Messaging Protocols

Unit 2

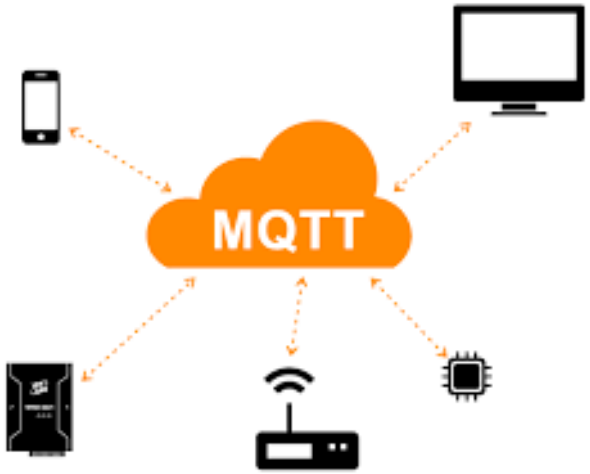
Subject : Intelligent System and Security

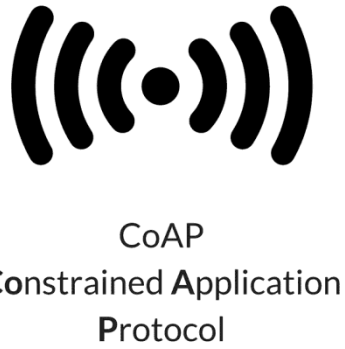
Course : M.Tech. AIDS – SEM2

Messaging Protocols in IoT

- 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.

1. Message Queuing Telemetry Transport (MQTT)
2. Constrained Application Protocol (CoAP)
3. Extensible Messaging 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.
- MQTT is widely used in IoT applications as it is a lightweight protocol.
- Here, lightweight means, it can work with minimal resources and does not require any specific hardware architecture or additional resources.

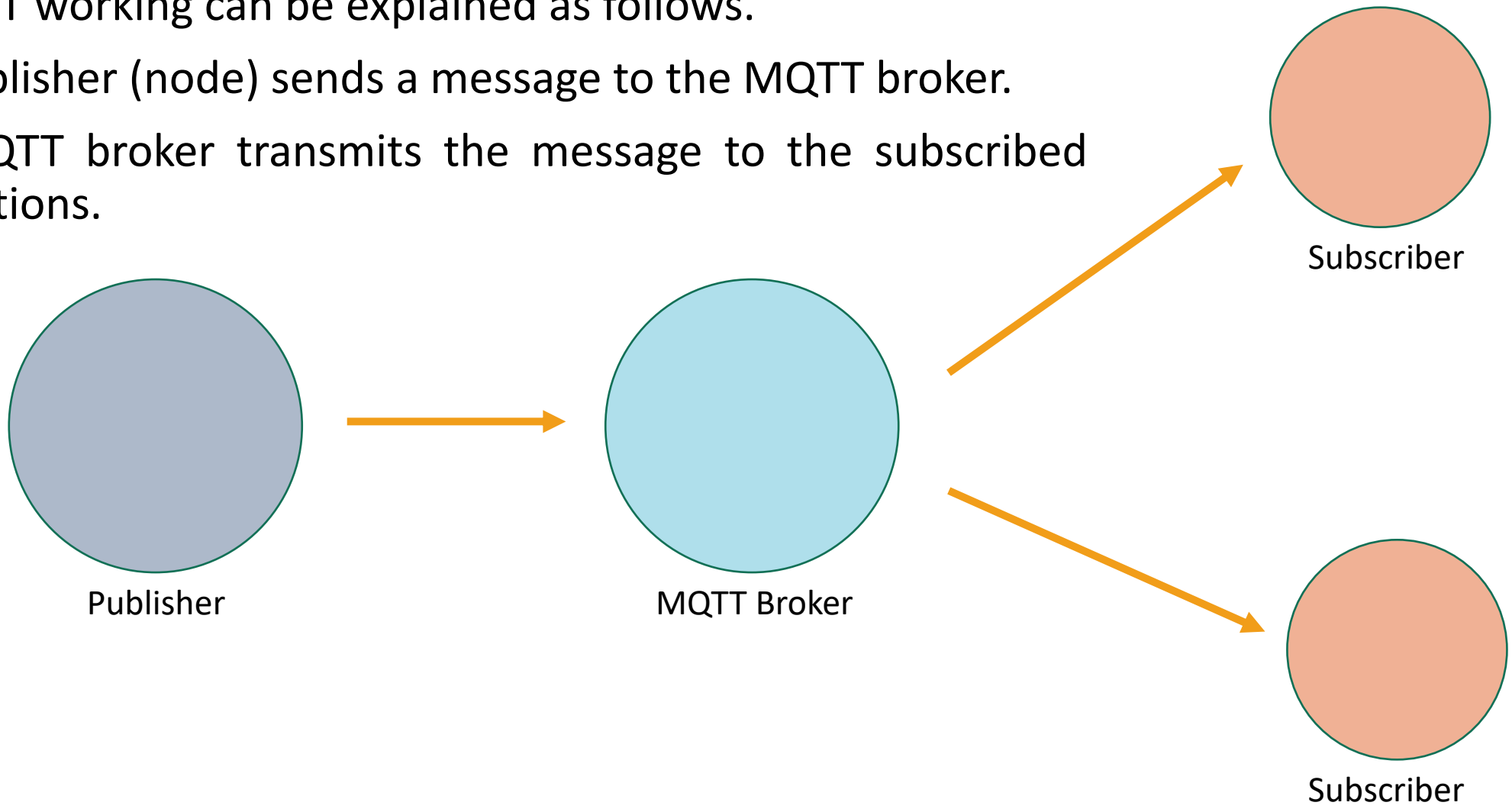
Communication model

1. Request-Response communication model
2. Publish-Subscribe communication model
3. Push-Pull communication model
4. Exclusive Pair communication model

