Module 6 : Network Industry Trends & Automation

Introduction to software defined networking, OPENFLOW Why network automation? Simplified Architectures, Deterministic outcomes, Business Agility, Types of network automation, Device Provisioning, Data collection, Migrations, Configuration Management, Compliance, Reporting, Troubleshooting, Evolving from the management plane from SNMP to device APIs--- Impact of open networking, Network Automation in the SDN era. Introduction to Internet of Things (IoT): Definition and characteristics of IoT, Physical design of IoT: Things in IoT, IoT Protocols.

Q.Explain Software Define Networking

SDN stands for Software Defined Network which is a networking architecture approach. It enables the control and management of the network using software applications. Through Software Defined Network (SDN) networking behavior of the entire network and its devices are programmed in a centrally controlled manner through software applications using open APIs.

To understand software-defined networks, we need to understand the various planes involved in networking.

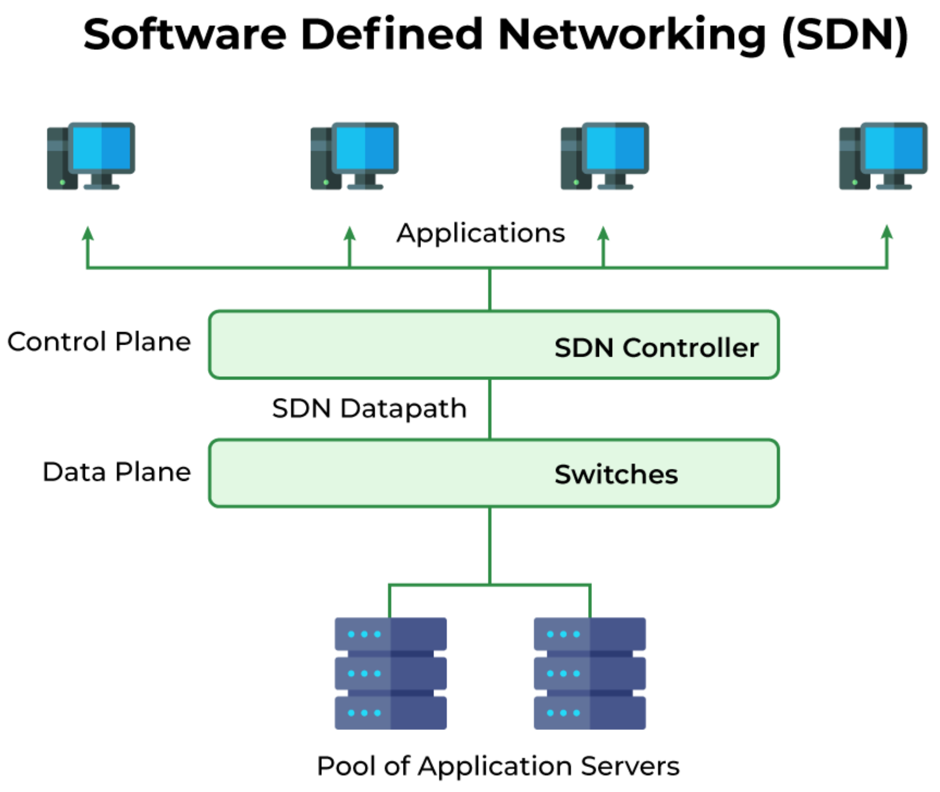
1. Data Plane
2. Control Plane

Data plane: All the activities involving as well as resulting from data packets sent by the end-user belong to this plane. This includes:

- Forwarding of packets.
- Segmentation and reassembly of data.
- Replication of packets for multicasting.

Control plane: All activities necessary to perform data plane activities but do not involve end-user data packets belong to this plane. In other words, this is the brain of the network. The activities of the control plane include:

- Making routing tables.
- Setting packet handling policies.



Software Defined Networking

Why SDN is Important?

- **Better Network Connectivity:** SDN provides very better network connectivity for sales, services, and internal communications. SDN also helps in faster data sharing.
- **Better Deployment of Applications:** Deployment of new applications, services, and many business models can be speed up using Software Defined Networking.
- **Better Security:** Software-defined network provides better visibility throughout the network. Operators can create separate zones for devices that require different levels of security. SDN networks give more freedom to operators.
- **Better Control with High Speed:** Software-defined networking provides better speed than other networking types by applying an open standard software-based controller.

In short, it can be said that- SDN acts as a “Bigger Umbrella or a HUB” where the rest of other networking technologies come and sit under that umbrella and get merged with another platform to bring out the best of the best outcome by decreasing the traffic rate and by increasing the efficiency of data flow.

Where is SDN Used?

- Enterprises use SDN, the most widely used method for application deployment, to deploy applications faster while lowering overall deployment and operating costs. SDN allows IT administrators to manage and provision network services from a single location.
- Cloud networking software-defined uses white-box systems. Cloud providers often use generic hardware so that the Cloud data center can be changed and the cost of CAPEX and OPEX saved.

Components of Software Defining Networking (SDN)

The three main components that make the SDN are:

1. SDN Applications: SDN Applications relay requests or networks through SDN Controller using API.
2. SDN controller: SDN Controller collects network information from hardware and sends this information to applications.
3. SDN networking devices: SDN Network devices help in forwarding and data processing tasks.

SDN Architecture

In a traditional network, each switch has its own data plane as well as the control plane. The control plane of various switches exchange topology information and hence construct a forwarding table that decides where an incoming data packet has to be forwarded via the data plane. Software-defined networking (SDN) is an approach via which we take the control plane away from the switch and assign it to a centralized unit called the SDN controller. Hence, a network administrator can shape traffic via a centralized console without having to touch the individual switches. The data plane still resides in the switch and when a packet enters a switch, its forwarding activity is decided based on the entries of flow tables, which are pre-assigned by the controller. A flow table consists of match fields (like input port number and packet header) and instructions. The packet is first matched against the match fields of the flow table entries. Then the instructions of the corresponding flow entry are executed. The instructions can be forwarding the packet via one or multiple ports, dropping the packet, or adding headers to the packet. If a packet doesn't find a corresponding match in the flow table, the switch queries the controller which sends a new flow entry to the switch. The switch forwards or drops the packet based on this flow entry.

