SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

INTERNET OF THINGS (2040233303)

Topic: IoT Communication Protocols

BRANCH: BCA

SEMESTER: V

PREPARED BY: RAKESH PATEL

Rakeshkumar Manilal Patel

Assistant Professor

Bachelor of Computer Applications

Silver Oak College of Computer Application (SOCCA)

Expertise in Internet of Things

WHAT WE WILL BE COVERING IN THIS TOPIC ?

- Link Layer Protocols
- Network/Internet Layer Protocols
- Messaging Protocols- MQTT, CoAP & XMPP
- Transport Layer Protocols
- Application Layer Protocols
- Basics of Sensor Network Topologies

LEARNING OBJECTIVES

- Understand the Fundamentals of IoT Protocols.
- Understand how these protocols manage data link establishment, data transfer, and error correction.
- Describe the function of the Network Layer in routing packets across different networks.
- Describe the characteristics and use cases of MQTT, CoAP, and XMPP.
- Describe the Transport Layer and Application Layer Protocols
- To know and understand various sensor network topologies

What is IoT Protocol ?

"An IoT communication protocol is a set of ***rules*** and ***standards*** that define how data is transmitted and received between IoT devices and networks".

IoT Connections for communication !

1. DEVICE TO DEVICE COMMUNICATION

2. DEVICE TO GATEWAY COMMUNICATION

3. GATEWAY TO CLOUD OR DATA SYSTEMS

4. BETWEEN DATA SYSTEMS

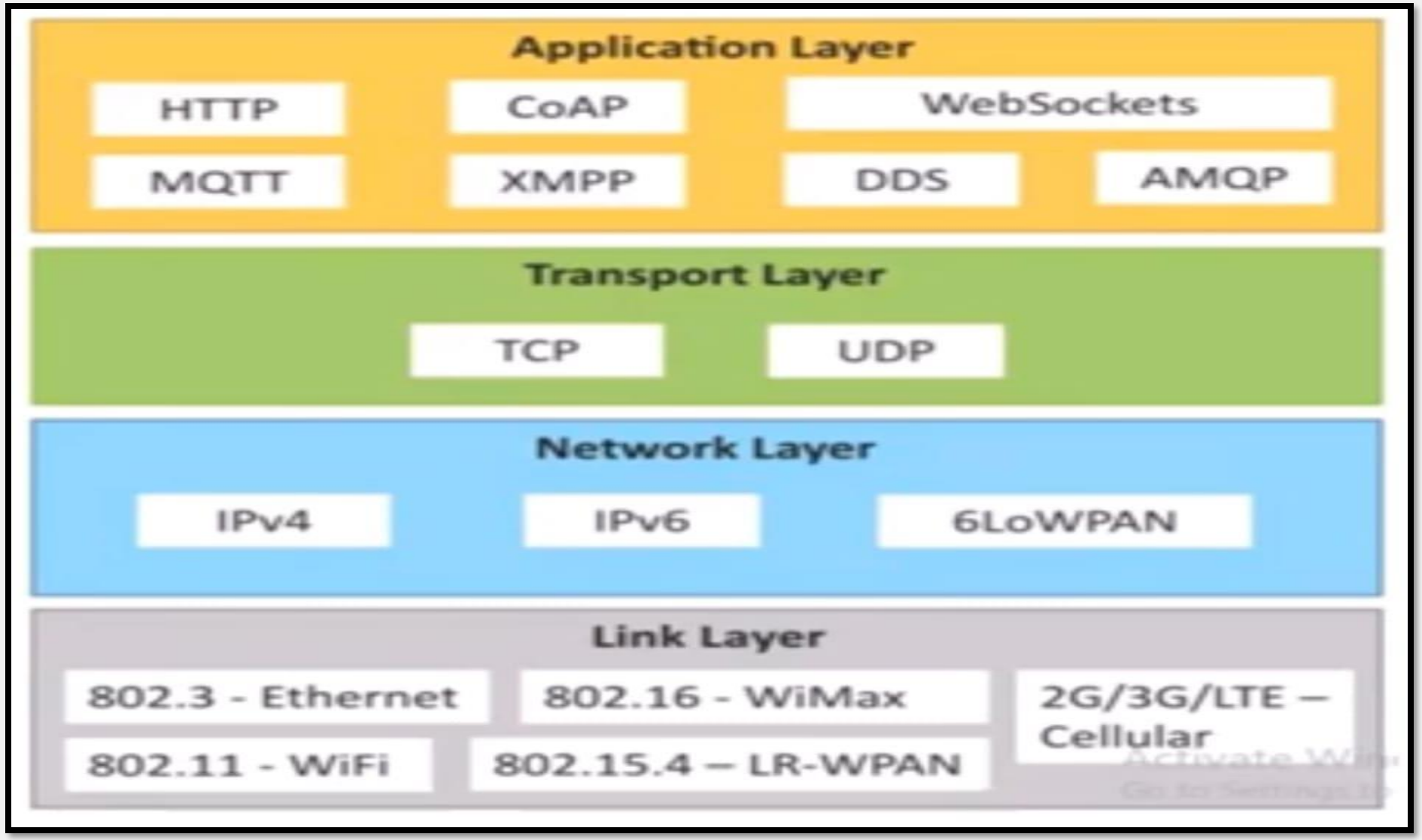


Figure: Generalized Stack of IoT Communication Protocols

Link Layer Protocols

- Link layer protocol determines how the data is physically sent over the network's physical layer.
- Link layer determine how the packets are coded and signaled by the hardware device over the medium to which the host is attached.
- Link layer protocols are Ethernet, 802.11, 802.16, 802.15.4, Bluetooth, Zigbee, Wi-fi, mobile communication etc.

Highlights-Link Layer Protocols

- The link layer is responsible for establishing and maintaining the physical and logical connection between devices.
- **Names of Protocols**: Bluetooth, Wi-Fi, Ethernet etc…
- **Example Scenario**: Imagine you are connecting your smartphone to a Bluetooth speaker. The Bluetooth protocol at the link layer manages the connection, allowing the devices to communicate and transmit audio data wirelessly.

Link Layer Protocols (Wi-Fi)

- **Function**: Wi-Fi is a wireless protocol used to connect devices to a local area network (LAN) or the internet without cables.
- **Example Scenario (Real World)**: In a smart home, Wi-Fi connects devices like smart thermostats and security cameras to the home network for remote control.
- **Key Points**:
  1. **Frequency Bands**: Operates on 2.4 GHz (wider range) and 5 GHz (faster speeds).
  2. **Data Rate**: Typical speeds range from 150 Mbps to over 9.6 Gbps, depending on the Wi-Fi standard.
  3. **Range**: Indoor range is typically 30-50 meters.

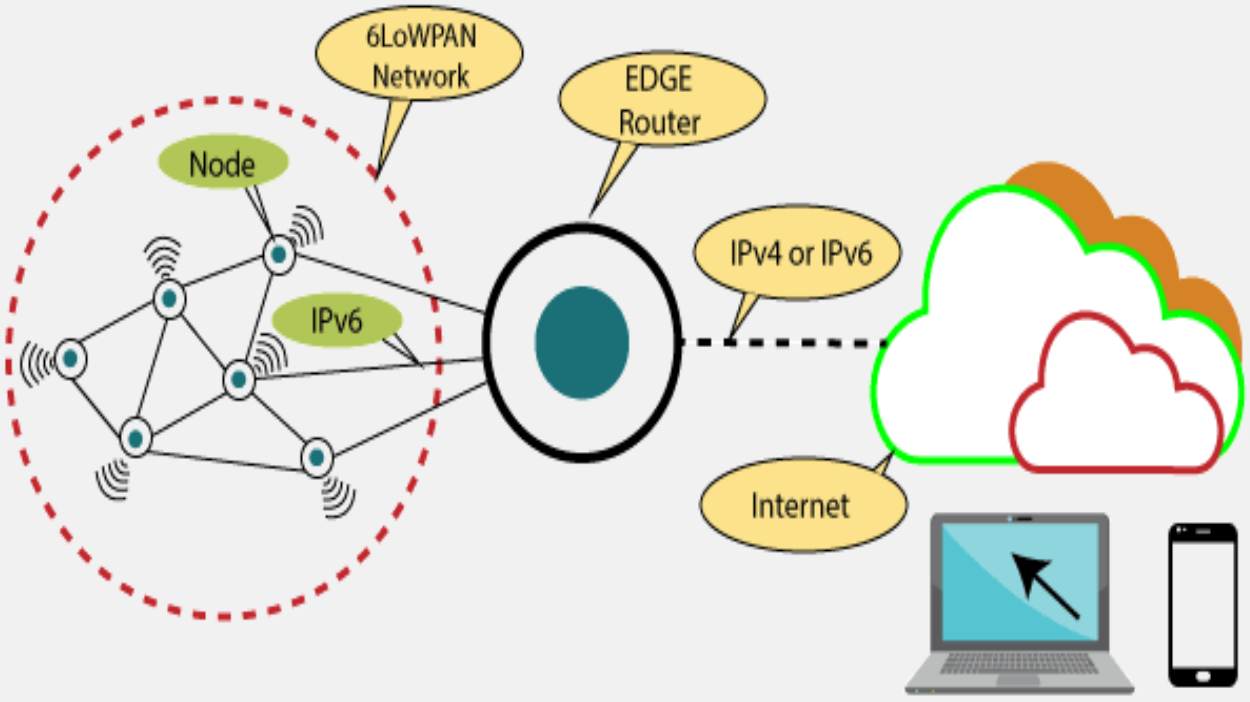
Link Layer Protocols (Ethernet)

- **Function**: Ethernet provides wired networking, offering high-speed, reliable communication over physical cables.
- **Example Scenario (Real World)**: In offices, Ethernet connects computers and servers for stable and fast communication.
- **Key Points**:
  1. **Speed**: Common speeds are 100 Mbps (standard) to 1 Gbps (Fast Ethernet).
  2. **Reliability**: Offers low latency and minimal interference.
  3. **Range**: Limited to 100 meters with standard twisted-pair cables.

Link Layer Protocols (Bluetooth)

- **Function**: Bluetooth is used for short-range wireless communication between devices over a short distance.
- **Example Scenario (Real World)**: Bluetooth is commonly used to connect wireless headphones to smartphones or fitness trackers to apps.
- **Key Points**:
  1. **Frequency Band**: Operates on the 2.4 GHz ISM band.
  2. **Data Rate**: Bluetooth Classic offers up to 3 Mbps, while Bluetooth Low Energy (BLE) is optimized for low power consumption.
  3. **Range**: Typically up to 10 meters (Bluetooth Classic) or 100 meters (BLE).

Network/Internet Layer Protocols



- The network layer responsible for the delivery of packets from the source to destination.
- Network layer uses IP address to chose one host among millions of host.
- In network layer, datagram needs a destination IP address for delivery and a source IP address for a destination reply.

Highlights-Network/Internet Layer Protocol

- The network layer handles the routing and forwarding of data packets between devices on different networks. It determines the best path for data to travel across interconnected networks.
- **Names of Protocols**: 6LoWAN, IPv4, IPv6.
- **Example Scenario**: When a smart agriculture sensor sends data to a remote server over a LoWAN network, the network layer protocols (e.g., IPv6) ensure that the data is properly routed from the sensor to the server, even if it needs to travel across different networks.
- **Key Points**:
    1. **6LoWAN**: Low-power, wide-area network protocol designed for IoT devices with limited power and long-range communication needs.
    2. **IPv4**: Older IP addressing system, still widely used but limited by a smaller address space.
    3. **IPv6**: Newer IP addressing system, with a much larger address space and better support for IoT devices.

