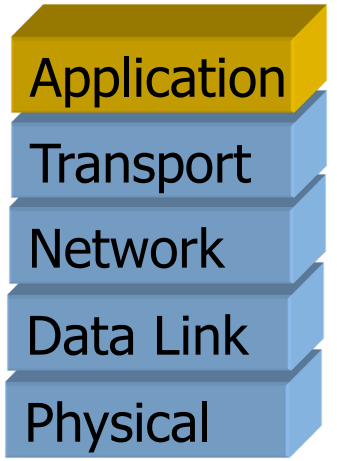
APPLICATION LAYER

APPLICATION LAYER



- Recall, the transport layer is responsible for reliable data delivery required by the application layer between sensor nodes and the sink(s).

We asserted that due to the energy, computation, and storage constraints of sensor nodes, traditional transport protocols cannot be applied directly to sensor networks without modification.

- A lot remains to be done in this layer. Some potential applications have been suggested as listed below but little work of substance has been reported on any particular area.

APPLICATION LAYER

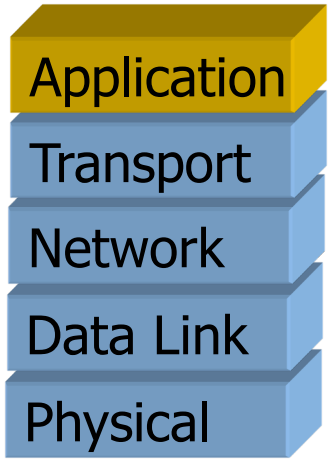
The application layer contains a variety of application layer protocols to generate various sensor network applications. This layer performs various sensor network applications, such as:

- ✓ Query dissemination.
- ✓ Node localization
- ✓ Time synchronization, and
- ✓ Network security, Authentication, key distribution,
- ✓ Sensor movement management

For example, the Sensor Management Protocol (SMP) is an application – layer management protocol that provides software operations to perform a variety of tasks, for example, exchanging location – related data, synchronizing sensor nodes, moving sensor nodes, scheduling sensor nodes, and querying the status of sensor nodes. Turning sensors on and off.

APPLICATION LAYER

- Interest Dissemination – Interest is sent to a sensor or a group of sensors. The interest is expressed in terms of an attribute or a triggering event.
- Advertisement of Sensed Data – Sensor nodes advertise sensed data in a concise and descriptive way and users reply with requests of data they are interested in receiving



SENSOR MANAGEMENT PROTOCOL (SMP)

An application layer management protocol makes the hardware and software of the lower layers transparent to the sensor network management applications.

System administrators interact with sensor networks by using sensor management protocol (SMP).

Unlike many other networks, sensor networks consist of nodes that do not have global identification, and often infrastructureless.

Therefore, SMP needs to access the nodes by using attribute-based naming and location-based addressing,

SIMPLE NETWORK MANAGEMENT PROTOCOL?

SNMP can't be adopted. Why?

- Sensor-specific failures are not handled
- Difficult to find the failed nodes
- Physical connections are not utilized
- Commonly, there is not a management agent
- Specifying nodes is difficult
- Network is self-configured, so that management server doesn't have all information of sensor nodes

- **Challenges**
  - Presents many and drastically different challenges. For example:
    - Deployment of nodes, Discarding of nodes
  - Requires augmentation to (or new approaches over) traditional network and service management techniques
  - Needs to take into account specific characteristics of WSNs (e.g., energy waste)

MANAGEMENT REQUIREMENT

- **Fault tolerance**
  - **Handle loss of nodes** - Lack of Power, Physical damage, Environmental interference
- **Scalability**
  - **Handle high density of nodes** - The number of sensor nodes is an extreme value of millions
- **Production costs**
  - **Make them low cost** - Cost of a single node is very important to justify the overall cost of the network
- **Operating environment**
  - **Survive and maintain communication** - The bottom of an ocean, biologically contaminated field, battlefield
- **Transmission media**
  - **Wireless** - Radio, infrared, optical media
- **Hardware constraints**
  - **Nodes are tiny** - Very small size, very light node, limited memory, limited battery
- **Power consumption**
  - **Limited Tx, computation, lifetime** - Replenishment of power is impossible
- **Changing Topology**
  - **Nodes** - Nodes moving, new nodes, loss nodes

