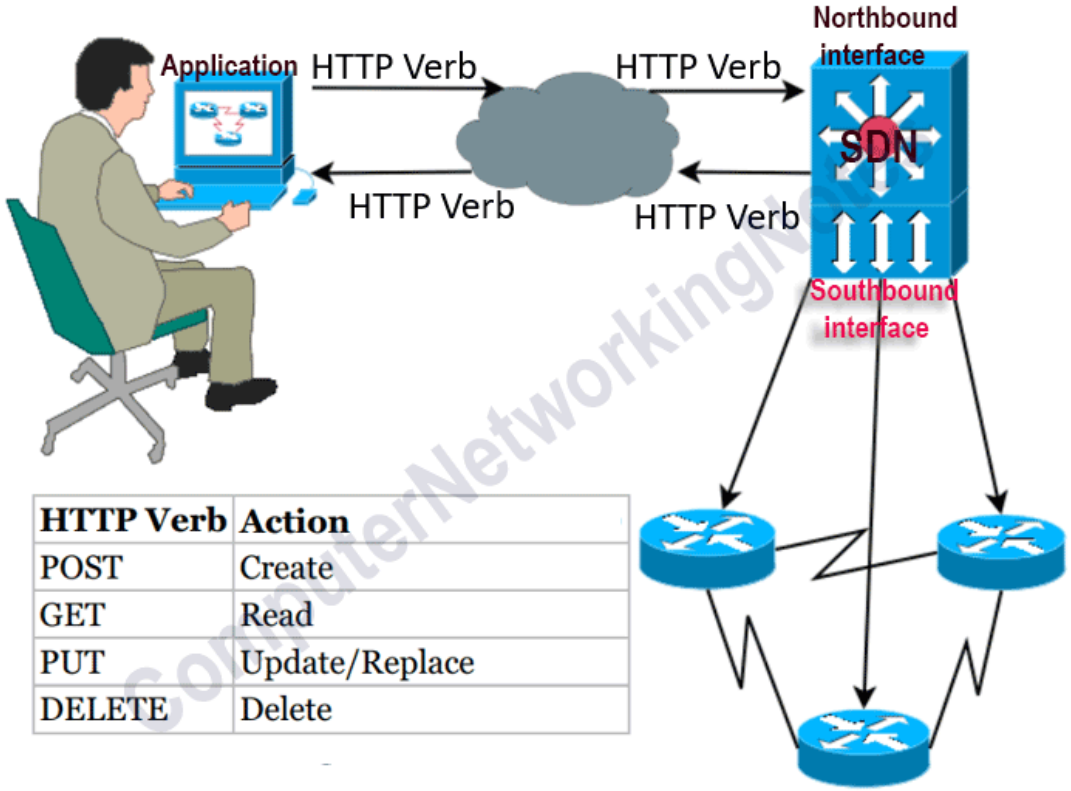
HTTP Verb	Action
POST	Create
GET	Read
PUT	Update/Replace
DELETE	Delete

- The controller sends back an HTTP GET response message with the object. The response message includes the information the application needs.
- The main difference between a browser and an SDN application's HTTP requests is the format of the data they retrieve. Browsers receive web pages, while SDN applications receive structured data.

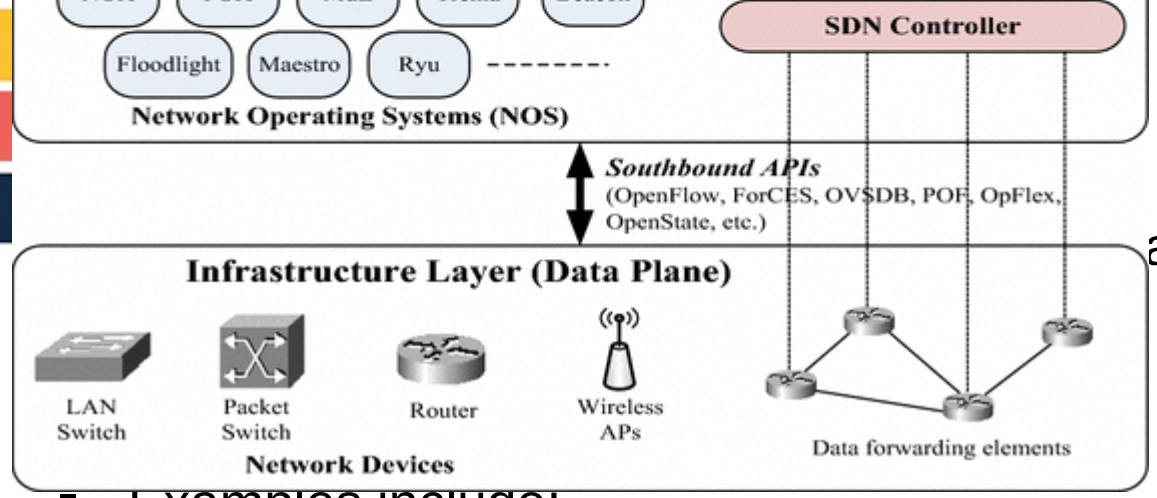
SDN (Software Defined Network)

SDN Architecture : Network OS

- The software platform that runs on the SDN controller.
- **Role:**
  - Provides the **abstraction layer** of the underlying network devices.
  - Offers services like topology discovery, statistics collection, and device management.
  - Acts as the **middleware** between hardware (data plane) and applications (application plane).
- **Examples:**
  - OpenDaylight's Karaf container, ONOS (Open Network Operating System).