Network Layer Protocols (6LoWAN-Low-Power Wide-Area Network)

- **Function:** LoWAN is designed for long-range communication between low-power IoT devices over large geographic areas. It supports devices that need to send small amounts of data intermittently over a wide area while conserving battery life.
- **Example Scenario (Real World):** LoWAN is commonly used in smart agriculture for monitoring soil moisture and weather conditions across large fields, where sensors need to transmit data over long distances to a central hub.
- **Key Points:**
  1. **Frequency Bands:** Operates in sub-GHz bands (e.g., 868 MHz in Europe, 915 MHz in the US).
  2. **Data Rate:** Typically low, ranging from 0.3 kbps to 50 kbps, optimized for energy efficiency and long-range communication.
  3. **Range:** Can reach up to 15-20 kilometers in rural areas and 2-5 kilometers in urban areas.

Network Layer Protocols
(IPv4 -Internet Protocol Version 4)

- **Function:** IPv4 is the fourth version of the Internet Protocol, responsible for routing and addressing packets of data across networks. It uses 32-bit addresses to identify devices on a network.
- **Example Scenario (Real World):** Most home networks and websites still use IPv4 addresses. For instance, your home router may have an IPv4 address like **192.168.1.1**, which helps it route data between your devices and the internet.
- **Key Points:**
  1. **Address Bits:** 32-bit address, allowing for approximately 4.3 billion unique addresses.
  2. **Exhaustion:** The limited address space has led to IPv4 address exhaustion, necessitating techniques like Network Address Translation (NAT).
  3. **Compatibility**: Widely supported and still in use, despite the transition to IPv6.

Network Layer Protocols
(IPv6 -Internet Protocol Version 6)

- **Function:** IPv6 is the successor to IPv4, providing a larger address space and better support for modern networks, including IoT. It uses 128-bit addresses, allowing for a vast number of unique IP addresses. Eg. IPv6 addresses are represented in hexadecimal and consist of eight groups of four hexadecimal digits, separated by colons.
  2001 : 8934 : DB11 : 010A : 12FC : 0370: 0000: 56DE
- **Example Scenario (Real World):** In smart cities, IPv6 can be used to assign unique addresses to every connected device, such as traffic lights, street cameras, and public Wi-Fi hotspots, enabling efficient management and data collection.
- **Key Points:**
  1. **Address Bits:** 128-bit address, allowing for approximately 3.4×10^38 unique addresses.
  2. **Features:** Built-in support for autoconfiguration and better security features (e.g., IPsec).
  3. **Scalability:** Designed to accommodate the exponential growth of connected devices, especially in IoT.

Messaging Protocols

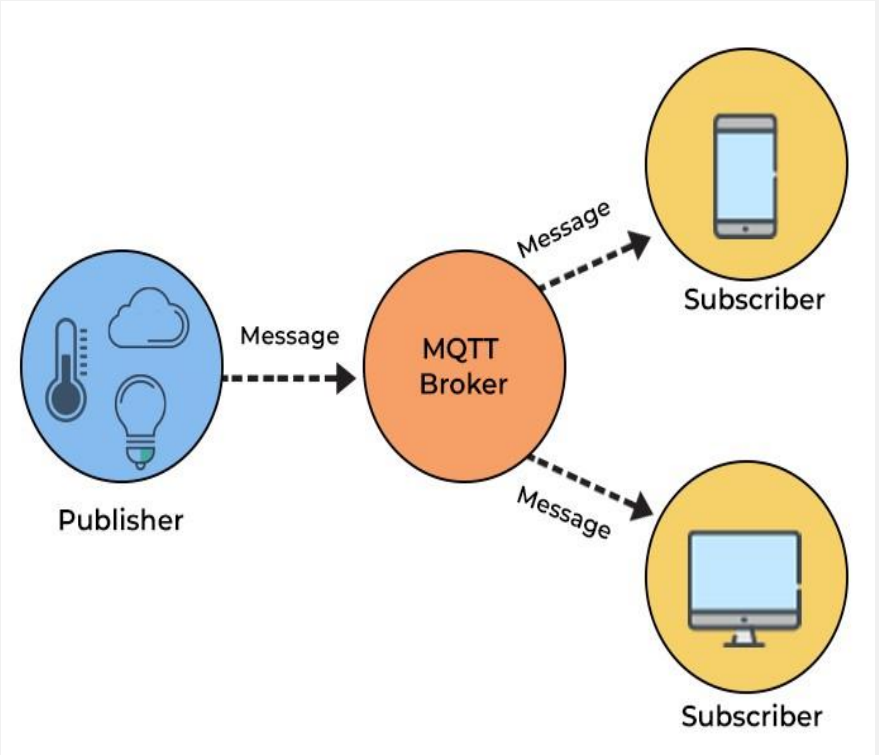
- Messaging protocols are very important for the transfer of data in terms of messages.
- They are useful for send/receive of a message to/from the cloud in IoT applications.
- In the section, two messaging protocols are discussed.
  - Message Queuing Telemetry Transport (MQTT)
  - Constrained Application Protocol (CoAP)
  - Extensible Messaging and Presence Protocol (XMPP)

Message Queuing Telemetry Transport (MQTT)

- As we know, IoT has one of the biggest challenges of resource constraints, the lightweight protocol is more preferred.
- It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
- Here, lightweight means, it can work with minimal resources and does not require any specific hardware architecture or additional resources.
- MQTT was introduced by IBM in1999 and standardized by OASIS in 2013. ISO standard (ISO/IECPRF20922).
- Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and network sand communications on the other side.

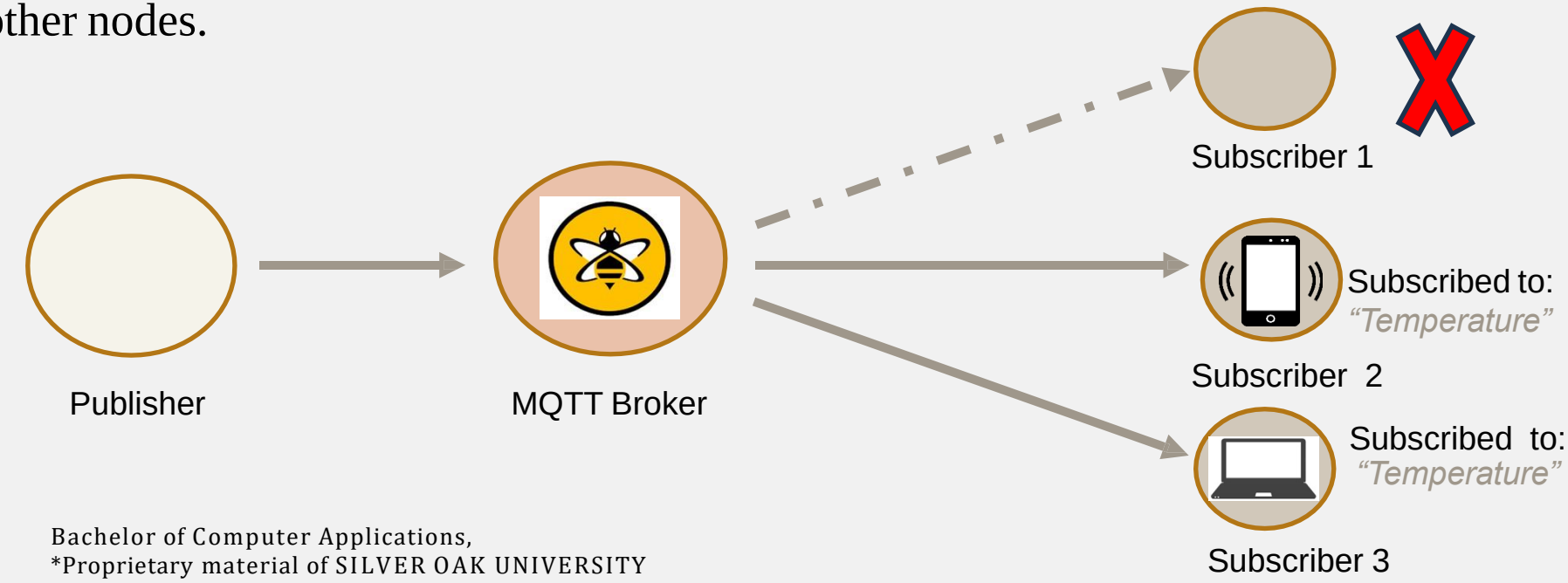
Message Queuing Telemetry Transport (MQTT)

- A message broker controls the publish-subscribe messaging pattern.
- A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker.
- The MQTT working can be explained as follows.
  1. The publisher (node) sends a message to the MQTT broker.
  2. The MQTT broker transmits the message to the subscribed destinations.



MQTT-Example

- Suppose a temperature sensor is connected to any system and we want to monitor that temperature.
- The controller connected to the temperature sensor publishes the temperature value to the MQTT broker with a **topic** name: "**Temperature**". Hence, it is considered a publisher.
- The MQTT broker transmits the temperature value to the subscribers.
- Note that the only nodes which have subscribed to the **topic** will capture the data and not the other nodes.



MQTT-Example

- Can publisher and subscriber interchange their role?
- Let us take an example of a Driverless car.
- A GPS sensor is connected to the car which gives the current location of the car.
- The system in car, publishes the location to MQTT broker with topic name – Current Location.
- The server/computer subscribes to the topic Current Location and receives the current location of the car.

Publisher: Current Location

MQTT Broker

Subscriber: Current Location

MQTT-Example

- Based on the currently selected destination, the server/computer computes the **direction** of the car.
- This data is published to an MQTT broker with a topic named – **Direction**.
- To get the command from the server, the system in the car has to subscribe to the topic named **Direction**.
- According to the command received from the server, the car will run in a particular direction.