MQTT

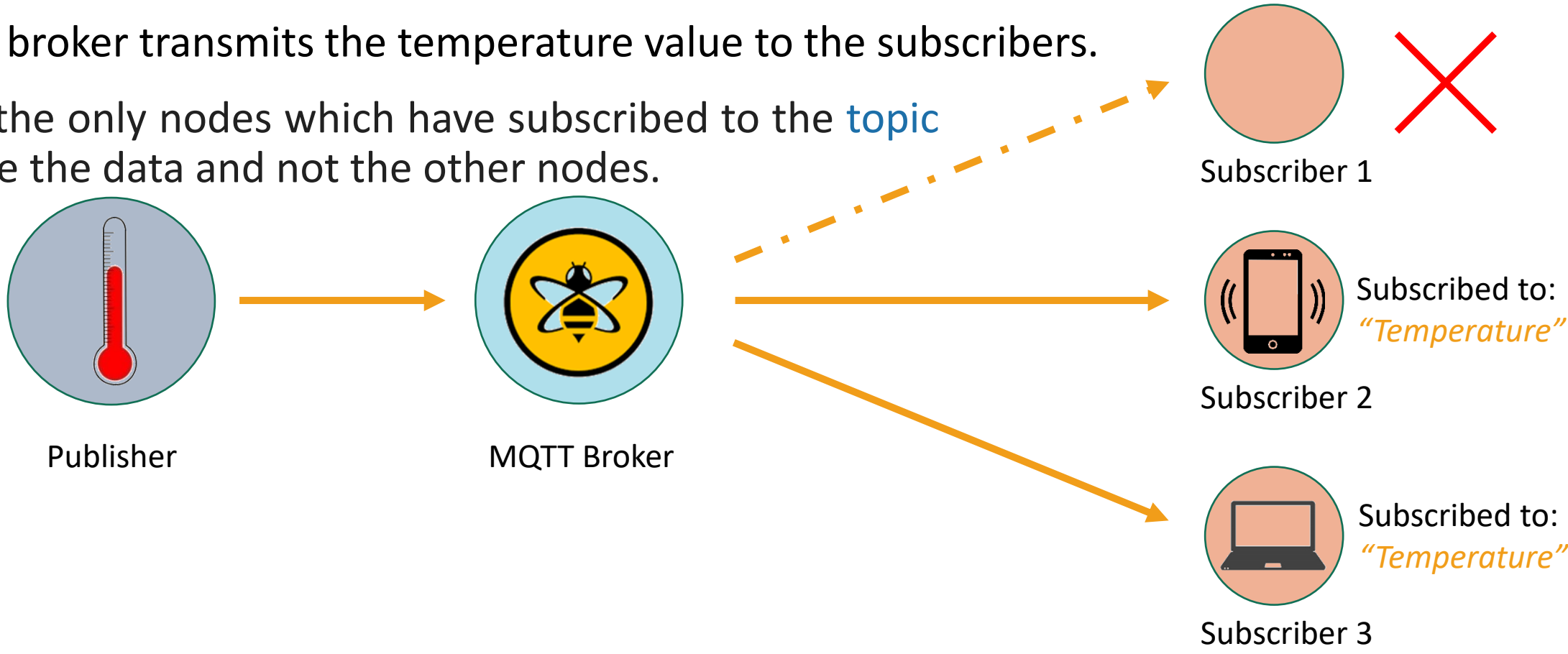
- MQTT works with “publish – subscribe” pattern.
- The MQTT working can be explained as follows.

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 – CurrentLocation.
- The server/computer subscribes to the topic CurrentLocation and receives the current location of the car.

Publisher: CurrentLocation

MQTT Broker

Subscriber: CurrentLocation

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

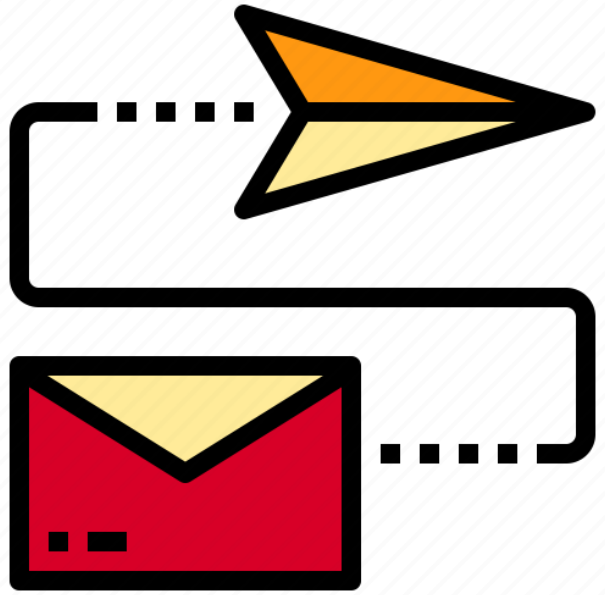
MQTT Broker

Subscriber: CurrentLocation

Publisher: Direction

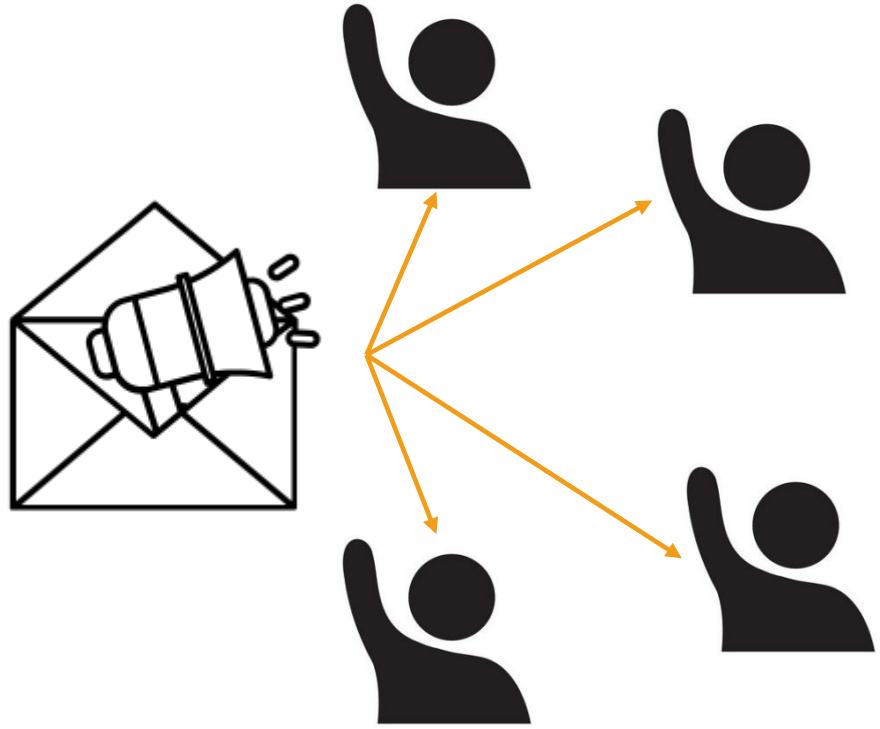
MQTT - Working

- In MQTT, clients do not have any address like a typical networking scheme.
- The broker filters the messages based on the subscribed topics.
- The messages will then be circulated to respective subscribers.
- The topic name can be anything in string format.
- The MQTT working can be explained with an example of the television broadcast.