HTTP Verb	Action
POST	Create
GET	Read
PUT	Update/Replace
DELETE	Delete

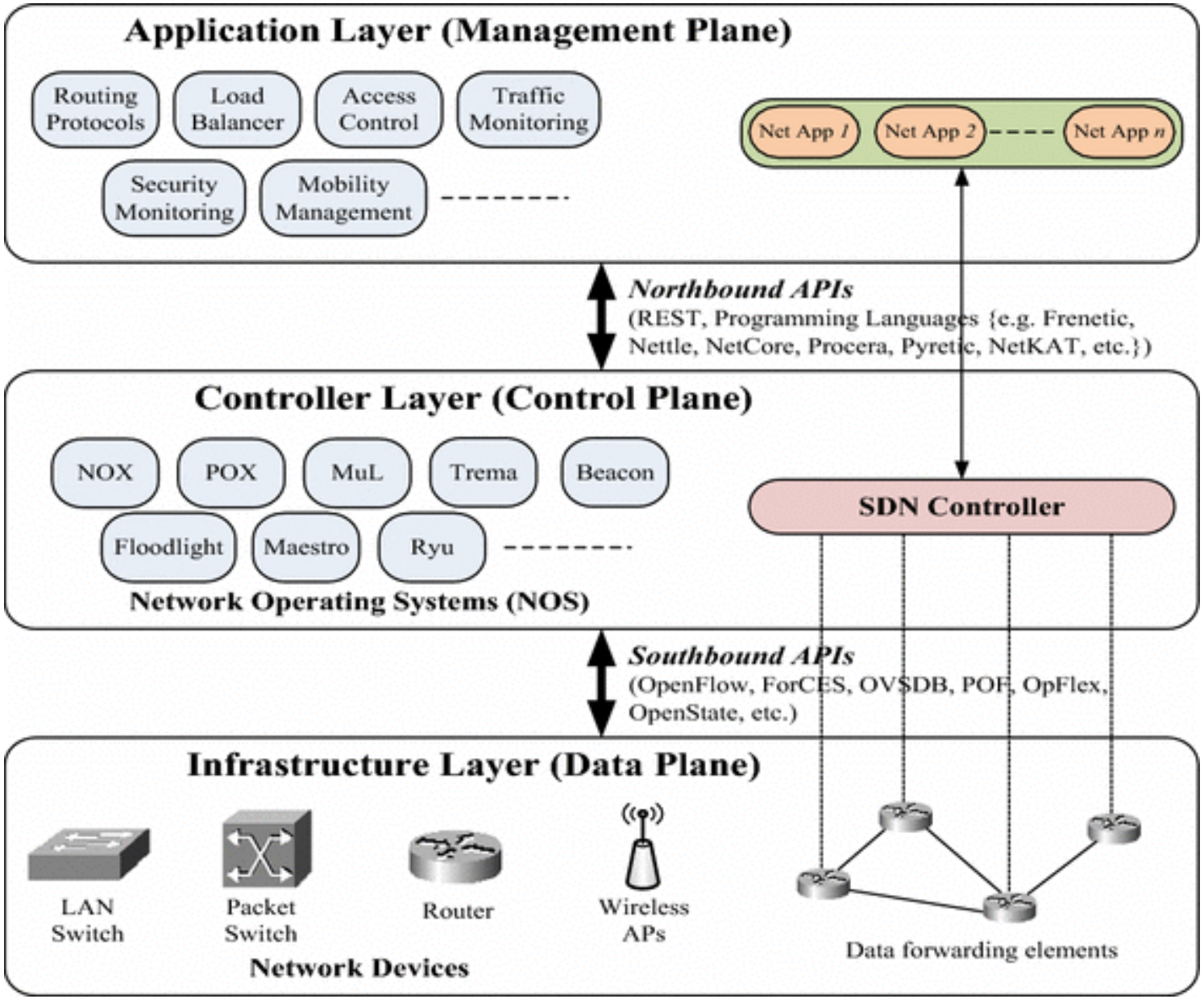


applications run.

program the network behavior.

