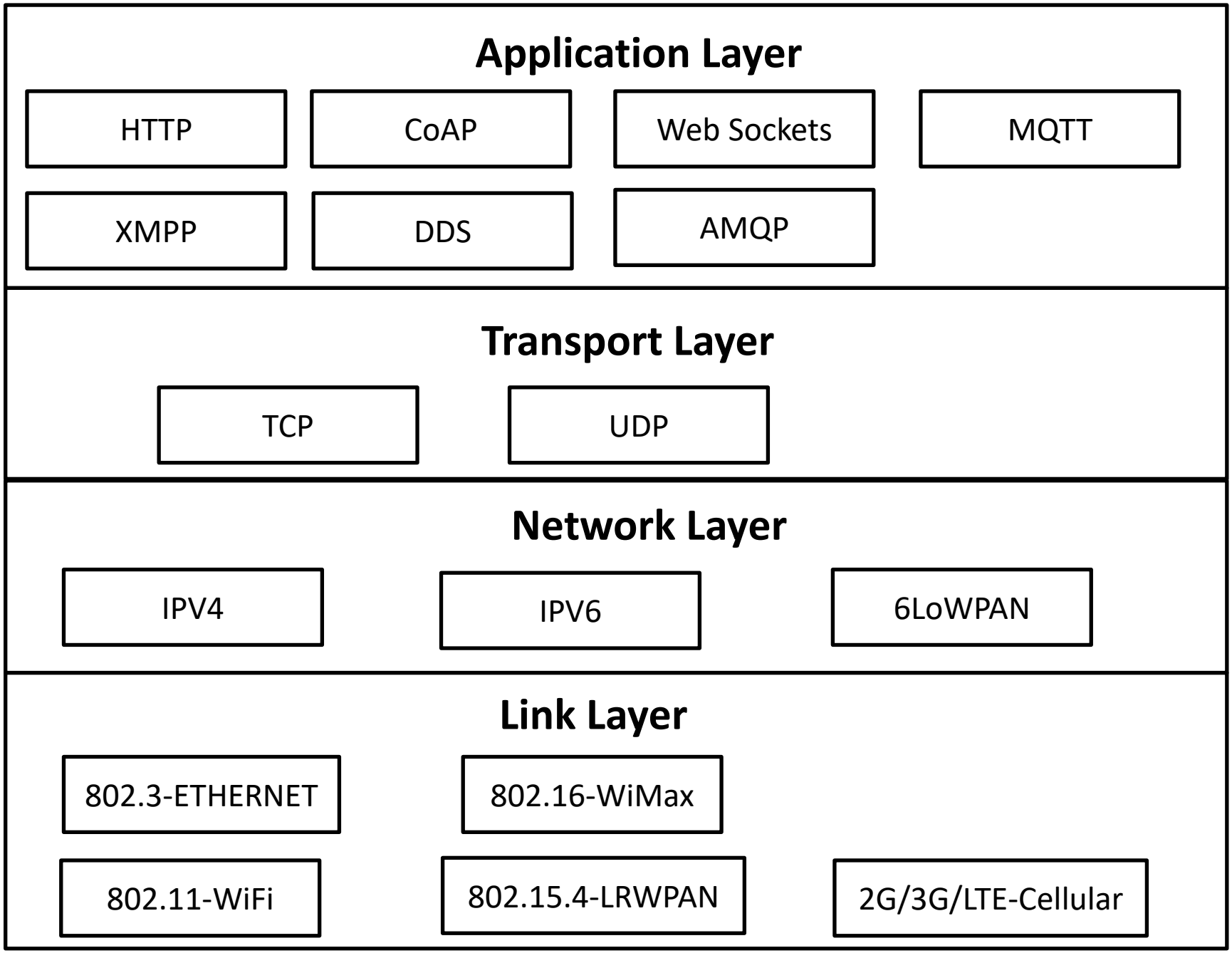
IoT protocols

- IoT protocols are a set of rules and standards that govern communication between devices on the Internet of Things (IoT).
- The main purpose of these technologies is to ensure smooth communication between different IoT devices, regardless of manufacturer or platform.
- They are the foundation of data transmission and are critical to the successful implementation of IoT projects.
- Without these proper protocols, IoT applications would not be able to communicate properly with each other.

Difference between conventional protocols and IoT Protocols

- While traditional protocols such as HTTP and FTP were designed for Internet applications, IoT standards are specifically tailored to the needs of IoT devices. These include range, data volume, and energy efficiency.
- Another difference is the type of communication. Traditional standards mainly use connection-oriented protocols such as TCP, however, IoT technologies support both connection-oriented and connectionless communication, allowing for flexible and adaptable data transfer in different IoT applications.
- IoT communication is based on a layered architecture that efficiently organizes data exchange across different protocols. Data exchange between devices goes through several layers from storage, to processing and to the user interface.