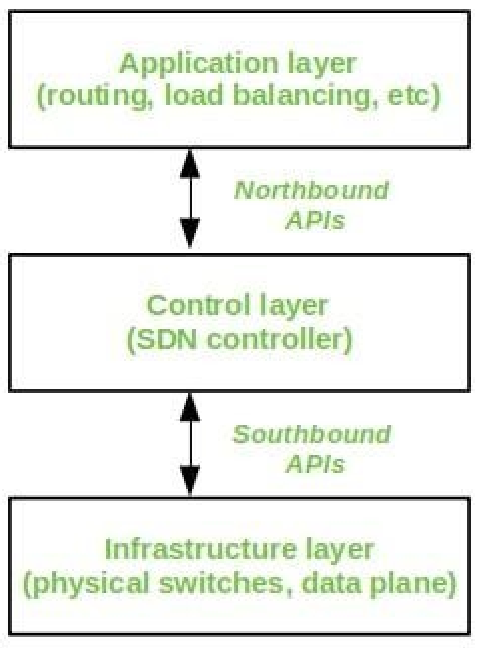
A typical SDN architecture consists of three layers.

• Application layer: It contains the typical network applications like intrusion detection, firewall, and load balancing

• Control layer: It consists of the SDN controller which acts as the brain of the network. It also allows hardware abstraction to the applications written on top of it.

• Infrastructure layer: This consists of physical switches which form the data plane and carries out the actual movement of data packets.

The layers communicate via a set of interfaces called the north-bound APIs(between the application and control layer) and southbound APIs(between the control and infrastructure layer).



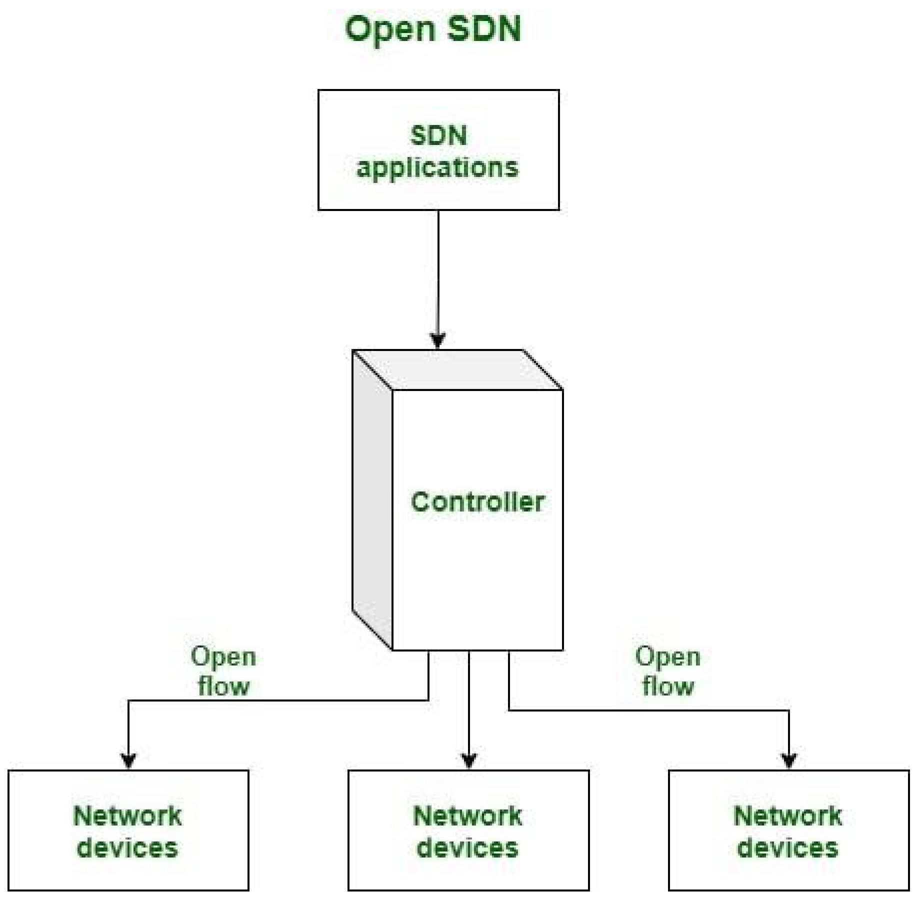
SDN Architecture

Different Models of SDN

There are several models, which are used in SDN:

1. Open SDN
2. SDN via APIs
3. SDN via Hypervisor-based Overlay Network
4. Hybrid SDN

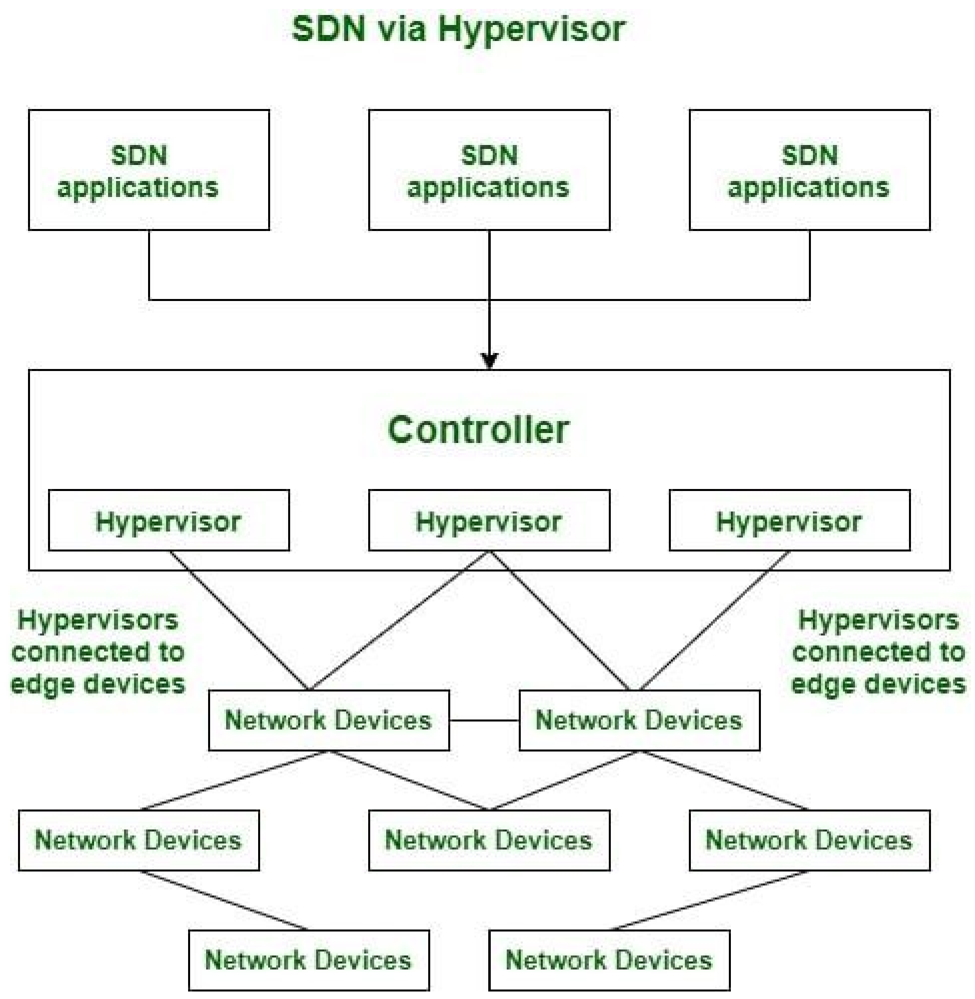
1. Open SDN: Open SDN is implemented using the OpenFlow switch. It is a straightforward implementation of SDN. In Open SDN, the controller communicates with the switches using south-bound API with the help of OpenFlow protocol.



Open SDN

2. SDN via APIs: In SDN via API, the functions in remote devices like switches are invoked using conventional methods like SNMP or CLI or through newer methods like Rest API. Here, the devices are provided with control points enabling the controller to manipulate the remote devices using APIs.

3. SDN via Hypervisor-based Overlay Network: In SDN via the hypervisor, the configuration of physical devices is unchanged. Instead, Hypervisor based overlay networks are created over the physical network. Only the devices at the edge of the physical network are connected to the virtualized networks, thereby concealing the information of other devices in the physical network.

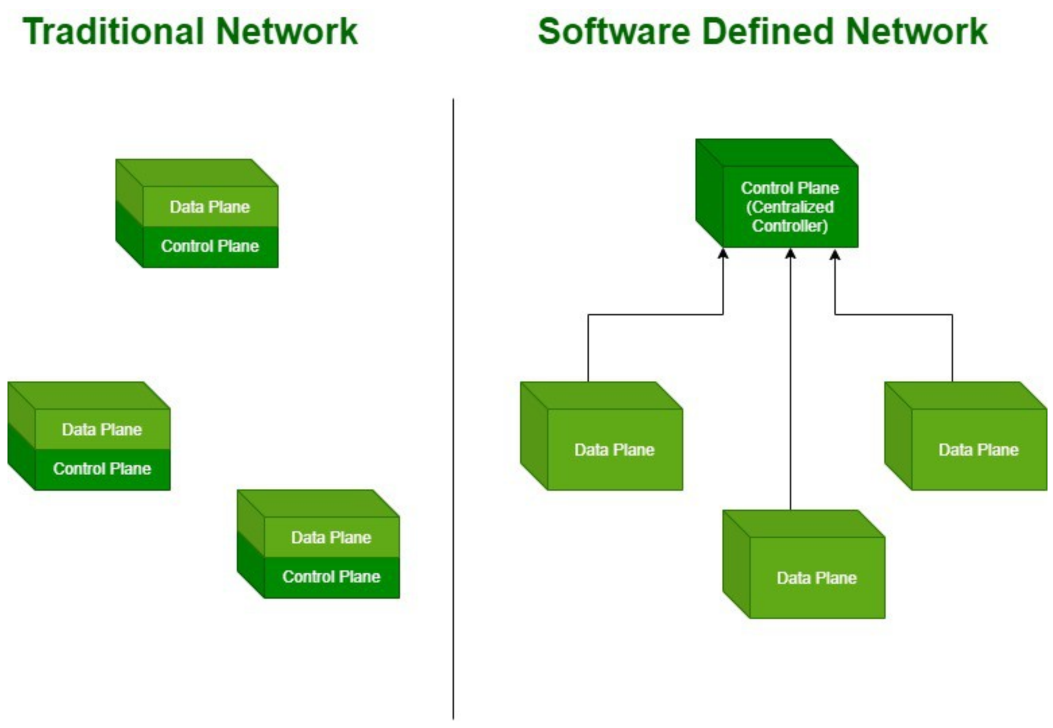


SDN via Hypervisor-based Overlay Network

4. Hybrid SDN: Hybrid Networking is a combination of Traditional Networking with software-defined networking in one network to support different types of functions on a network.

Difference between SDN and Traditional Networking

Software Defined Networking	Traditional Networking
Software Defined Network is a virtual networking approach.	A traditional network is the old conventional networking approach.
Software Defined Network is centralized control.	Traditional Network is distributed control.
This network is programmable.	This network is nonprogrammable.
Software Defined Network is the open interface.	A traditional network is a closed interface.
In Software Defined Network data plane and control, the plane is decoupled by software.	In a traditional network data plane and control plane are mounted on the same plane.



Difference between SDN and Traditional Networking

Advantages of SDN

- The network is programmable and hence can easily be modified via the controller rather than individual switches.
- Switch hardware becomes cheaper since each switch only needs a data plane.
- Hardware is abstracted, hence applications can be written on top of the controller independent of the switch vendor.
- Provides better security since the controller can monitor traffic and deploy security policies. For example, if the controller detects suspicious activity in network traffic, it can reroute or drop the packets.

