IoT Communication Protocols

Unit - IV

Topics to be covered

- Need of various Protocols for IoT Communication
- Messaging Protocols: MQTT, CoAP, XMPP
- Transport Protocols: BLE, Li-Fi
- Sensor Network Topology: Point-to-point, Mesh, Ring, Star

Need of protocols in IoT Communication

- IoT systems are generally installed in some remote areas, whereas data processing and controlling is done at some other place.
- So there is need for data transmission from one place to another, which is wireless or wired.
- But every-time when the communication occurs between different devices, some common set of rules needs to be established
- Protocols work for providing a proper communication channel through all the layers
  - **Messaging Protocols:** For transfer of data.
  - **Transport Protocols:** For transferring the message from Source to Destination.

Messaging Protocols

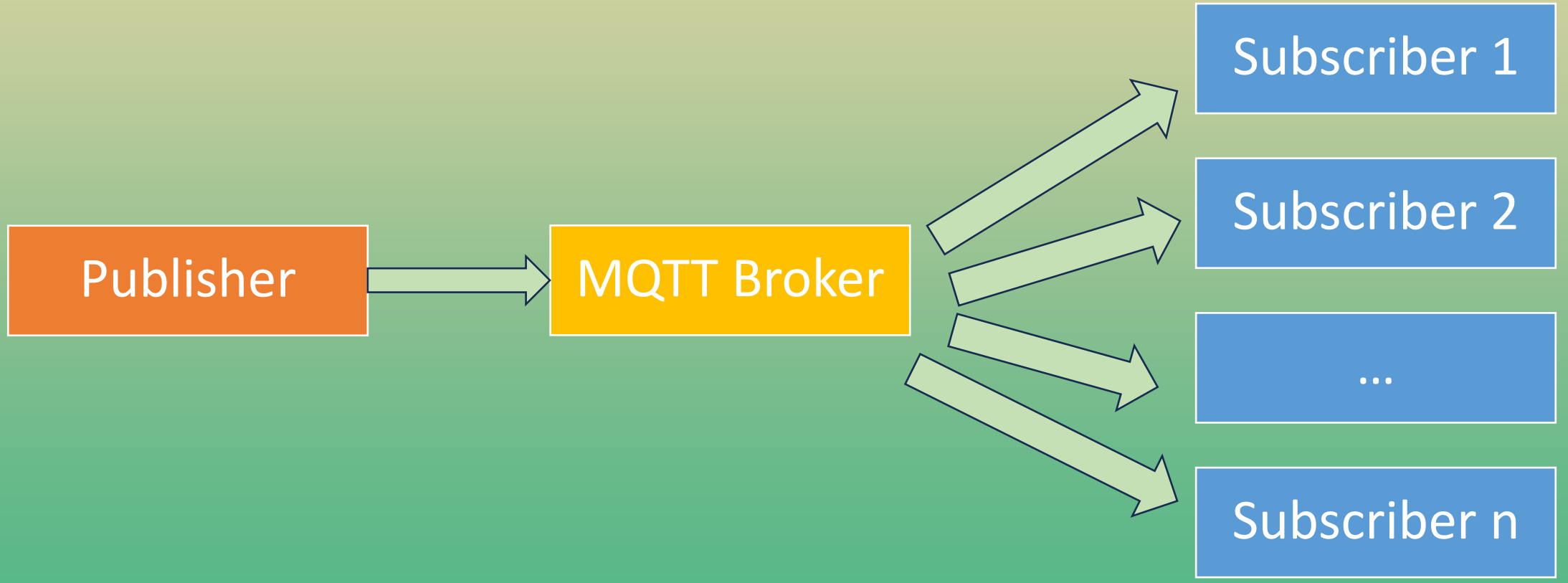
- Messaging protocols are very important for the transfer of data in terms of messages.
- They are useful for send/receive of a message to/from the cloud in IoT applications.
- 3 protocols are going to be discussed here
  - **MQTT:** Message Queuing Telemetry Transport
  - **CoAP:** Constrained Application Protocol
  - **XMPP:** Extensible Messaging and Presence Protocol

Messaging Protocols: MQTT

- Message Queuing Telemetry Transport
- In IoT biggest challenge is Resource Constraint, so light-weight protocols are needed.
- MQTT is one of the widely used protocols which is a very lightweight, meaning this can work with minimal resources and does not require any specific hardware architecture.

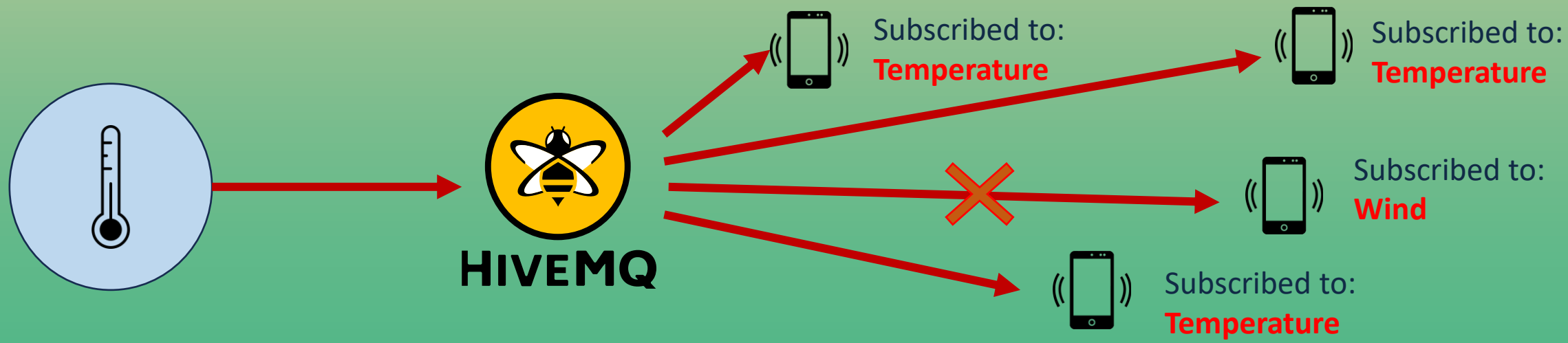
Messaging Protocols: MQTT Working

- Works on “Publish – Subscribe” pattern.
  - The publisher (node) sends a message to the MQTT Broker
  - MQTT Broker transmits the message to the destinations who have subscribed.



Messaging Protocols: MQTT

- Suppose a temperature sensor is connected to any system and we want to monitor that temperature.
- First of all the sensor will sense the temperature and send it to the controller.
- Controller then reads and then it publishes the temperature to the MQTT Broker, where the registration was previously done under a particular ID (ex. Temperature).
- Then MQTT Broker will transmit the value of that temperature to all the subscribers who have subscribed to this ID (Temperature).
- Here an MQTT Broker can handle multiple IDs, so only those subscribers will get the data of Temperature who have subscribed for this data.

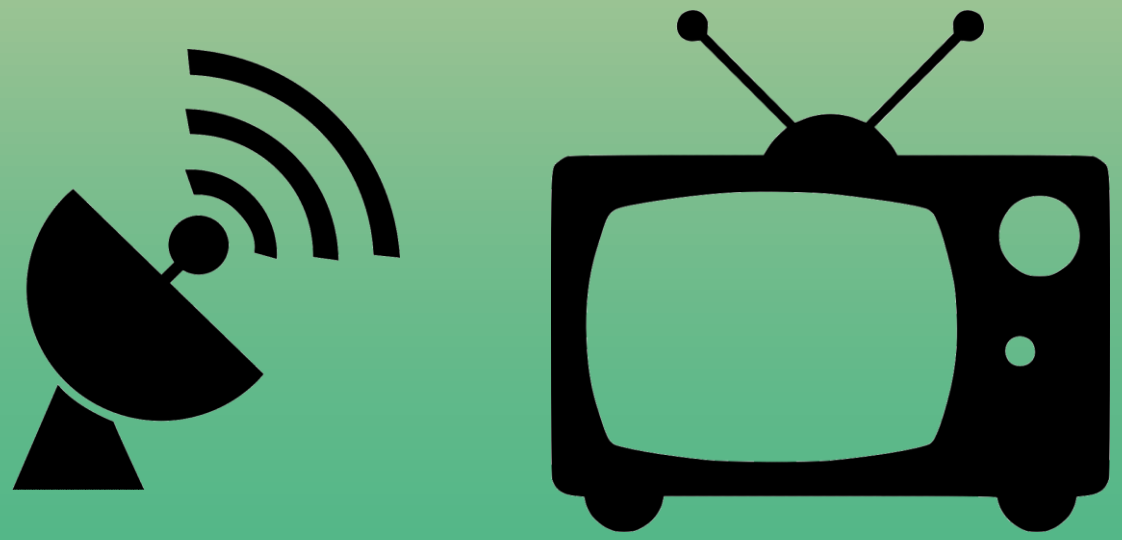


Messaging Protocols: MQTT

- Communication can also be done in a 2-way manner, i.e. Subscriber of one data can be a Publisher of some other data as well.
- Self Driving car will Publish location to broker and then to the system/server which is the subscriber of “CurrentLocation”.
- Based on the selected destination and current location received, the system/server will publish “Direction” data to MQTT Broker and then to the car (Subscriber of: Direction).
- Then the car will turn to the desired direction as per the data received.

Messaging Protocols: MQTT

- Here client does not have any address like typical networking.
- Broker simply filters messages based on subscribed “Topics”.
- Then Message is circulated to respective subscribers like Television broadcasting.
- In television broadcasting, a broadcaster will broadcast all the channels and the subscriber can view only channels for which they’ve subscribed.
- Topic name is given in a String.



Messaging Protocols: MQTT

- Node(s) in MQTT collects the information from sensors or some I/O devices, if it is a publisher.
- Connects it to the messaging server MQTT Broker.
- A special string – Topic is used to publish and identify the message, this message is relayed to all the subscribers.
- Any node can be a subscriber or publisher or both, those are also known as Clients.