IoT Protocols



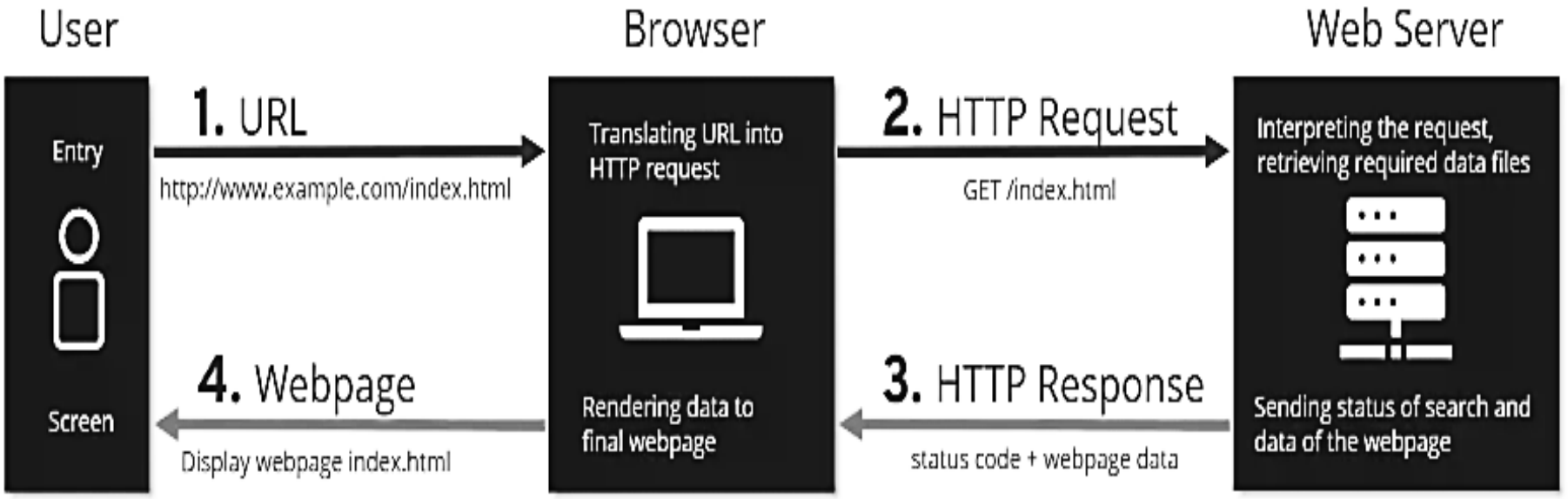
HTTP (hyper text transfer protocol)

- It is an application layer protocol.
- It is a client server protocol also known as request response protocol.
- http requests the webpage to the server and after getting response from the server it closes the connection.
- A web browser is an http client whereas a web server is the http server.

How HTTP Works?

- **Open Web Browser:** First, you open your web browser and type a website URL (e.g., www.example.com).
- **DNS Lookup:** Your browser asks a Domain Name System (DNS) server to find out the IP address associated with that URL. Think of this as looking up the phone number of the website.
- **Send HTTP Request:** Once the browser has the website's IP address, it sends an HTTP request to the server. The request asks the server for the resources needed to display the page (like text, images, and videos).
- **Server Response:** The server processes your request and sends back an HTTP response. This response contains the requested resources (like HTML, CSS, JavaScript) needed to load the page.
- **Rendering the Web Page:** Your browser receives the data from the server and displays the webpage on your screen.

- After the page is loaded, the connection between the browser and server is closed. If you request a new page, a new connection will be made.



CoAP (Constrained Application Protocol)

- CoAP is an IoT protocol.
- It is designed to join IoT devices through low bandwidth restricted network.
- CoAP is designed for M2M and IoT applications.
- It is an application layer protocol so follow request response model.
- CoAP runs over UDP protocol. (i.e. at transport layer UDP is present)
- It uses few resources than HTTP.
- CoAP clients can use GET, PUT and DELETE commands.

- **GET:** when client wants to get some data it uses GET command.
- **PUT:** when client wants to upload some data it uses PUT command.
- **DELETE:** if client wants to delete some data it uses DELETE command.

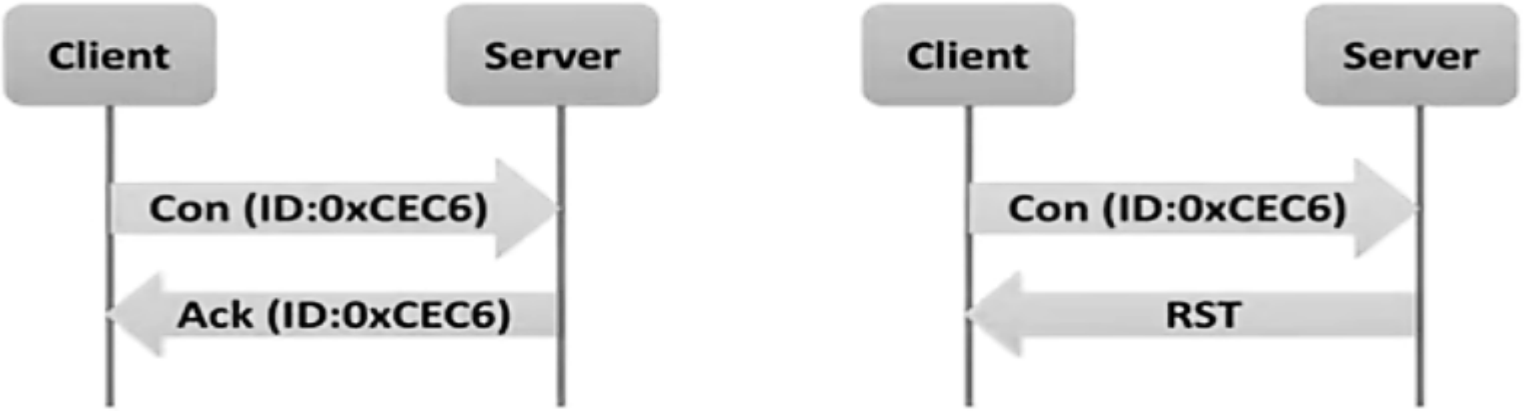
CoAP Layers

- **CoAP is divided into two layers.**

1. **Application Layer:** it is designed for communication based on request response model.
2. **UDP Layer or lower layer:** it is designed to deal with UDP and asynchronous messages.

Types of messages in CoAP

1. **Confirmable message or reliable message (con):** the client keep on sending message until the acknowledgement is received from server or the number of request are exhausted.