Disadvantages of SDN

- The central dependency of the network means a single point of failure, i.e. if the controller gets corrupted, the entire network will be affected.
- The use of SDN on large scale is not properly defined and explored.

Q. Write a short note on Open FloW

Software Defined Networking (SDN), often referred to as a revolutionary new idea in computer networking, promises to dramatically simplify network control, management, and enable innovation through network programmability.

The separation of the forwarding hardware from the control logic allows easier deployment of new protocols and applications, straightforward network visualization and management, and consolidation of various middle boxes into software control. Instead of enforcing policies and running protocols on a convolution of scattered devices, the network is reduced to simple forwarding hardware and the decision-making network controller(s). The forwarding hardware consists of the following:

1. A flow table containing flow entries consisting of match rules and actions that take on active flows.
2. A transport layer protocol that securely communicates with a controller about new entries that are not currently in the flow table.

While OpenFlow has received a considerable amount of industry attention, it is worth mentioning that the idea of programmable networks and decoupled control plane (control logic) from data plane has been around for many years. The Open Signaling Working Group (OPENSIG) initiated a series of workshops in 1995 to make ATM, Internet, and mobile networks more open, extensible, and programmable. Motivated by these ideas, an Internet Engineering Task Force (IETF) working group came up with General Switch Management Protocol (GSMP), Network initiative proposed the idea of a network infrastructure that would be programmable for customized services. However, Active Network never gathered critical mass, mainly due to practical security and performance concerns.

Characteristics of the Internet of Things

The Internet of Things (IoT) is characterized by the following key features that are mentioned below.

1. Connectivity

Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connect, this should be guaranteed at all times. For example, the connection between people through Internet devices like mobile phones, and other gadgets, also a connection between Internet devices such as routers, gateways, sensors, etc.

2. Intelligence and Identity

The extraction of knowledge from the generated data is very important. For example, a sensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.

3. Scalability

The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as an outcome is enormous, and it should be handled appropriately.

4. Dynamic and Self-Adapting (Complexity)

IoT devices should dynamically adapt themselves to changing contexts and scenarios. Assume a camera meant for surveillance. It should be adaptable to work in different conditions and different light situations (morning, afternoon, and night).

5. Architecture

IoT Architecture cannot be homogeneous in nature. It should be hybrid, supporting different manufacturers ' products to function in the IoT network. IoT is not owned by anyone engineering branch. IoT is a reality when multiple domains come together.

6. Safety

There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also be at risk. Therefore, equipment safety is also critical.

For more, refer to Challenges to IoT.

7. Self Configuring

This is one of the most important characteristics of IoT. IoT devices are able to upgrade their software in accordance with requirements with a minimum of user participation. Additionally, they can set up the network, allowing for the addition of new devices to an already-existing network.

8. Interoperability

Interoperability is critical for the success of IoT, as it enables different devices and systems to work together seamlessly and provides a seamless user experience. Without interoperability, IoT systems would be limited to individual silos of data and devices, making it difficult to share information and create new services and applications.

To achieve interoperability, IoT devices, and systems use standardized communication protocols and data formats. These standards allow different devices to understand and process data in a consistent and reliable manner, enabling data to be exchanged between devices and systems regardless of the technology used.

Explain Challenges in IoT

Reliability

The system should be perfectly working and delivering all of its specifications correctly. It is a very critical the requirement in applications that requires emergency responses.

In IoT applications, the system should be highly reliable and fast in collecting data, communicating them, and making decisions, and eventually, wrong decisions can lead to disastrous scenarios.

Management

Managing all These devices and keeping track of the failures, configurations, and performance of a such large number of devices is a challenge in IoT.

Providers should manage Faults, Configuration, Accounting, Performance, and Security (FCAPS) of their interconnected devices and account for each aspect.

Interoperability

Interoperability means that heterogeneous devices and protocols need to be able to inter-work with each other.

This is challenging due to the large number of different platforms used in IoT systems. Interoperability should be handled by both the application developers and the device manufacturers to deliver the services regardless of the platform or hardware specification used by the customer.

Mobility

IoT devices need to move freely and change their IP address and networks based on their location. Thus, the routing protocol, such as RPL has to reconstruct the DODAG each time a node goes off the network or joins the network which adds a lot of overhead.

In addition, mobility might result in a change of service provider which can add another layer of complexity due to service interruption and changing gateway.

Explain IoT Architecture

IoT architecture can be explained in either three or five layers. Many agree that the three-layer model is simple to understand but leaves out some key details regarding how the data is used. Below is a breakdown of each layer, what it does, and why it’s important.

Perception

The perception layer represents the physical IoT devices themselves. This can include health monitors, lighting systems, autonomous vehicles, robotics, and security systems. Each IoT device collects data that requires processing.

> IoT sensors are inexpensive and can collect hundreds of different data points for processing. The data collected will vary depending on the goals of the organization. For example, agricultural IoT devices can record atmospheric data, moisture levels, solid temperature, and airflow to improve crop yield and revenue.

Transport

The transport layer is responsible for sending collected data to the cloud or edge device for processing. The transport layer relies on internet gateways to move data from the physical perception layer into the processing phase.

Administrators typically rely on cellular and Wi-Fi networks to move data through the transport layer. There are a few different technologies system admins can use during this stage:

- Cellular 4G LTE / 5G
- Wi-Fi

- Bluetooth
- Low-Power Wide-Area Networks

Administrators can design their IoT architecture with a mix and match of transport protocols. Ultimately the transport protocol you use should be able to reliably support data from the sensor to the closest internet gateway. We cover this concept in more detail in our IoT connectivity article.

Processing

Once the data reaches the cloud or edge device the server can transform this data into information. Modern IoT architectures leverage machine learning and artificial intelligence that create value by analyzing this data.

For example, if an IoT sensor is recording a high fluctuation in temperature, artificial intelligence can alert to this anomaly by monitoring the current temperature compared to its baseline. In this case, the server might be able to send a command to an HVAC unit to lower the temperature and resolve the issue.