- Examples include:
  - **Load Balancing Apps** (distribute traffic efficiently)
  - **Firewall Apps** (security enforcement)
  - **Traffic Engineering Apps** (optimize network performance)
  - **Monitoring Apps** (collect stats and visualize network health)

SDN (Software Defined Network)



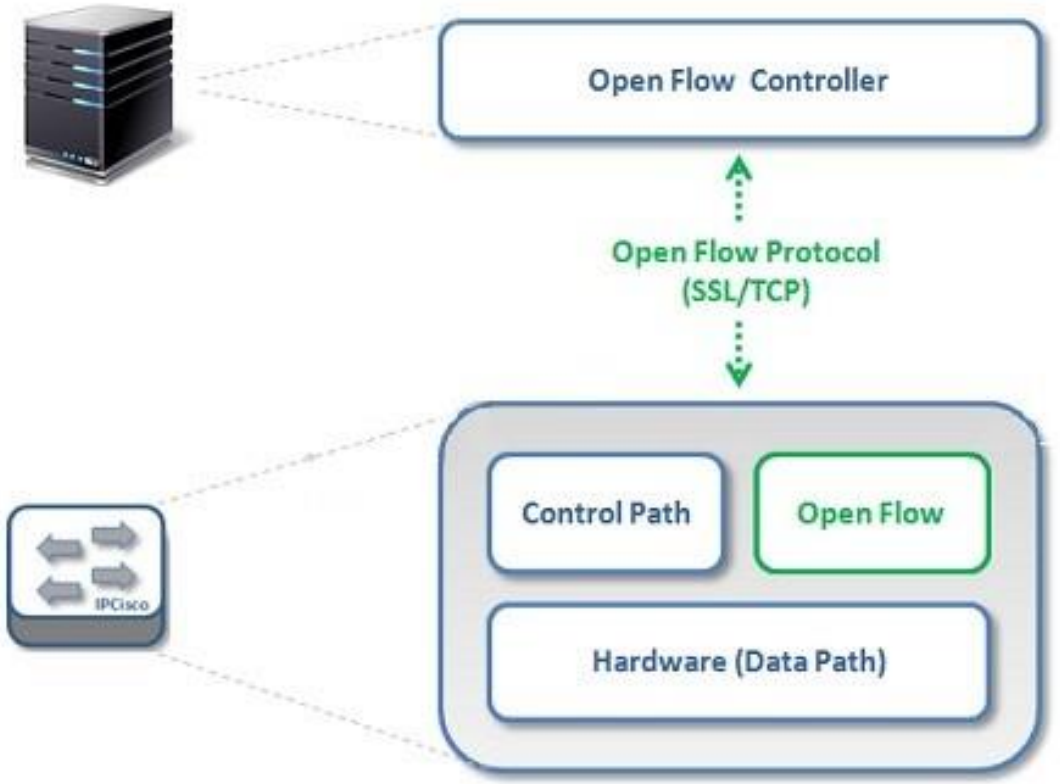
SDN Terminology

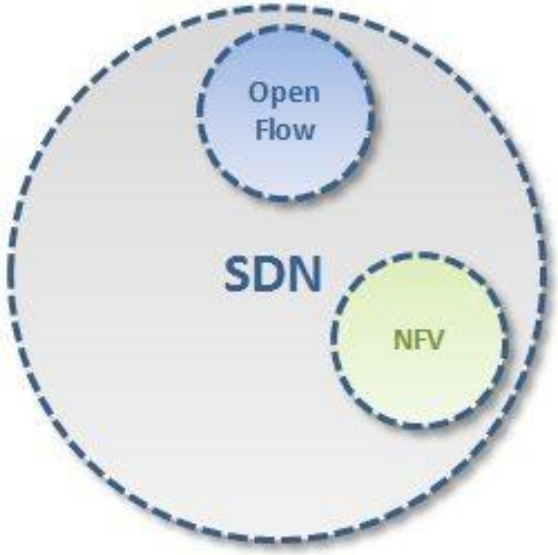
SDN Protocol Terms

- **NFV : Network Functions Virtualization.** Virtualization of the different skilled physical network devices with their virtual counterparts.
- **Flow :** Sequence of packet between the source and the destination.
- **Flow Rules :** Actions set for the flow.
- **The Flow Table :** Tables on the switch that handles the traffic flow.
- **Open Flow :** Southbound Interface **API**, that provide the communication of SDN Controller and SDN Data Plane devices.

Open Flow Overview

- Open Flow is a Standard based Layer 2 Communication protocol used between Controller and Switch in SDN.
- It allows to access to the Forwarding Plane of the network devices.
- Open Flow was standardized by a non-profit consortium, Open Networking Foundation (ONF) that has many network vendors as a member.
- This protocol had firstly developed for testing the new protocols in campus networks. But later, it has evolved to used in SDN.
- Generally, Open Flow is mixed with SDN. But this is not true, SDN is not Open Flow. Open Flow is a subset of SDN. Let's see the difference of these two terms.