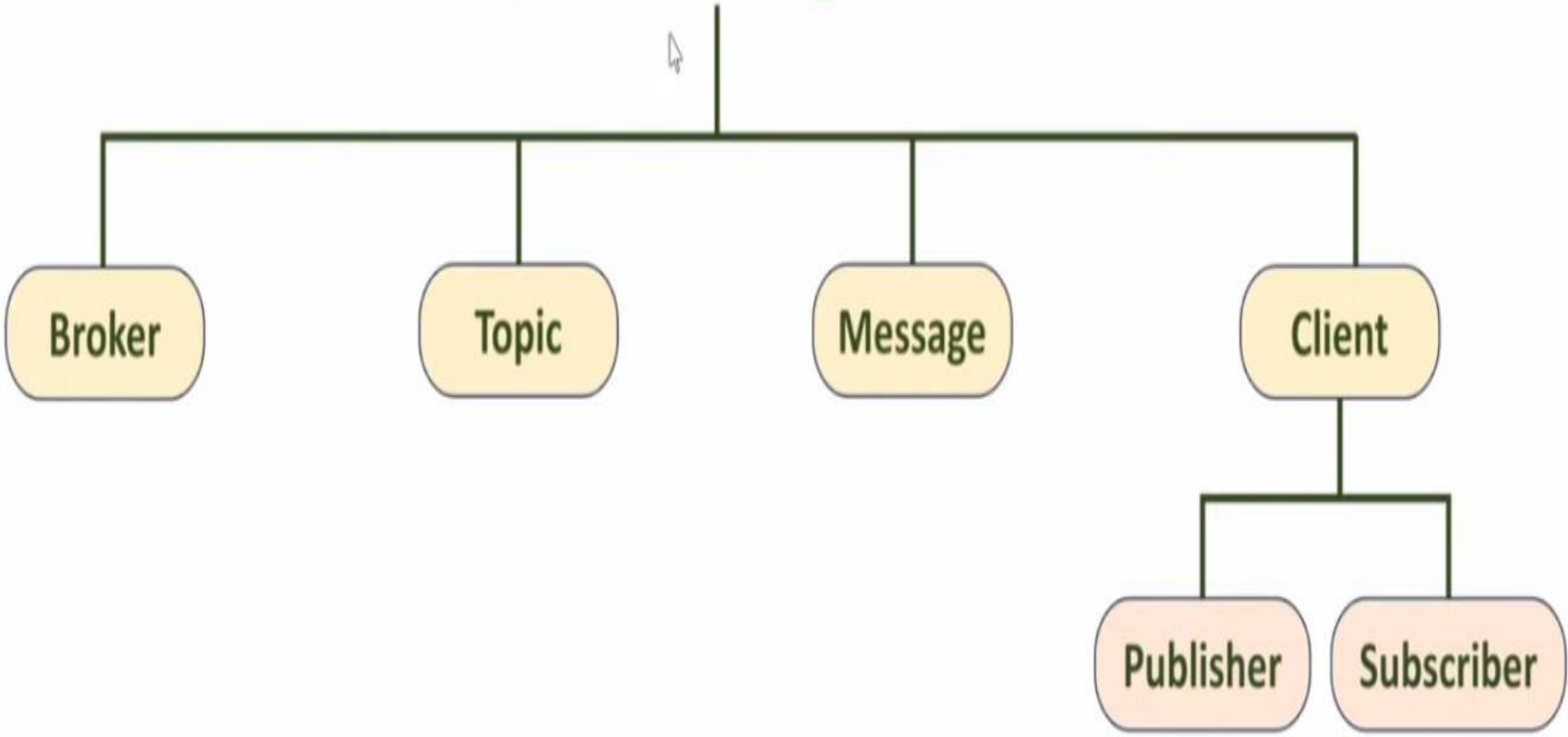
2. If server is having trouble in confirmation or receiving message it can send **Reset (RST) message.**
3. **Non-confirmable message (non):** these messages doesn't have important data so no need of acknowledgement is required.
4. **Acknowledgement message (ACK**): it is sent by the server to acknowledge the confirmable message.

*for acknowledgement, a confirmation ID of 2 byte is sent and the same id is received as a confirmation message.

MQTT (Message Queuing Telemetry Transport)

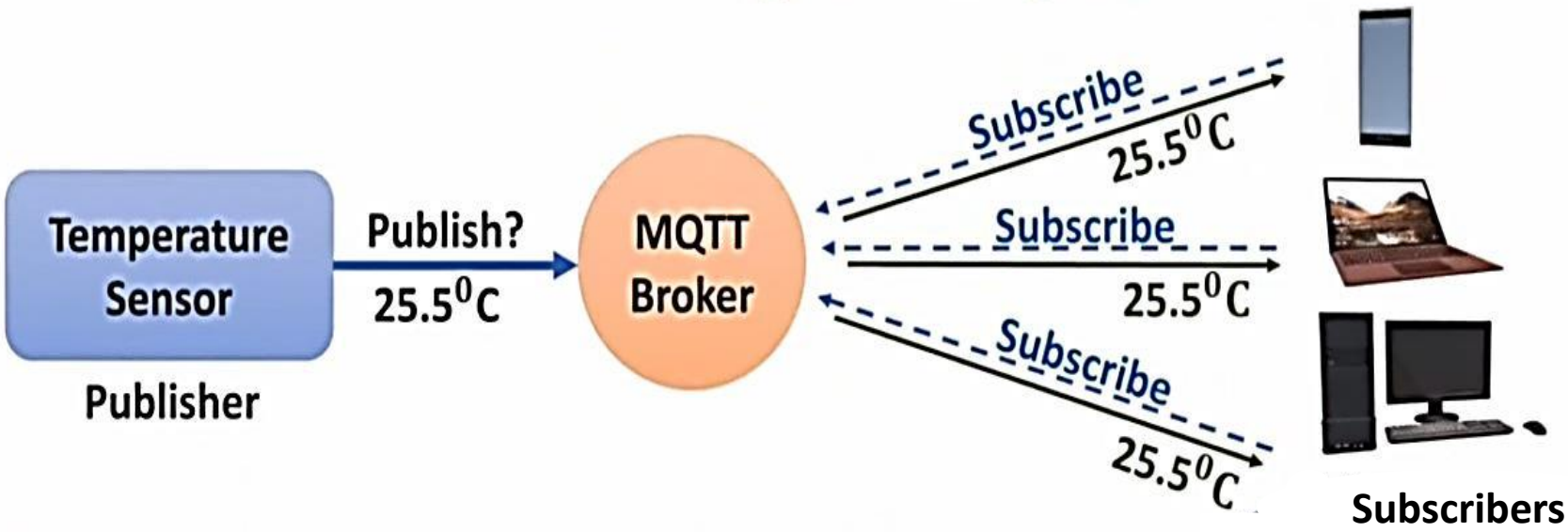
- MQTT – Message Queuing Telemetry Transport
- MQTT is used for M2M and IoT connectivity.
- It is a publish-subscribe-based messaging protocol that transports messages between devices.
- It is very lightweight and thus suited for M2M (Mobile to Mobile, WSN – Wireless Sensor Network) and IoT sensors, where communication nodes communicate with applications through the MQTT message broker.
- It usually runs over TCP/IP Protocol.
- MQTT was developed by IBM and Eurotech.
- MQTT is designed for limited devices, high latency, and low bandwidth communication.

