MQTT, CoAP, TCP

-brief overview

-Mahbub Ul Alam

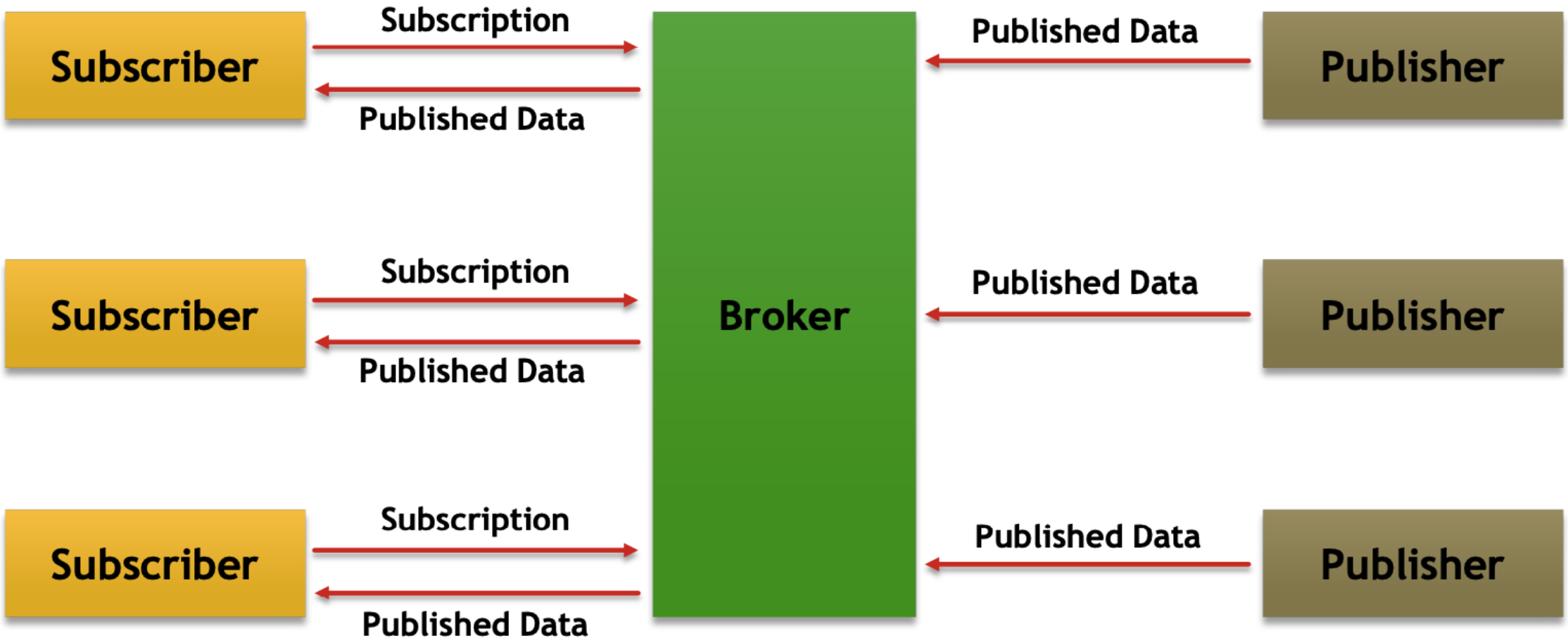
MQTT (Message Queuing Telemetry Transport)

- MQTT is a M2M (Machine-To-Machine) connectivity protocol.
- MQTT is based on **Publisher/Subscriber** model.
- It was created to connect to the systems situated in remote place and to get data from those sensor.
- Decoupling can be done because of **Publisher/Subscriber** model which is hard to achieve in **Client/Server** model.

MQTT Components

- MQTT has 3 basic components
  - Publisher (e.g. Motor, Mobile Devices, etc.)
  - Subscriber (e.g. Temperature sensor, Motion Sensor, etc.)
  - Broker

MQTT Diagram



MQTT Publisher

- System or sensors which collects data and send it to the broker which further sends it to multiple subscribers.
- Example motion sensor, water level sensor, etc.
- Publisher publish data in following formats
  - Binary
  - JSON
  - SDC Record
  - Text