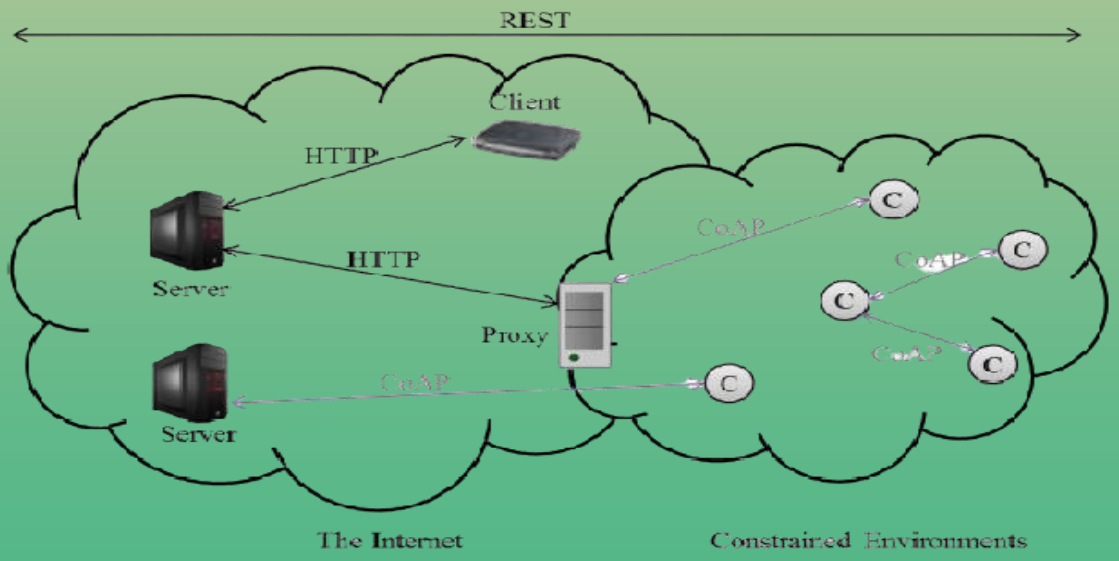
Messaging Protocols: MQTT

- MQTT server has different configurations to manage **QoS (Quality of Service)**.
- There are 3 levels of QoS:
  - Level 0: **At most once** (Best Effort, No Acknowledgement)
  - Level 1: **At least once** (Acknowledged, retransmitted until acknowledged)
  - Level 2: **Exactly once** (Request to send, Clear to send)
- Some Cloud based web service providers/brokers for MQTT are available.
  - HiveMQ
  - Eclipse MQTT
  - EMQX
  - jMQTT
  - Mosquitto
  - Cloud MQTT etc.

Messaging Protocols: CoAP

CoAP

- Constrained Application Protocol, designed by IETF (Internet Engineer Task Force), to work in a constrained environment such as those with low-bandwidth, high-latency, or limited processing power.
- It is a one-to-one communication protocol.
- It is lightweight like MQTT and uses less resources than HTTP.
- HTTP runs over TCP, Connection-Oriented, where CoAP runs over UDP, Connection-less.



Messaging Protocols: CoAP

CoAP

- CoAP is based on the RESTful architecture (Representational State Transfer).
- REST approach ensures the secure, fault tolerant and scalable system.
- The CoAP optimizes the length of the datagram.
- CoAP is connectionless protocol and it requires retransmission support.

Messaging Protocols: CoAP

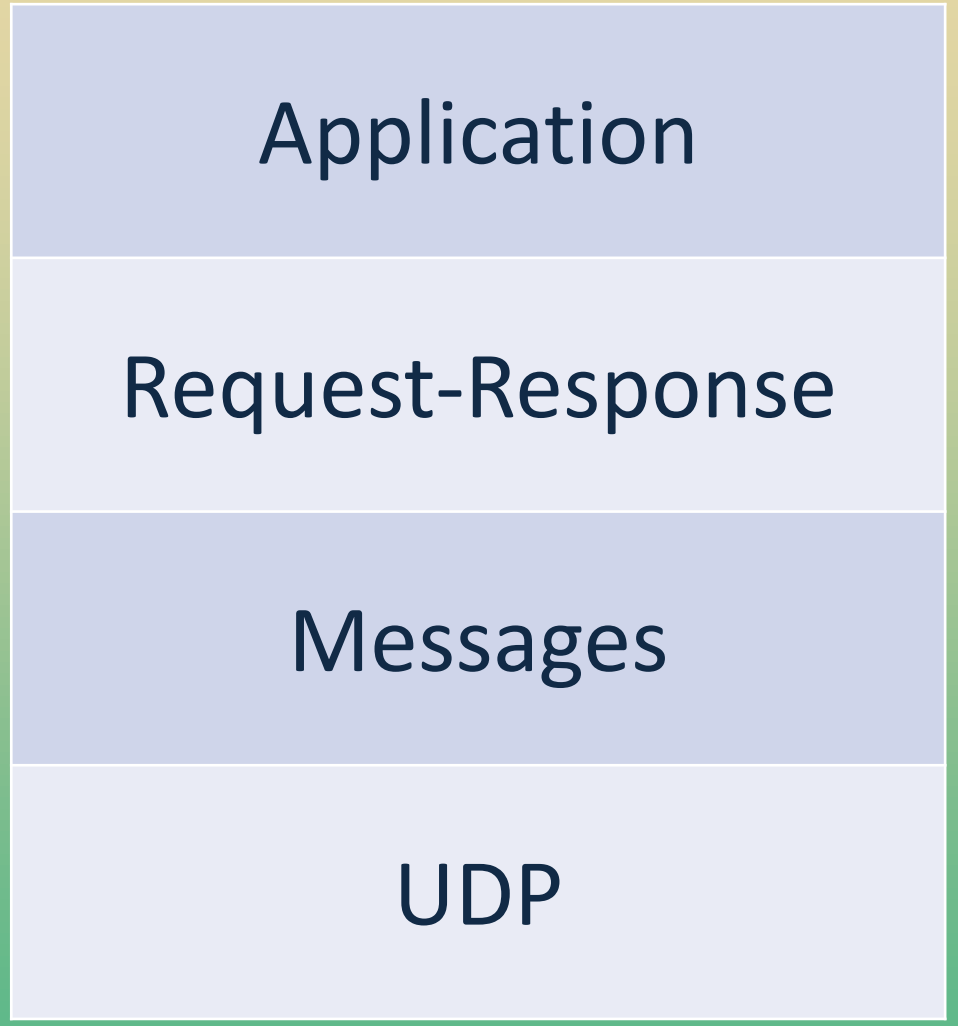
CoAP

- Some characteristics include:
  - **RESTful protocol:** Works on REST Principles, hence suitable for building scalable and resource-oriented applications.
  - **UDP as Transport protocol:** Lightweight and connectionless, suitable for IoT applications.
  - **Low Overhead:** Protocol specific overhead is less to conserve resources, designed to be efficient int terms of message size and processing requirements.
  - **Request-Response model:** Similar to HTTP – Client -> Requests, Server -> Responds
  - **Observing Resource:** Allows clients to receive message from server, whenever a resource changes its state.
  - **URI Support:** URI model is used to identify resources over the network, similar to URLs in HTTP.
  - **Security:** Uses DTLS (Datagram Transport Layer Security), ensuring Confidentiality and Integrity.
  - **Multicast Support:** Allows single message to be sent to multiple clients.

Messaging Protocols: CoAP

CoAP

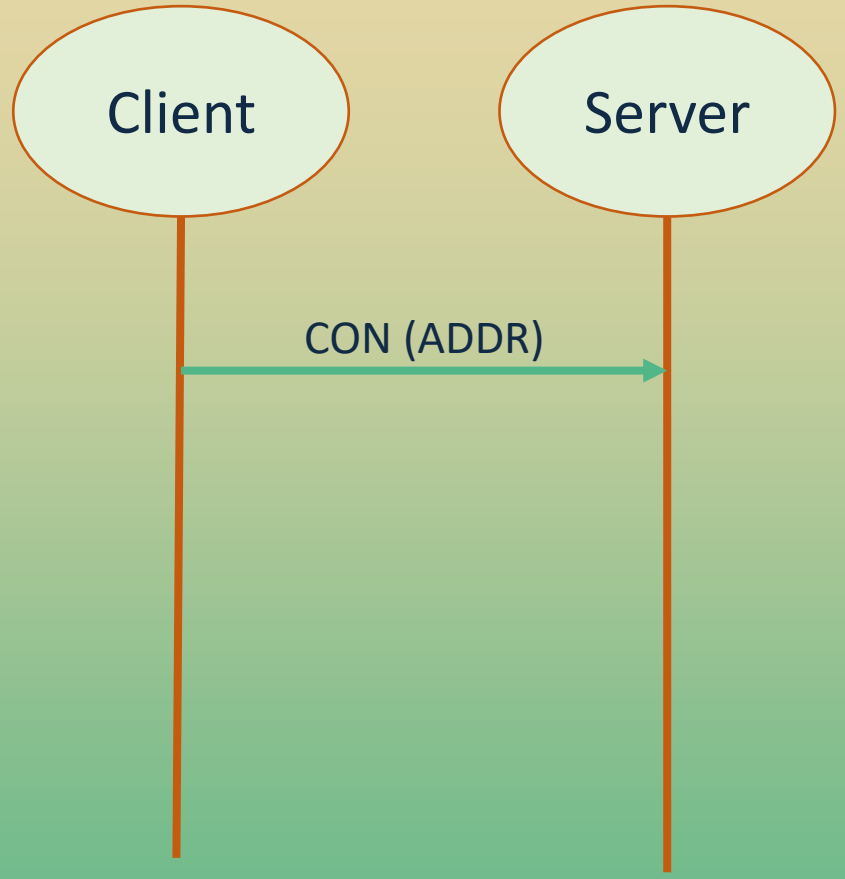
- CoAP has 4 layers
  - **UDP :** Responsible for Transmitting/ Receiving the messages
  - **Messages:** Deals with UDP layer and asynchronous switching
  - **Request-Response:** Concerns communication methods to deal with messages, client can use GET/PUT/DELETE methods to transmit the message.
  - **Application:** The higher layer for interaction with the user