Open Flow Overview

Key Components of OpenFlow

- **Flow Table** (inside the switch)
  - A set of **flow entries** that define what to do with packets.
  - Each entry has:
    - **Match fields** – like source IP, destination IP, TCP port, VLAN ID, etc.
    - **Counters** – how many packets/bytes matched.
    - **Actions** – forward to port, drop, modify header, etc.
- **SDN Controller**
  - A centralized software that decides network policies.
  - Installs, updates, or removes flow entries on switches.
- **OpenFlow Channel**
  - A secure communication path (often over TCP/TLS) between controller and switches.

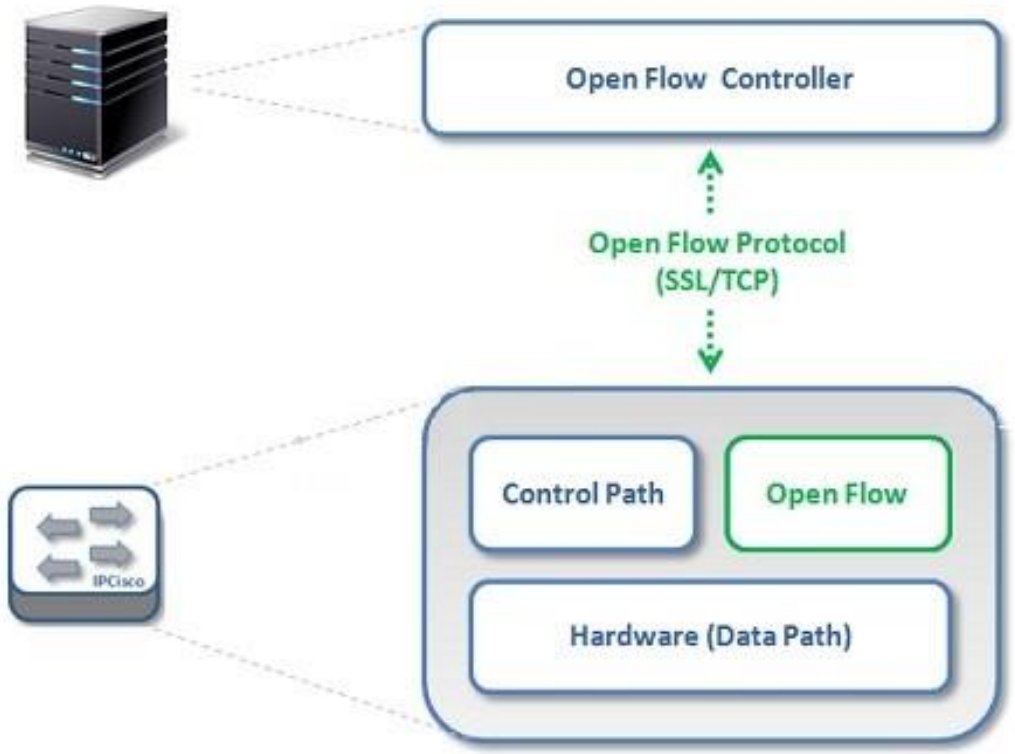
Open Flow Overview

How OpenFlow Works (Step-by-Step)

- **A packet arrives** at a switch.
- The switch **checks its flow table**:
  - If a matching rule is found → take the specified action (forward/drop/modify).
  - If **no match** → the switch sends a **Packet-In message** to the controller.
- **Controller decides** what to do and sends back a **Flow-Mod message**:
  - Installs a new rule in the switch's flow table.
  - Tells the switch what to do with that packet (and future packets in the same flow).
- **Switch applies the action** (forwards packet, drops it, etc.).

Open Flow Overview

- Open Flow is the protocol used in SDN, that is used to communicate forwarding plane and control plane of the network. In other words, the communication between Controller and the Network Devices are done with Open Flow.
- It allows operations, manipulations on Network Devices over Open Flow Interface.
- **Open Flow** has some common roles. A nice summary of these roles are given below:
  - Open Flow allows separation of control and data planes in the network.
  - It provides centralization of the control.
  - Open Flow uses Flow based control mechanism.
  - Takes advantage routing tables in Ethernet switches and routers.



NFV (Network Function Virtualization)

BACKGROUND

Traditional physical network hardware has always been difficult to change and upgrade. Introducing a software patch or rolling out a new service on a physical network can take months to complete. This is both time consuming and costly.

As a result, Over-The-Top, OTT operators have been able to gain a significant market foothold as they only utilize the network and do not inherit its problems. The OTT operators can launch a software-based internet platform that can be deployed almost immediately and run over the top of existing hardware.

NFV, network functions virtualization started in with mobile telecommunications networks with the promise of making networks more flexible and far easier to upgrade and change.

The concept of network functions virtualization was first introduced this concept in November 2012 as part of the ETSI ISG to provide hardware-related CAPEX and OPEX reductions.