Application

Processing typically occurs without human intervention, but humans still need to tell the server what to do when certain rules are met or thresholds are broken. The application layer is where administrators manage IoT device orchestration, create rule sets, and set service-level agreements for their IoT architecture.

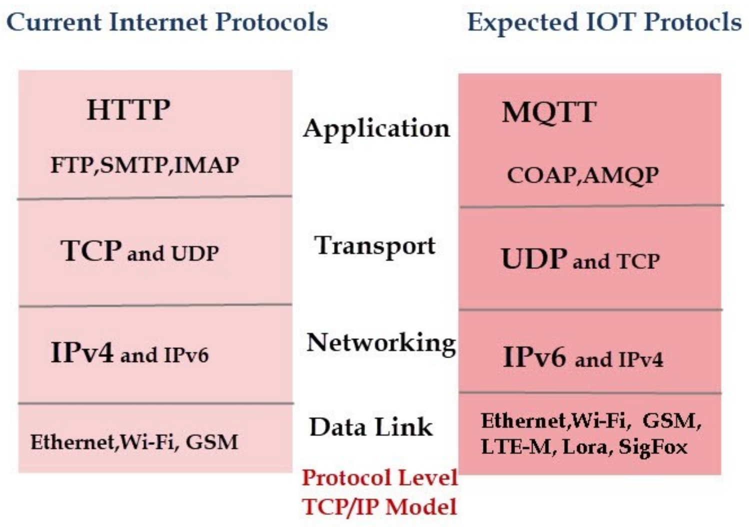
> If you’ve ever used an app to turn your lights on at home, you’ve used the application layer to do so. Reliable IoT architectures use the application layer to control and manage their networks from a centralized dashboard. This centralization reduces complexity, especially in enterprise IoT networks, which in turn improves efficiency and security.

Business

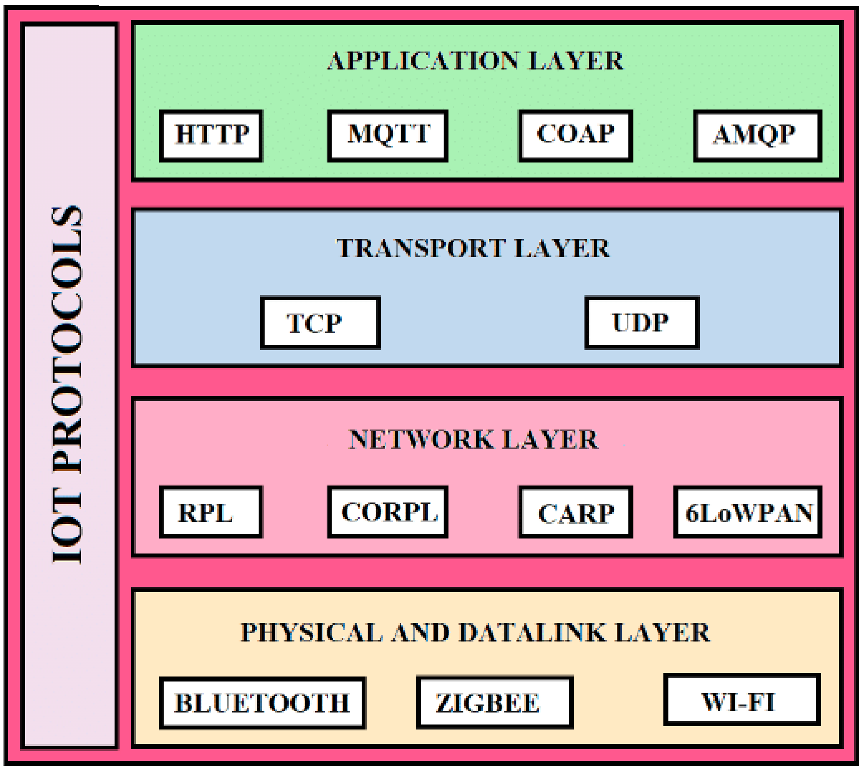
Finally, we arrive at the business layer, where information is transformed into business intelligence that drives decision-making. Stakeholders and executives can use the insights collected at the application layer to make better business decisions.

The business layer typically relies on reports and live dashboards for business intelligence. Information collected from the application layer can be enriched further at this level through other integrations. For example, business intelligence analysts can correlate cost savings based on electricity consumption before and after smart lighting sensors were installed.

Q. Define IoT. Explain IoT protocol layer diagram with protocols at each layer



IOT and Internet Protocols



Physical layer protocols: IoT network protocols connect medium to high power devices over the network. End-to-end data communication within the network is allowed using this protocol. HTTP, LoRaWAN, Bluetooth, Zigbee are a few popular IoT network protocols.

Network Layer Protocol:RPL : RPL is Routing protocol for Low Power Lossy Network. It is special routing protocol design for IOT to find the shortest path to reach to the destination

IoT Application Layer protocols: IoT data protocols connect low power IoT devices. Without any Internet connection, these protocols can provide end-to-end communication with the hardware. Connectivity in IoT data protocols can be done via a wired or cellular network. MQTT, CoAP, AMQP, XMPP, DDS are some popular IoT data protocols.