Messaging Protocols: CoAP

CoAP

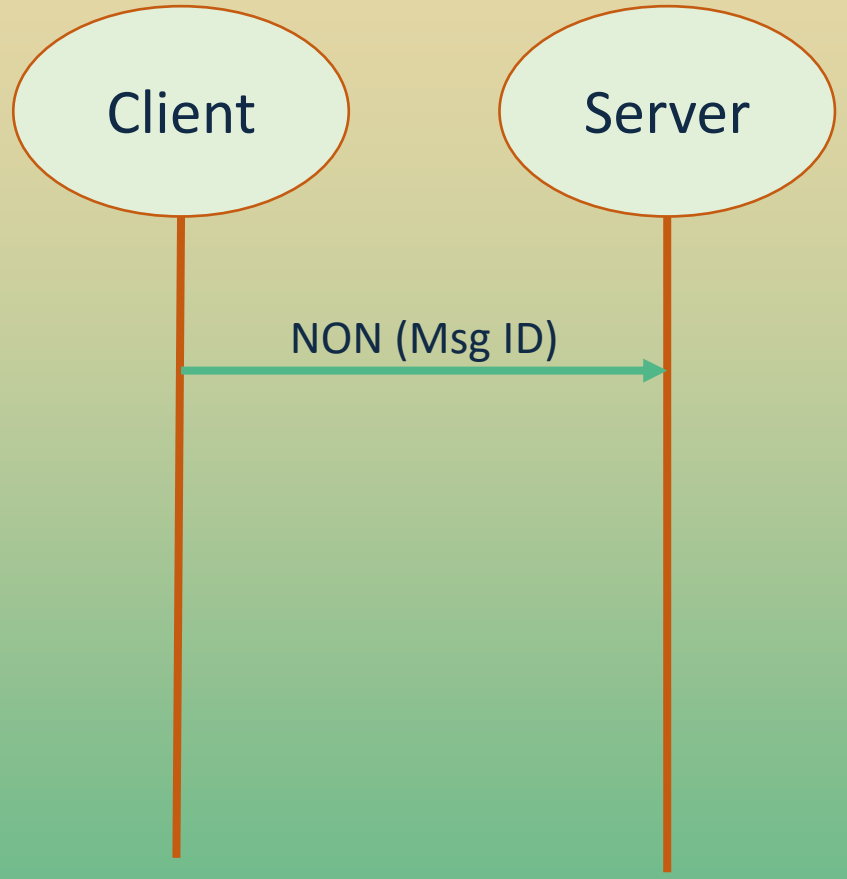
- CoAP message layer supports 4 types of messages:
- **CONfitmable – Reliable Messagging (CON):**
  - Here Acknowledgement is recieved from the Server.
  - If timed-out or Fail of Acnowledgement occurs, RST message from the server is sent as a response and the client will retransmit.
  - Each message has a unique ID to Identify the acknowledgement received.



Messaging Protocols: CoAP

CoAP

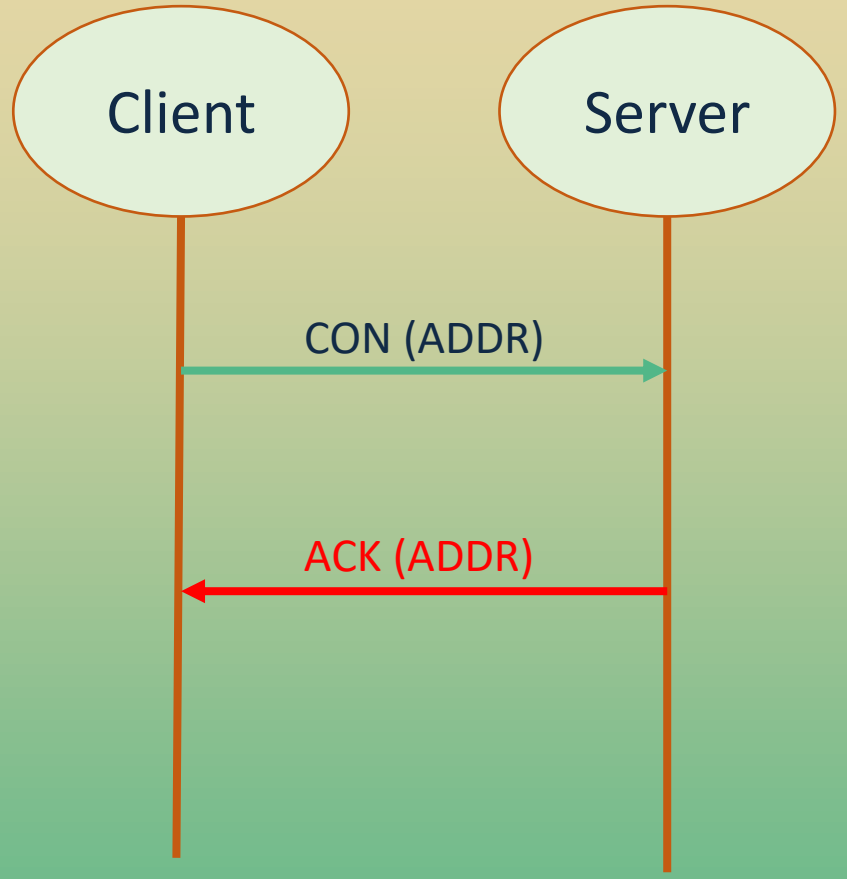
- **NON-Confirmable – Non-reliable Messaging (NON):**
  - No reliability is ensured, as there is no acknowledgement by the server.
  - Message has ID to unique each one individually.
  - If the message is not processed by server, RST response is transmitted.



Messaging Protocols: CoAP

CoAP

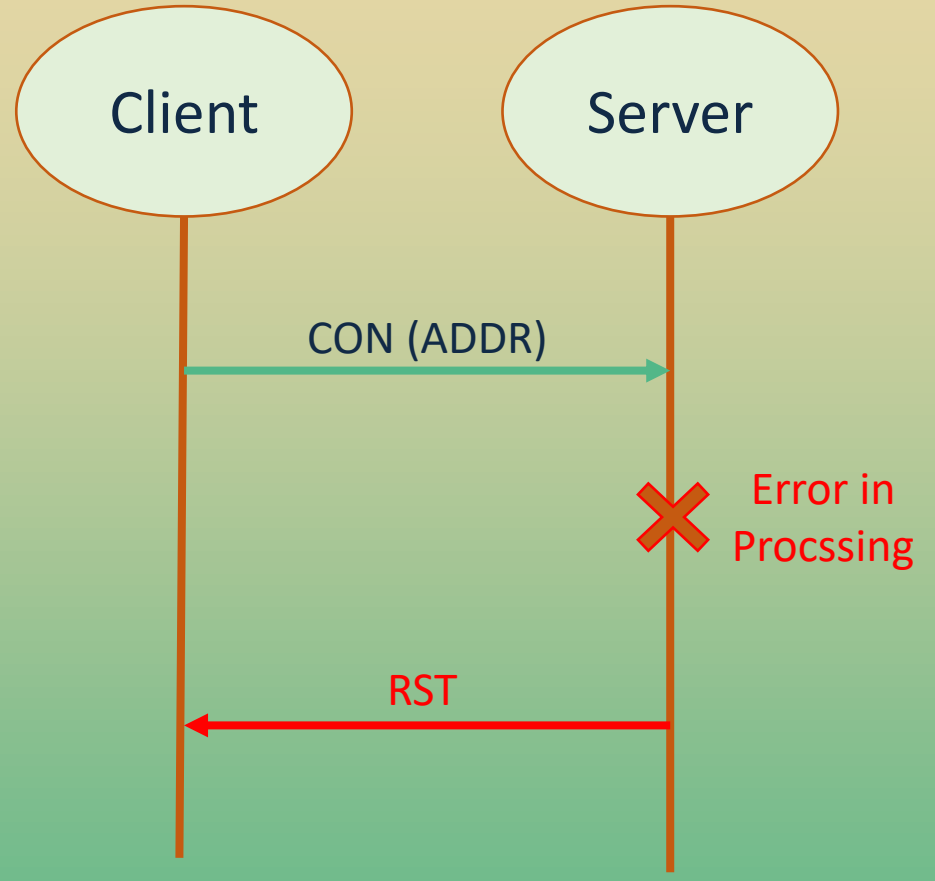
- **ACKnowledgement (ACK):**
  - Like traditional messaging, an acknowledgement message is sent to the client.
  - Similar to simple handshaking.
  - ACK denotes that the message is successfully processed in server without any errors/issues.



Messaging Protocols: CoAP

CoAP

- **ReSeT (RST):**
  - If server encounters some issue processing the message, server sends and RST message to the client.
  - It informs the client that the message with the ID which was last sent should be disregarded.



Messaging Protocols: CoAP

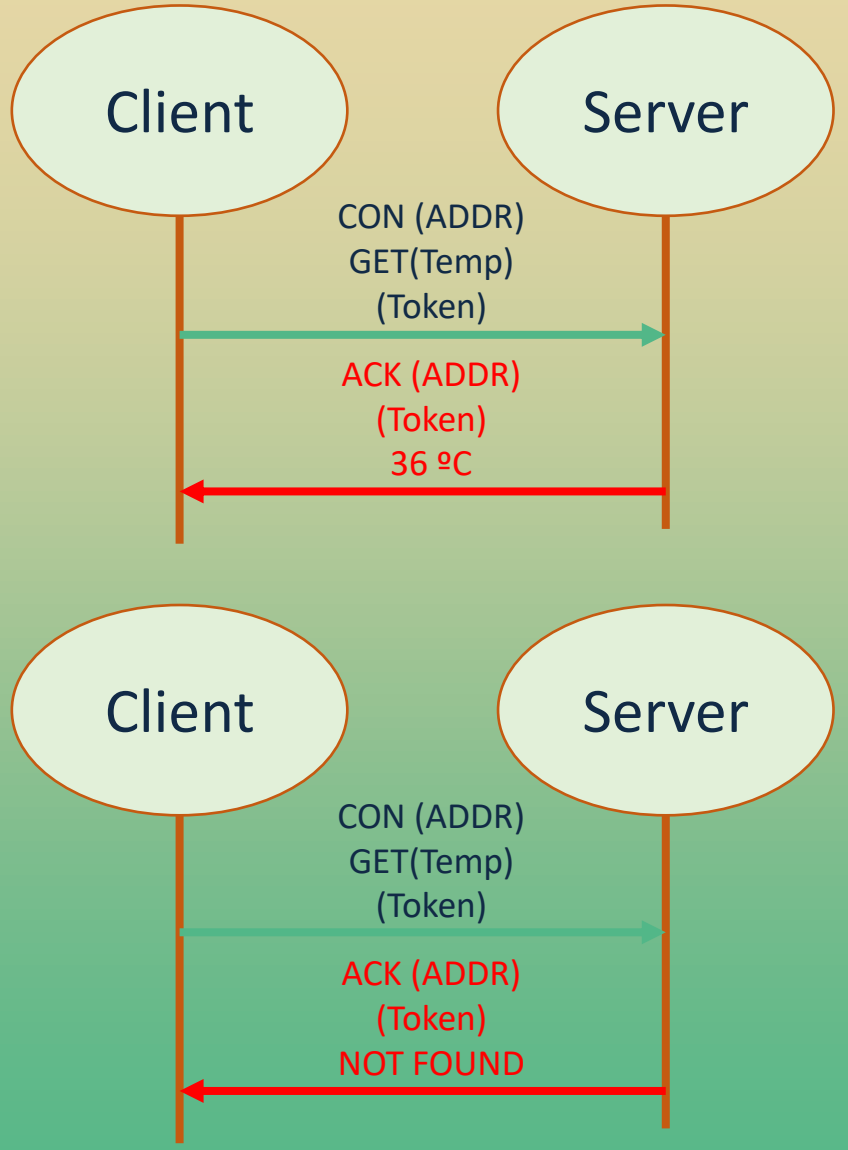
CoAP

- 3-Modes of communication in Request-Response Layer
  - **Piggy-Backed**
  - **Separate Response**
  - **Non-Confirmable Request and Response**

