AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Communication Technologies – MQTT, CoAP

22AIE211 Introduction To Communication & IoT

MQTT

- Message Queue Telemetry Transport.
- ISO standard (ISO/IEC PRF 20922).
- It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
- Lightweight = Low network bandwidth and small code footprint
- MQTT was introduced by IBM in 1999 and standardized by OASIS in 2013.
- Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and networks and communications on the other side.
- Client libraries are available in almost all popular languages now.

  https://eclipse.org/paho/

- The current MQTT specification is available at:
  - http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
  - http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

Features of MQTT

- It supports publish/subscribe message pattern to provide one-to-many message distribution and decoupling of applications
- Three qualities of service for message delivery:
- "At most once" , where messages are delivered according to the best efforts of the operating environment. Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
- "At least once", where messages are assured to arrive but duplicates may occur.
- "Exactly once", where message are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.
- A small transport overhead and protocol exchanges minimized to reduce network traffic.
- A mechanism to notify interested parties when an abnormal disconnection occurs. (Last Will and Testament)

Last Will and Testament

- The protocol provides a method for detecting when clients close their connections improperly by using keep-alive packets. So when a client crashes or it's network goes down, the broker can take action.
- Clients can send a Last Will and Testament (LWT) message to the broker at any point. When the broker detects the client has gone offline (without closing their connection), it will send out the saved LWT message on a specified topic, thus letting other clients know that a node has gone offline unexpectedly.

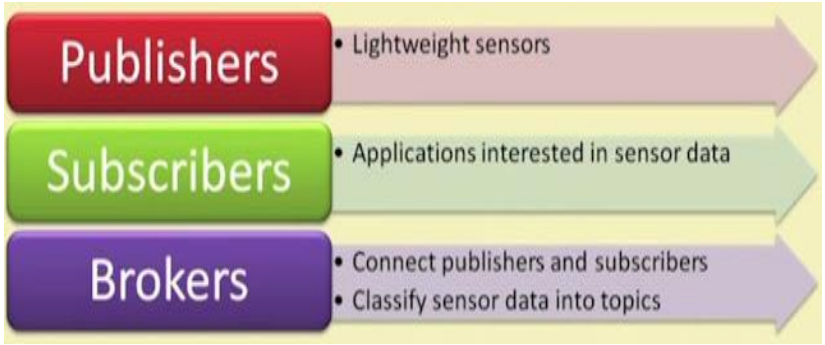
MQTT Pub/Sub Protocol

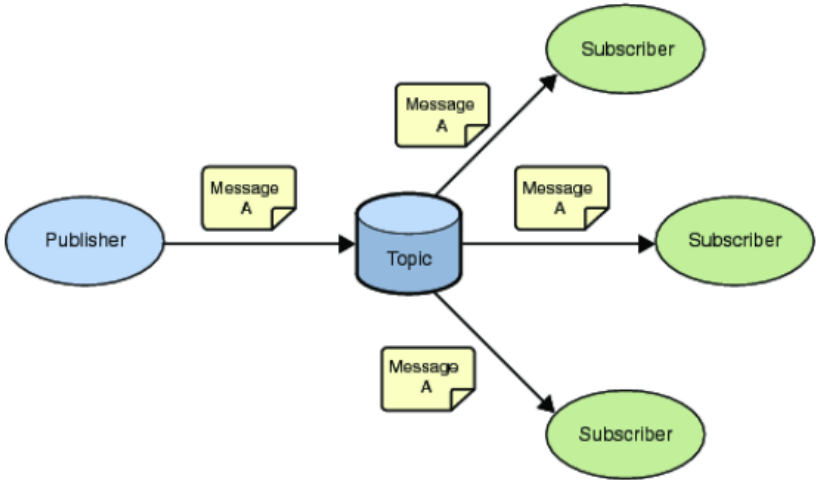
- MQ Telemetry Transport (MQTT) is a lightweight broker-based publish/subscribe messaging protocol.
- A message broker controls the publish-subscribe messaging pattern.
- A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker.
- Designed for:
- Remote connections
- Limited bandwidth
- Small-code footprint