As more developments have been made with NFV, service agility has become one of the main drivers for the development of network functions virtualization

NFV (Network Function Virtualization)

Network Functions Virtualization, NFV is an approach to telecommunications networking where the network entities that traditionally used dedicated hardware items are now replaced with computers on which software runs to provide the same functionality.

By running a network based around NFV, Network Functions Virtualization techniques, it is easier to expand and modify the network, and it is able to provide considerably more flexibility as well as being able to standardize on much of the hardware as it consists of additional computing power. In this way costs can be considerably reduced.

NFV (Network Function Virtualization)

NFV, network functions virtualization is a concept that virtualizes major elements of a network. In this way, rather than having a dedicated item of hardware to provide a given function, software running on a computer/server is used.

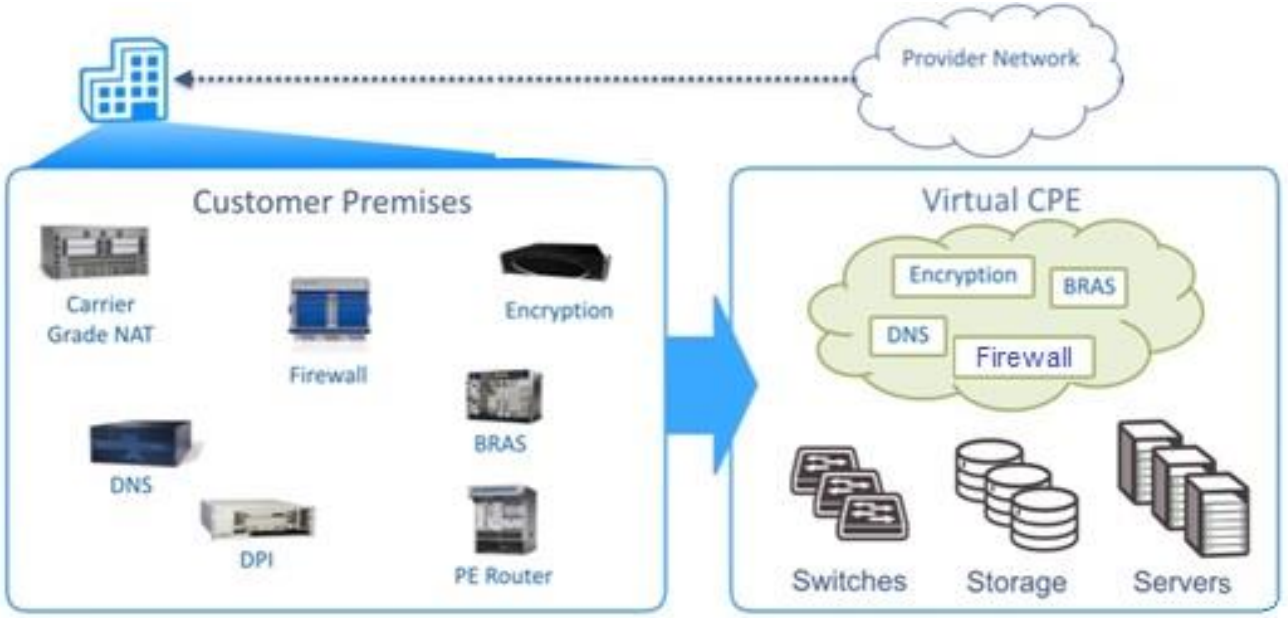
In this way entire classes of network node functions can be set up as building blocks that can be connected to create overall telecommunications networks.

NFV utilizes traditional server virtualization, but extends the concept significantly. In this way one or more virtual machines running different software and providing different processes, on top of industry standard high volume servers, are able to provide the functions of switches and storage, or even cloud computing infrastructure, instead of having custom hardware appliances for each network function.

Examples of the virtualized functions that can be provided include: virtualized load balancers, firewalls, intrusion detection devices, WAN accelerators, routers, access control and billing.

NFV (Network Function Virtualization)

The role of NFV is to relocate network functions from dedicated appliances to generic servers. NFV allows network operators to implement network policy without worrying about where to place the functions in the network and how to route traffic through these functions.



NFV moves all the functions as listed below in a common x86 architecture using virtualization.
• Router • Firewall • Web server • Load Balancer • Switch • Media Server