MQTT Components



- **MQTT Client:** A client can be either the publisher or the subscriber. Subscriber can subscribe to topics and receive messages. Publishers can publish the topics.
- **MQTT Broker:** It is a central point of communication. It is responsible for dispatching all messages between clients. It receives subscription from clients on topics and based on subscriptions it forwards the messages to clients.
- **Topic:** Topic is an identifier for subscribers (temperature, pressure, humidity, gases etc.).
- **Message:** Subscribers receives messages based on topics of subscription from broker.

Working of MQTT



- Step – 1: Publishers can publish topics. {Ex. Temperature}
- Step – 2: Subscribers can subscribe to one or more topics.
- Step – 3: Publisher gives messages on topics to MQTT Broker.
- Step – 4: MQTT Broker publishes the messages to subscribed users.

MQTT Methods

Connect – The client requests a connection with the MQTT Broker.

Disconnect – The client requests a disconnection with the MQTT Broker.

Subscribe – The client subscribes to the topic.

Unsubscribe – The client unsubscribes from the topic.

Publish – The client publishes messages.

MQTT Advantages

- It is a simple protocol for IoT devices.
- It is a lightweight protocol, so it is easy to implement in software.
- Low Network usage.
- Low Power usage.
- Lower data rate for protocol implementation.

MQTT Applications

- Home Automation
- Factory Automation
- Medical & Healthcare
- Transport & Logistics

Comparison b/w CoAP & MQTT

Parameters	CoAP	MQTT
Full Form	❖ Constrained Application Protocol	❖ Message Queuing Telemetry Transport
Architecture Model	❖ Request Response Model	❖ Public Subscribe Model
Messaging Mode	❖ Asynchronous and Synchronous	❖ Asynchronous only
Communication Mode	❖ One to One	❖ Many to Many
Transport Layer	❖ Mainly UDP	❖ Mainly TCP
Header Size	❖ 4 Byte	❖ 2 Byte
Message Labelling	❖ Yes (2byte ID with message)	❖ No
LNN Effectiveness	❖ Excellent	❖ Low
Sorting of Data	❖ Universal Resource Identifier (URI)	❖ Broker Topic
Port Used	❖ 61631	❖ Secure (8883) and Nonsecure (1883)
Persistence Support	❖ No Support	❖ It supports live data Communication
Reliability	❖ It takes care of confirmable messages, non-confirmable messages, acknowledgments, and retransmissions.	❖ Three quality services: 1. Delivery not Guaranteed 2. Delivery confirmation 3. Delivery double confirmation

Web Sockets

- Unlike http, Web Socket is a persistent connection between the client and the server. It is a bi-directional full-duplex connection.
- It is used for real time live data communication.
- A web browser can upgrade its connection from http to Web Socket connection but for that the browser has to send an http request and if request agreed the client server will handshake to establish a Web Socket connection. E.g.. Fetching trading data in browser.

HTTP vs Web Sockets

- HTTP is stateless whereas Web Socket is stateful.
- HTTP is unidirectional whereas Web Socket is Bidirectional.
- HTTP is half-duplex and Web Socket is full-duplex.

HTTP

VS

Duplex
Half

Messaging pattern
Request-response model

Protocol
Stateless

Overhead
Each request and response has a moderate overhead (headers, cookies, etc.)

Scalability
Stateless protocols tend to scale well horizontally, although it depends on your situation.

Use cases
Retrieving static content and resources, implementing REST APIs.

Built-in features
Compression, multiplexing, authentication, and more.

Event-driven
No

Realtime updates
No

Bidirectional updates
No

WebSockets

Duplex
Full

Messaging pattern
Request-response model

Protocol
Stateful

Overhead
Low

Scalability
WebSockets are challenging to scale horizontally due to their stateful nature.

Use cases
Implementing realtime updates like live news updates or bidirectional communication such as chat or multiplayer collaboration.

Built-in features
None

Event-driven
Yes

Realtime updates
Purpose built for realtime updates.

Bidirectional updates
Yes

XMPP (eXtensible Messaging and Presence Protocol)

XMPP – eXtensible Messaging and Presence Protocol

It is a protocol (standard) that is used to build chat systems.

It uses XML to exchange data between client and server.

eXtensible: The protocol can be {has been} extended with new features.

Messaging: Send one-to-one and group messages.