Publish Subscribe Messaging

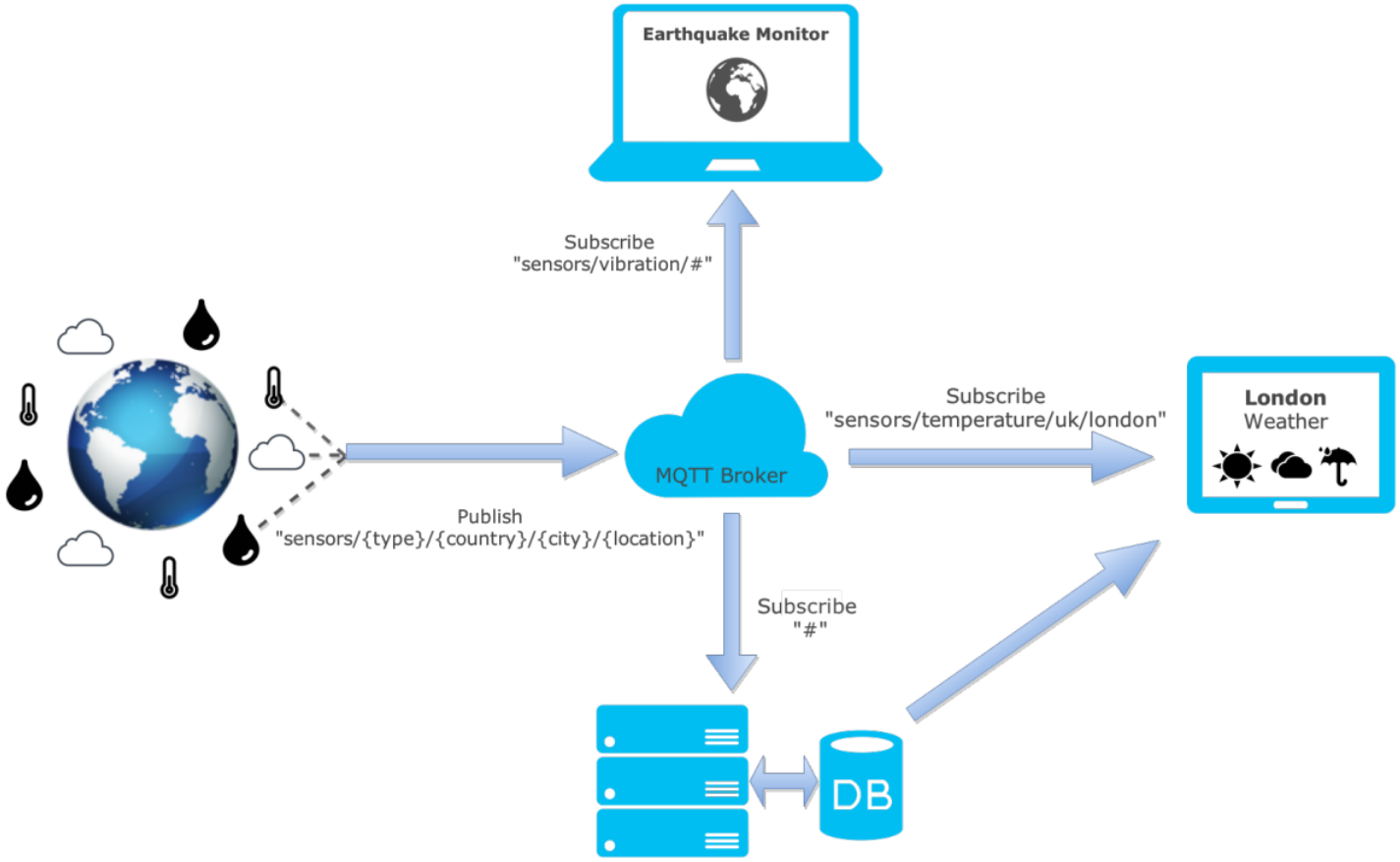
- A Publish Subscribe messaging protocol allows a message to be published once and multiple consumers (applications / devices) to receive the message providing decoupling between the producer and consumer(s)
- A producer sends (publishes) a message (publication) on a topic (subject)
- A consumer subscribes (makes a subscription) for messages on a topic (subject)
- A message server/broker matches publications to subscriptions
  - If no matches the message is discarded
  - If one or more matches the message is delivered to each matching subscriber/consumer

MQTT Components





Broker/Publish/Subscribe



Publish Subscribe Messaging (MQTT Topics)

- A topic is a simple string that can have more hierarchy levels, which are separated by a slash.
- A sample topic for sending temperature data of the living room could be *house/living-room/temperature*.
- A subscriber can subscribe to an absolute topic or can use wildcards:

➢ Single-level wildcards "+" can appear anywhere in the topic string

- The subscription to *house/+/temperature* would result in all messages sent to the previously mentioned topic house/living-room/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.