Message queue telemetry transport (MQTT) protocol

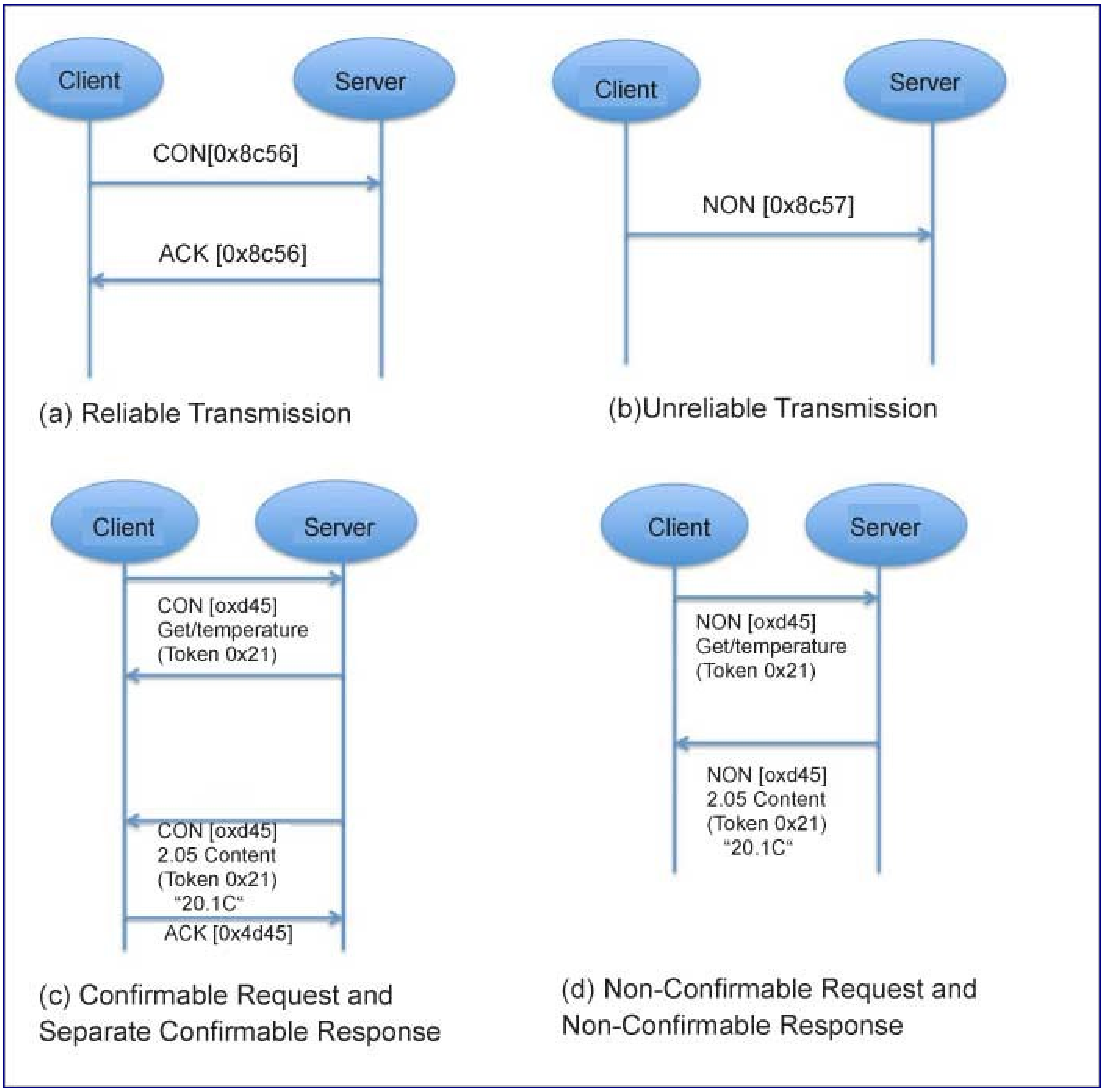
This open source publish/subscribe messaging transport protocol is very lightweight and ideal for connecting small devices to constrained networks. It was designed to work in low-bandwidth conditions, such as sensors and mobile devices, on unreliable networks. This capability makes it the most preferred protocol for connecting devices with small code footprint, as well as for wireless networks with varying levels of latency due to bandwidth constraints or unreliable connections. It works on top of transmission control protocol/Internet protocol (TCP/IP) to provide reliable delivery of data. MQTT has three main components:

- Subscriber
- Publisher
- Broker

The basic workflow of this protocol involves the publisher being responsible for generating and transmitting information to subscribers through a broker. The main function of the broker is to ensure security by checking the authorisation of subscribers and publishers. This protocol is preferred for IoT devices because it provides well-organised information routing functions to small, cheap, low memory, power devices and low bandwidth networks. To ensure message reliability, MQTT uses three levels of quality of service (QoS).

- *QoS0 (At most once)*: This is the least reliable but the quickest mode. The message is sent without any confirmation received.
- *QoS1 (At least once):* Ensures that the message is delivered at least once, even if duplicate messages may be received.
- *QoS2 (Exactly once):* This is the most reliable but also the most bandwidth-consuming mode. Duplicate messages are controlled to ensure that it gets delivered only once.

MQTT is a bi-directional communication protocol, where the clients can produce and consume data by publishing messages and subscribing to topics. Two-way communication enables IoT devices to send sensor data and simultaneously receive configuration information and control commands. Through MQTT, encrypting messages using TLS and verifying clients using modern authentication protocols becomes much easier.


Client
Server
CON[0x8c56]
ACK [0x8c56]
(a) Reliable Transmission
Client
Server
NON [0x8c57]
(b)Unreliable Transmission
Client
Server
CON [oxd45] Get/temperature (Token 0x21)
CON [oxd45] 2.05 Content (Token 0x21) "20.1C"
ACK [0x4d45]
(c) Confirmable Request and Separate Confirmable Response
Client
Server
NON [oxd45] Get/temperature (Token 0x21)
NON [oxd45] 2.05 Content (Token 0x21) "20.1C"
(d) Non-Confirmable Request and Non-Confirmable Response

Constrained application protocol (CoAP)