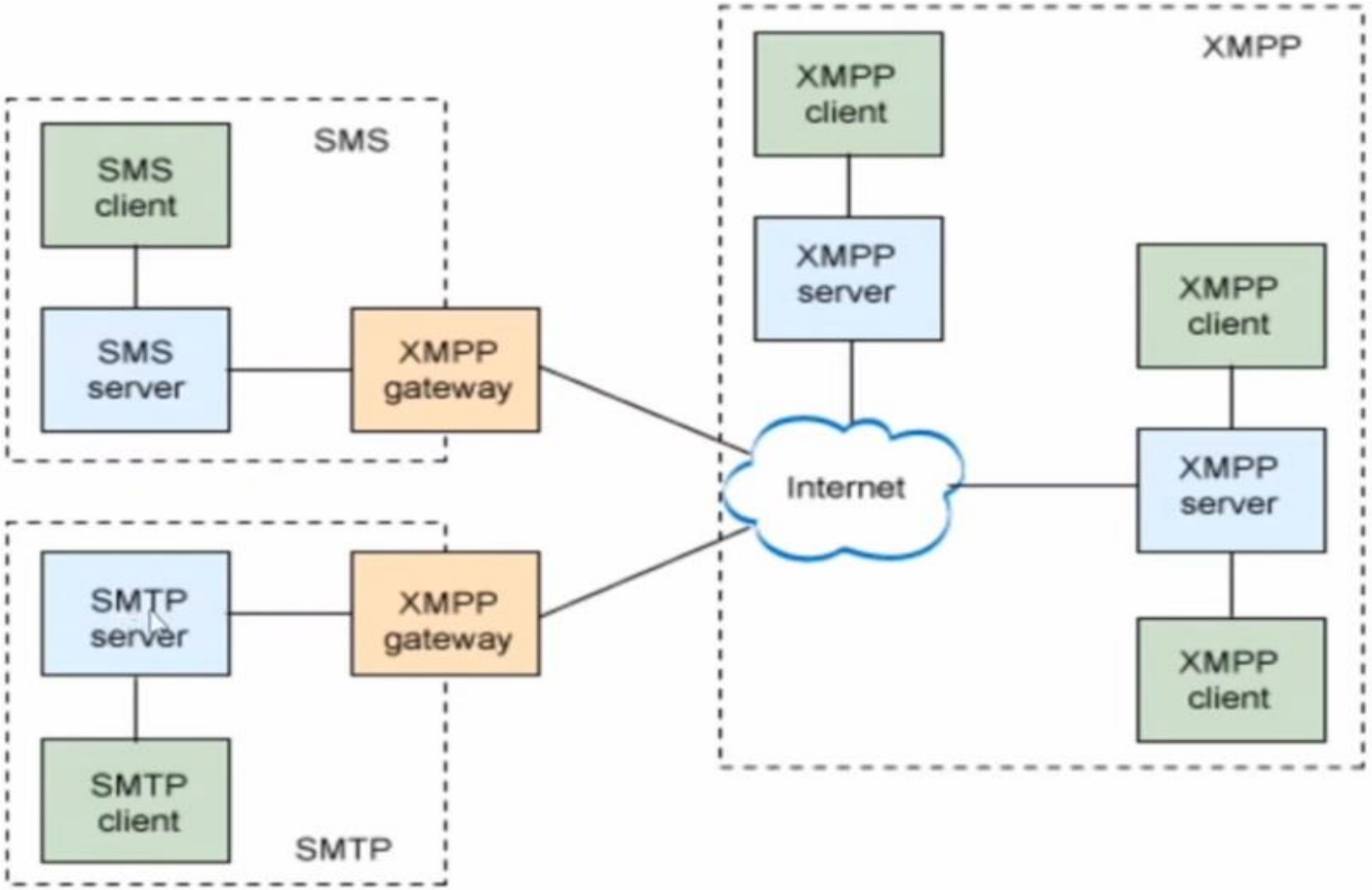
Presence: See your contact's online status.

XMPP Stanzas

- Stanza: It is the basic unit of communication in XMPP.
- There are three types of stanzas in XMPP
  - Message: used when you want to send messages
  - Presence: used to send online status information and to control subscription status between contacts.
  - IQ (Info/Query): used to [get] some information from the server or to [set] or apply some settings.

Stanzas are the fundamental data structures for exchanging information, such as messages, presence information, and structured data requests.

XMPP Architecture



XMPP Features

Peer-to-Peer Sessions: XMPP supports machine-to-machine or peer-to-peer across diverse set of networks.

Multi-User chat: XMPP supports group and conference chat.

Encryption: Point-to-point encryption is done by TLS {Transport Layer Security} and OTR {Off The Record messaging} is an extension of XMPP that enables encryption of messages and data.

Connection to other protocols: using XMPP Gateways other protocols like SMS and SMTP can also be connected with XMPP servers for different services.

XMPP use cases

- Real-Time Web Chat
- Instant Messaging
- Real-Time Group Chat
- IoT Device Control
- Online Gaming
- VoIP
- Geo Location
- System Control
- Push Notification

DDS (Data Distribution Service)

- It is an IoT protocol developed for M2M communication developed by OMG (object management group).
- DDS is a standard protocol for real-time, reliable, and scalable data exchange between devices and systems.
- It enables efficient communication in distributed IoT environments, particularly where low latency and high throughput are critical.
- It enables data exchange via publish-subscribe methodology.
- It develops a broker-less architecture unlike MQTT.

DDS Architecture

https://store.outrightcrm.com/blog/dds-protocol-in-iot/

AMQP (Advanced Message Queuing Protocol)

- AMQP is more advanced protocol than MQTT.
- It is more reliable and have better security support.
- AMQP enables encrypted and interoperable messaging between organizations and applications.
- Used to send banking transaction messages between servers and users.
- At the other end there will be an acknowledgment of message acceptance so it is a reliable.
- The protocol is used in client/server messaging and IoT device management.