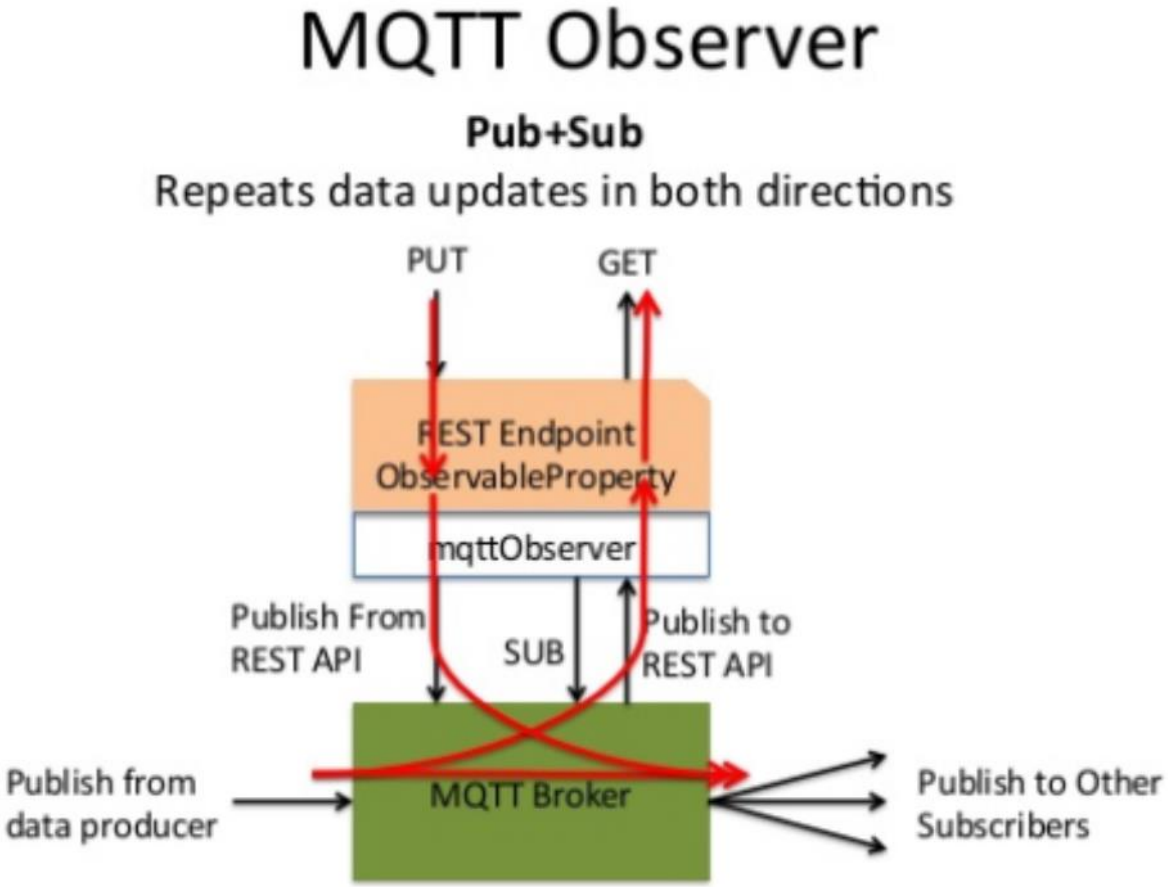
➢ If more than one level needs to be subscribed, such as, the entire sub-tree, there is also a multilevel wildcard (#).

- Multi-level wildcards "#" must appear at the end of the string
- It allows to subscribe to all underlying hierarchy levels.
- For example *house/#* is subscribing to all topics beginning with house.