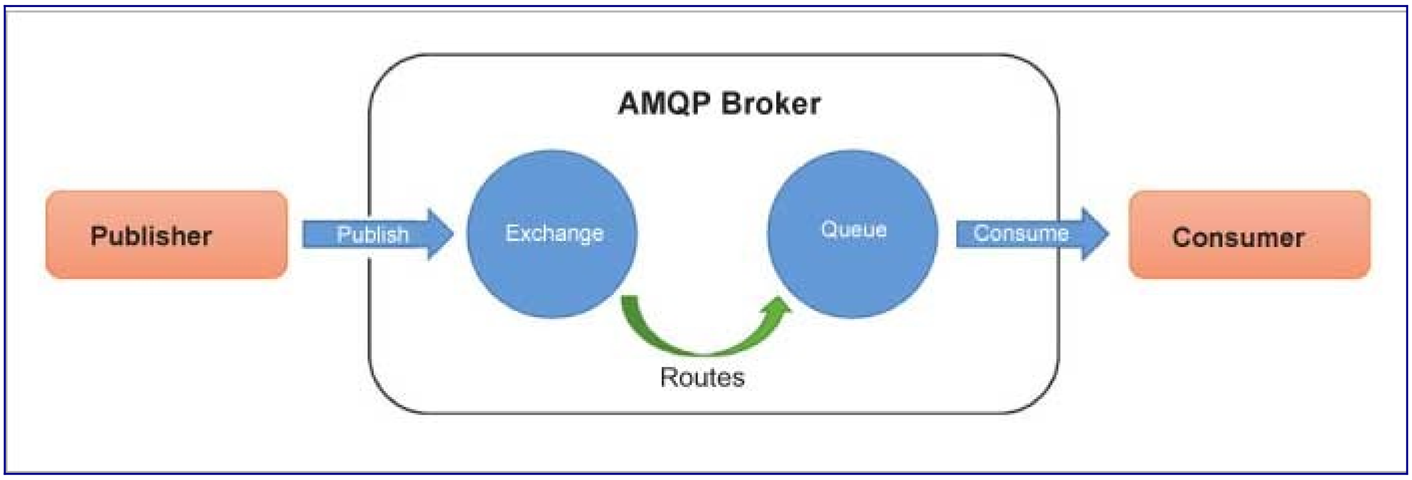
CoAP is a Web transfer protocol for constrained devices and networks in IoT. It can be implemented over a user datagram protocol (UDP), and is designed for applications with limited capacity to connect using light-weight machine-to-machine (LWM2M) communication — such as smart energy and building automation. LWM2M allows remote management of IoT devices, and provides interfaces to securely monitor and regulate them. CoAP architecture is based on the famous REST model, where servers make resources available under a URL, and clients access these resources using methods such as GET, PUT, POST, and DELETE. The CoAP and HTTP protocols have many similarities, with the difference that CoAP is enhanced for IoT, and more specifically for M2M. It has low overhead, along with proxy and caching capabilities, and exchanges messages asynchronously. The architecture of CoAP is divided into two main categories: messaging, which is responsible for the reliability and duplication of messages; and request/ response, which is responsible for communication.

The message layer is on top of the UDP, and is responsible for exchanging messages between the IoT devices and Internet. CoAP has four different types of messages — confirmable, non-confirmable, acknowledgment and reset. A confirmable message (CON) is a reliable message when exchanged between two endpoints. It is sent over and over again until the other end sends an acknowledge message (ACK). The ACK message has the same message ID as that of a CON. If the server faces issues managing the incoming request, it can send back a reset message (RST) instead of an ACK. For exchanging non-critical messages, unreliable NON messages can be used, where the server does not acknowledge the message. NON messages are assigned message IDs to detect duplicate messages.

Request/response is the second layer of the CoAP abstraction layer, which uses CON or NON messages to send requests. In scenarios where a server can respond immediately, a request is sent using a CON message, with an ACK message containing the response, or the error code is sent back by the server. The request and the response both have the same token, different from the message ID. In places where the server cannot respond immediately, it sends an ACK message with no content as the response. Once the response is ready, a new CON message containing the response is sent back to the client and the client acknowledges the response received.

Advanced message queuing protocol (AMQP)

AMQP is an open standard application layer protocol designed for higher security, reliability, easy provisioning, and interoperability. It is a connection-oriented protocol, which means the client and the broker need to establish a connection before they transfer data because TCP is used as a transport protocol. AMQP offers two levels of QoS for reliable message delivery — unsettle format (similar to MQTT QoS0) and settle format (similar to MQTT QoS1). The main difference between AMQP and MQTT standards is that in AMQP the broker is divided into two main components — exchange and queues. Exchange is responsible for receiving publisher messages and sending them to queues. Subscribers connect to those queues, which basically represent the topics, and receive the sensory data whenever they are available.



Features	CoAP	MQTT	AMQP
Messaging pattern	request/ response	publish-subscribe	request/response; publish-subscribe
Architecture	tree	tree	star
Transport	UDP	TCP, MQTT-S: UDP	TCP
Network layer	IPv6	IPv4 or IPv6	IPv4 or IPv6
QoS level	2 levels	3 levels	3 levels
Communication scope	Device-to-device	Device-to-cloud	Device-to-device, Device-to-cloud, Cloud-to-cloud
Addressing	URI	topic only	queue, topic/Routing Key
Security	DTLS, IPsec	TLS	SASL/TLS
Interoperability	Semantic	Foundational	Structural
[Broker/server] implementation language	Java, C, C#, C++, Erlang, Go, Python, JavaScript, Ruby, Rust, Swift	Java, C, C#, C++, JavaScript, Erlang, Go, Lua, Python	Java, C, C#, C++, Python, Ruby
Header size	4 bytes	2 bytes	8 bytes
Governing body	IETF	OASIS	OASIS

Table 1: A comparative analysis of IoT protocols

Network Automation

What is Network Automation

Network automation is the process of automating the configuring, managing, testing, deploying, and operating of physical and virtual devices within a network. With everyday network tasks and functions automated and repetitive processes controlled and managed automatically, network service availability improves.

Any type of network can use network automation. Hardware- and software-based solutions enable data centers, service providers, and enterprises to implement network automation to improve efficiency, reduce human error, and lower operating expenses.

Why automate your network?

One of the biggest issues for network managers is the growth of IT costs for network operations. The growth of data and devices is starting to outpace IT capabilities, making manual approaches nearly impossible. Yet up to 95 percent of network changes are performed manually, resulting in operational costs 2 to 3 times higher than the cost of the network. Increased IT automation, centrally and remotely managed, is essential for businesses to keep pace in the digital world.