Comparison Chart

	MQTT	AMQP	HTTP	CoAP
Abstraction	Pub/Sub	Pub/Sub	Request/Reply	Request/Reply
Architecture	Brokered	P2P or Brokered	P2P	P2P
QoS	3	3	Provided by TCP	Confirmable and no Confirmable
Interoperability	Partial	Yes	Yes	Yes
Real-time	Yes	No	No	No
Transports	TCP	TCP	TCP	UDP
Subscription Control	hierarchical matching	Exchanges, Queues and bindings	N/A	support for Multicast addressing msgs.
Data Serialization	Undefined	AMQP type system or user defined	No	Configurable
Dynamic Discovery	No	No	No	Yes
Security	SSL	TLS	SSL/TLS	DTLS

Network Layer Protocols

IPV4

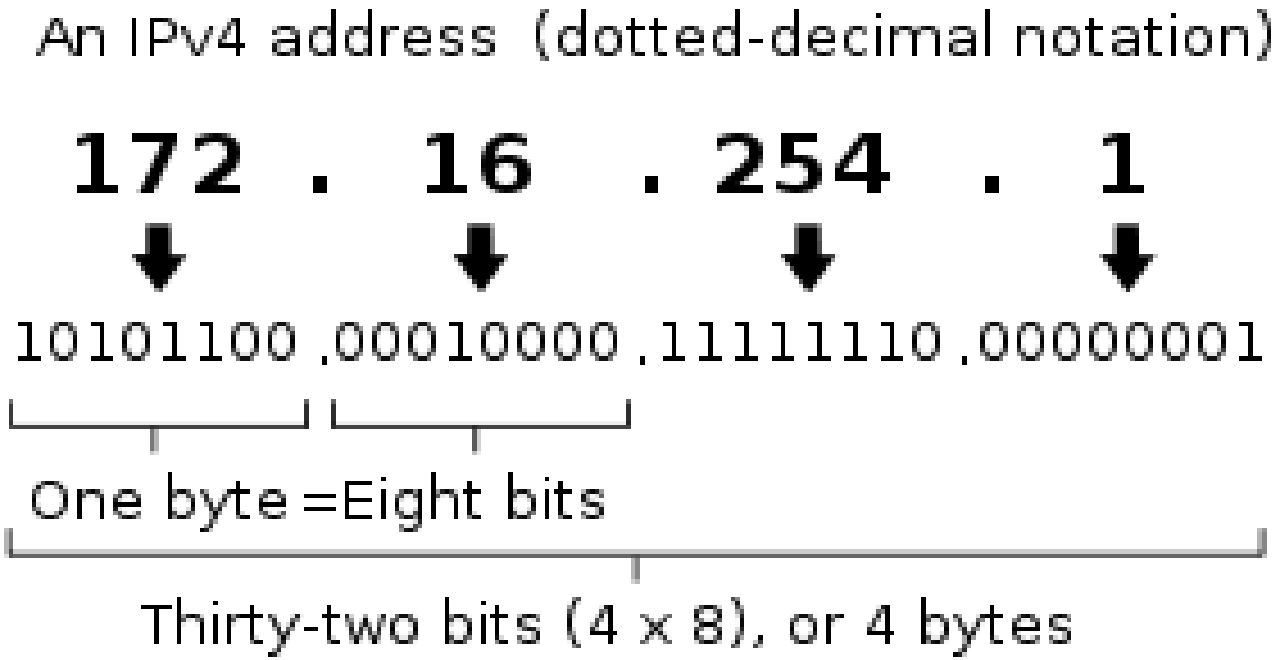
- **IP** stands for **Internet Protocol version v4** stands for **Version Four** (IPv4), is the most widely used system for identifying devices on a network.
- IP version four addresses are 32-bit integers which will be expressed in decimal notation.
- With 32 bit address we can have 4.29 Billion different nodes on a network.

Parts of IPv4

IPv4 addresses consist of three parts:

- **Network Part**: The network part indicates the distinctive variety that's appointed to the network. The network part conjointly identifies the category of the network that's assigned.
- **Host Part**: The host part uniquely identifies the machine on your network. This part of the IPv4 address is assigned to every host. **Subnet Number**: This is the non-obligatory part of IPv4. Local networks that have massive numbers of hosts are divided into subnets and subnet numbers are appointed to that

IPV4 Representation

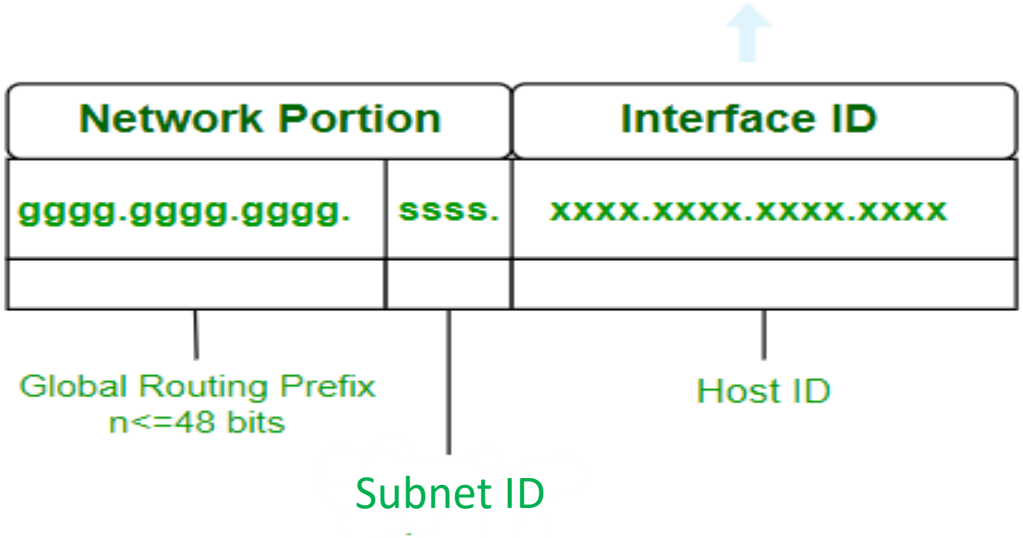


IPV6

- The most common version of the Internet Protocol currently in use, IPv4, will soon be replaced by IPv6, a new version of the protocol.
- Because so many connected devices are being used, the original IP address scheme, known as IPv4, is running out of addresses.
- An IPv6 address consists of eight groups of four hexadecimal digits.

for e.g.., 3001:0da8:75a3:0000:0000:8a2e:0370:7334

- With 128-bit address space, it allows 340 undecillion (340×10^{36}) unique address space.