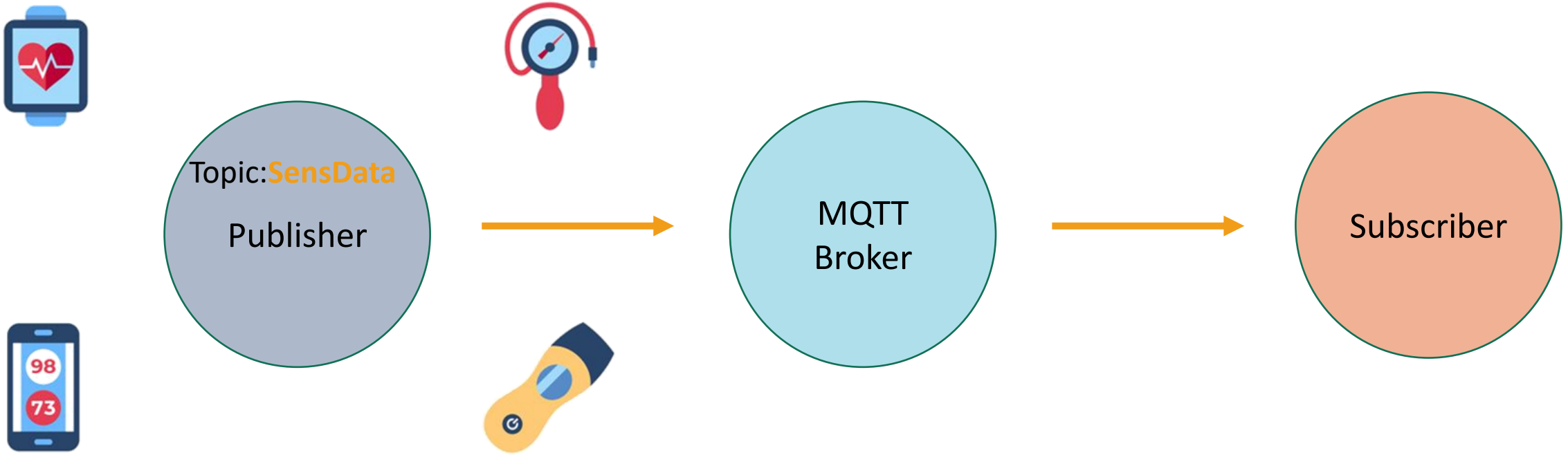
MQTT - Working

- 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.



QoS in MQTT

- MQTT supports different QoS (Quality of Service) levels.
- There are 3 layers of QoS supported by MQTT.
  - At most once (QoS 0): No Guarantee of Message Transmission
  - At least once (QoS 1): One Message is Guaranteed.
  - Exactly once (QoS 2): Handshake Mechanism Between the Sender and Receiver.

QoS 0 - At most once: Messages are delivered according to the best efforts of the operating environment. There is no guarantee of delivery. A message may be lost, and this is the fastest QoS level.

QoS 1 - At least once: The message is assured to arrive, but it may arrive more than once. The sender stores the message until it gets an acknowledgment from the receiver.

QoS 2 - Exactly once: This is the highest level of service. The message is assured to arrive exactly once. It involves a four-step handshake process between sender and receiver.

MQTT Topic

- The subscription to *house/+/temperature* would result in all messages sent to the previously mentioned topic *house/livingroom/ temperature*, as well as any topic with an arbitrary value in the place of living room, such as *house/kitchen/temperature*.
- The plus sign is a **single level wild card** and only allows arbitrary values for one hierarchy.
- If more than one level needs to be subscribed, such as, the entire sub-tree, there is also a **multilevel wildcard** (#).
- It allows to subscribe to all underlying hierarchy levels.
- For example *house/#* is subscribing to all topics beginning with *house*.

SMQTT

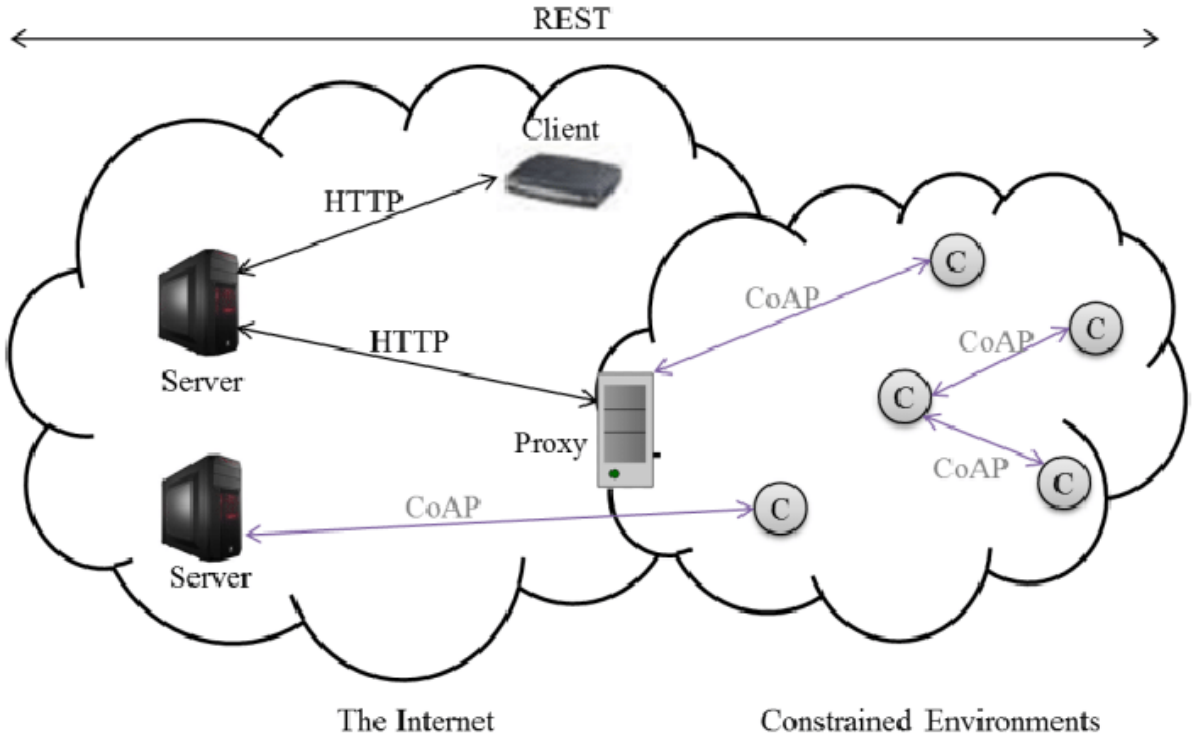
- **Secure MQTT** is an extension of MQTT which uses encryption based on lightweight attribute.
- The main advantage of using such encryption is the broadcast encryption feature, in which one message is encrypted and delivered to multiple other nodes, which is quite common in IoT applications.
- In general, the algorithm consists of four main stages: setup, encryption, publish and decryption.

SMQTT

- In the setup phase, the subscribers and publishers register themselves to the broker and get a master secret key according to their developer's choice of key generation algorithm.
- When the data is published, it is encrypted and published by the broker which sends it to the subscribers, which is finally decrypted at the subscriber end having the same master secret key.
- The key generation and encryption algorithms are not standardized.
- SMQTT is proposed only to enhance MQTT security features.