Messaging Protocols: CoAP

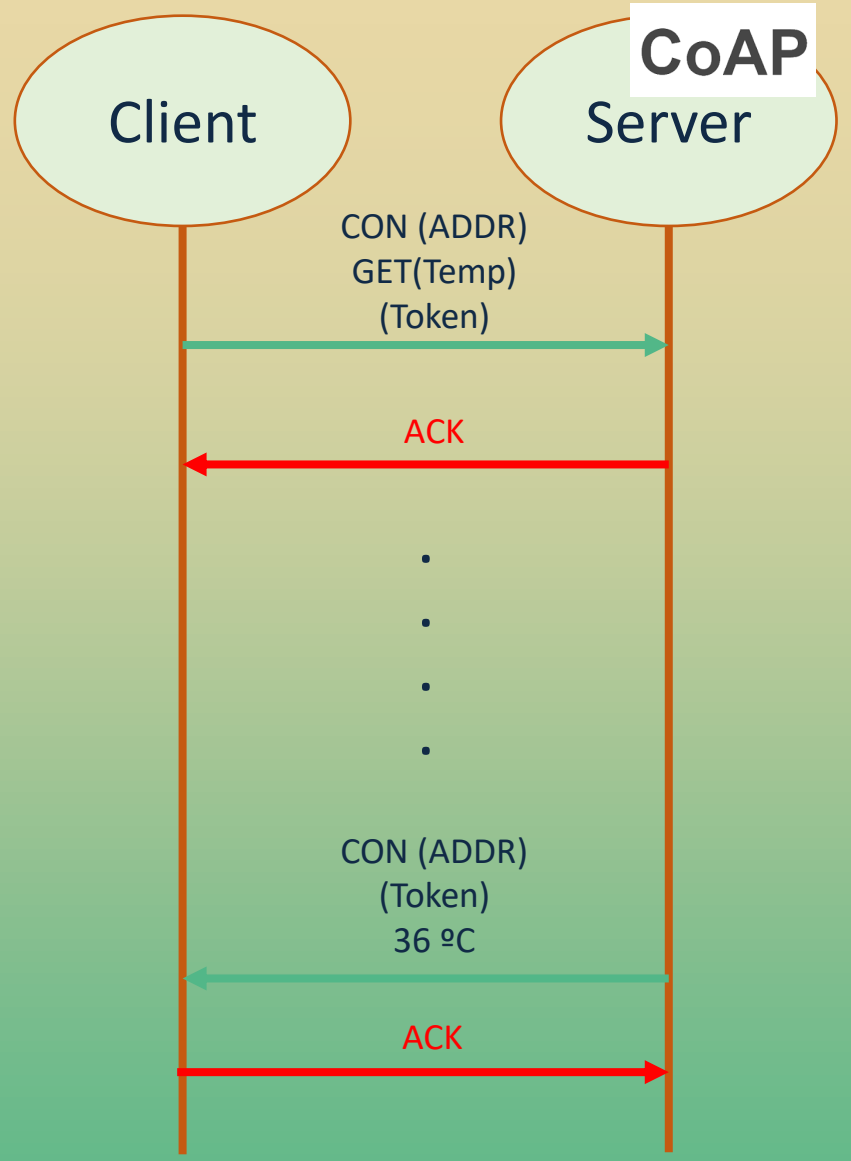
CoAP

- **Piggy-Baked:**
  - Client sends the data in a particular way (GET/PUT etc.) with token (ID) number and CON method.
  - ACK is transmitted by server immediately with corresponding token (ID) and message
  - If message is not received by the server or data is not available then server will send failure as acknowledgement



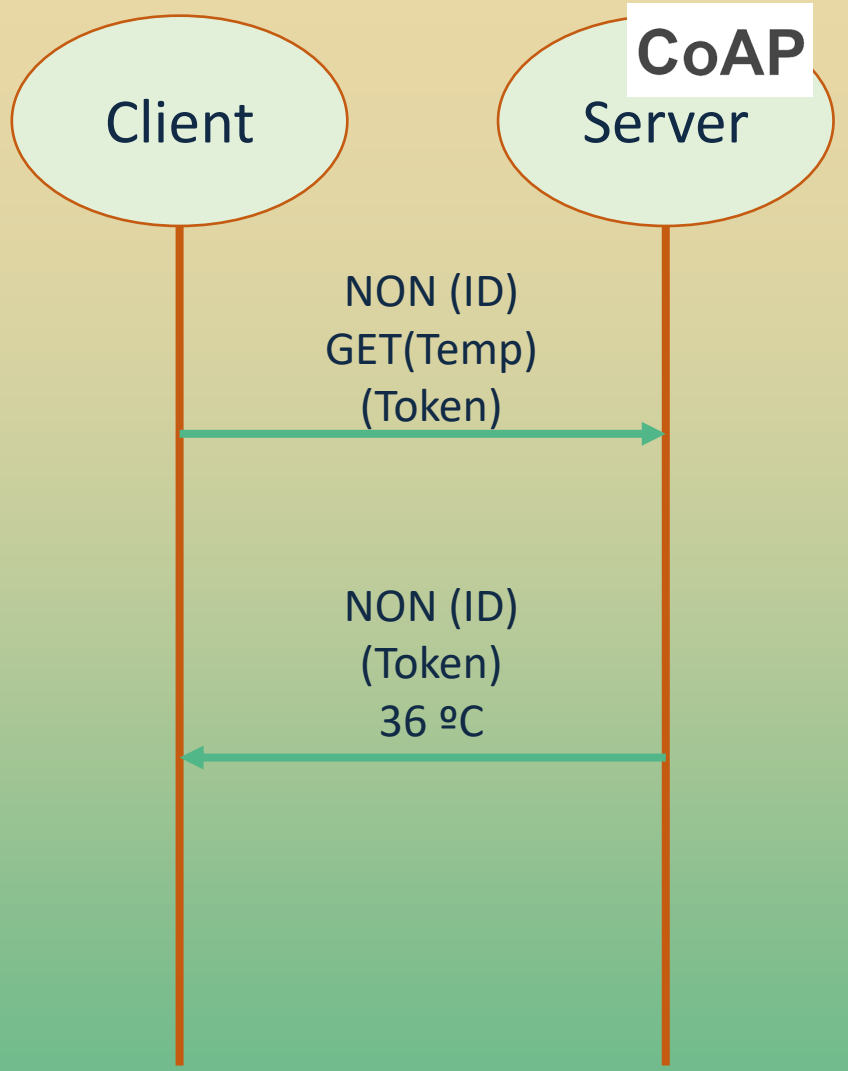
Messaging Protocols: CoAP

- **Separate Response:**
  - Client sends the data in a particular way (GET/PUT etc.) with token (ID) number and CON method. (Same as Piggy-Baked method)
  - If server is busy at the moment, and empty
  - After some time when the server has data ready, it sends a CON message with the data to client.
  - In response to that, client sends the ACK message to Server.



Messaging Protocols: CoAP

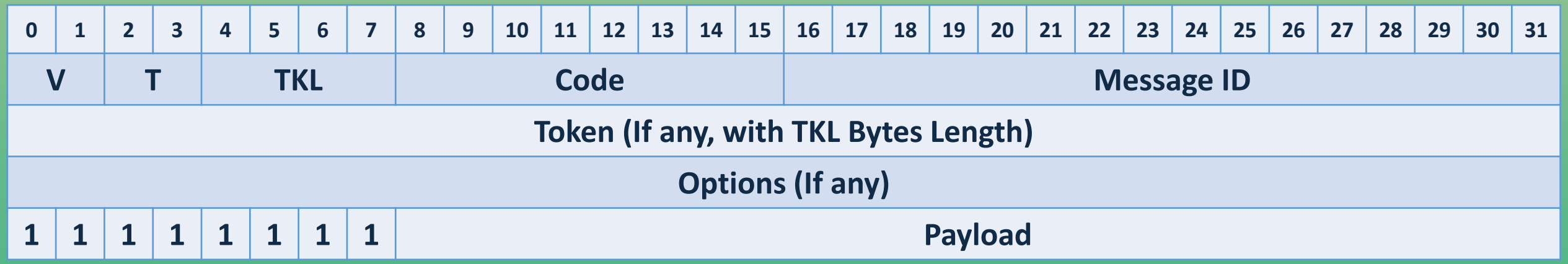
- **Non-Confirmable Request-Response:**
  - Here client sends a message with particular method (GET, PUT etc.) with a token (ID) as a NON message.
  - In response, server sends a NON message with a token (ID) and its data