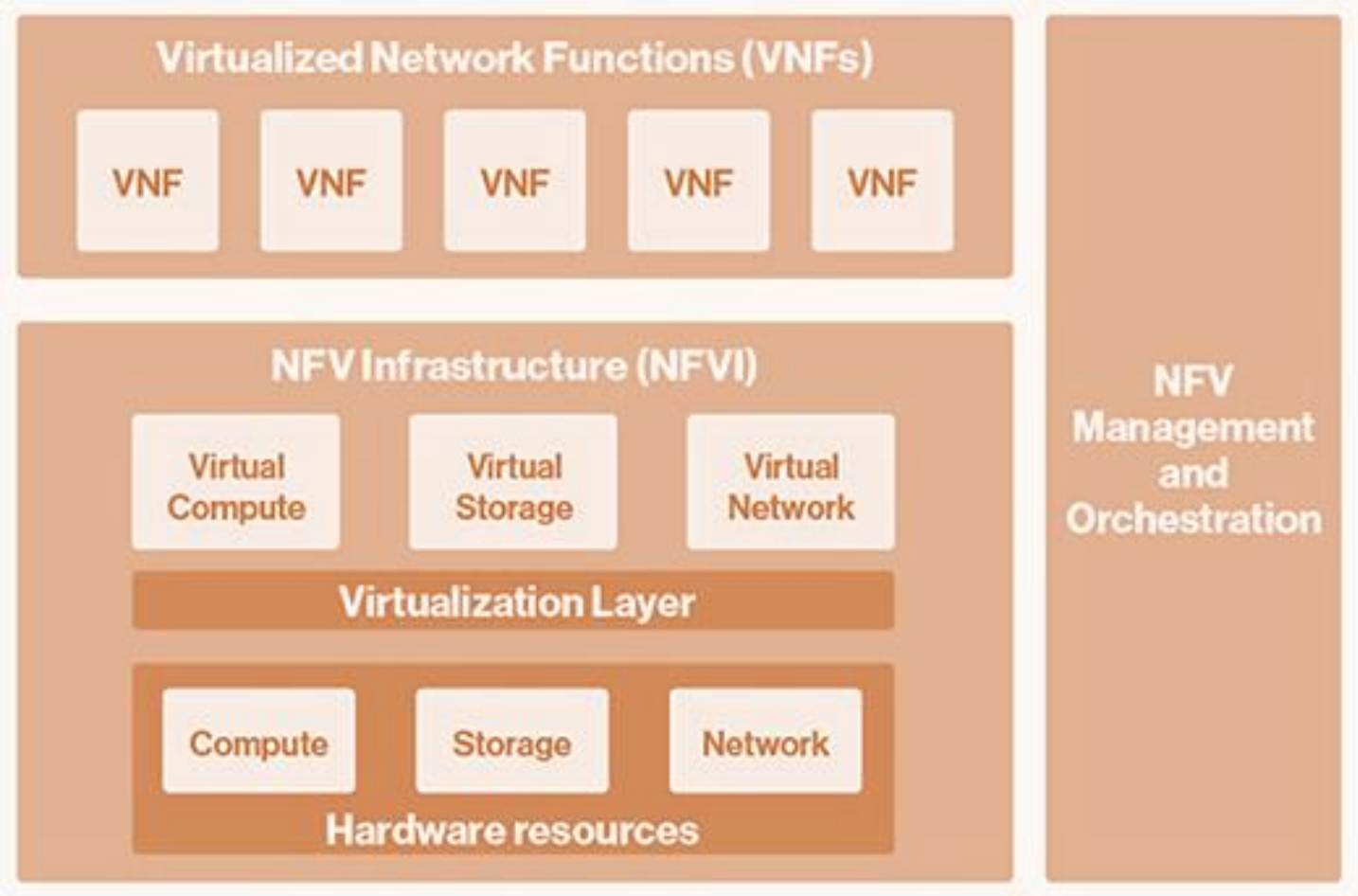
NFV helps in achieving following benefits for the operators.
• Standard hardware • Less complex • very flexible • reduced power • lower CapEx • lower OpEx • Test new applications • Low risk • Reduced TTM • Open market to software suppliers

NFV Framework

NFV FRAMEWORK

As with any system, a network using NFV techniques can be broken down into a number of elements. Those for Network Functions Virtualization are:

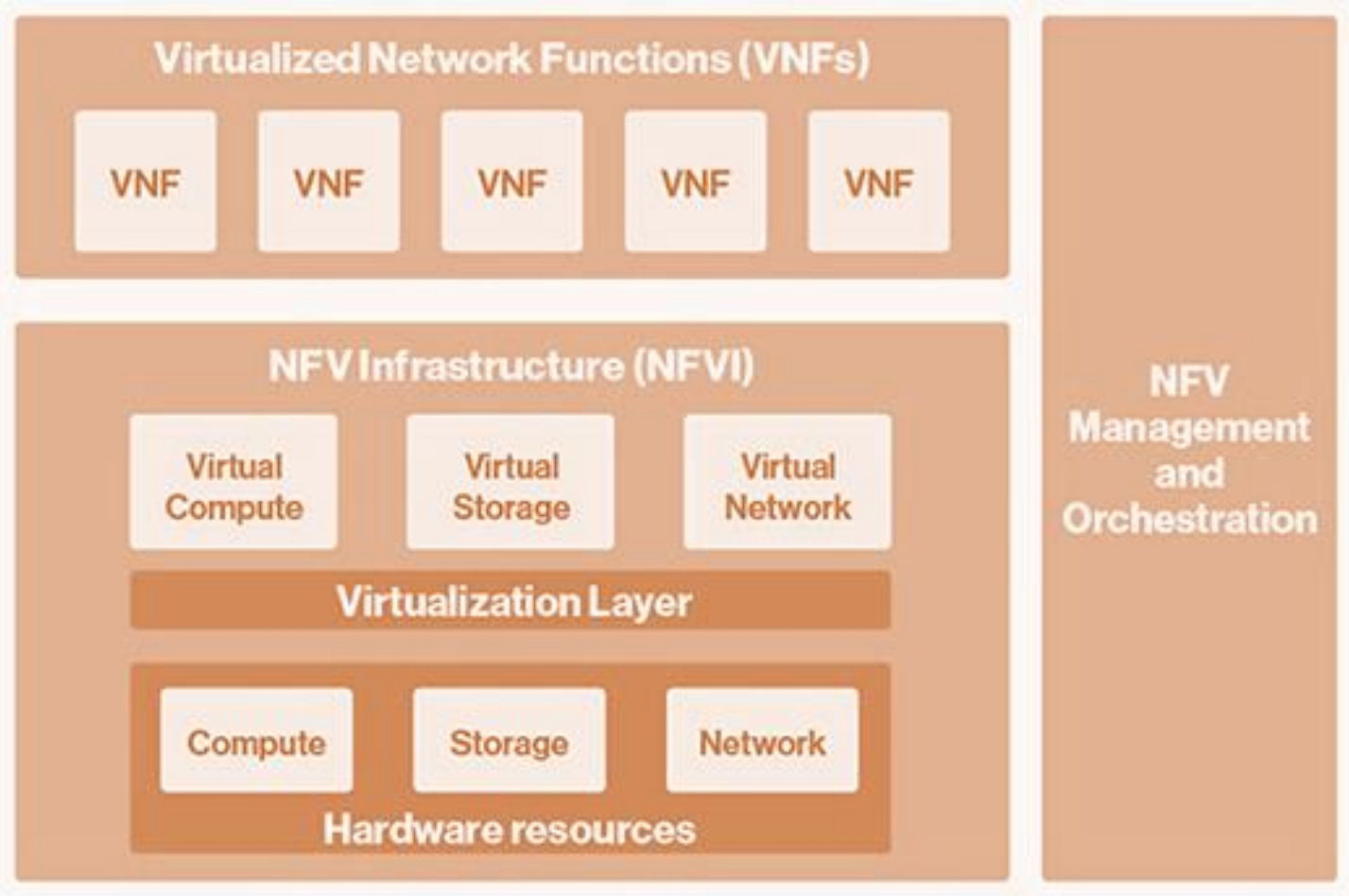
Virtualized Network Functions, VNF: The virtualized network functions comprise the software used to create the various network functions in their virtualized format. These are then deployed onto the hardware, i.e. the Network Function Virtualization Infrastructure.



NFV Framework

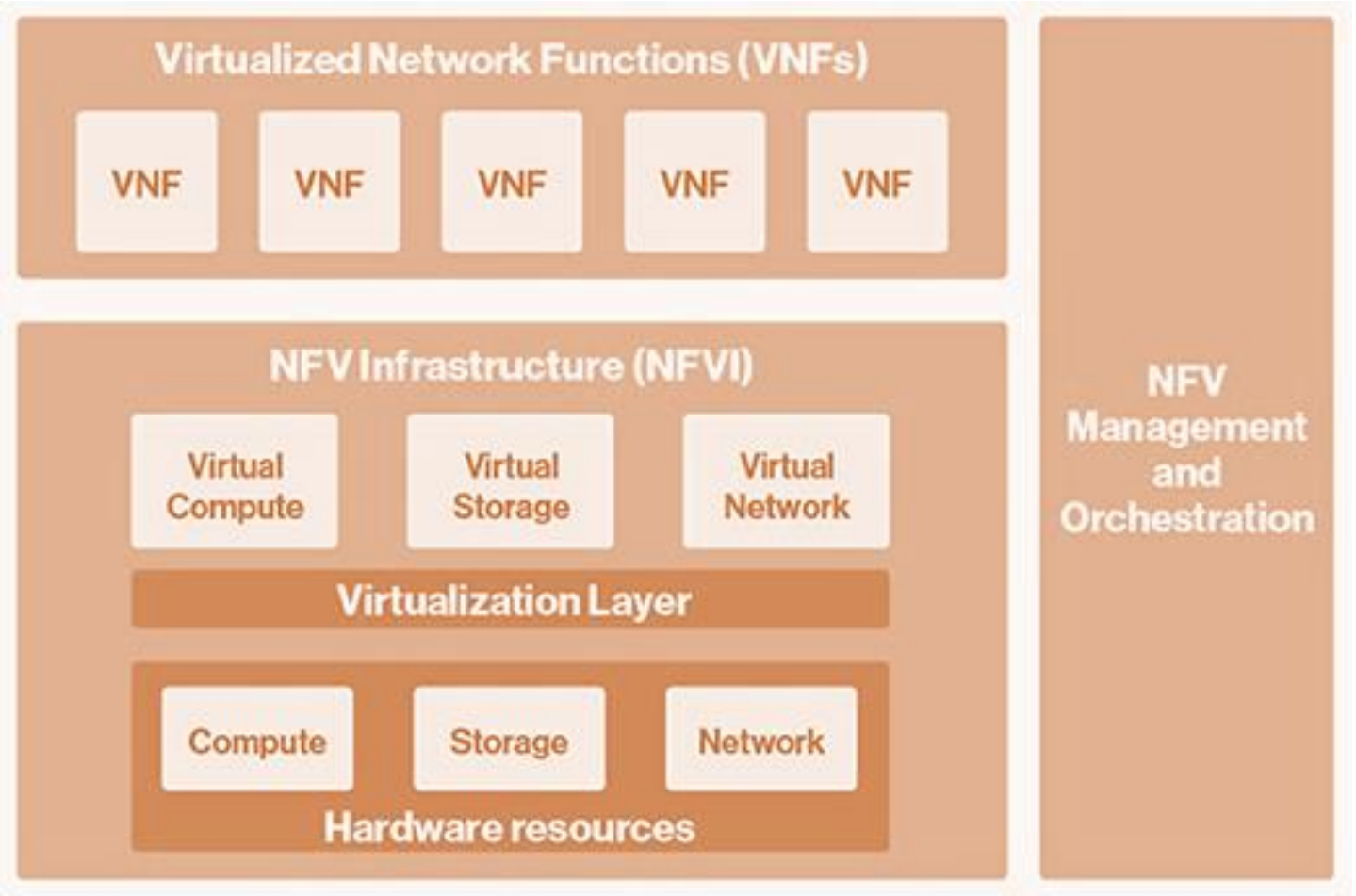
Network function virtualization infrastructure, NFVI: The NFVI consists of all the hardware and software components which are contained within the environment in which VNFs are deployed.