Functional Areas

	Functions
Fault	▪ Faults in USNs are not an exception and tend to occur frequently, thus fault management is a critical function. ▪ This is one of the reasons that make WSN management different from traditional network management ▪ Self-diagnostic: the network monitors itself and find faulty or unavailable nodes ▪ Self-healing: the network prevents disruptions or that acts to recover itself or the node after the self-diagnostic
Configuration	▪ Self-organization: is the property which the sensor nodes must have to organize themselves to form the network ▪ Self-configuration: nodes setup and network boot up must occur automatically
Accounting	▪ It includes functions related to the use of resources and corresponding reports ▪ It establishes metrics, quotas and limits that can be used by functions of other functional areas ▪ It must provide self-sustaining functionalities
Performance	▪ There is a trade-off to be considered: the higher the number of managed parameters, the higher the energy consumption and the lower the network lifetime ▪ On the other hand, if enough parameter values are not obtained, it may not be possible to manage the network appropriately
Security	▪ Security functionalities for USNs are intrinsically difficult to be provided because of their ad-hoc organization, intermittent connectivity, wireless communication and resource limitations ▪ A WSN is subject to different safety threats: internal, external, accidental, and malicious

Specific Management Functions: Power Management

- **Manages how a sensor node uses its power**
- **Example**
  - Sensor node may turn off its receiver after receiving a message from one of its neighbors
    - avoid getting duplicated messages
  - When the power level of the sensor node is low
    - Broadcasts to its neighbor when it is low in power
    - Cannot participate in routing messages
    - Reserve the remaining power for sensing
- **Requirements**
  - Using battery
  - Limited Power
  - Expand the life time of sensor node
  - Reduce the overhead

Simple Routing ⟺ **Processing**

Power Management cont.

Power Management in Protocol Layer

- **Physical layer**
  - Low Power Modulation Scheme
  - Transceiver, Sensor, Process : Small, Low Power, Low Cost
- **Data link layer**
  - Energy efficiency MAC protocol
    - Adaptive duty cycling – S-MAC, ASCENT, SPAN
    - Wake up on-demand – STEM, Wake-on-Wireless
  - Reduce the collision, signaling, frame overhead
  - Power saving mode (ex. On/Off mode)
- **Network Layer**
  - Energy-efficiency routing
  - Energy-efficiency data aggregation algorithms
  - Location based routing
- **Transport Layer**
  - Use UDP message protocol between Sink and Sensor node
  - Limited memory and processing power
- **Application Layer**
  - Energy-efficiency Applications

UDP (User Datagram Protocol) is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP). UDP is an alternative to the Transmission Control Protocol

Topology Management

- **Goal**
  - is to coordinate the sleep transitions of all nodes, while ensuring adequate network connectivity, such that data can be forwarded efficiently to the data sink.
- **Requirements**
  - Heterogeneous node
  - Data discovery & data dissemination
  - Limited memory & power constraint
  - Application requirements
  - Node mobility
- **Ad-hoc Self-organization**
  - LCA (Linked Cluster Algorithm)
  - LAA (Link Activation Algorithm)
  - DEA (Distributed Evolution Algorithm)

Security Management

- **Requirements**
  - Peanut CPU (slow computation rate)
  - Battery power: trade-off between security and battery life
  - Limited memory
  - High latency: conserve power, turn on periodically
- **Security Management in USN**
  - Applications need security (privacy)
  - Absence of security enables attacks such as spoofing & replay attacks, resulting in DoS or system compromise
  - Intrusion prevention : First line of defense
  - Intrusion detection : Second line of defense
- **Main Security Threats in USN**
  - Radio links are insecure
  - Sensor nodes are not temper resistant
- **Attacker types**
  - Mote-class
  - Outside / inside

- **Countermeasures**
  - Link layer encryption – selective forwarding
  - Using a counter – Replay attacks
  - Limiting the number of neighbors per node – Insider attacks
  - Bi-directionality of the link – HELLO flood
  - Geographically routing – Wormhole attacks

- **Attacks**
  - Physical attack
  - Denial-of-service
  - Battery exhaustion
  - Clock synchronization
  - Location discovery
  - Attacks on routing
    - spoofed, altered, or replayed routing information
    - selective forwarding
    - sinkhole attack
    - sybil attack
    - wormholes
    - HELLO flood attacks
    - acknowledgment spoofing

APPLICATION LAYER MESSAGING PROTOCOLS

Otman A. Basir

PROCESS2PROCESS COMMUNICATION

It also defines how the applications interface with lower layer protocols to send data over the network. Enables process-to-process communication using ports.
Protocols:

- HTTP: Hyper Text Transfer Protocol that forms foundation of WWW. Follow request- response model Stateless protocol.
- CoAP: Constrained Application Protocol for machine-to-machine(M2M) applications with constrained devices, constrained environment and constrained n/w. Uses client- server architecture.
- WebSocket: allows full duplex communication over a single socket connection.
- MQTT: Message Queue Telemetry Transport is light weight messaging protocol based on publish-subscribe model. Uses client server architecture. Well suited for constrained environment.