Publisher: CurrentLocation

Subscriber: Direction

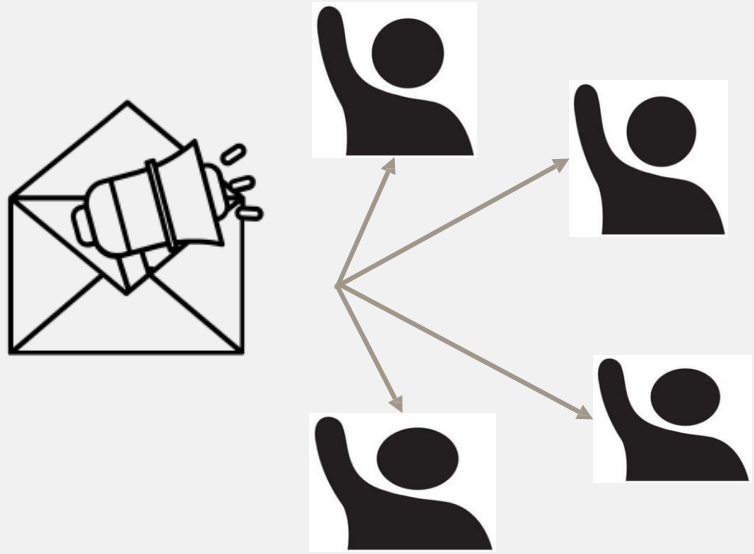
MQTT Broker

Subscriber: CurrentLocation

Publisher: Direction

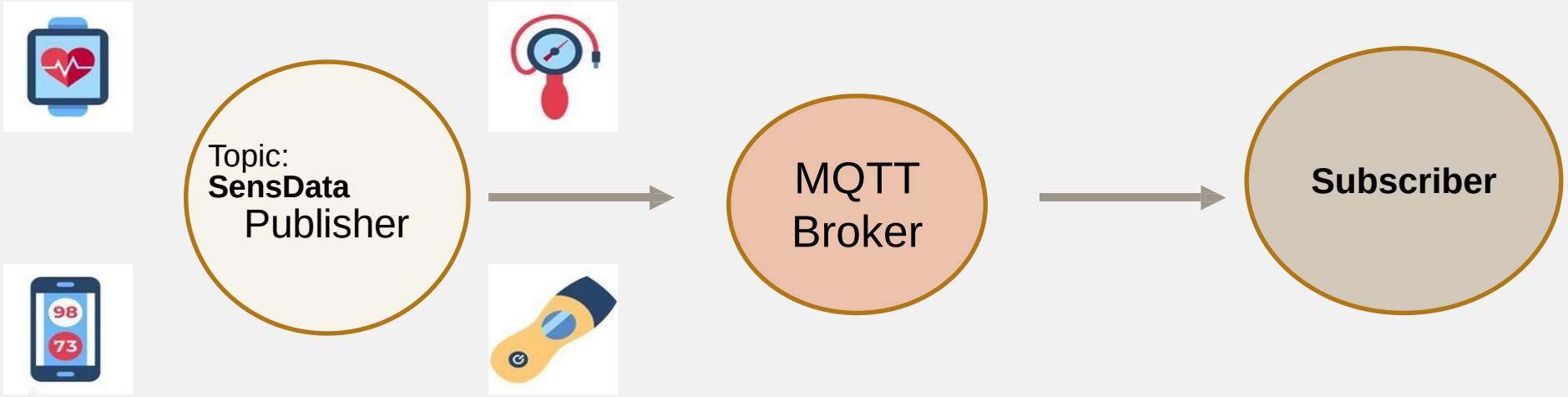
MQTT- Working

- Television broadcaster station broadcasts all the channels.
- The viewers subscribe to their favorite channels only.
- Similarly, in MQTT, the publisher node publishes data with the topic name and interested subscribers subscribe to the topic.
- Note that there can be multiple subscribers of a same topic as well as a single subscriber can subscribe to multiple topics.



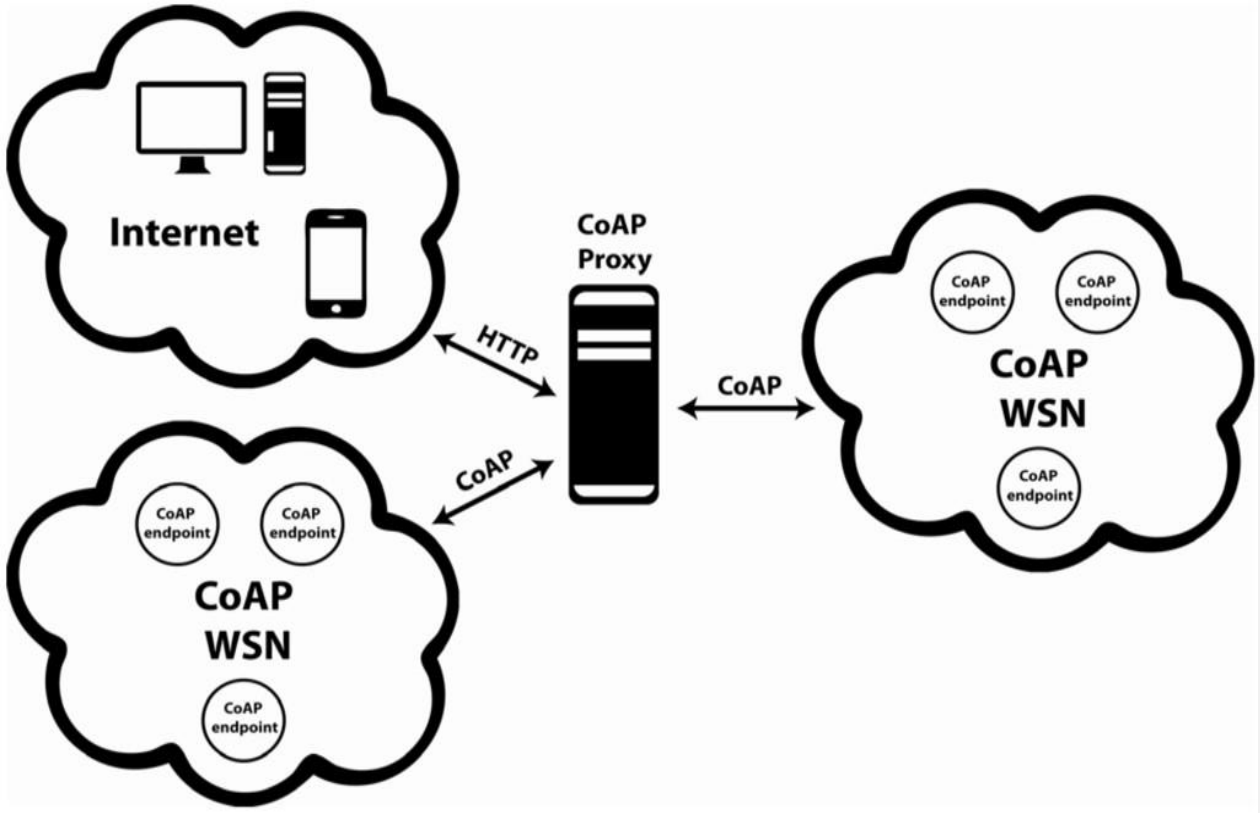
MQTT- Node(s)

- Node(s) in MQTT collects the information from sensors or other i/p devices in case of the publisher.
- Connects it to the messaging server known as the MQTT broker.
- A specific string – Topic is used to publish the message and let other nodes understand the information. These nodes are considered subscribers.
- Any node can be publisher or subscriber and node is also referred to as client.



Constrained Application Protocol (CoAP)

- CoAP is designed by IETF (Internet Engineer Task Force) to work in constrained environment.



- It is a one to one communication protocol.
- CoAP is also light weight like MQTT protocol and it uses less resources than HTTP.
- HTTP runs over TCP and is connection oriented while CoAP runs over UDP and it is connection less.

Constrained Application Protocol (CoAP)

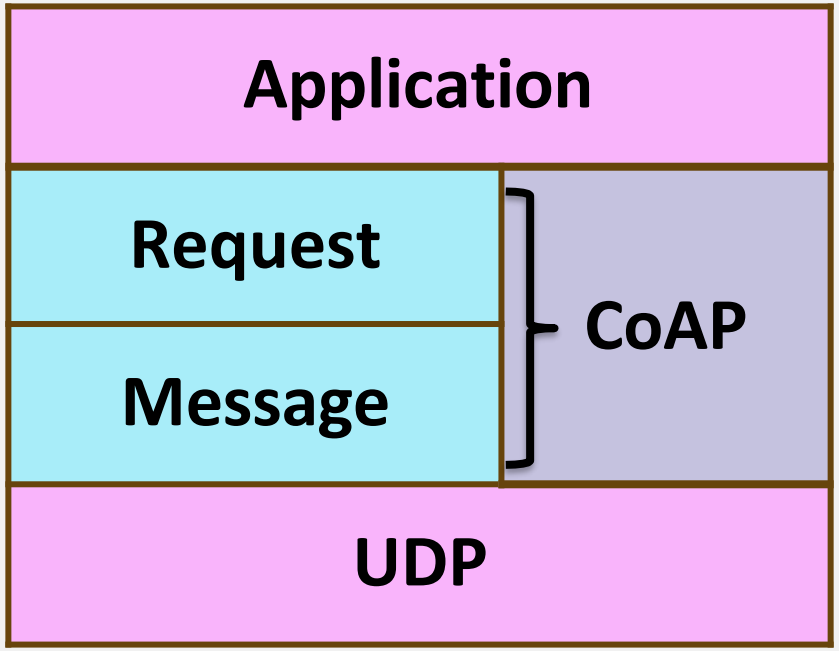
- Web transfer protocol for use with constrained node sand networks.
- Designed for Machine to Machine (M2M) applications such as smart energy and building automation.
- Based on Request - Response model between end-points
- Client-Server interaction is asynchronous over a datagram oriented transport protocol such as UDP.
- The Constrained Application Protocol (CoAP) is a session layer protocol designed by IETF Constrained RESTful Environment (CoRE) working group to provide lightweight RESTful (HTTP) interface.
- Representational State Transfer (REST) is the standard interface between HTTP client and servers.

Constrained Application Protocol (CoAP)

- Lightweight applications such as those in IoT, could result in significant over head and power consumption by REST.
- CoAP is designed to enable low-power sensors to use RESTful services while meeting their power constraints.
- Built over UDP, instead of TCP (which is commonly used with HTTP) and has a light mechanism to provide reliability.
- CoAP has four messaging modes:
    1. Confirmable
    2. Non-confirmable
    3. Piggyback
    4. Separate

CoAP Layers

- CoAP has four layers:
  - UDP
  - Messages
  - Request-Response
  - Application
- The message layer is designed to deal with UDP and asynchronous switching.



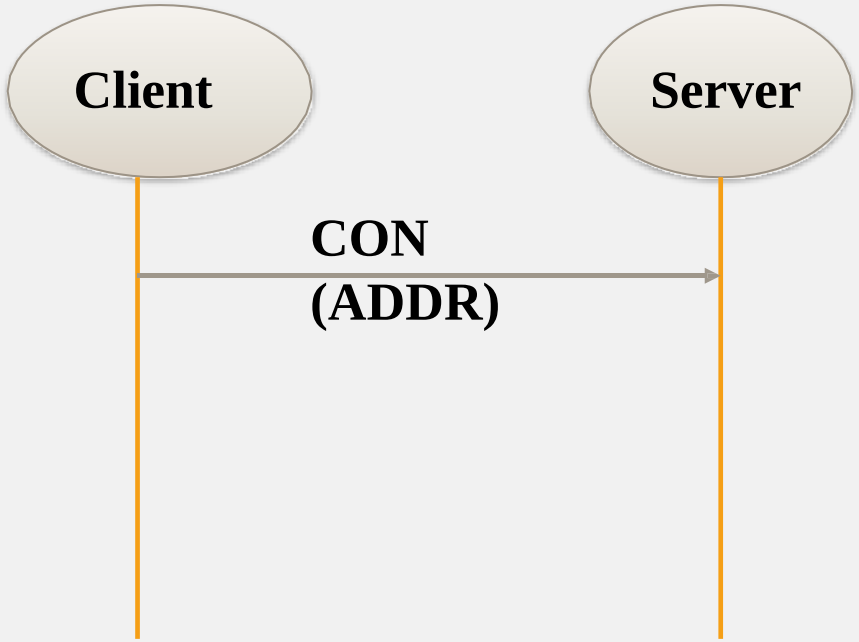
- The request/response layer concerns communication method and deal with request/response messages.
- In the request/response layer, clients may use GET/PUT/DELET methods to transmit the message.