Publish Subscribe Messaging

- A subscription can be durable or non durable
  - Durable:
    - Once a subscription is in place, a broker will forward matching messages to the subscriber:
      - ¨ Immediately the subscriber may get connected
      - ¨ If the subscriber is not connected, messages are stored on the server/broker until the next time the subscriber connects
  - Non-durable: The subscription lifetime is the same as the time the subscriber is connected to the server / broker
- A publication may be retained
  - A publisher can mark a publication as retained
  - The broker / server remembers the last known good message of a retained topic
  - The broker / server gives the last known good message to new subscribers
    - i.e. the new subscriber does not have to wait for a publisher to publish a message in order to receive its first message

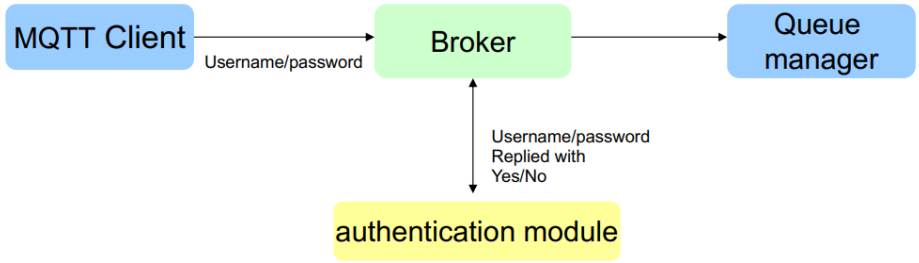
MQTT Observer



MQTT Methods

SMQTT

- Secure MQTT is an extension of MQTT which uses encryption based on lightweight attribute based encryption.
- 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
- 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



Broker Software

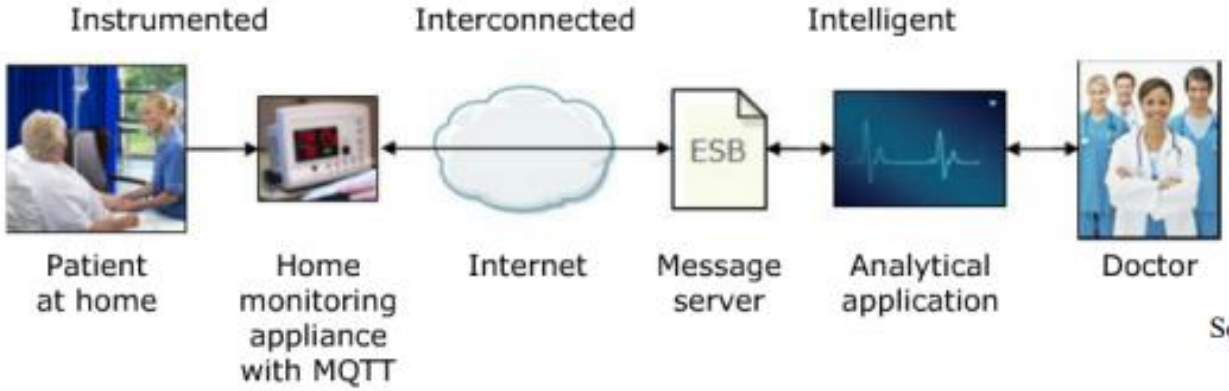
- Mosquitto - One of the earliest production ready brokers, Mosquitto is written in C and offers high performance with a lot of configurability.
- Mosca - Written in Node.js, this can be embedded in a Node application or run as a standalone executable. Easy configuration and extensibility, also very performant.
- RSMB - IBM’s implementation of the MQTT protocol. This is one of the less popular options but is a mature system, written in C.
- HiveMQ - HiveMQ is a relatively new player, and is oriented towards enterprise environments.

Projects that Implement MQTT

- Amazon Web Services announced Amazon IoT based on MQTT in 2015.
- Microsoft Azure IoT Hub uses MQTT as its main protocol for telemetry messages.
- Node-RED supports MQTT nodes as of version 0.14.
- Facebook has used aspects of MQTT in Facebook Messenger for online chat. Facebook messenger uses MQTT to minimize battery usage.
- The EVRYTHNG IoT platform uses MQTT as an M2M protocol for millions of connected products.