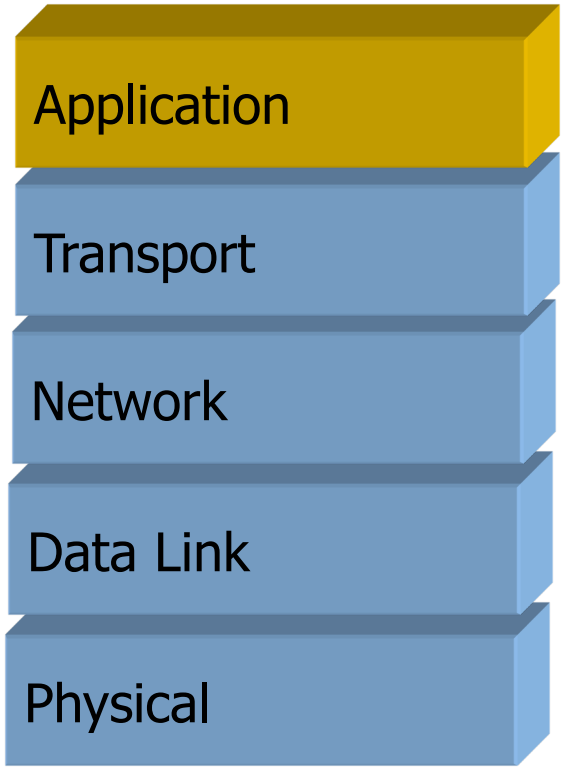
APPLICATION LAYER

- XMPP: Extensible Message and Presence Protocol for real time communication and streaming XML data between network entities. Support client–server and server–server communication.
- DDS: Data Distribution Service is data centric middleware standards for device–to–device or machine–to–machine communication. Uses publish–subscribe model.
- AMQP: Advanced Message Queuing Protocol is open application layer protocol for business messaging. Supports both point–to–point and publish–subscribe model.

APPLICATION LAYER

With HTTP's protocol overhead, how will billions of low–power, low–cost IoT devices communicate on the Internet?

TABLE 1: IETF INTERNET PROTOCOL SUITE		
Layer	**Full Internet**	**Description**
Application	HTTP	Defines TCP/IP application protocols and the interface to transport layer services.
Transport	TCP / UDP	Provides communication session management. Defines the level of service and status of the connection.
Internet	IP	Performs IP routing with source and destination address information.



IETF: Internet Engineering Task Force

APPLICATION LAYER

- Two of the most promising messaging protocols for small devices are MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol) .

Both MQTT and CoAP:

- Are open standards
- Are better suited to constrained environments than HTTP
- Provide mechanisms for asynchronous communication
- Run over IP
- Have a range of implementations

- MQTT gives flexibility in communication patterns and acts purely as a pipe for binary data, while CoAP is designed for interoperability with the web.

MQTT (MESSAGE QUEUING TELEMETRY TRANSPORT)