Message Layer in CoAP

- CoAP message layer supports four types of messages:

1. Confirmable – Reliable Messaging (CON):

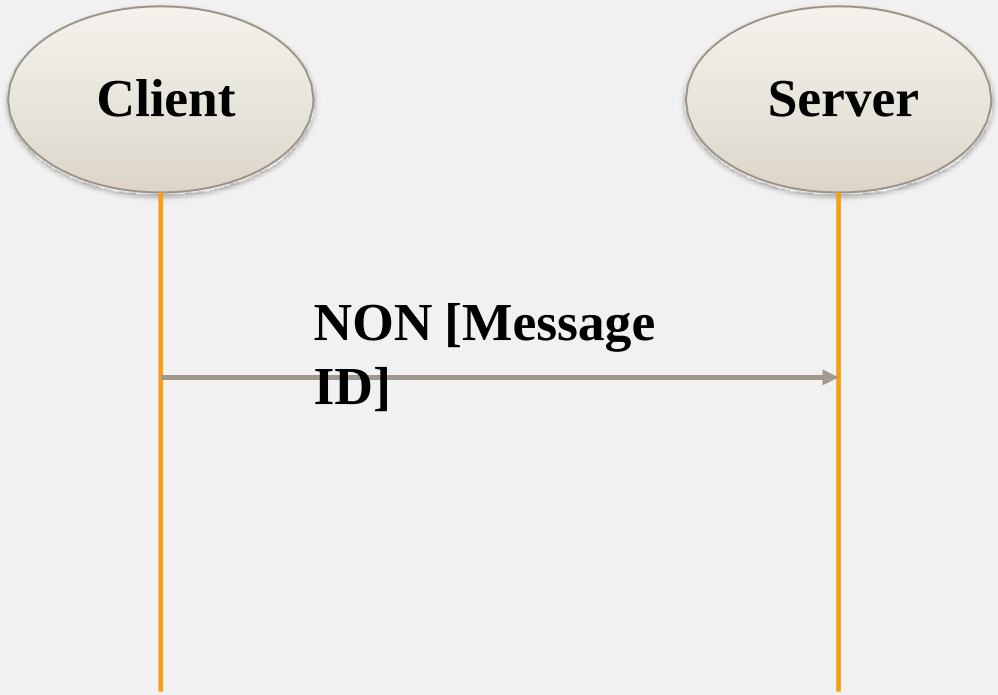
✓ This is reliable approach because the retransmission of message occurs until the acknowledgement of the same message ID is received.



✓ If there is a timeout or fail of acknowledgement, the RST message will be sent from the server as a response.

✓ Hence, the client will retransmit the message, this resolves the retransmission and it is considered a reliable approach.

Message Layer in CoAP (Contd.)



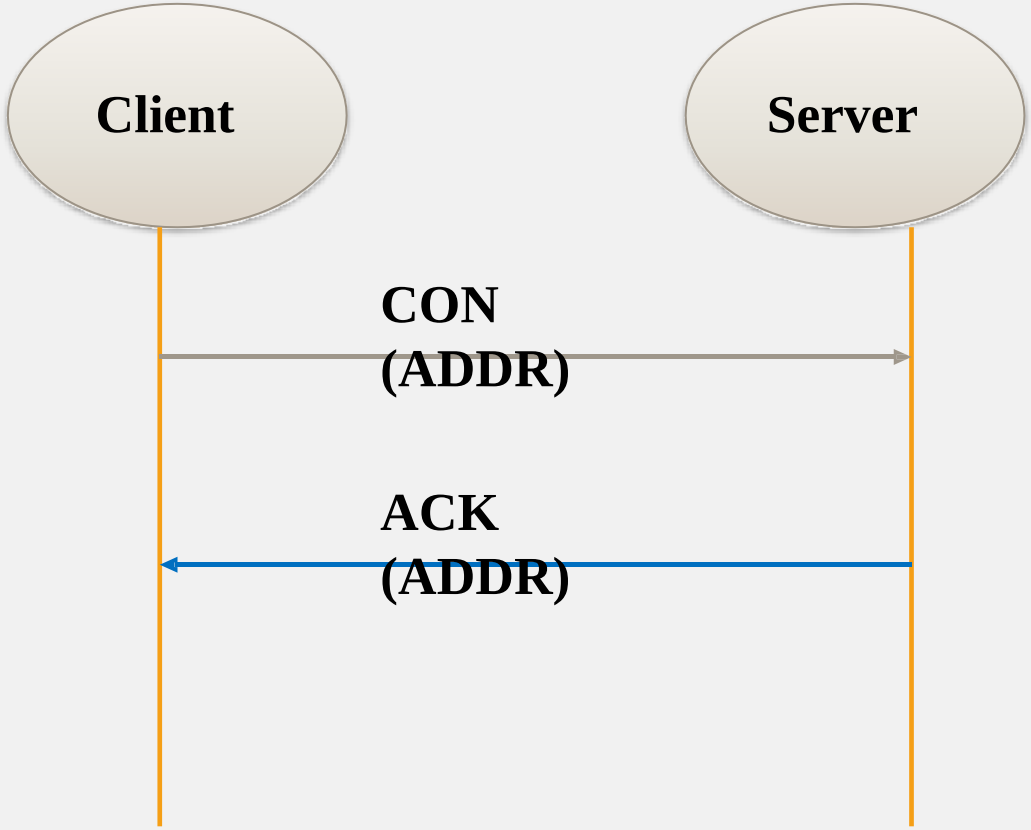
2. Non-confirmable – Non-reliable Messaging (NON):

- ✓ In this message transmission style, there is no reliability is ensured.
- ✓ The acknowledgement is not issued by the server.
- ✓ The message has ID for supervision purpose, if the message is not processed by the server it transmits the RST message.

Message Layer in CoAP (Contd.)

3. Acknowledgement (ACK):

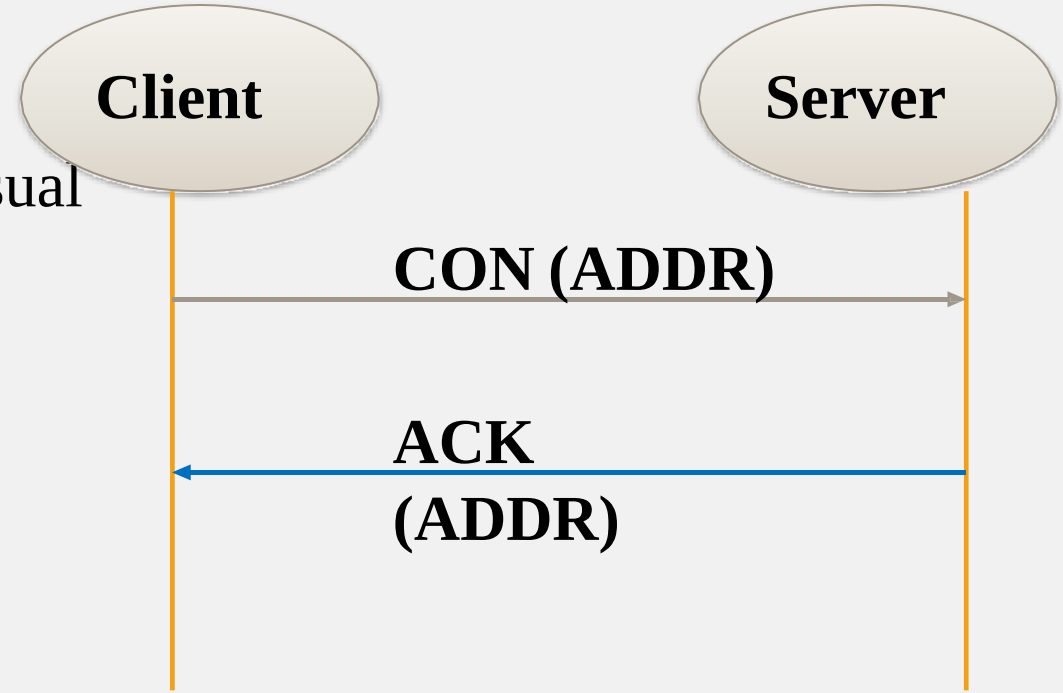
- ✓ In this messaging, the traditional acknowledgement message sent as usual protocol.
- ✓ We can compare this with regular handshaking scheme.
- ✓ The handshake is automated process that establishes a link for communication before actual data transfer begins.



Message Layer in CoAP (Contd.)

3. Acknowledgement (ACK):

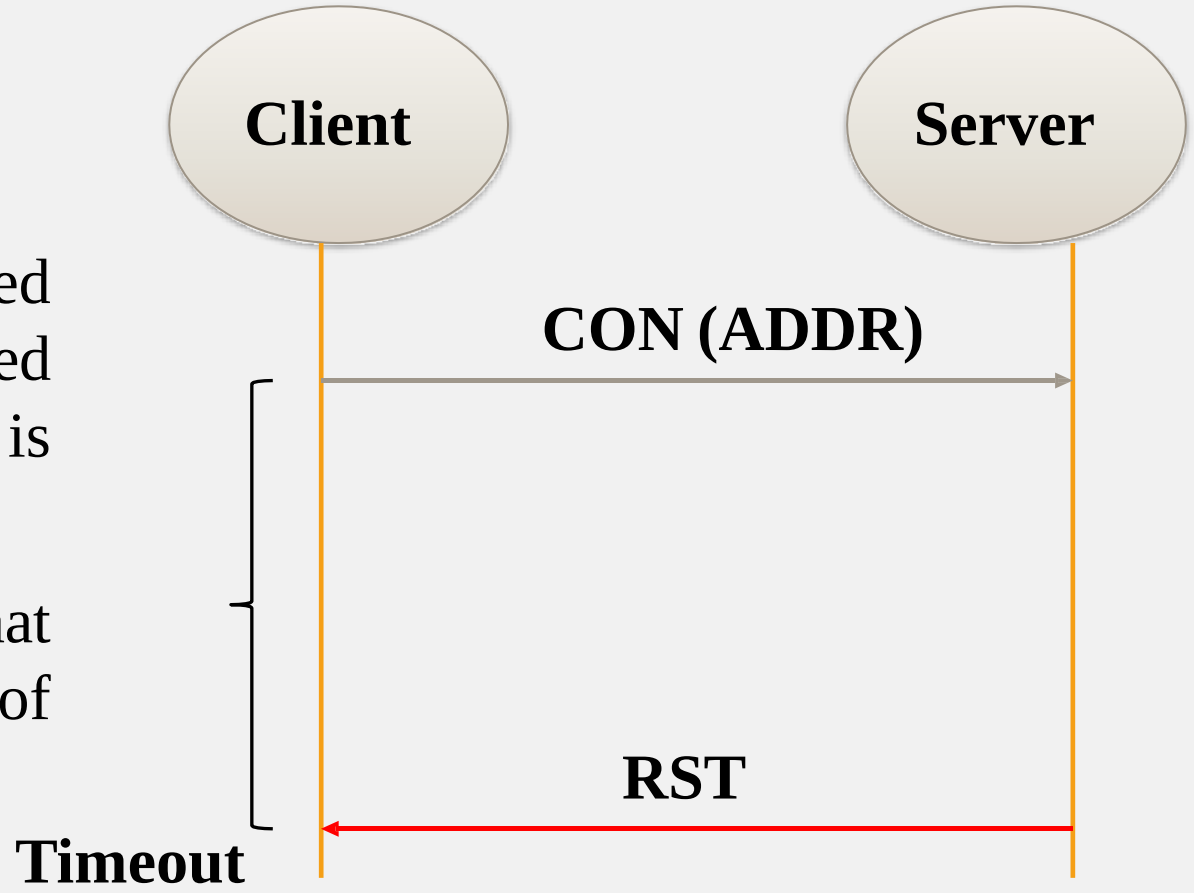
- ✓ In this messaging, the traditional acknowledgement message sent as usual protocol.
- ✓ We can compare this with regular handshaking scheme.
- ✓ The handshake is automated process that establishes a link for communication before actual data transfer begins.



Message Layer in CoAP (Contd.)