Messaging Protocols: CoAP

CoAP

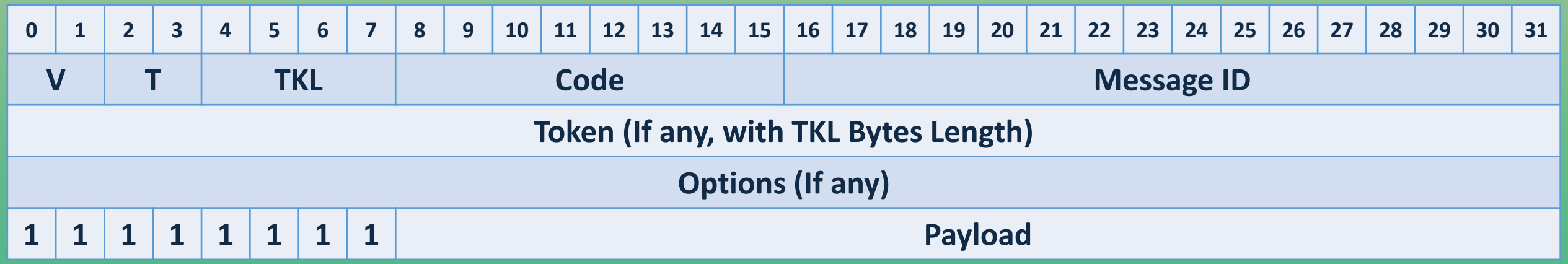
- CoAP message format is of 16-Bytes consisting various fields.
  - **V:** CoAP Version, 2-bits unsigned int
  - **T:** Message Type, 2-bits unsigned int
    - 0: CON
    - 1: NON
    - 2: ACK
    - 3: RST
  - **TKL:** Token length, 4-bits unsigned int



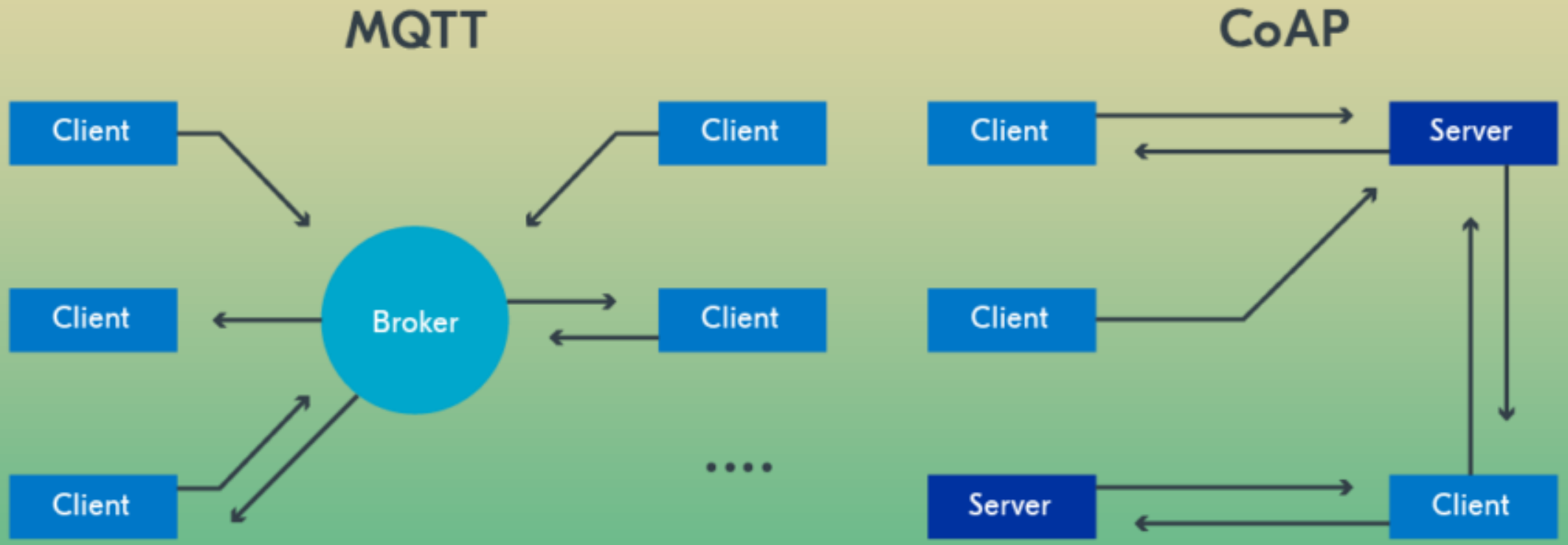
Messaging Protocols: CoAP

CoAP

- **Code:** Response Code of message
- **Message ID:** Message Identifier, to avoid duplication.
- **Token:** Optional field, size of the field is defined in TKL
- **Options:** 4-bytes Options field is available to indicate any options available.
- **Payload:** refers to the Payload Data (Actual Data) to be transmitted.



MQTT v/s CoAP


MQTT
Client
Client
Client
Broker
Client
Client
....
CoAP
Client
Server
Client
Server
Client

Messaging Protocols: XMPP

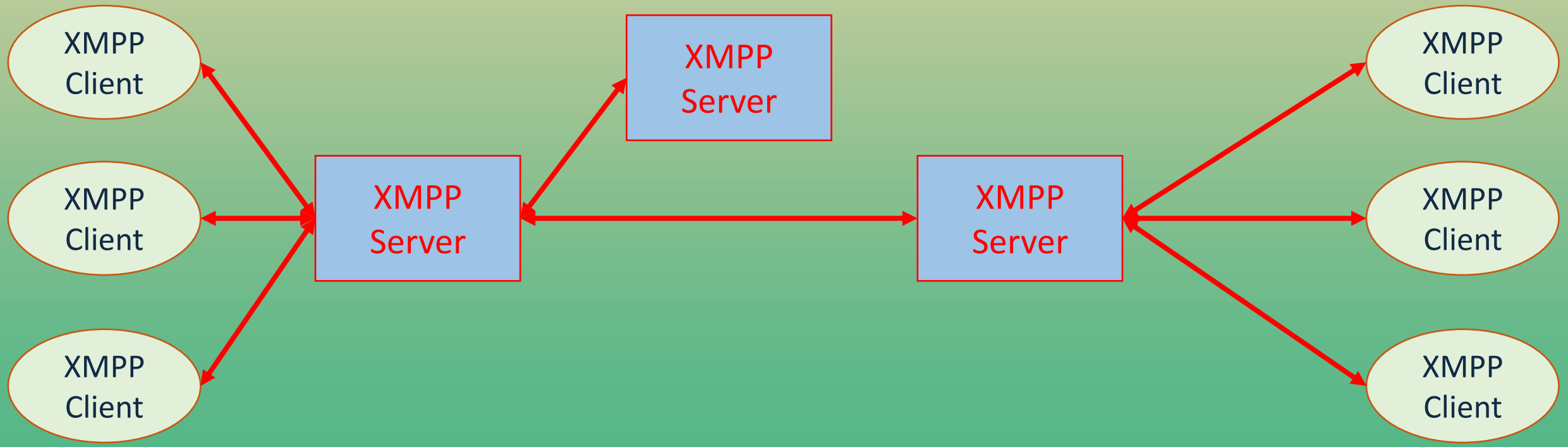
- eXtensible Messaging and Presence Protocol
- Designed for Messaging, Chat, Video & Voice calls and Collaboration.
- Used for messaging services.
- Previously known as Jabber.
- **X – eXtensible:**
  - Designed to accept and accommodate any changes.
  - It is Extensible because of the open standard approach.
- **M – Messaging:**
  - Supports sending messages to the recipients in real-time.
- **P – Presence:**
  - Used to identify status of recipients as “Online”, “Offline”, “Busy”.
  - This helps understand, whether the recipient is ready to receive or not.
- **P – Protocol:**
  - Defines set of standards for messaging and presence, hence protocol.

Messaging Protocols: XMPP

- User requires following inbuilt features:
  - Message Sending or Receiving.
  - Understanding of the Presence / Absence of Recipient.
  - Managing Subscription details.
  - Managing Contacts List.
  - Message Blocking service.

Messaging Protocols: XMPP

- Architecture is as depicted in the figure.
- XMPP client can communicate bidirectionally with XMPP server.
- More than one clients can be connected to a single XMPP server.
- XMPP Server may be connected to another XMPP Client or Servers.



Messaging Protocols: XMPP

- Initially XMPP was with TCP using open ended XML.
- Later on, XMPP was developed based on HTTP.
- XMPP has 2 different methods of communication while working with HTTP.
- **Polling (Pull Method):** Message are stored in server and fetched/pulled by client using GET/POST HTTP.
- **Binding (Push method):** Bidirectional-streams Over Synchronous HTTP (BOSH) enables the servers to push the messages to the clients when they are required.
- Binding method is more effective than Polling.
- Also both methods uses HTTP, so fetch and post activities can be done through firewall, which is allowed in a secured manner.

Transport Protocols

- To transmit the data generated in a node through out the network, some set of transport protocols are available like TCP, UDP which are already used in various protocols.
- But for some part of IoT system, an IoT device might not be capable of handling requirement of processing, transmission or reception of the messages.
- For that some specific protocols are designed with some minor modifications in the already available protocols.
  - **Bluetooth 4.0 – BLE**
  - **Light Fidelity – Li-Fi**