GLOBAL UNICAST	2000::/3	Publicly routable
UNIQUE LOCAL	FC00::/7	Routable in the LAN
LINK LOCAL	FE80::/10	Not routable
MULTICAST	FF00::/8	Addresses for groups
ANYCAST	2000::/3	Shared address

- **Global Routing Prefix:** The Global Routing Prefix is the portion of an IPv6 address that is used to identify a specific network or subnet within the larger IPv6 internet.
- **Subnet Id:** The portion of the address used within an organization to identify subnets. This usually follows the Global Routing Prefix.
- **Host Id:** The last part of the address, is used to identify a specific host on a network.

- **Types of IPv6 Address**
- Now that we know about what is IPv6 address let's take a look at its different types.
- **Unicast Addresses :** Only one interface is specified by the unicast address. A packet moves from one host to the destination host when it is sent to a unicast address destination.
- **Multicast Addresses:** It represents a group of IP devices and can only be used as the destination of a datagram.
- **Anycast Addresses:** The multicast address and the anycast address are the same. The way the anycast address varies from other addresses is that it can deliver the same IP address to several servers or devices. Keep in mind that the hosts do not receive the IP address. Stated differently, multiple interfaces or a collection of interfaces are assigned an anycast address.

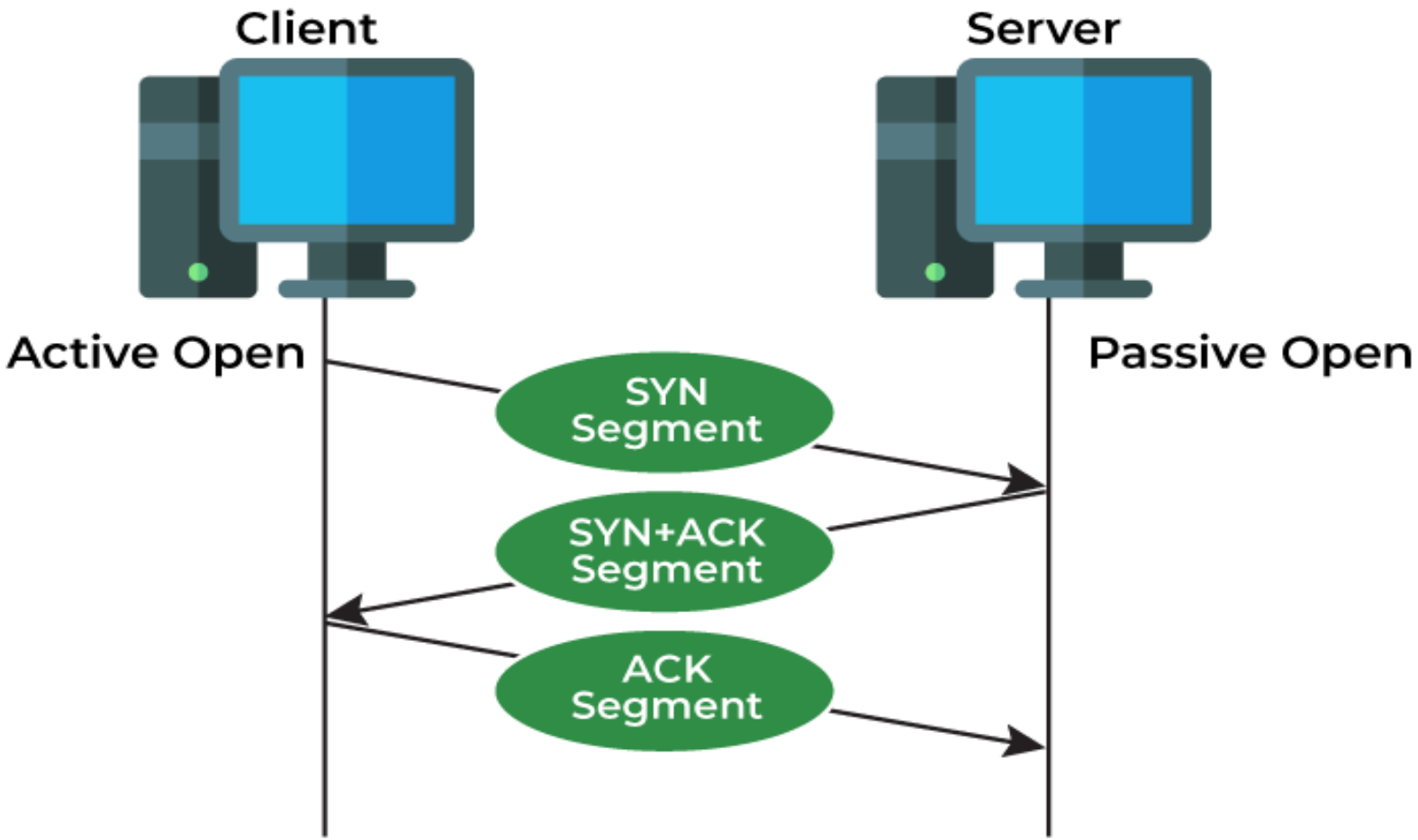
- **Advantages**
- **Faster Speeds:** IPv6 supports multicast rather than broadcast in IPv4.This feature allows bandwidth-intensive packet flows (like multimedia streams) to be sent to multiple destinations all at once.
- **Stronger Security:** IPSecurity, which provides confidentiality, and data integrity, is embedded into IPv6.
- Routing efficiency
- Reliability
- Most importantly it's the final solution for growing nodes in Global-network.
- The device allocates addresses on its own.
- Internet protocol security is used to support security.
- Enable simple aggregation of prefixes allocated to IP networks; this saves bandwidth by enabling the simultaneous transmission of large data packages.