4. Reset (RST):

- ✓ The receiver is expecting the message from sender of a particular message ID.
- ✓ If no message is processed or received before specific amount of time (called timeout), the message called RESET is transmitted from receiver.
- ✓ This message informs the sender that there is a trouble in transmission of messages.

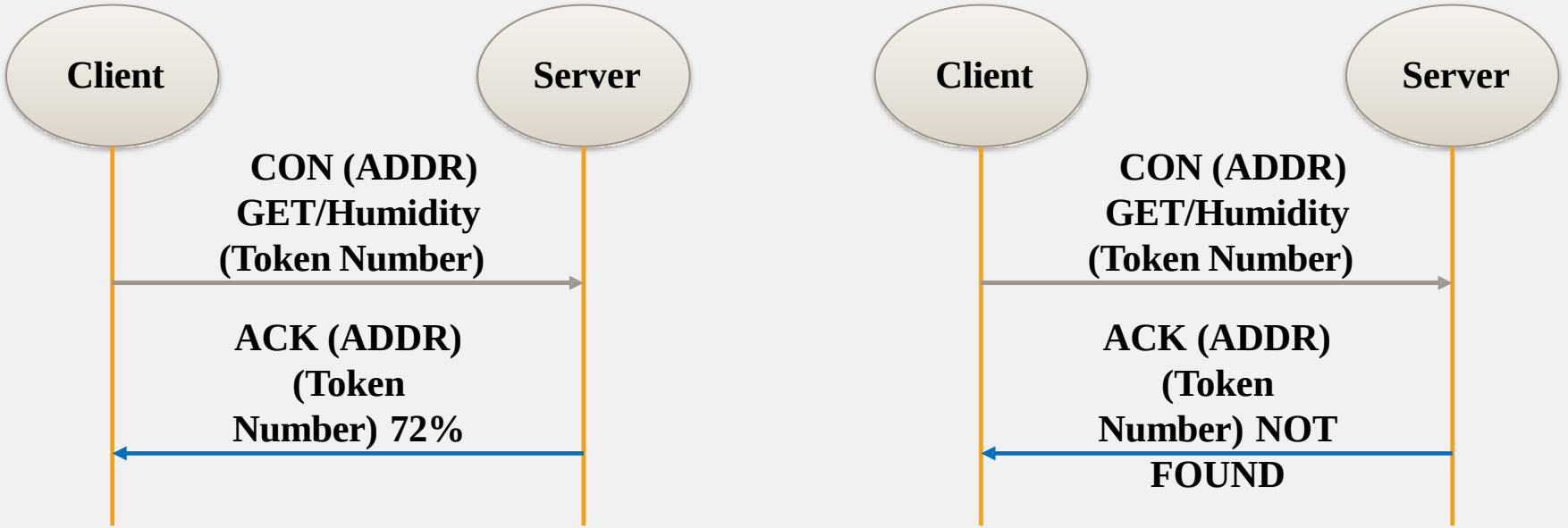


Request-Response Layer in CoAP

There are three modes in CoAP request-response layer.

1. Piggy-Backed:

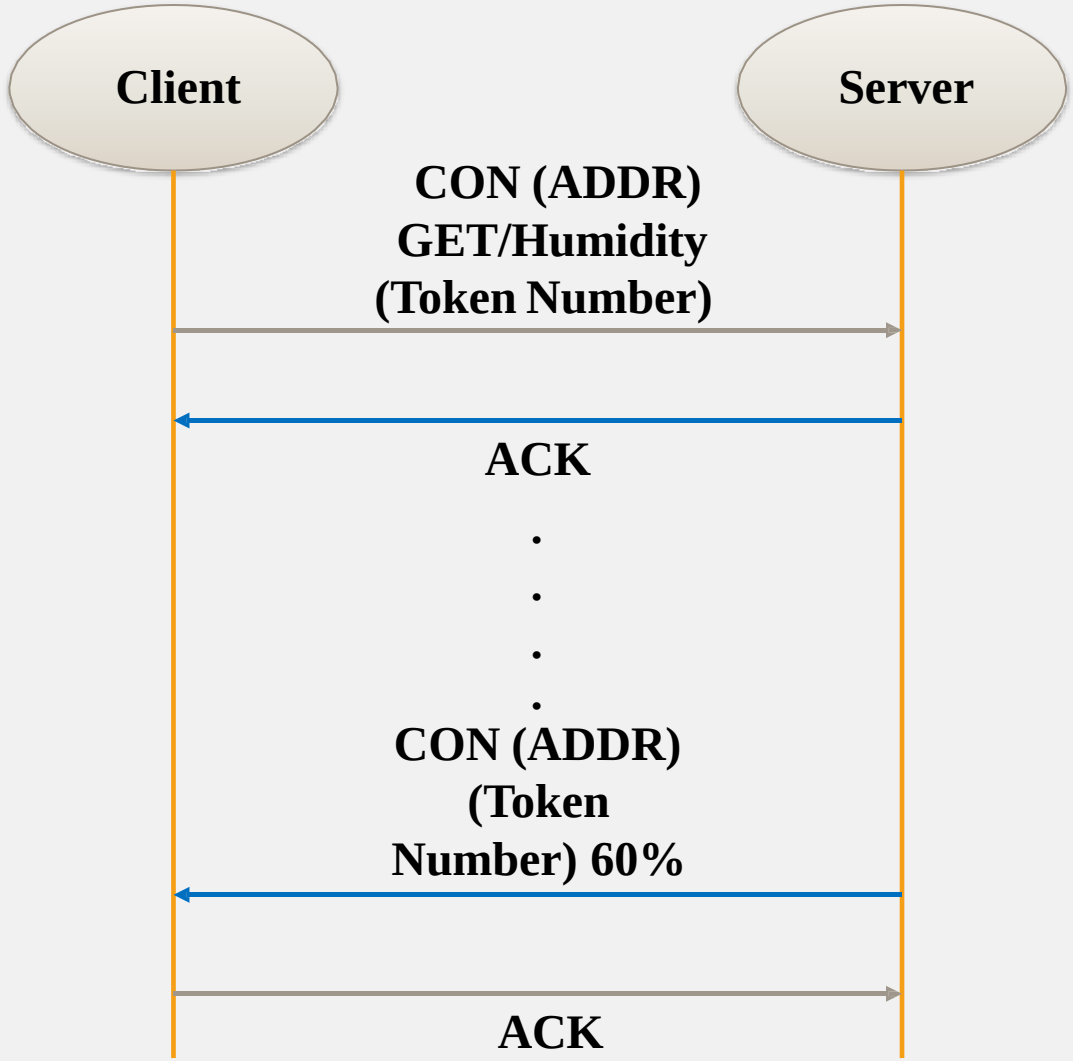
- In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and CON messaging method.
- The ACK is transmitted by server immediately with corresponding token number and message.
- If the message is not received by server or data is not available then the failure code is embedded in ACK.



Request-Response Layer in CoAP (Contd.)

2. Separate Response:

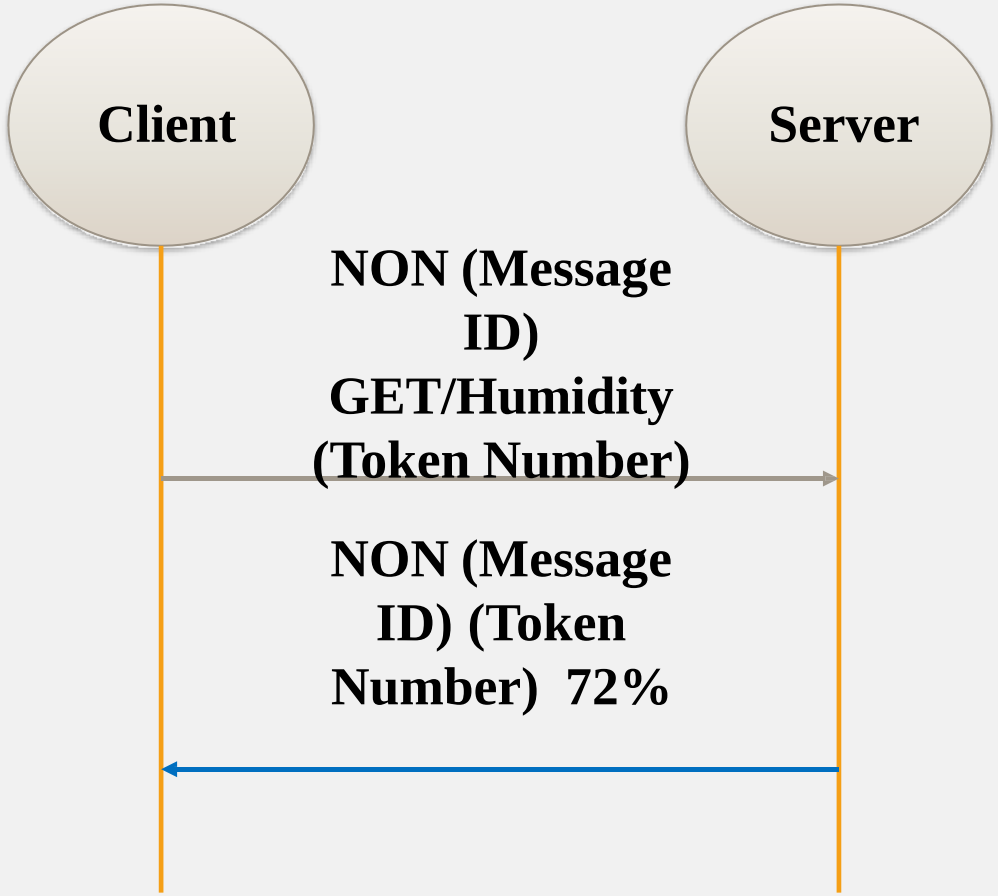
- In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and CON messaging method.
- If the server is unable to respond immediately, an empty ACK will be reverted.
- After some time, when the server is able to send the response, it sends a CON message with data.
- In response to that, ACK is sent back from client to server.



Request-Response Layer in CoAP (Contd.)