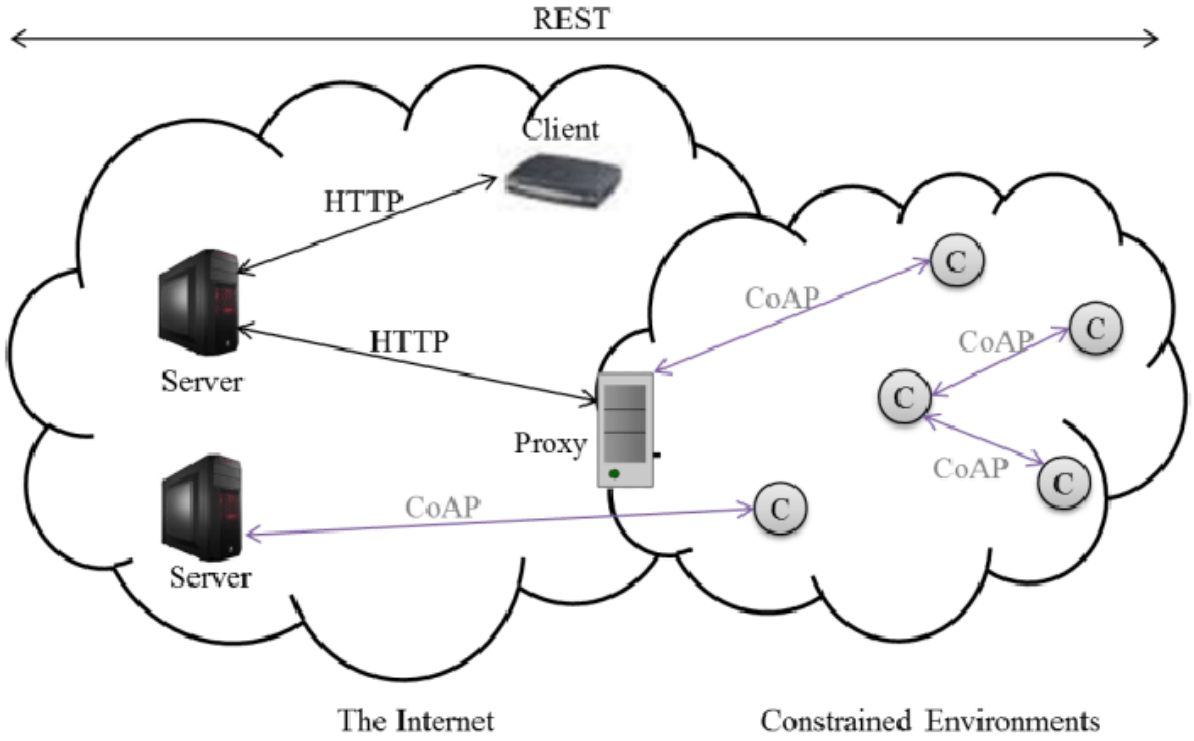
Constrained Application Protocol (CoAP)

- 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.



CoAP Architecture

- 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.

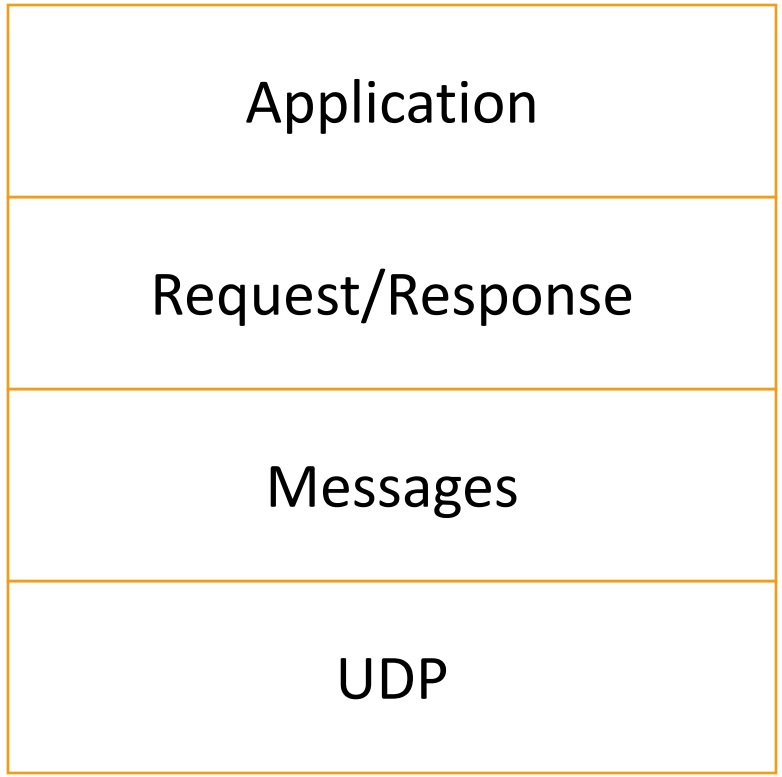


CoAP Layers

- CoAP has four layers:

1. UDP
2. Messages
3. Request-Response
4. 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

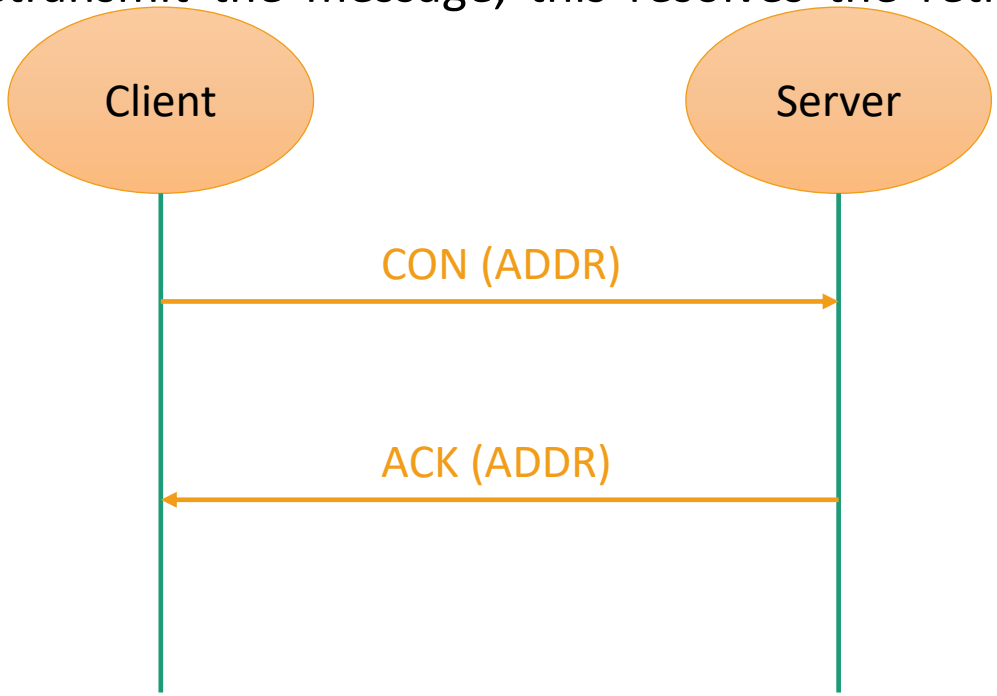
- CoAP message layer supports four types of messages:

1. Confirmable – Reliable Messaging (CON)
2. Non-confirmable – Non-reliable Messaging (NON)
3. Acknowledgement (ACK)
4. Reset (RST)

Message Layer in CoAP

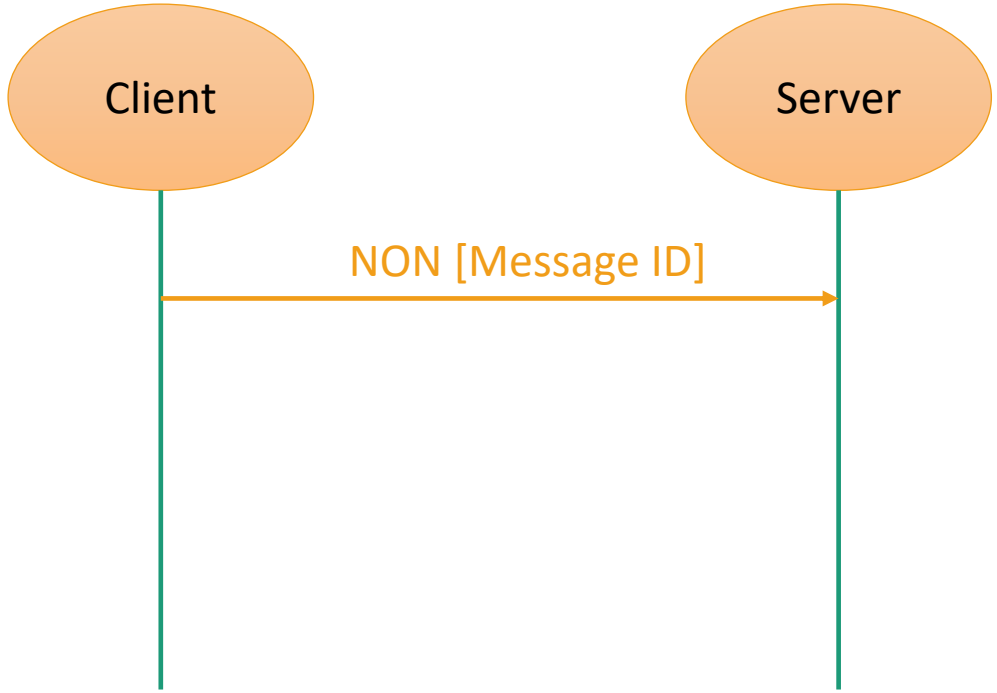
- CoAP message layer supports four types of messages:

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.



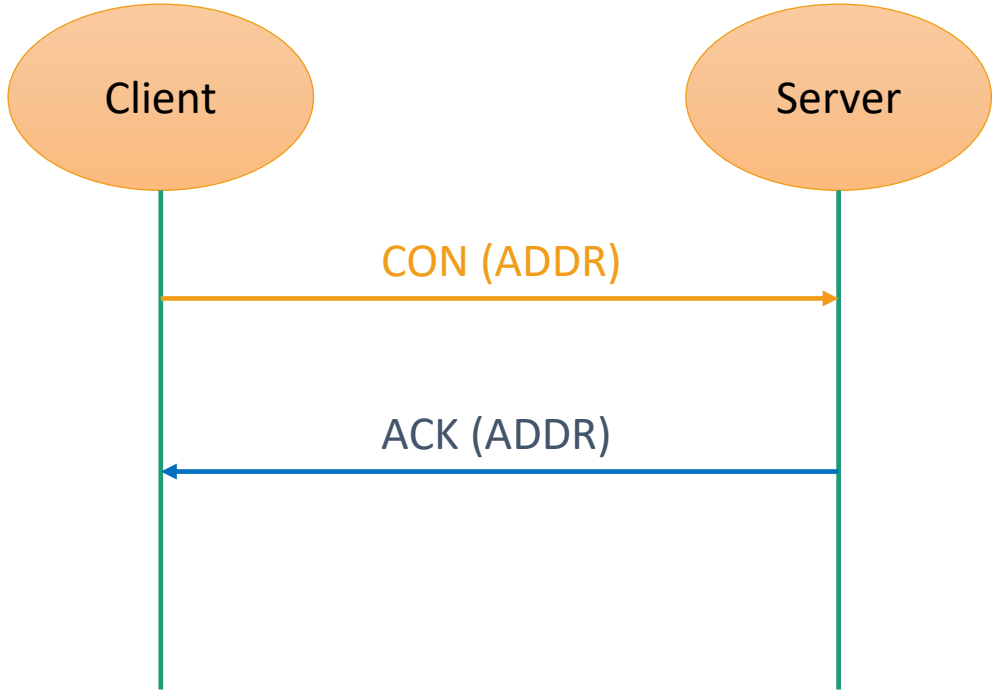
Message Layer in CoAP (Contd.)

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.



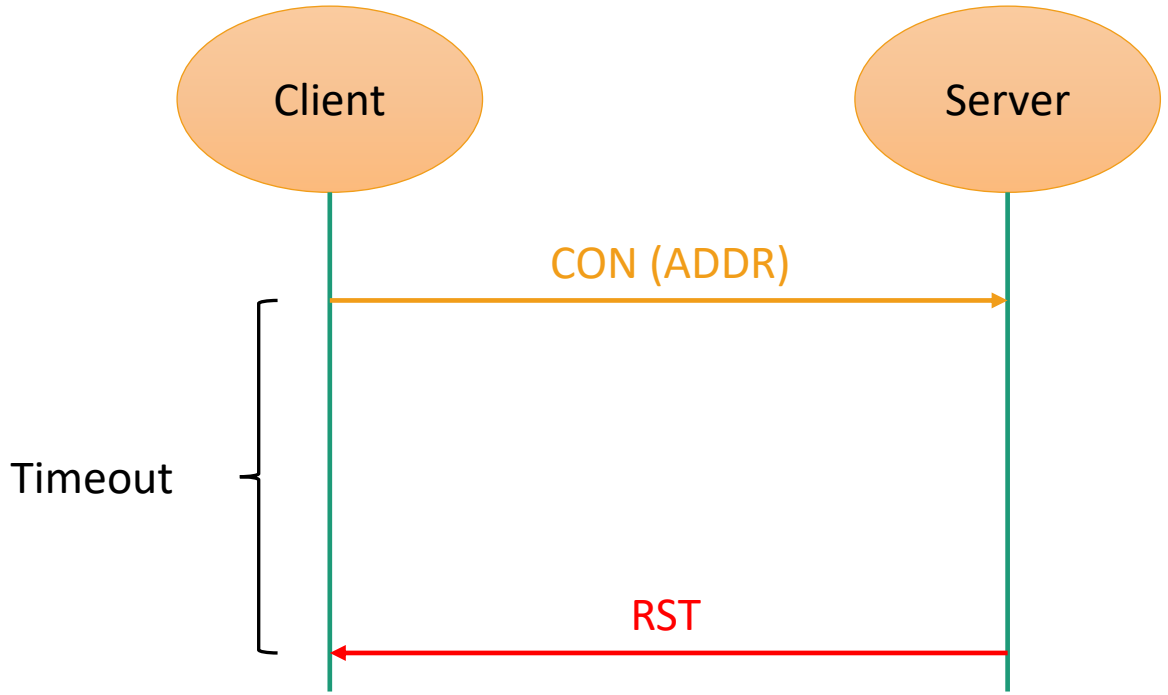
Message Layer in CoAP (Contd.)