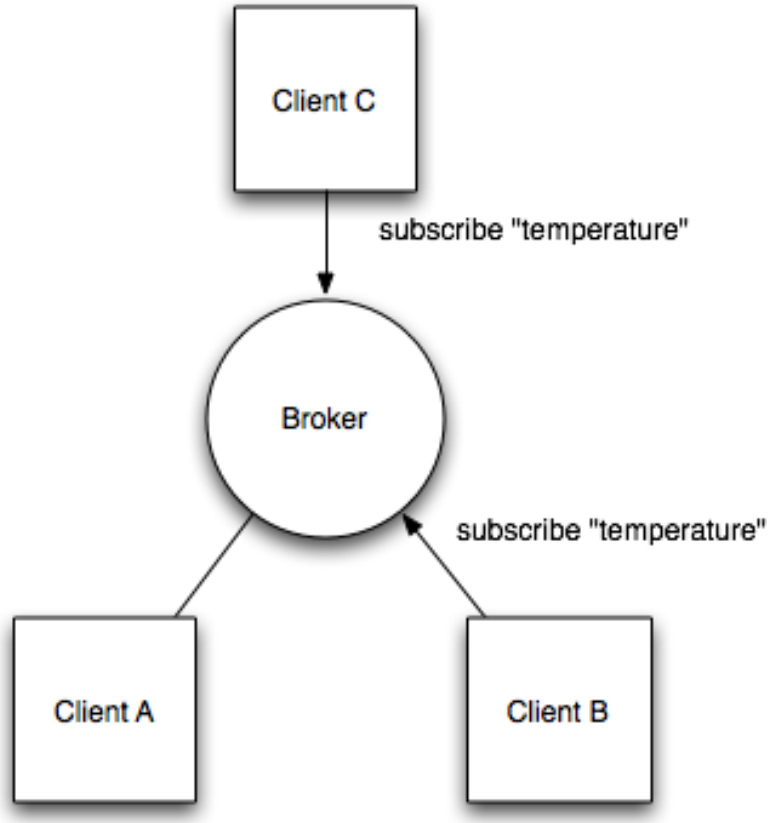
- A publish/subscribe messaging protocol designed for lightweight M2M communications, originally developed by IBM and is now an open standard by the OASIS (Organization for the Advancement of Structured Information Standards) open-standards and (ISO/IEC 20922), MQTT v3.1.1.

- MQTT uses the concept of a message broker, where message brokers handle subscriptions and distribution to interested clients

- MQTT specifically targets constrained environments
  - Small code foot print in order to fit onto embedded devices
  - Limited network bandwidth to work in constrained networks
- Mostly, MQTT is used to funnel sensor readings into a collection point (Broker)

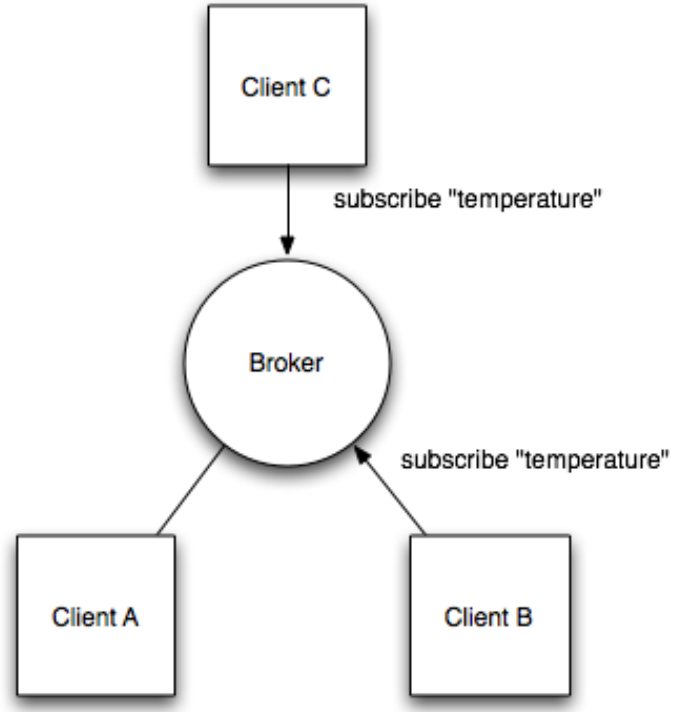
MQTT ARCHITECTURE

- MQTT has a client/server model, where every sensor is a client and connects to a server, known as a broker, over TCP.
- MQTT is message oriented. Every message is a discrete chunk of data, can be opaque to the broker.
- Every message is published to an address, known as a topic. Clients may subscribe to multiple topics. Every client subscribed to a topic receives every message published to the topic.
- The publisher subscriber model allows MQTT clients to communicate one-to-one, one-to-many and many-to-one.