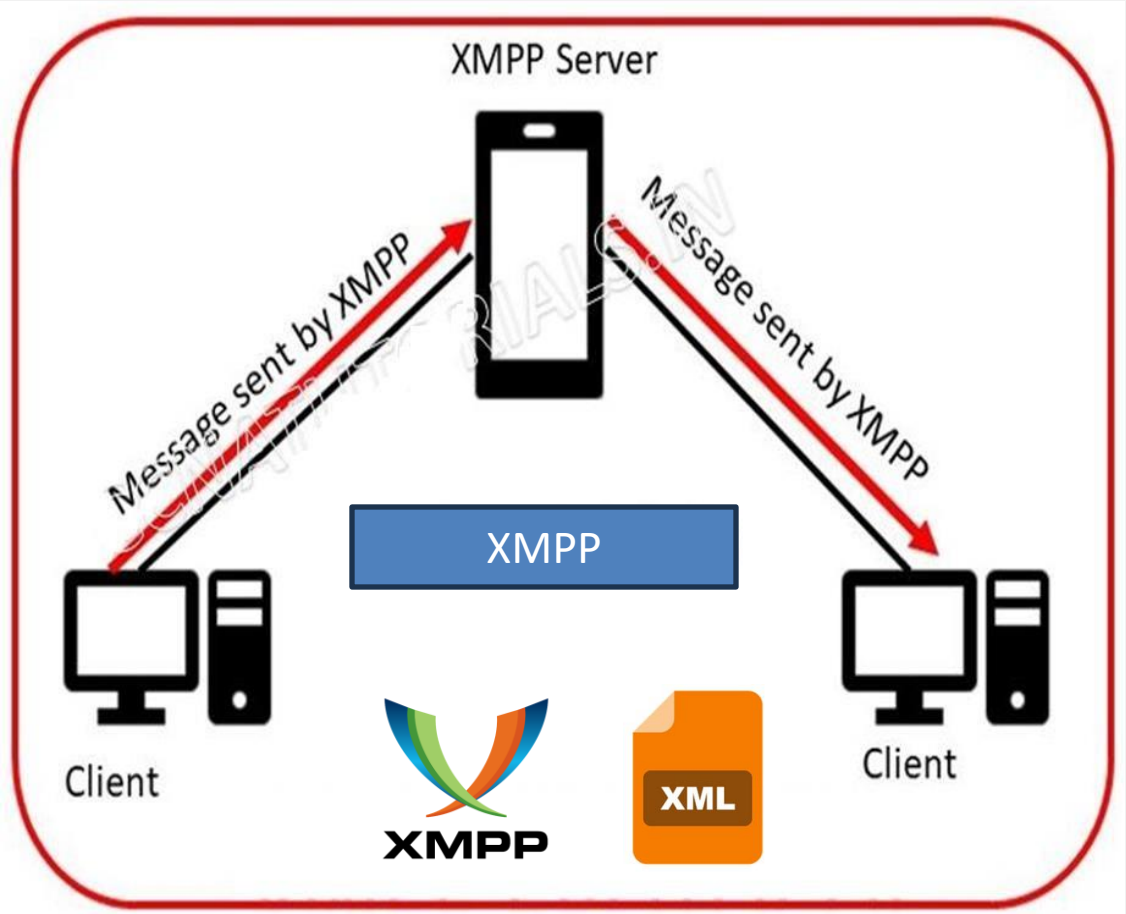
3. Non-Confirmable Request Response:

- In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and NON messaging method.
- The server does not give ACK in NON messaging method, so it will send a NON type response in return with token number and its data.



XMPP Protocol

- The full form of XMPP is Extensible Messaging and Presence Protocol.
- It is designed for messaging, chat, video, voice calls and collaboration.
- The fundamental base of this protocol is XML and it is used for messaging services such as WhatsApp.
- The XMPP was originally named as Jabber and later known as XMPP.



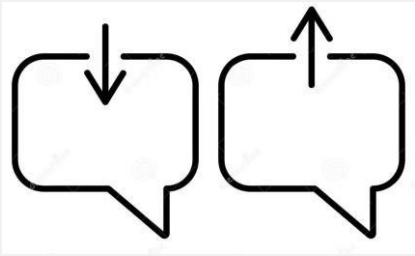
XMPP denotation

- The denotation for XMPP can be given as following:
- X – X denotes eXtensible.
- It is considered extensible as it is designed to accept and accommodate any changes.
- The protocol is defined in open standard and it is extensible because of open standard approach.
- M – M denotes Messaging.
- XMPP supports sending message to the recipients in real time.
- P – P denotes Presence.
- It is helpful to identify the status such as “Offline” or “Online” or “Busy”.
- It also helps in understanding if the recipient is ready to receive the message or not.
- P – P denotes Protocol.
- The set of standards together acting as a protocol.

Messenger Requirements

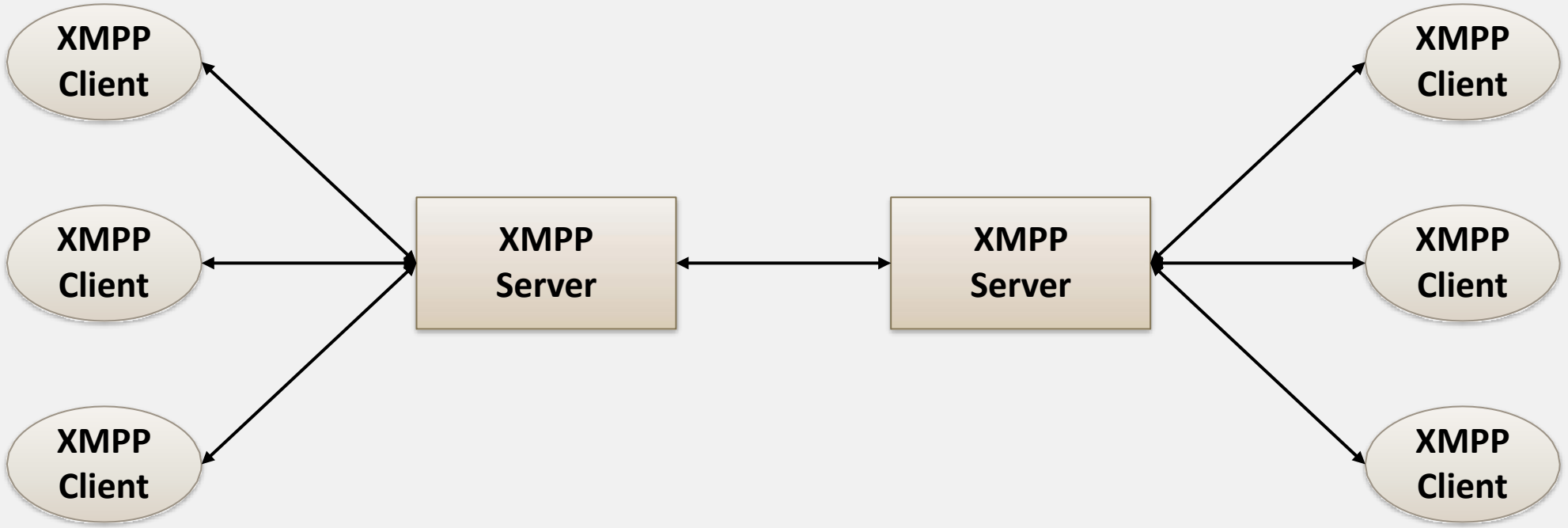
- Any messenger requires the following features inbuilt :
- Message sending and receiving features.
- Understanding the presence/absence status.
- Managing subscription details.
- Maintaining the contact list.
- Message blocking services.
- Note that all the listed features are built in XMPP.



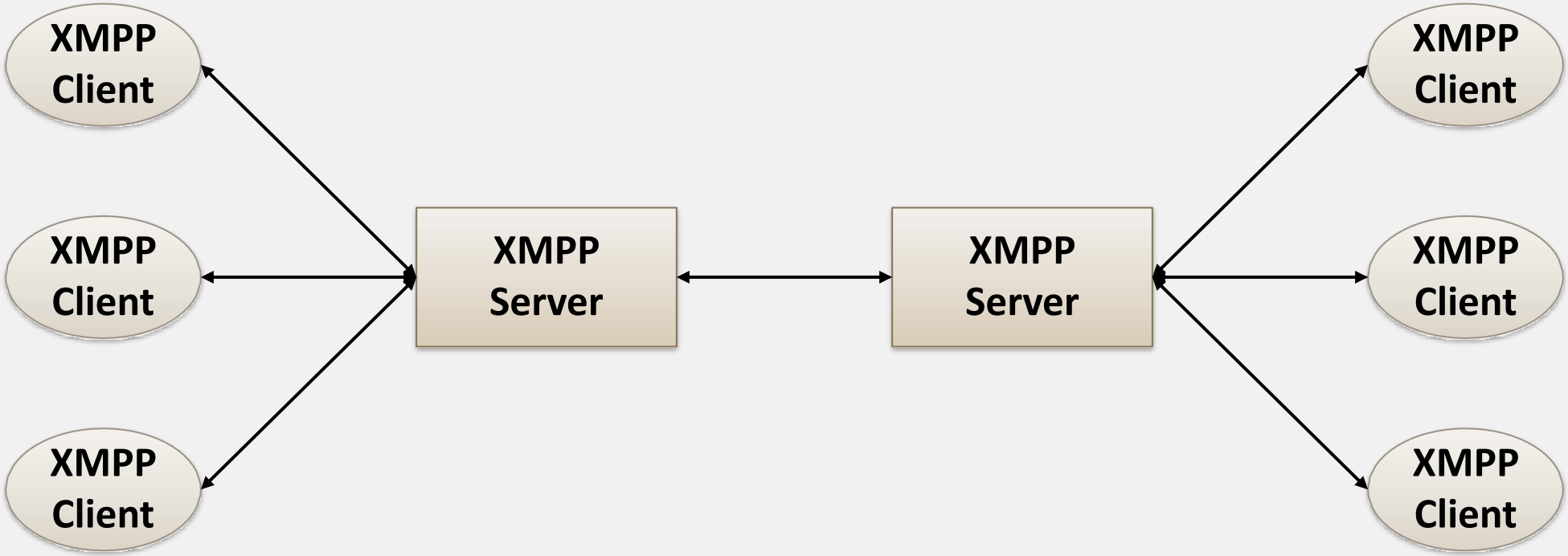

XMPP Architecture

- The architecture of XMPP protocol is shown in the figure :
- The XMPP clients communicate with XMPP server bidirectional.
- There can be any numbers of clients connected to XMPP server.
- The XMPP servers may be connected to the other clients or other XMPP servers.



XMPP Architecture (Contd.)

- The initial version of the XMPP was with TCP using open ended XML.
- After certain period of time, the XMPP was developed based on HTTP.
- The XMPP can work with HTTP is through two different methods Polling and Binding.
- What is Polling and Binding?



XMPP Architecture (Contd.)

- In polling method, the messages stored in the server are pulled or fetched.
- The fetching is done by XMPP client through HTTP GET and POST requests.
- In binding method, Bidirectional-streams Over Synchronous HTTP (BOSH) enables the server to push the messages to the clients when they are sent.
- The binding approach is more effective than polling.
- Also both method uses HTTP, hence the firewall allows the fetch and post activity without any difficulties.

Transport Layer Protocols

- **Function of Layer**: The transport layer is responsible for end-to-end communication, ensuring data is delivered error-free, in sequence, and with no losses.
- **Names of Protocols**: TCP, UDP.
- **Example Scenario**: When you stream a video from a server, TCP might be used to ensure that all data packets arrive correctly, whereas UDP might be used for real-time applications like online gaming where speed is more important than perfect accuracy.