MQTT Application Examples

- Home pacemaker monitoring solution
  - Sensors on patient
  - Collected by a monitoring equipment in home (broker) using MQTT
  - Subscribed by a computer in the hospital
  - Alerts the doctor if anything is out-of-order



Source: Lampkin 2012

CoAP(Constrained Application Protocol)

- Web transfer protocol for use with constrained nodes and networks.
- The Constrained Application Protocol (CoAP) is defined by IETF CoRE WG for the manipulation of resources on a device that is on the constrained IP 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.
- 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.
- CoAP is designed to enable low-power sensors to use RESTful services while meeting their power constraints

Constrained IP Networks

- A constrained IP network has limited packet sizes, may exhibit a high degree of packet loss, and may have a substantial number of devices that may be powered off at any point in time but periodically "wake up" for brief periods of time.
- These networks and the nodes within them are characterized by severe limits on throughput, available power, and particularly on the complexity that can be supported with limited code size and limited RAM per node.
- Low-Power Wireless Personal Area Networks (LoWPANs) are an example of this type of network. Constrained networks can occur as part of home and building automation, energy management, and the Internet of Things.

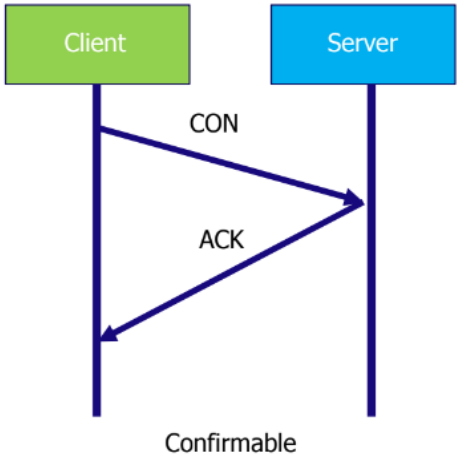
Devices on Constrained Networks

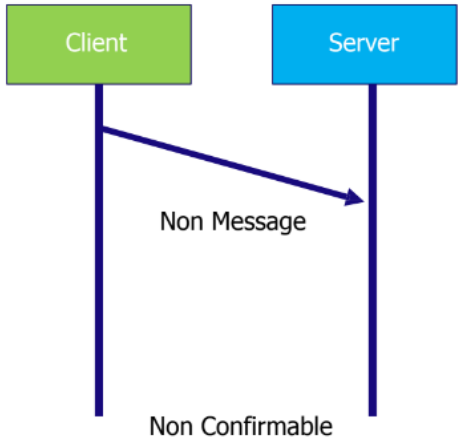
- The general architecture consists of nodes on the constrained network, called Devices, that are responsible for one or more Resources that may represent sensors, actuators, combinations of values or other information.
- Devices send messages to change and query resources on other Devices.
- Devices can send notifications about changed resource values to Devices that have subscribed to receive notification about changes.
- A Device can also publish or be queried about its resources

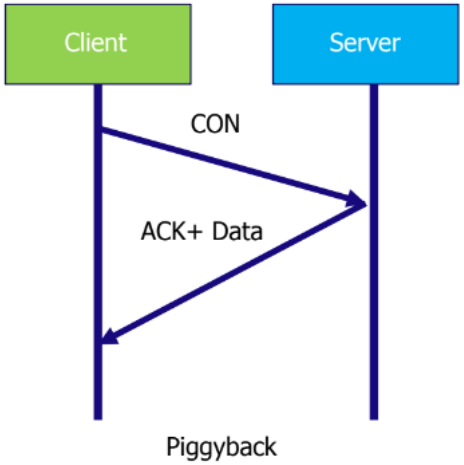
CoAP messages

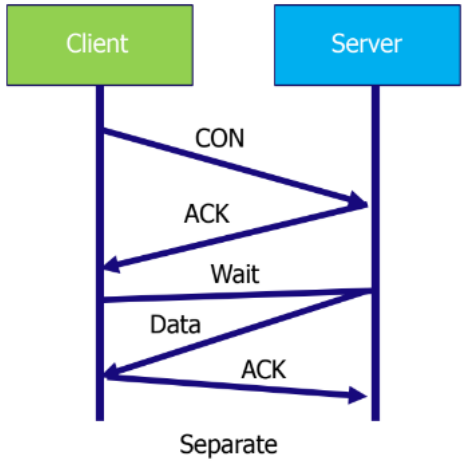
- CoAP architecture is divided into two main sub-layers:
  - Messaging
  - Request/response.
- The messaging sub-layer is responsible for reliability and duplication of messages, while the request/response sub-layer is responsible for communication.
- CoAP has four messaging modes:
  - Confirmable
  - Non-confirmable
  - Piggyback
  - Separate