Types of network automation

Any network type, including LANs, WANs, data center networks, cloud networks, and wireless networks, can always use automation. Automation is possible for any network resource that a CLI or API can manage.

Script-driven network automation

Script-driven network automation uses programming and scripting languages to carry out activities; ideally, these jobs should have exact triggers and repeatable processes. Due to their familiarity, legacy languages like Perl and Tcl are often used in network automation.

Newer open-source programming languages like Ansible, Python, and Ruby have gained traction as networks become more complicated because of their usability and versatility. Bash and Go are additional programming languages for network automation.

Software-based network automation

An administrator portal coordinates software-based network automation, also known as intelligent automation, which does away with the need to write commands manually. Typically, these systems offer templates for task creation and execution based on plain language rules.

Intent-based network automation

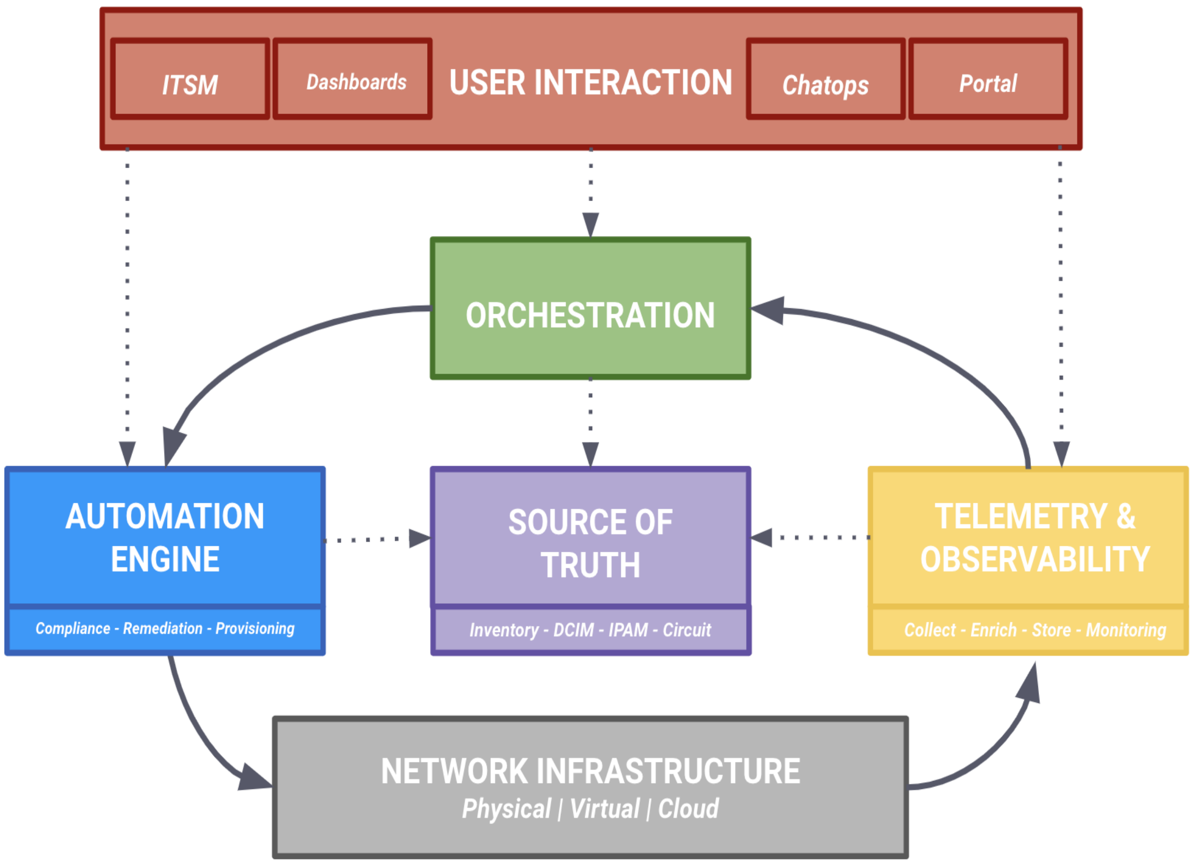
The most recent automation tools use machine learning and artificial intelligence (AI) to understand the user and business intent and automatically change how network policy is applied. Network administrators set performance service levels for users and applications. When these service levels are not met, the network automatically makes adjustments to restore acceptable performance levels for business-critical applications to the best of its ability.

Q. Explain Network Automation Architecture

A network automation solution is rooted in transforming (and improving) a network operation workflow, making some (or all) steps automated. Every step should have a purpose (if not, it’s time to drop it). This means that we will have a list of tasks/functionalities to be automated.

The goal of this architecture is to help you answer the question: Where should I start?

We propose an architecture with six blocks, each of which has a clear goal. This helps to follow a systematic approach. First, mapping the workflow steps into these blocks, and finally, finding the right tooling to implement them. Any block may contain multiple tools, or a tool can be applied to multiple blocks. Also it’s okay to skip blocks if there are no related tasks needed in a network automation workflow. This is only a reference framework.


ITSM
Dashboards
USER INTERACTION
Chatops
Portal
ORCHESTRATION
AUTOMATION ENGINE
Compliance - Remediation - Provisioning
SOURCE OF TRUTH
Inventory - DCIM - IPAM - Circuit
TELEMETRY & OBSERVABILITY
Collect - Enrich - Store - Monitoring
NETWORK INFRASTRUCTURE
Physical | Virtual | Cloud

We will cover these blocks in more detail throughout this blog series. Here, we simply introduce them, with a one-line description:

- **Network Infrastructure:** Includes all the components of your network: physical devices, virtual functions, and cloud services.
- **User Interactions:** Offers proper entry point for the network automation solutions.
- **Source of Truth:** Defines the intended state of the network.
- **Automation Engine:** Implements tasks related to changing the state of the network.
- **Telemetry and Observability:** Obtains and monitors operational state data.
- **Orchestration:** Connects all the automation steps to implement the desired workflows.