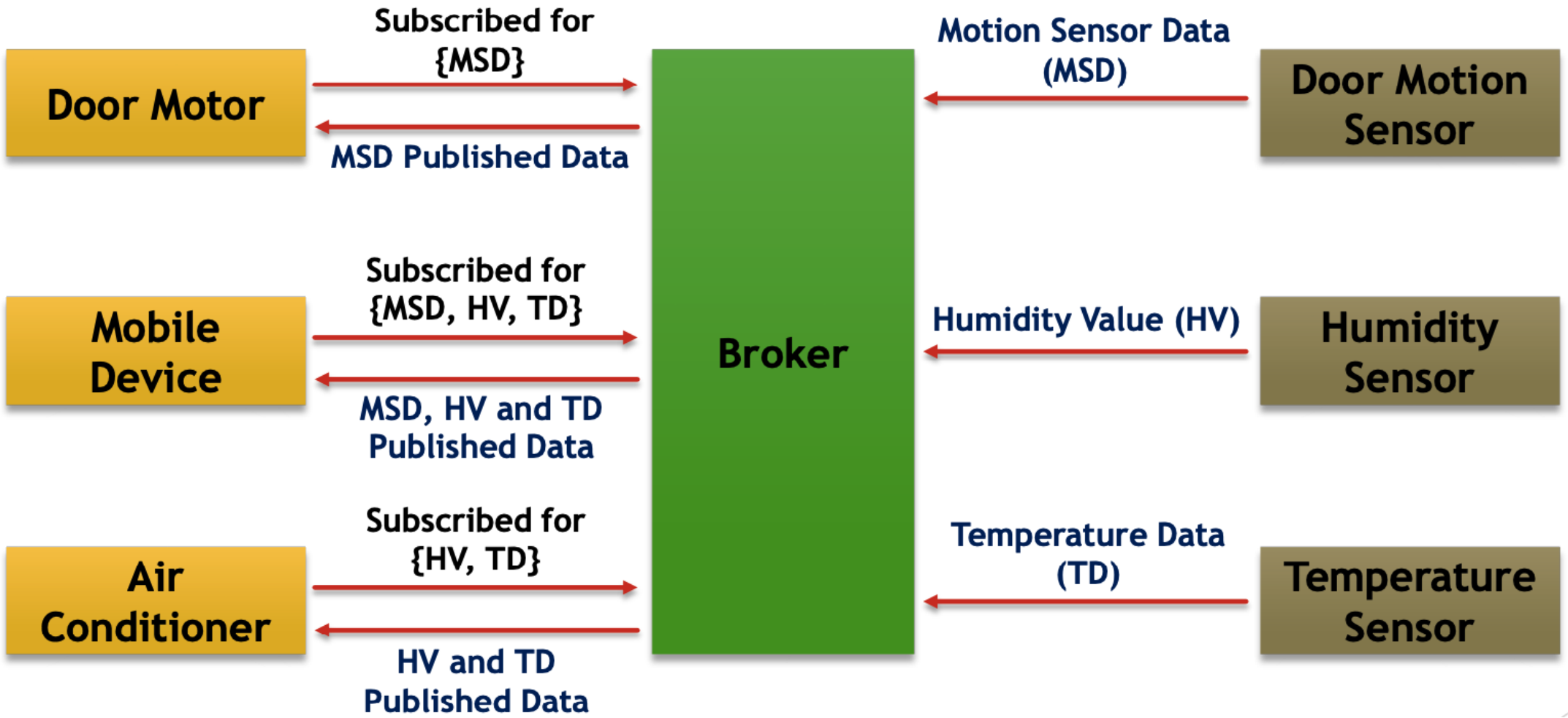
MQTT Subscriber

- Subscriber can be a mobile device, data server, monitoring stations, etc. which receives publish data from broker so that it can act according to it or monitor it or store it.
- Subscriber send request to broker to send required publisher’s data.
- Broker has the table in which it maintains all subscriptions request, and send
- publish data according to it.
- Example Mobile devices, Monitoring system, etc.

MQTT Broker

- Broker is the component which take care of receiving data from Publisher and sending it to Subscriber accordingly.
- Broker has to filter messages, broker can filter messages in following ways.
  - Subject based
  - Content based
  - Type based
- Broker has a subscription table in which it store all the request from the subscriber for the publisher’s data.
- Broker sends publish data to multiple subscriber according to this list.

Simple MQTT Example



Decoupling in Pub/Sub

- Space decoupling
  - In MQTT there no need for publisher and subscriber to know each other, since it is Pub/Sub model there no need to exchange IP address.
- Time decoupling
  - Since it is Pub/Sub model there is no need for publisher and subscriber to run on same time.
- Synchronization decoupling
  - Operations on both publisher and subscriber do not need to be interrupt during the publishing and receiving.

CoAP (Constrained Application Protocol)

- CoAP is developed BY **IETF** (Internet Engineering Task Force)
- As the name suggest CoAP has been developed for data transmission in **constrained network**.
- In network where devices are low power, and network is lossy, we need protocols like UDP for transmission, but UDP has its own disadvantages, here comes CoAP in scene which uses UDP but it is more reliable, secured, etc.
- CoAP has become more important for IoT and M2M because of its features.
- CoAP runs on UDP enable devices, it support **less memory devices** and also **low power devices**.