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.)

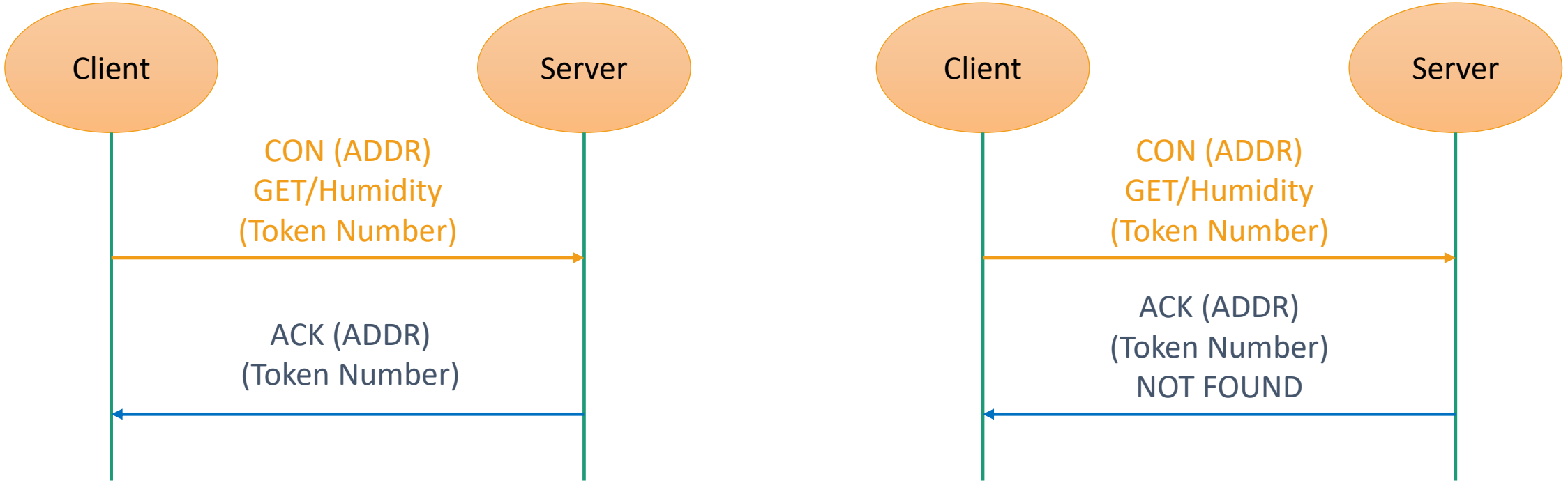
4. Reset (RST):
   - The receiver is expecting the message from sender of a particular message ID.
   - If no message is processed or received before specific amount of time (called timeout), the message called RESET is transmitted from receiver.
   - This message informs the sender that there is a trouble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



Request-Response Layer in CoAP

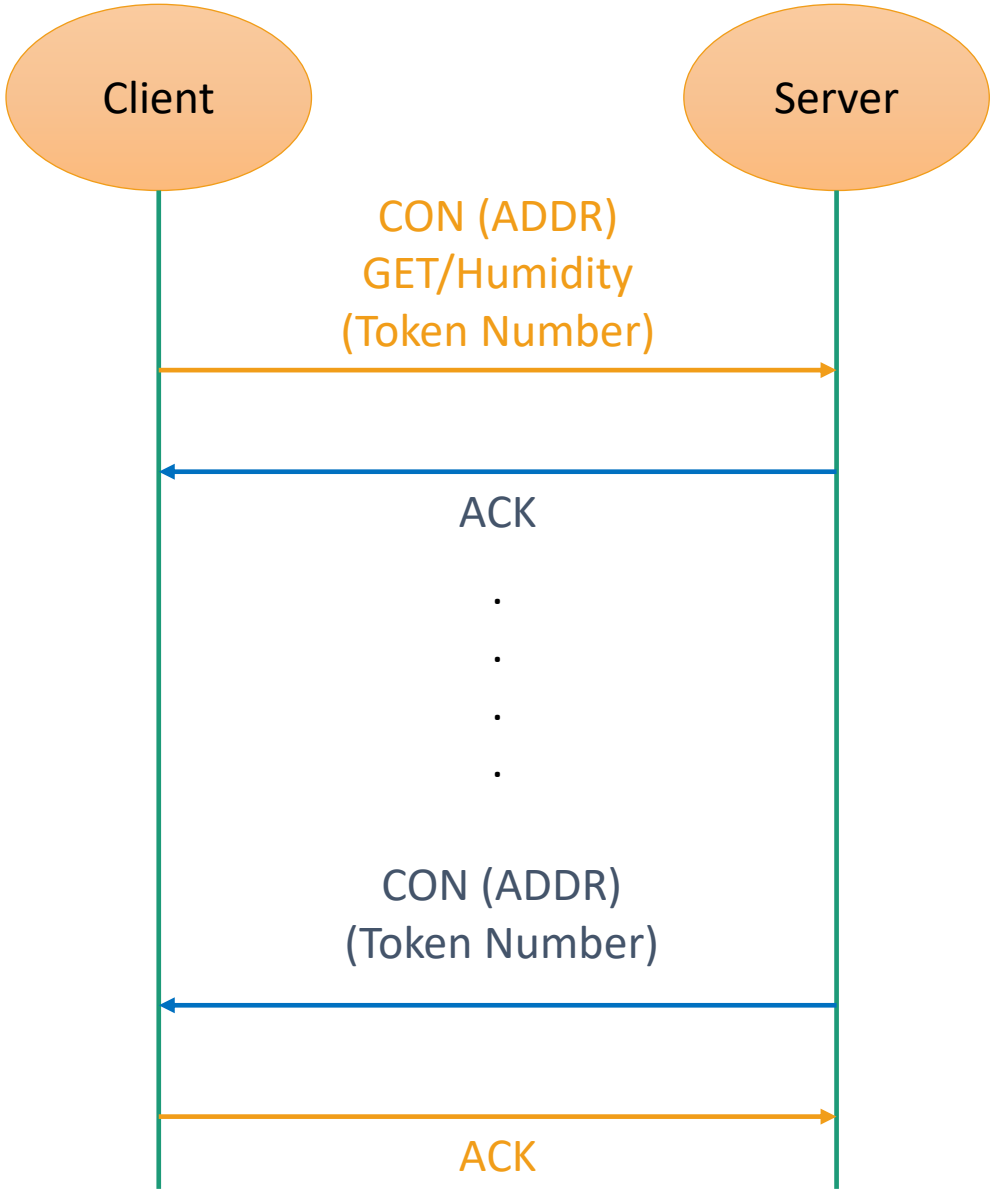
- There are three modes in CoAP request-response layer.