Transport Protocols: BLE

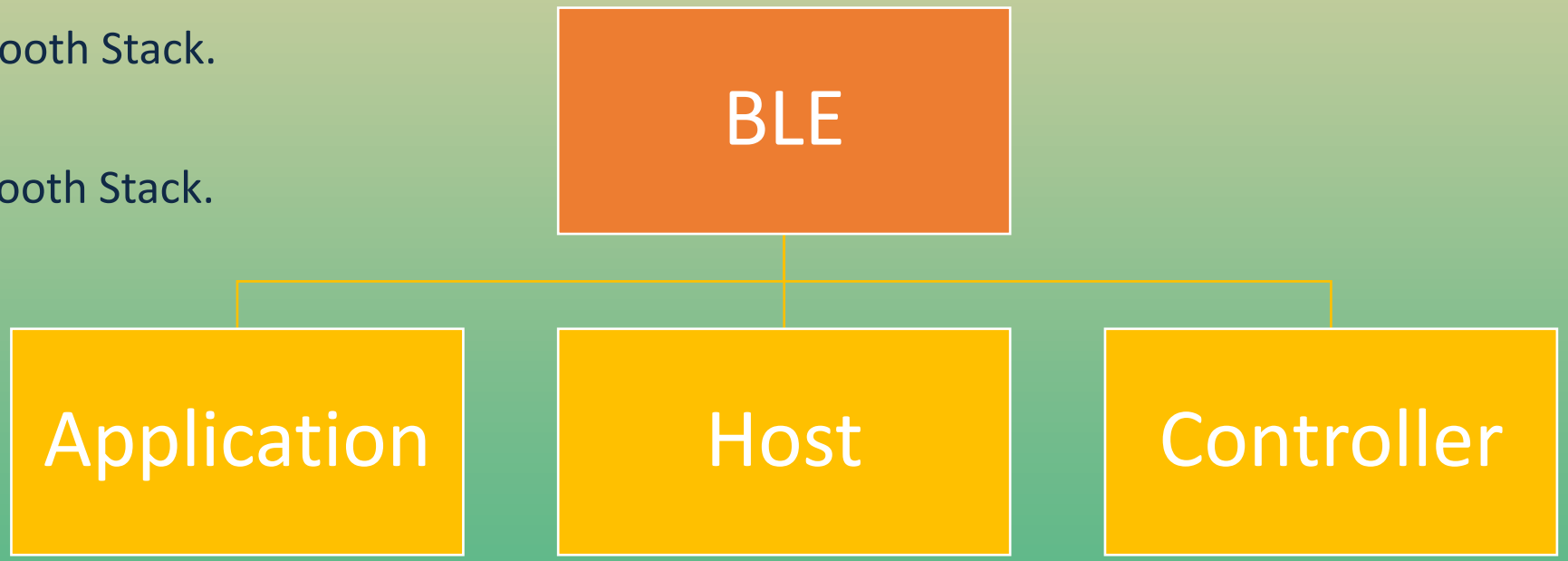
- Bluetooth Low Energy
- BLE is open low energy short range radio communication technology.
- This protocol is designed by Bluetooth Special Interest Group (SIG)
- Working is same as Bluetooth in PAN (Personal Area Network)
- BLE is considered superior to the Bluetooth due to some features like,
  - **Power Saving**
  - **Supported by Android, iOS, BlackBerry etc.**
  - **Supported by Computer OS like, Windows, MAC, Linux variants.**
- Because of this, BLE is widely used in areas like,
  - Healthcare
  - Environment
  - Fitness (Wearable Devices)
  - Proximity related applications
  - Tracking Devices

Transport Protocols: BLE

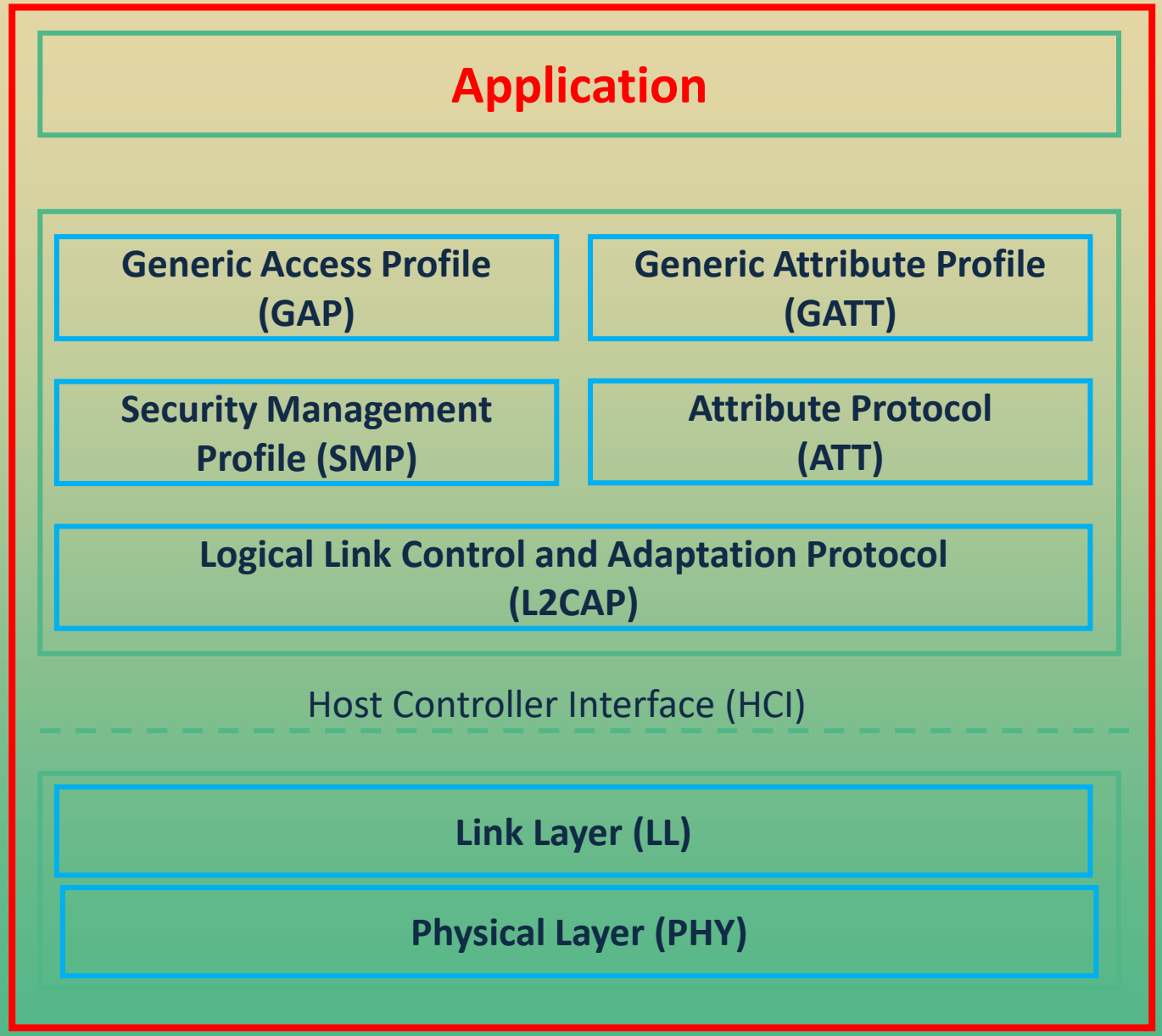
- There are some key features of BLE that makes it more suitable with IoT,
  - Licence free and no additional cost is required.
  - No restrictions to manufacturer
  - BLE modules are inexpensive and affordable
  - Modules are small in size, and less power requirements
  - Range of BLE is higher than classic Bluetooth.

Transport Protocols: BLE

- Architecture of BLE consists of mainly 3 components
  - Application Block
    - Here user application is accommodated.
    - Interacts directly with Bluetooth Stack.
  - Host Block:
    - Upper layer of Bluetooth Stack.
  - Controller Block
    - Lower layer of Bluetooth Stack.

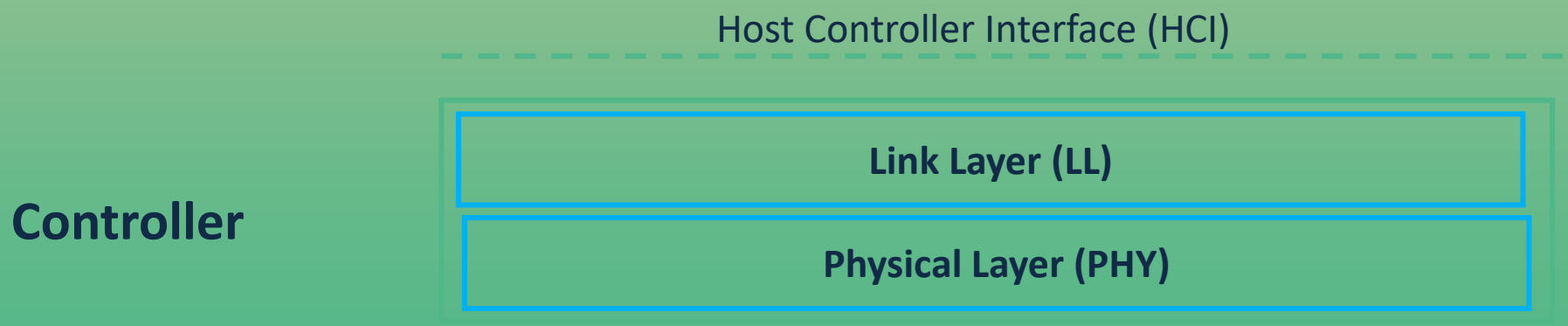


Transport Protocols: BLE



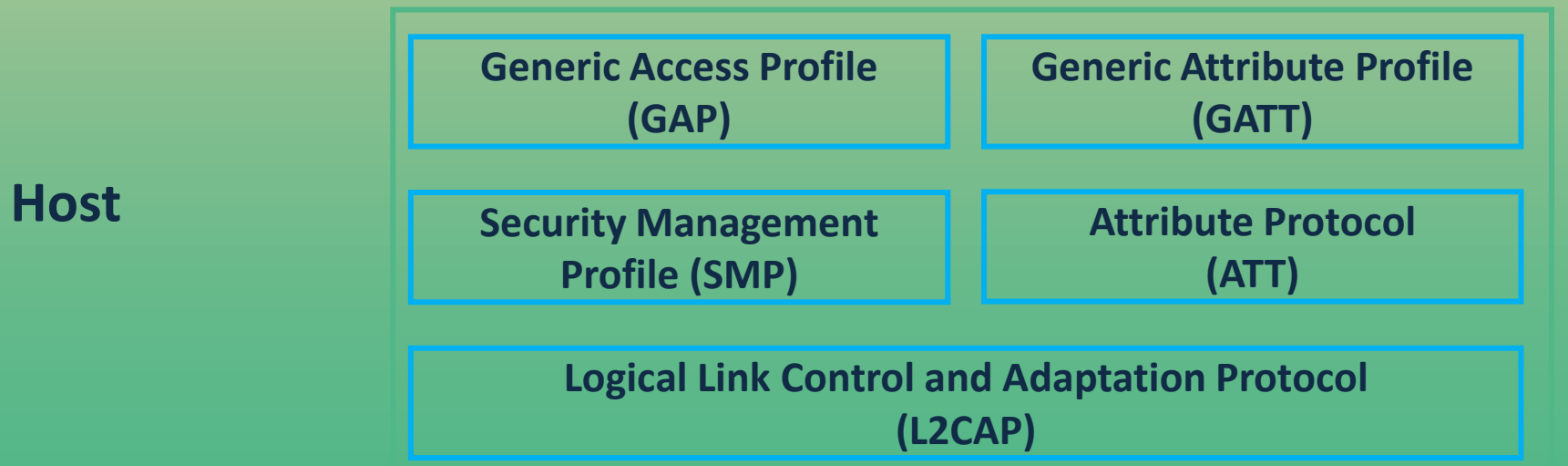
Transport Protocols: BLE

- **Controller**: It has 3 Components
  - **Host Controller Interface (HCI):** Used to enable interoperability between hosts and controllers assembled by different manufacturers.
  - **Link Layer (LL):** Defines the packet structure.
  - **Physical Layer (PHY):** It takes care of transmission/reception, modulation/demodulation and analog-to-digital and vice-versa conversion.