CoAP (Constrained Application Protocol)

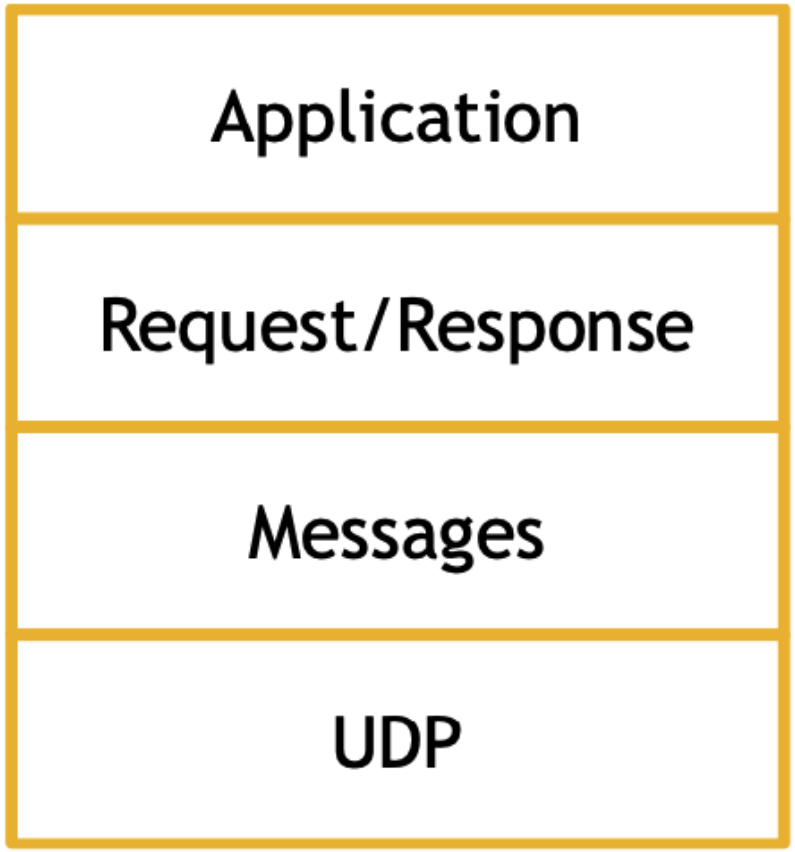
- Unlike MQTT it is **1 to 1** protocol.
- CoAP is based on **Client/Server model**.
- It supports in **RESTful** web services.
- It uses less resources then HTTP.
- HTTP runs on TCP and it not suitable for devices with constraints, hence CoAP runs on UDP, although it is connection less protocol.

CoAP (Constrained Application Protocol)

- In CoAP client can use **GET, PUT, DELETE** methods during request.
- CoAP provides support to optimize the datagram length.
- It also support **IP multicast**
- Re-transmission is supported by TCP and not by UDP, but CoAP supports **re-transmission**.
- CoAP also supports **Piggy-backed** package.
  - When server sends response with Acknowledgement it is known as Piggy-backed.

CoAP Layers

- Mostly it is divided in two layers.
  - Upper Layer (Request and Response)
  - Lower Layer (Message)
- There are 4 types of message
  - Confirmable (CON)
  - Non Confirmable (NON)
  - Acknowledgement (ACK)
  - Reset (RST)
- Reset (RST) – RST message is sent when recipient fails to send response in time.



Transmission Control Protocol (TCP)

- Connection oriented
  - Explicit set-up and tear-down of TCP session
- Stream-of-bytes service
  - Sends and receives a stream of bytes, not messages
- Reliable, in-order delivery
  - Checksums to detect corrupted data
  - Acknowledgments & retransmissions for reliable delivery
  - Sequence numbers to detect losses and reorder data
- Flow control
  - Prevent overflow of the receiver’s buffer space
- Congestion control
  - Adapt to network congestion for the greater good

An Analogy: Talking on a Cell Phone

- Alice and Bob on their cell phones
  - Both Alice and Bob are talking
- What if Alice couldn't understand Bob?
  - Bob asks Alice to repeat what she said
- What if Bob hasn't heard Alice for a while?
  - Is Alice just being quiet?
  - Or, have Bob and Alice lost reception?
  - How long should Bob just keep on talking?
  - Maybe Alice should periodically say "uh huh"
  - … or Bob should ask "Can you hear me now?" ☺

Transmission Control Protocol

- TCP must perform typical transport layer functions:
  - Segmentation → breaks message into packets
  - End-to-end error control →since IP is an unreliable Service
  - End-to-end flow control → to avoid buffer overflow
  - Multiplexing and demultiplexing sessions
- TCP is [originally described in RFC 793, 1981]
  - Reliable
  - Connection-oriented → virtual circuit
  - Stream-oriented → users exchange streams of data
  - Full duplex → concurrent transfers can take place in both directions
  - Buffered → TCP accepts data and transmits when appropriate (can be overridden with “push”)