1. Piggy-Backed:
   - In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and CON/NON 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.



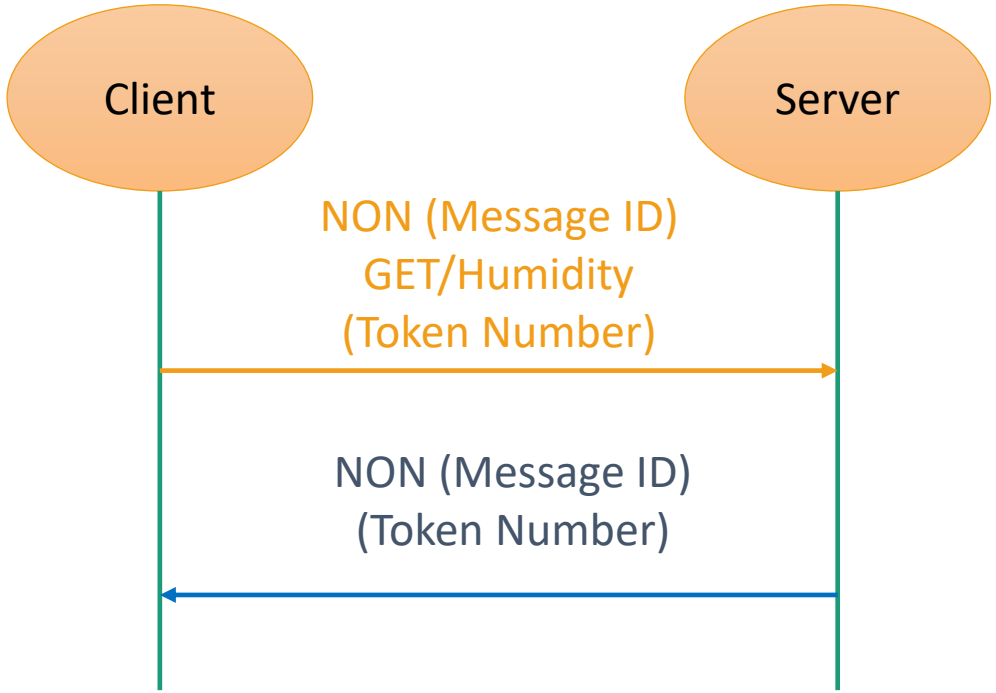
Request-Response Layer in CoAP (Contd.)

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.



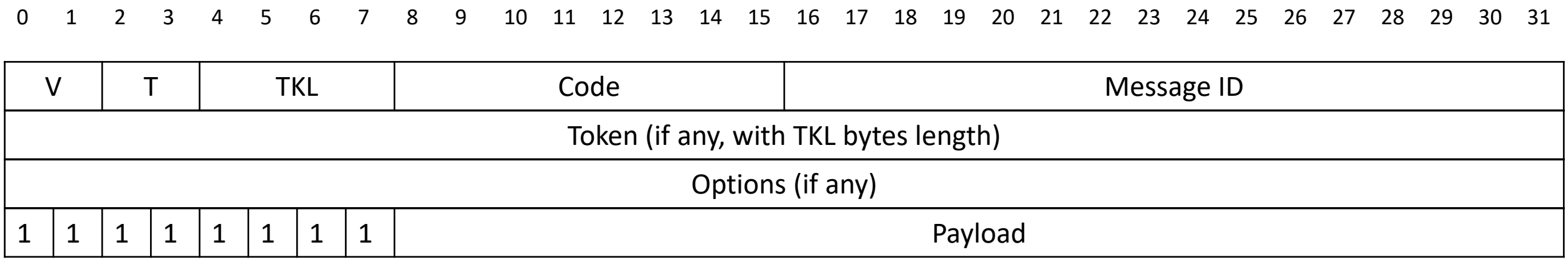
Request-Response Layer in CoAP (Contd.)

3. Non-Confirmable Request and 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.



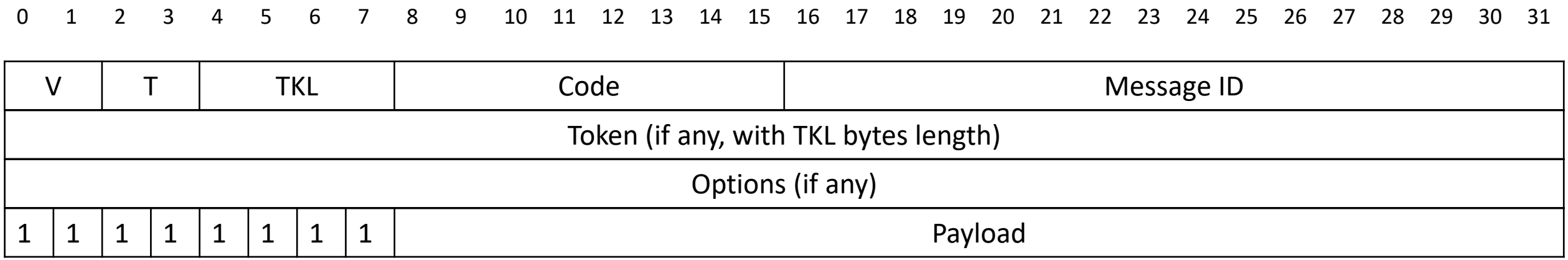
CoAP Message Format

- The CoAP message format is of 16 Bytes consisting various fields.
- V : It refers to version of CoAP. It is two bit unsigned integer.
- T : It refers to message type. It is also two bit unsigned integer.
  - Confirmable (0)
  - Non-Confirmable (1)
  - ACK (2)
  - RESET (3)
- TKL : It refers to the token length and is four bit unsigned integer.
- Code: It refers to the response code.



CoAP Message Format (Contd.)

- Message ID: It is identifier of the message and to avoid duplication.
- Token : Optional field whose size is indicated by the Token Length field.
- Options : This field is used if any options available and having length of 4 Bytes.
- Payload : This field refers to the payload data where actual data is transmitted.



Difference between MQTT & CoAP

	MQTT	CoAP
Underlying protocol	TCP (connection oriented)	UDP (connectionless)
Communication	**M:N (Many to many)**	**1:1 (One-to-one)**
Power	Higher than CoAP. Lesser than other protocols.	Lowest, consumes less power than MQTT, making it the best for point to point communication.
Model	**Publisher/Subscriber**	**Request-response (RESTful)**

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.

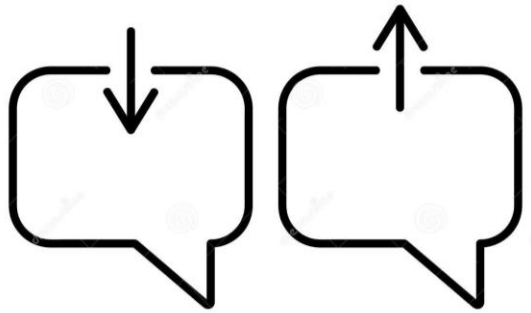
XMPP denotation

- 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.