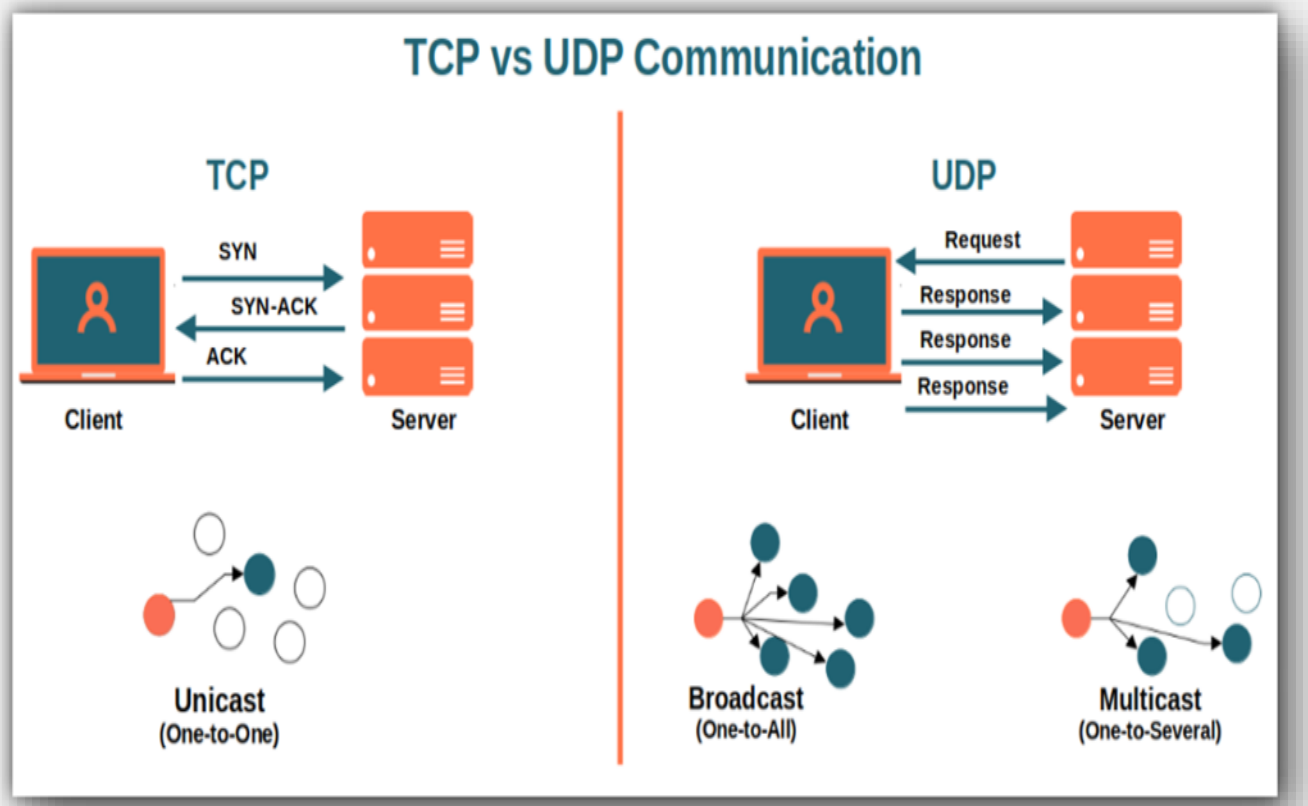


Figure: Comparison TCP Vs UDP Communication

Transport Layer Protocol (TCP-Transmission Control Protocol)

- **Function:** TCP is a connection-oriented protocol that ensures reliable, ordered, and error-checked delivery of data between applications. It establishes a connection between sender and receiver before transmitting data and ensures that all packets arrive correctly and in order.
- **Example Scenario (Real World):** When you download a file from a website or stream a video, TCP is used to ensure that all parts of the file or video arrive correctly and in the right sequence, providing a smooth and error-free experience.
- **Key Points:**
  1. **Connection-Oriented:** Establishes a connection before data transfer and ensures reliability.
  2. **Data Integrity:** Provides error checking and guarantees data is received in the correct order.
  3. **Data Rate:** The data rate is dependent on network conditions and can vary. TCP itself does not specify a data rate; it's about ensuring reliable delivery.

Transport Layer Protocol (UDP-User Datagram Protocol)

- **Function:** UDP is a connectionless protocol that allows applications to send messages (datagrams) without establishing a connection first. It provides a faster but less reliable method of communication compared to TCP. It does not guarantee delivery, ordering, or error checking.
- **Example Scenario (Real World):** UDP is used in real-time applications like online gaming or video conferencing where low latency is crucial, and occasional data loss is acceptable. For instance, in a live video call, UDP helps transmit video and audio quickly, even if some packets are lost.
- **Key Points:**
  1. **Connectionless:** No connection is established before data transfer, leading to lower latency.
  2. **No Reliability:** Does not guarantee delivery, order, or error checking. Packets may be lost or received out of order.
  3. **Data Rate:** The data rate can be higher due to lower overhead compared to TCP, but it depends on the application and network conditions.

Application Layer Protocols

- The application layer is the interface between the end-user and the network. It provides protocols that allow software applications to communicate with each other over the network.
- **Names of Protocols**: HTTP, MQTT, CoAP, XMPP, Websockets, DDS AMQP etc...
- **Example Scenario**: A smart thermostat sends temperature data to a cloud server using MQTT, allowing the server to process and display the data to the user on a mobile app.

Note: MQTT, CoAP and XMPP are already covered in Messaging Protocols so refer previous slides.

Application Layer Protocol (HTTP)

- **Function:** HTTP is a protocol used for transmitting hypermedia documents, such as HTML. It is the foundation of data communication on the World Wide Web, enabling clients (e.g., web browsers) to request and receive data from servers.

- **Example Scenario (Real World)**:
  When you enter a URL in your web browser, the browser sends an HTTP request to the server where the website is hosted. The server responds with the requested web page, which your browser displays.

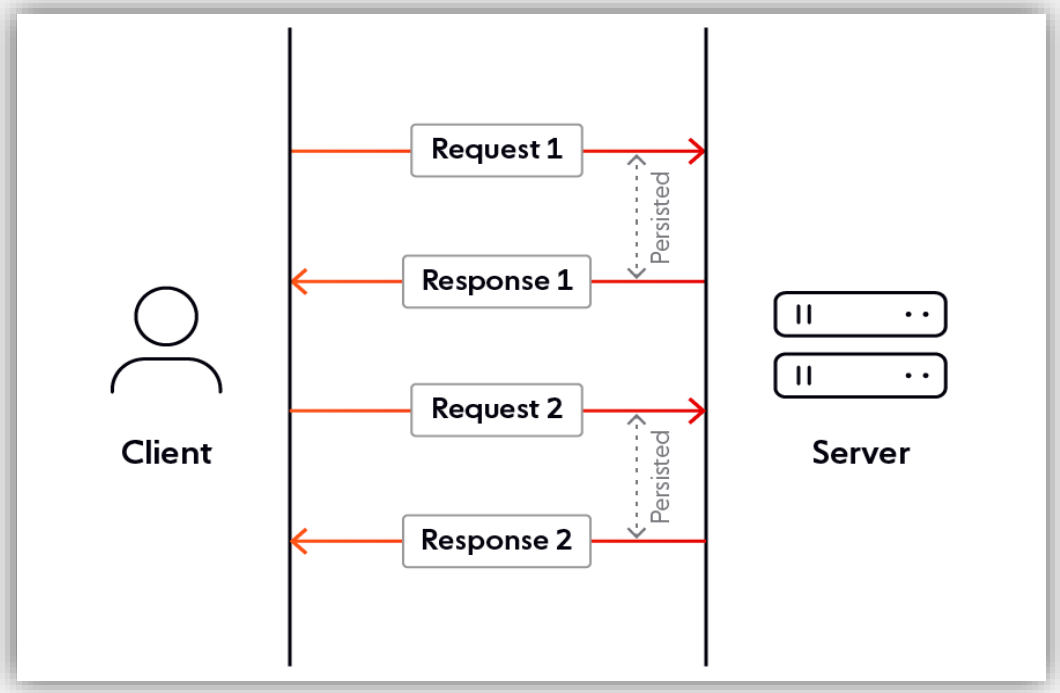


Figure: HTTP Working

Application Layer Protocol (HTTP)

- **Key Points**:

1. Request-Response Model: HTTP operates on a request-response model. The client sends a request (GET, POST, etc.), and the server responds with the requested data or an error message.

2. Stateless Protocol: HTTP is stateless, meaning each request is independent and the server does not retain any session information between requests.

3. Layered on TCP/IP: HTTP runs on top of the TCP/IP protocol, ensuring reliable data transfer between client and server.

Application Layer Protocol (Websockets)

- **Function:** WebSockets provide a full-duplex communication channel over a single, long-lived connection between a client and a server. This allows for real-time data exchange without the need for repeated HTTP requests.

- **Example Scenario (Real World)**:
  In a live chat or gaming application, WebSockets allow messages to be sent and received instantly between users, without the delay and operates fast.

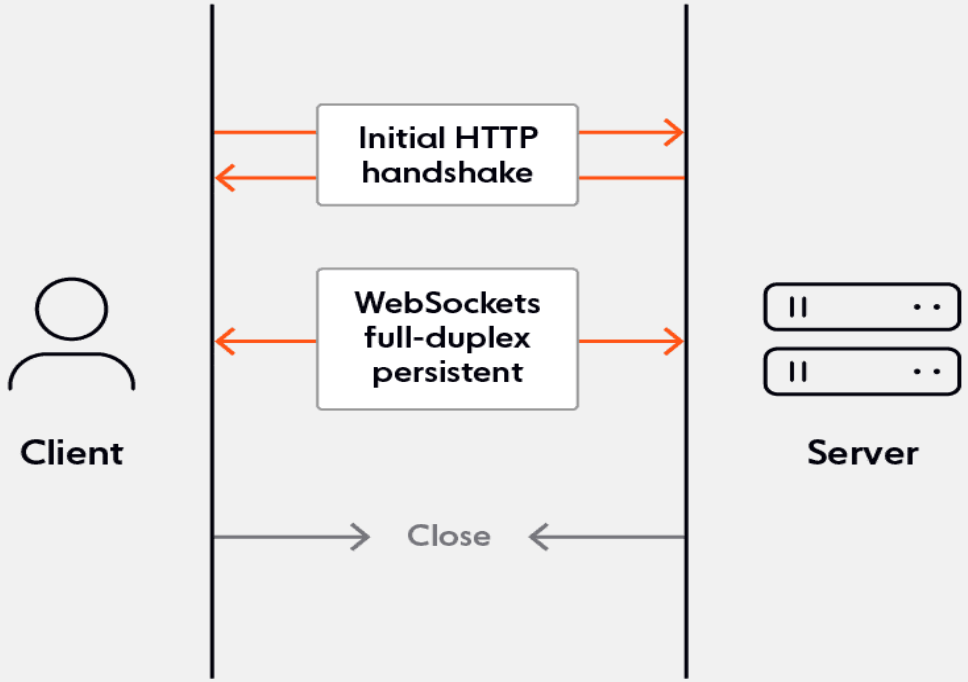


Figure: Websocket Working

Application Layer Protocol (Websockets)

Key Points:

1. Persistent Connection: Once established, the WebSocket connection remains open, allowing continuous data flow between client and server.

2. Bidirectional Communication: Unlike HTTP, WebSockets allow both client and server to send data to each other at any time, enabling real-time updates.

3. Low Overhead: WebSockets reduce the overhead associated with establishing multiple HTTP connections, making them more efficient for real-time applications like live chat, gaming, or financial tickers.

Basics of Sensor Network Topologies

In the context of the Internet of Things (IoT), *sensor networks consist of multiple sensor nodes that collect data from the environment and communicate with each other to transmit this data to a central system*. The way these sensor nodes are arranged and communicate with one another is referred to as the sensor network topology.