One of the advantages of NFV is that the NFV-Infrastructure, NFVI can be located across several physical locations, allowing operators to typically place their centers at the most convenient locations. The network providing connectivity between these locations is part of the NFV-Infrastructure.



NFV Framework

Network functions virtualization management and orchestration architectural framework, NFV-MANO Architectural Framework: NFV-MANO consists of the various functional blocks in whatever form that enables the exchange information, manipulation and storage needed manage and run the NFVI and VNFs, the network to operate correctly and provide significant improvements in efficiency and performance over other forms of network.



The NFVI and the NFV-MANO areas of the network are built within the overall NFV platform. This platform implements carrier-grade features used to manage and monitor the various components, recover from failures and provide effective security. These functions are all needed to run a public carrier network.

VNF (Virtualized Network Functions)

- Virtualized network functions or VNFs are the software realizations of the various network functions that can be deployed on a Network Functions Virtualization Infrastructure and needed to enable the network to operate.

- In this way, a virtual network function or VNF handles a specific network function that run on one or more virtual machines on top of the hardware networking infrastructure.

- The individual virtual network functions, VNFs, can be considered to be building blocks and they can be connected or combined together, providing all the capabilities required to provide a complete networking communication service.

VNF (Virtualized Network Functions)

Examples of various virtual network functions can be found within all areas of a telecommunications network and they can include:

- Switching: BNG, CG-NAT, routers.
- Tunneling gateway elements: IPSec/SSL VPN gateways.
- Traffic analysis: DPI, QoE measurement.
- Signaling: SBCs, IMS.
- Application-level optimization: CDNs, load Balancers.
- Home routers and set top boxes.
- Mobile network nodes: HLR/HSS, MME, SGSN, GGSN/PDN-GW, RNC.
- Network-wide functions: AAA servers policy control, charging platforms.
- Security functions: firewalls, intrusion detection systems, virus scanners, spam protection.

In this way, it can be seen that a huge number of VNFs can be run on a network using Network Functions Virtualization.

SDN Vs VNF

Features	SDN	NFV
Focus or major role	SDN focuses on data center.	NFV focuses on service providers or operators.
Strategy	It splits the control and data forwarding planes.	It replaces hardware network devices with software.
protocol	Uses OpenFlow	Not finalized yet, does support OpenFlow
Where the applications will run?	Applications run on industry standard servers or switches	Applications run on industry standard servers
Prime initiative supporters	Vendors of enterprise networking software and hardware.	Telecom service providers or operators.
Business initiator	Corporate IT	Service provider
Customer benefit or end user benefit	Drives down complexity and cost and increases agility.	Drives down complexity and cost and increases agility.
Initial applications	Cloud orchestration and networking	Routers, Firewalls, Gateways, CDN, WAN Accelerators, SLA Assurance
Formalization body	Open Networking Foundation (ONF)	ETSI NFV Working Group