Transport Protocols: BLE

- **Host:** It has 6 Components
  - **Generic Access Profile (GAP):** Device discovery, Connection Management and Security.
  - **Generic Attribute Profile (GATT):** For Data Exchange process. Generic guidelines are given by GATT to Push data or read/write data.
  - **Attribute Protocol (ATT):** Protocol for Accessing data.
  - **Logical Link Control and Adaptation Protocol (L2CAP):** For Fragmentation/De-fragmentation of Application data. Also Multiplexing/de-multiplexing of channels over the shared Logical Link.
  - **Security Manager (SM):** Takes care of Pairing, Authentication and Encryption.
  - **Host Controller Interface (HCI):** Used to enable interoperability between hosts and controllers from different manufacturers



Transport Protocols: BLE

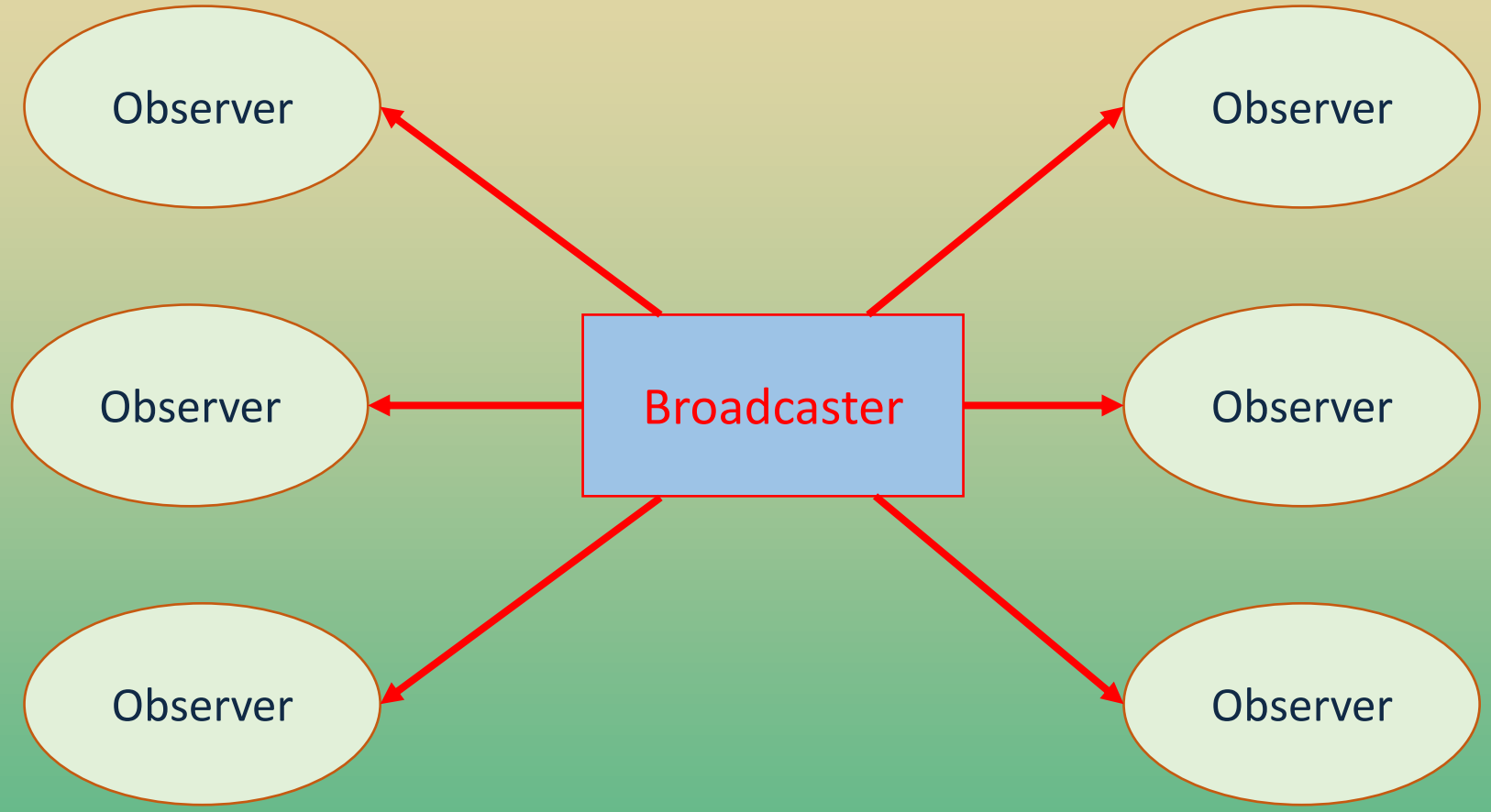
- **Application:** It has only 1 Component
  - **Application:** It is same as Application Layer in OSI Model. Top most layer, which is responsible for UI, Logic and Data Representation.

Application

Application

Transport Protocols: BLE

- Broadcasting in BLE:
  - Meaning sending a message to more than one recipient
  - Only One-way Comm.
  - If receiver is in the vicinity of the broadcaster then the message will be received
  - 2 – Parties are involved in this:
    - **Broadcaster**
    - **Observer**



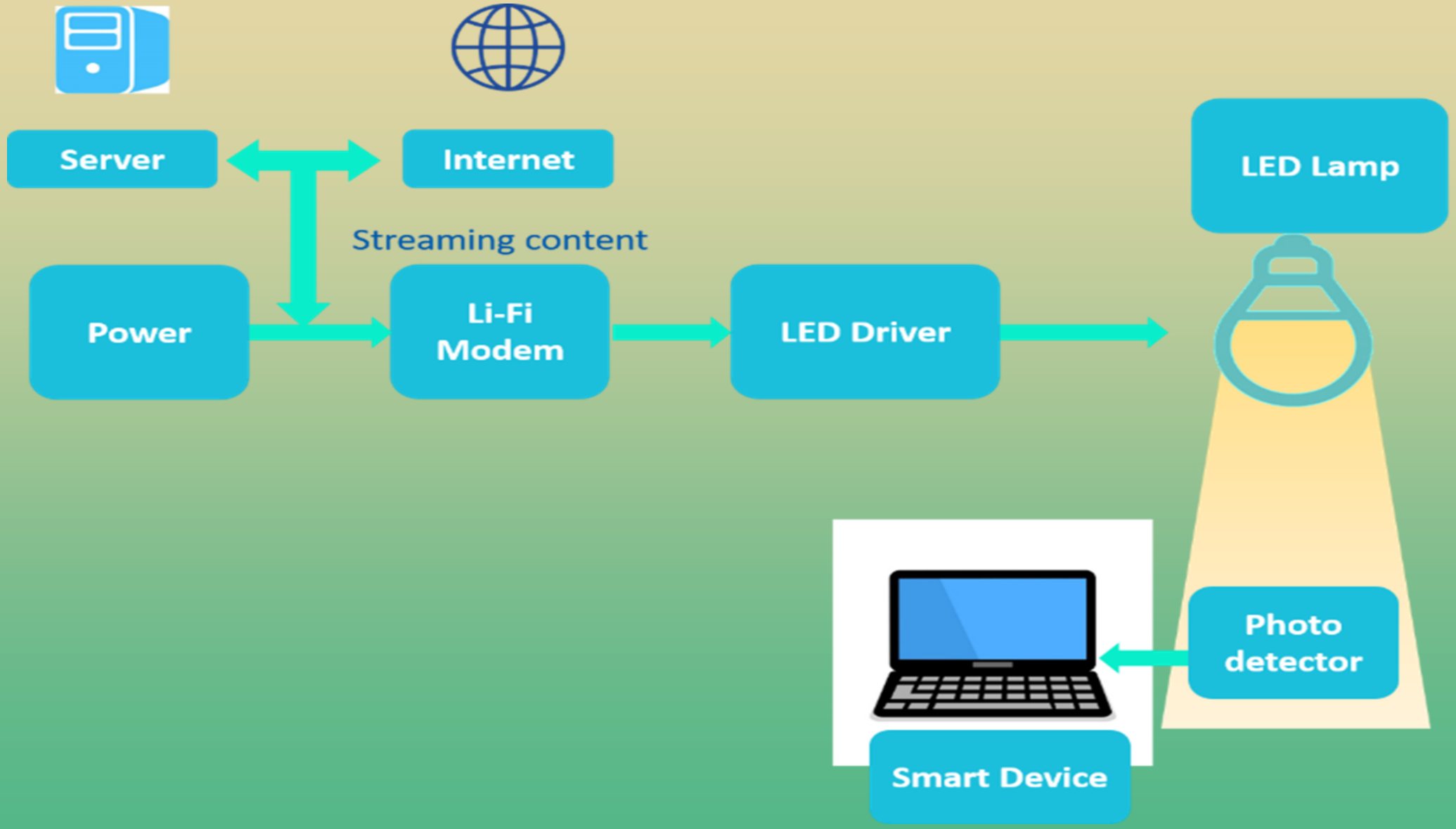
Transport Protocols: BLE

- **Broadcaster:**
  - Sends a non-connectable advertising packets in frequent intervals to all those who are willing to collect which is similar to radio broadcast.
- **Observer:**
  - The observer will keep scanning predefined frequency to receive any non-connectable packets.

Transport Protocols: Li-Fi

- Light Fidelity
- Fastest communication protocol in current time.
- It solves some problems of Wi-Fi Communications like,
  - More secured than Wi-Fi
  - More available bandwidth
  - Less/No Congestion with more number of nodes connecting to a single central device.