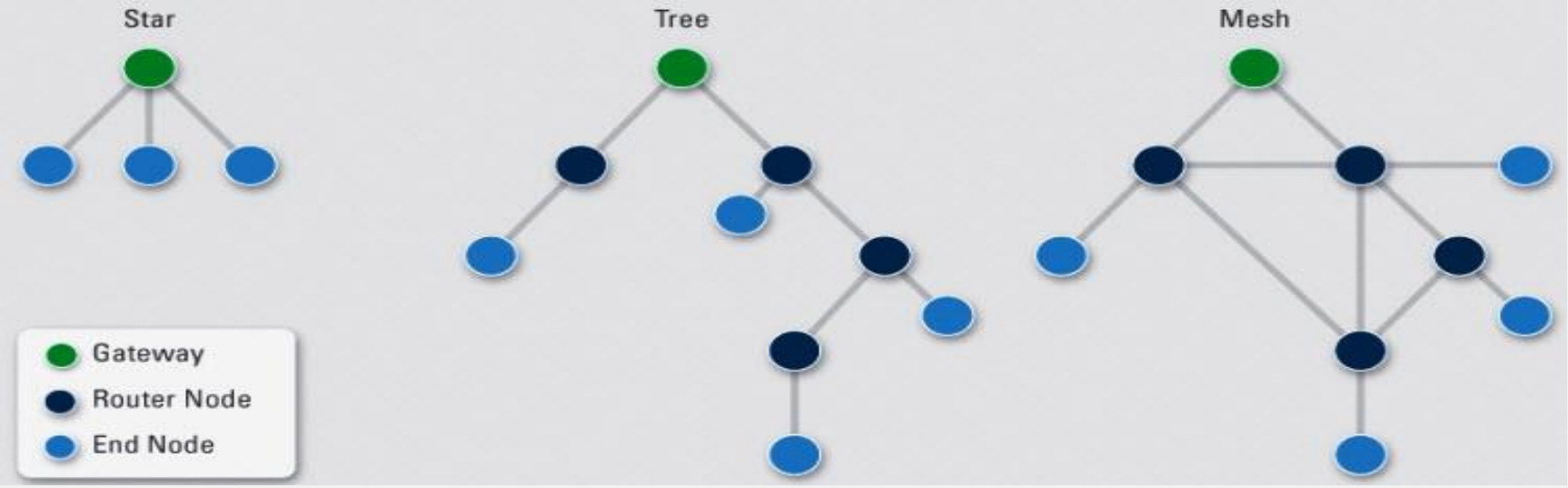


Figure: Examples of Sensor Network Topologies

Star Topology

- In a star topology, each sensor or node is connected to a central hub or gateway. The central hub manages all communication, so if one node needs to send data to another, it first sends the data to the hub, which then routes it to the target node.
- This topology is simple to implement and allows for easy addition or removal of nodes without affecting the entire network. However, if the central hub fails, the whole network is compromised.

- **Real-Life Example/Application**:
  Star topology is commonly used in home automation systems, where smart devices like lights, thermostats, and security cameras communicate with a central hub (e.g., a smart home controller or router).

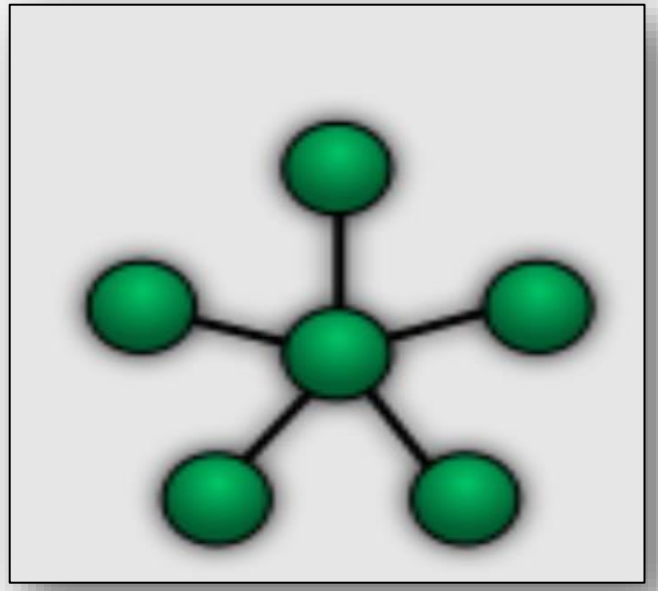


Figure: Star Topology

Mesh Topology

- In a mesh topology, every node is connected to multiple other nodes. This creates multiple paths for data to travel from one node to another, increasing redundancy and reliability.
- If one node fails, data can be rerouted through other nodes. This topology is highly resilient but can be more complex to configure and maintain, especially in larger networks.

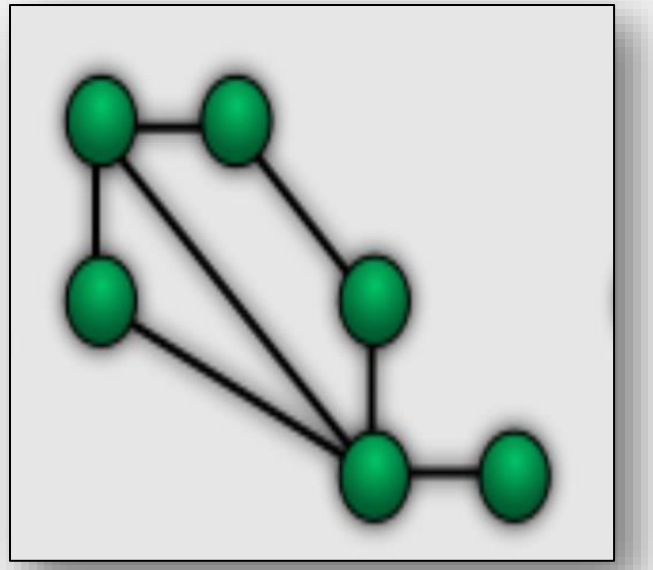


Figure: Mesh Topology

- **Real-Life Example/Application**:
  Mesh topology is often used in industrial IoT applications, such as smart factories, where sensors and devices need reliable communication across a large area with potential obstacles.

Tree Topology

- Tree topology is a hierarchical structure where nodes are connected in a parent-child relationship. Each parent node can have multiple child nodes, and data typically flows from the child nodes up to the parent nodes and eventually to a central hub.
- This topology is efficient for managing large networks and segmenting different parts of the network, but it can suffer from bottlenecks at the parent nodes.

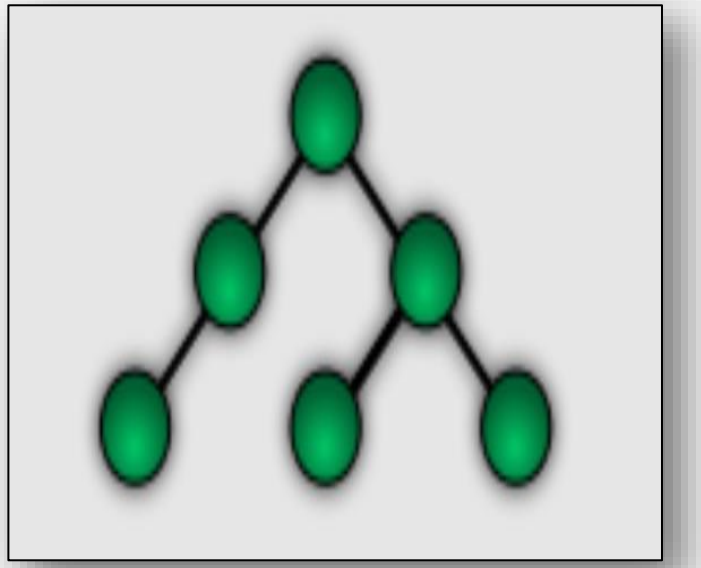


Figure: Tree Topology

- **Real-Life Example/Application**:
  Tree topology is often used in smart grid systems, where different levels of the grid (e.g., local substations, homes) communicate up the hierarchy to a central control system.

Bus Topology

- In a bus topology, all nodes are connected to a single communication line, known as the bus. Data travels along the bus, and each node checks if the data is addressed to it.
- This topology is cost-effective for small networks but can become inefficient as more devices are added, leading to collisions and slower communication.

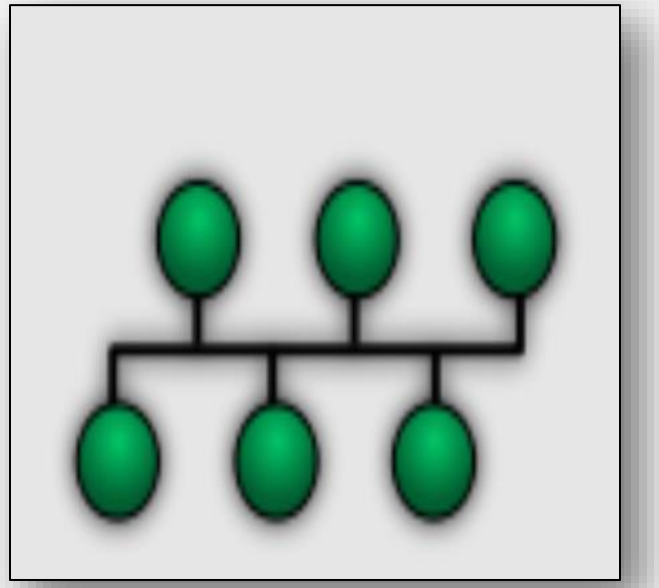


Figure: Bus Topology

- **Real-Life Example/Application**:
  Bus topology is often used in small-scale IoT networks, such as a simple sensor network for environmental monitoring in a small area, where all sensors share a common communication line.

Ring Topology

- In a ring topology, nodes are connected in a circular manner, with each node connected to exactly two other nodes, one on either side.
- Data travels in one direction (or both directions in a dual-ring setup) around the ring. This topology can be efficient for certain applications, but a failure in any single node or connection can disrupt the entire network.

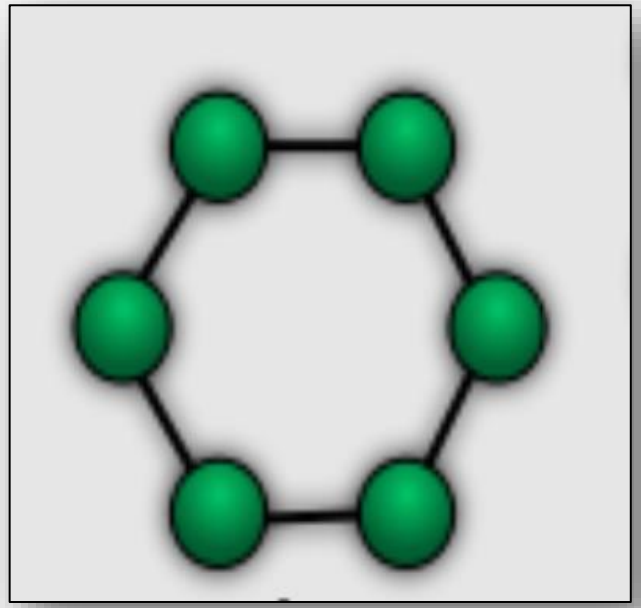


Figure: Ring Topology

- **Real-Life Example/Application**:
  Ring topology is used in some industrial control systems, where reliable, predictable communication paths are needed for real-time monitoring and control.

SUMMARY

- Link layer protocols handle raw data transmission between devices on a network, addressing framing, error correction, and access methods.
- Network/internet layer protocols route data packets across networks, using IP addressing and routing algorithms.
- Messaging protocols facilitate communication between applications. MQTT is lightweight, ideal for IoT devices with limited resources, using a publish-subscribe model.
- CoAP is also optimized for constrained environments, offering a web-like approach for IoT applications.
- XMPP, designed for real-time communication, supports features like presence and instant messaging, often used in enterprise and social applications.
- The transport layer ensures reliable data transfer between devices in IoT networks by managing end-to-end communication.
- The application layer enables communication between IoT devices and applications by translating data into formats understandable by software.

SUMMARY

- The transport layer ensures reliable data transfer between devices in IoT networks by managing end-to-end communication.
- The application layer enables communication between IoT devices and applications by translating data into formats understandable by software.
- Different sensor network topologies are used as the per the requirement or applications of the system.

REFERENCES

1. https://www.geeksforgeeks.org

2. I.A. Dhotre, “Internet of Things”, Technical Publications, 1st Edition 2021.

3. https://www.scaler.com

4. https://www.tutorialspoint.com

5. https://www.javatpoint.com

Thank You