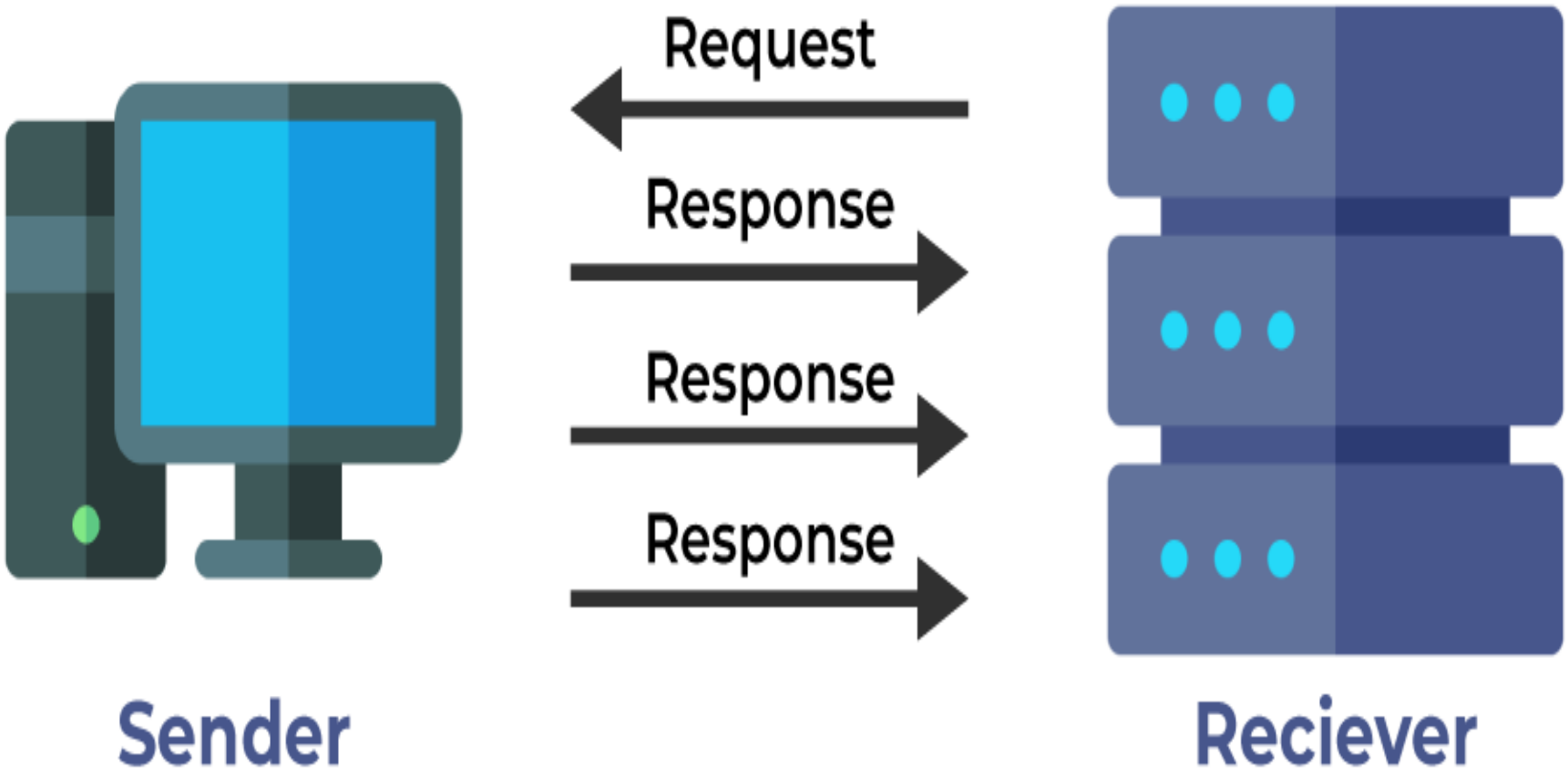
IP4 & IPV6

IPv6	IPv4
IPv6 has a 128-bit address length	IPv4 has a 32-bit address length
It supports Auto and renumbering address configuration	It Supports Manual and DHCP address configuration
The address space of IPv6 is quite large it can produce 3.4×10^{38} address space	It can generate 4.29×10^{9} address space
Address Representation of IPv6 is in hexadecimal	Address representation of IPv4 is in decimal
In IPv6 checksum field is not available	In IPv4 checksum field is available
IPv6 has a header of 40 bytes fixed	IPv4 has a header of 20-60 bytes.
IPv6 does not support VLSM	IPv4 supports VLSM(Variable Length subnet mask).

TCP & UDP

- TCP and UDP both are protocols of the Transport Layer.
- TCP is a connection-oriented protocol (It requires a logical connection to be established between the two processes before data is exchanged) whereas UDP is a connectionless protocol (IT allows data to be exchanged without setting up a link between processes.
- Unlike TCP, it is an unreliable and connectionless protocol.


Client
Server
Active Open
Passive Open
SYN Segment
SYN+ACK Segment
ACK Segment


Request
Response
Response
Response
Sender
Reciever

- **Advantages of UDP**
- It does not require any connection for sending or receiving data.
- Broadcast and Multicast are available in UDP.
- UDP can operate on a large range of networks.
- UDP has live and real-time data.
- UDP can deliver data if all the components of the data are not complete.

- **Disadvantages of UDP**
- We can not have any way to acknowledge the successful transfer of data.
- UDP cannot have the mechanism to track the sequence of data.
- UDP is connectionless, and due to this, it is unreliable to transfer data.
- In case of a Collision, UDP packets are dropped by Routers in comparison to TCP.
- UDP can drop packets in case of detection of errors.

- Where TCP is Used?
- Sending Emails
- Transferring Files
- Web Browsing
- Where UDP is Used?
- Gaming
- Video Streaming
- Online Video Chats

TCP offers dependable, orderly, and error-free data transmission, making it ideal for operations that require precision, such as file transfers and web browsing. UDP, on the other hand, provides quicker, connectionless communication that is excellent for real-time applications such as gaming and video streaming, when speed is critical and minor data loss is acceptable.