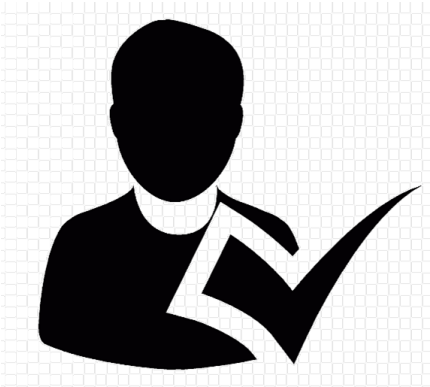
Messenger Requirements

- Any messenger requires the following features inbuilt :

1. Message sending and receiving features.
2. Understanding the presence/absence status.
3. Managing subscription details.
4. Maintaining the contact list.
5. Message blocking services.



- Note that all the listed features are built in XMPP.



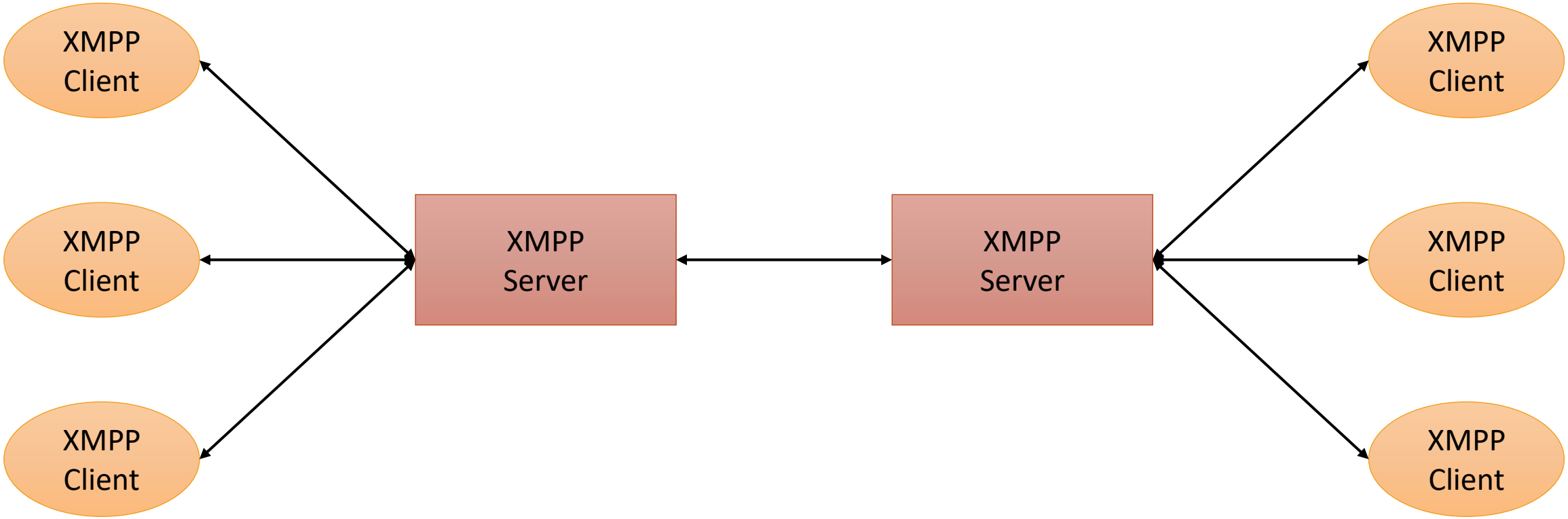
XMPP Features:

- The following are features of XMPP:

1. **Decentralized:** XMPP is similar to that of e-mail. Anyone can run their own XMPP server.
2. **Secure:** XMPP is very secure with end-to-end encryption.
3. **Stable & functionally proven:** It is functionally proven since its initial version Jabber launched in 1998.
4. **Standardization:** IETF published XMPP specifications as RFCs in RFC3920 & 3921 standards.
5. **Flexibility:** XMPP is not just messaging oriented; it is flexible enough for file sharing, gaming & even remote monitoring of systems.

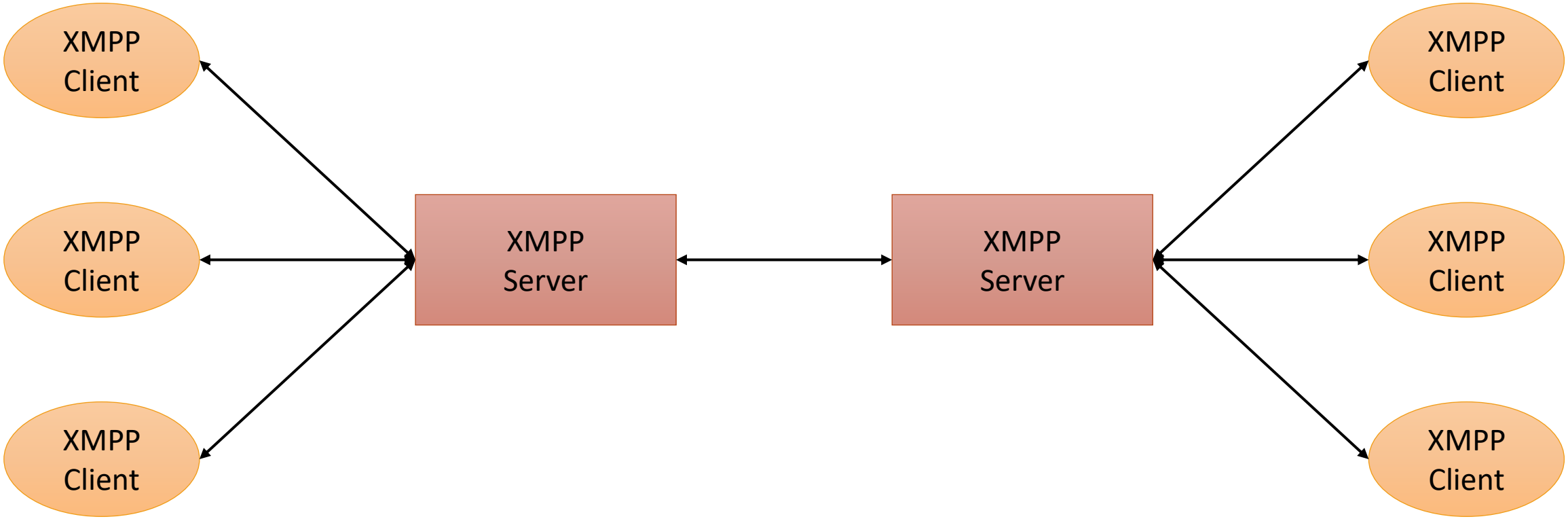
XMPP Architecture

- 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 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.