Transport Protocols: Li-Fi



Transport Protocols: Li-Fi

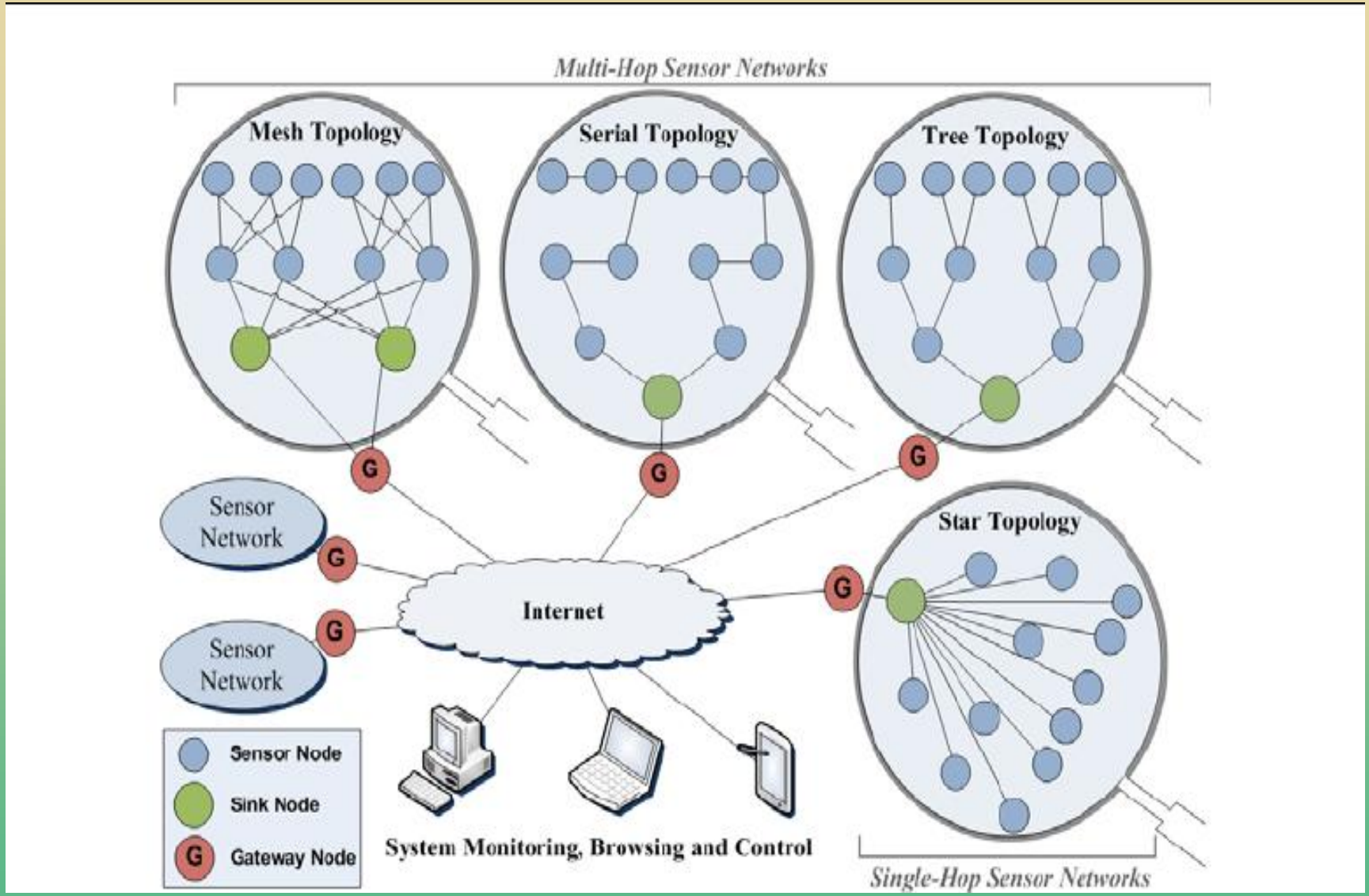
- The Server wants to transmit the data to the receiver with very high speed. Server needs to be connected to Internet or Intranet.
- The transmitted data generated will be converted into streaming content.
- Then that content is given to a Lamp Driver, powered by an external source.
- LED lamp is turned ON and OFF at a very high speed, which is at very high speed that human eyes can’t capture it.
- The light is received by the photo detector and is passed on to the demodulation process and amplification to generate the data stream sent by transmitter.
- The Received data stream is further given to the receiver (ex. PC, Mobile etc.) via some Application.
- All components used for receiving the data is known as receiver dongle.

Transport Protocols: Li-Fi

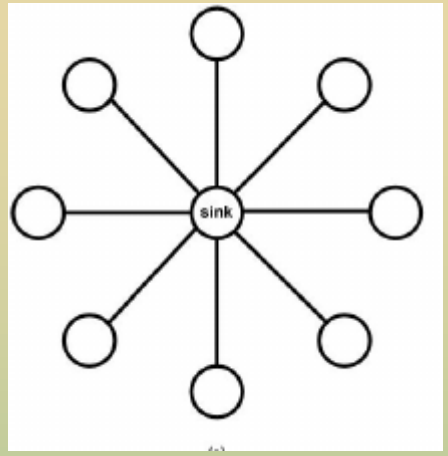
- **Advantages:**
  - Extremely secure as Light cannot penetrate through walls and no data hijacking
  - Fastest and Effective
  - Effective alternate to RF communication
  - Inexpensive, because light is generated through LEDs and detectors are also inexpensive.
- **Disadvantages:**
  - Can be implemented in short range only and no obstacle can be present between Lamp and Receiver dongle.
  - Set-up requires some time.
  - Bi-directional communication cannot be done, like sending data back from PC to the Server.

IoT Sensor Network Topologies

- Sensors may be connected as a **Single-hop Network**, where each sensor Node sends the data directly to the Sink Node.
- Or Multi-hop Network, where each Sensor Node relies on its neighbours to forward its sensory data to the Sink Node.

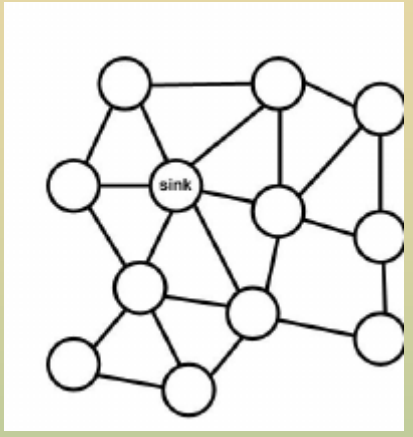


IoT Sensor Network Topologies



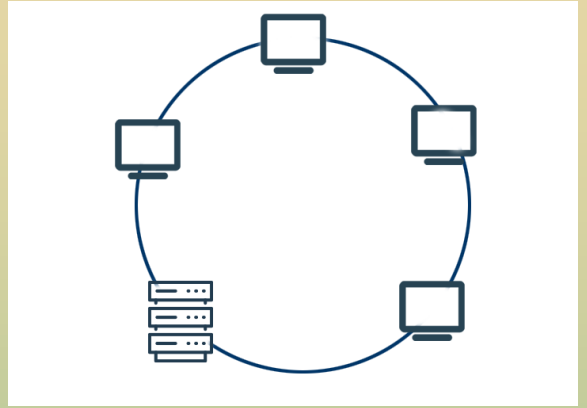
- **Point-to-Point (Star) Topology:**
  - **Centralised Control:**
    - Central hub or base station manages and controls communication.
    - Simplifies network administration and monitoring.
  - **Easy Expansion:**
    - Easy to add or remove nodes without disrupting the entire network.
    - Scalable for small to medium-sized networks.
  - **Limited Scalability:**
    - Becomes less efficient and scalable for large networks due to increased traffic at the central hub.
  - **Single Point of Failure:**
    - The central hub represents a potential single point of failure.
    - If the hub malfunctions, the entire network may be affected.

IoT Sensor Network Topologies



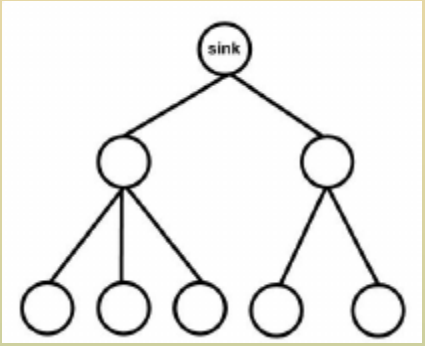
- **Mesh Topology:**
  - **Connectivity:**
    - Can dynamically reroute traffic in the event of node or link failures.
    - Offers self-healing capabilities to maintain connectivity.
  - **Redundancy:**
    - Redundant paths between nodes enhance reliability.
    - Multiple communication routes reduce the impact of node or link failures.
  - **Scalability Challenges:**
    - May become complex and difficult to manage as the number of nodes increases.
    - Routing algorithms must handle a potentially large number of connections.
  - **Increased Hardware Cost:**
    - Requires more hardware for interconnecting nodes.
    - Can lead to higher implementation costs.

IoT Sensor Network Topologies



- **Ring Topology:**
  - **Simplicity:**
    - Simple and straightforward to implement.
    - Data travels in a predictable direction.
  - **Predictive Latency:**
    - Predictable latency for data transmission in a circular fashion.
    - Suitable for applications with consistent timing requirements.
  - **Single Failure Impact:**
    - A break in the ring disrupts communication for the entire network.
    - The failure affects all nodes between the break and the endpoints.
  - **Limited Scalability:**
    - Scaling the network may become challenging, especially with an increasing number of nodes.

IoT Sensor Network Topologies



- **Tree Topology:**
  - **Hierarchical Structure:**
    - Organized in a hierarchical tree-like structure with a root node.
    - Efficient for applications requiring centralized control.
  - **Efficient Data Routing:**
    - Data travels along predetermined paths from leaf nodes to the root.
    - Effective for applications where data needs to be aggregated at a central point.
  - **Energy Efficiency:**
    - Can be energy-efficient as nodes closer to the leaves may operate on lower power.
    - Suitable for battery-powered devices.
  - **Scalability Challenges:**
    - Scalability is limited, especially with a deep hierarchy.
    - May face challenges with network expansion.