CoAP messages





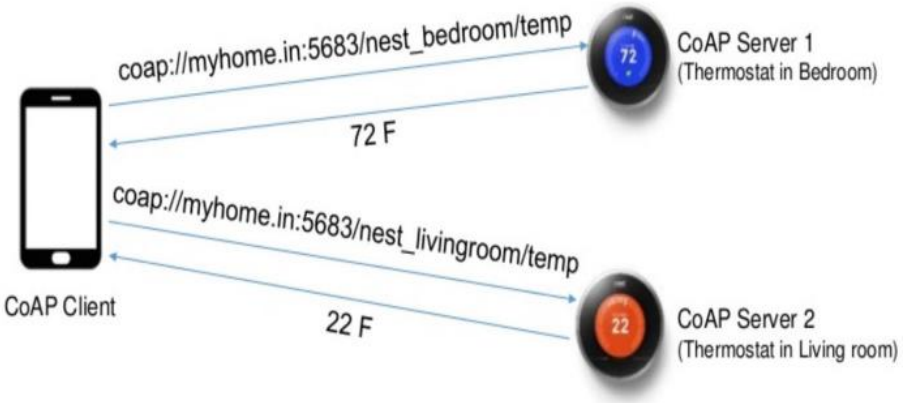


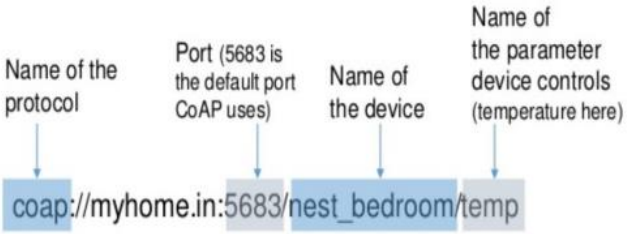


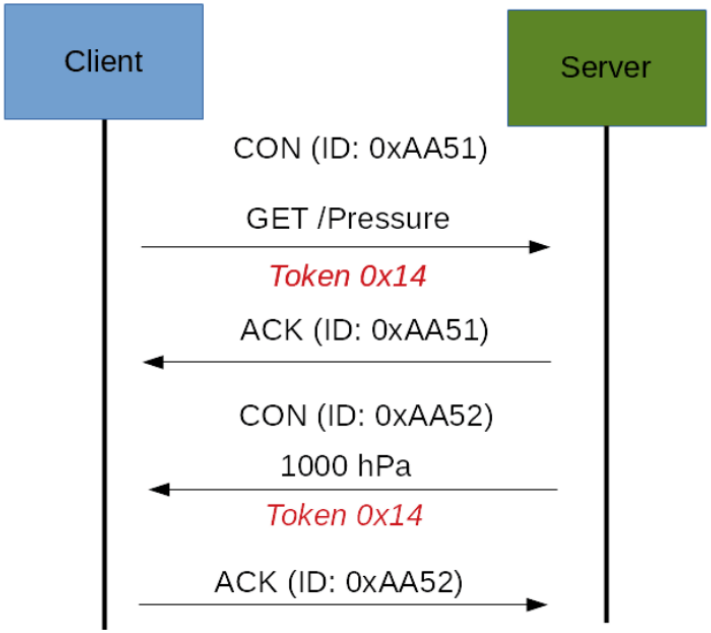
CoAP messages

- Confirmable and non-confirmable modes represent the reliable and unreliable transmissions, respectively, while the other modes are used for request/response.
- Piggyback is used for client/server direct communication where the server sends its response directly after receiving the message, i.e., within the acknowledgment message.
- On the other hand, the separate mode is used when the server response comes in a message separate from the acknowledgment, and may take some time to be sent by the server.

CoAP – Request Response







CoAP methods

- CoAP makes use of GET, PUT, POST, and DELETE methods in a similar manner to HTTP.
- New methods can be added, and do not necessarily have to use requests and responses in pairs.
  - For example: OBSERVE (embedded in GET method)

Application Scope of CoAP

- includes applications to monitor simple sensors (e.g. temperature sensors, light switches, and power meters), to control actuators (e.g. light switches, heating controllers, and door locks), and to manage devices

Summary

- MQTT
- CoAP