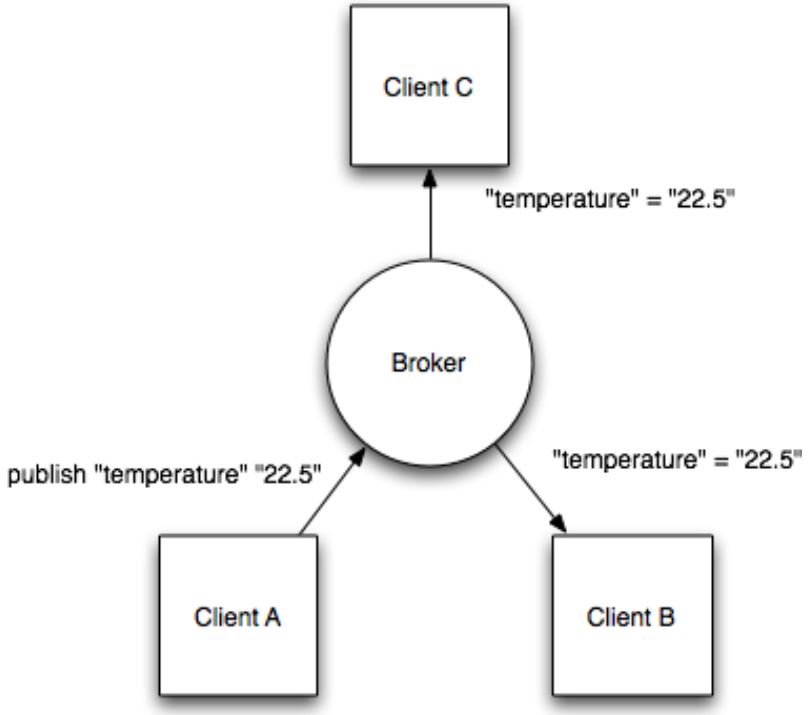


MQTT ARCHITECTURE

- Example, a simple network with three clients and a central broker.



All three clients open TCP connections with the broker. Clients B and C subscribe to the topic temperature .



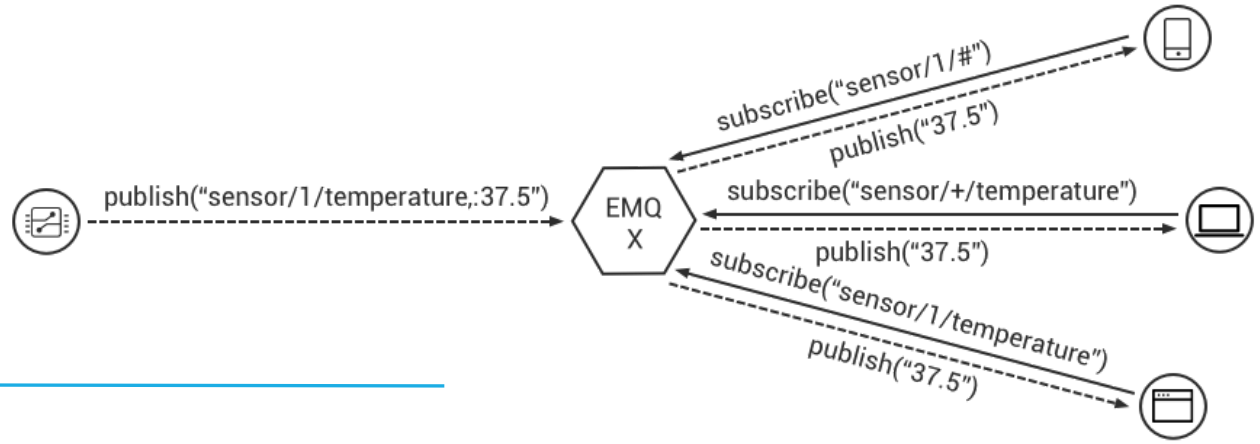
At a later time, Client A publishes a value of 22.5 for topic temperature . The broker forwards the message to all subscribed clients.

MQTT TOPICS

- A very important concept of MQTT is topic
  - each message has a topic, it decides which message is going to be received by whom.
  - Topic is the routing mechanism for MQTT.
  - A topic consists of one or more levels, levels are separated by slash(/). Sometimes a topic is comparable to a category or a directory.
  - Wildcards ("#" and "+") in a topic makes it easier for subscribers to subscribe to multiple topics at once. The "+" stands for one single level while the "#" stands for multiple levels. In most cases you can name a topic freely at your choice, but with one exception: Topics start with "$" are reserved for system use.
  - MQTT supports 1-to-many (broadcasting) messaging by its nature: A publisher publishes a message with a topic to a broker, the broker forwards this message to all clients who subscribed to this very topic.

MQTT TOPIC SUBSCRITPION EXAMPLE

- A sensor measures the patient's temperature and publishes it to the MQTT broker with the topic "sensor/1/temperature".
- A system that cares about everything of this patient can subscribe to "sensor/1/#".
- A device that is collecting temperature of every patients can subscribe to "sensor/+/temperature".
- A device that is only interested in this patient's temperature can precisely subscribe to the topic "sensor/1/temperature", without using any wildcard.



MQTT MESSAGES

- A Message is the information to exchange with other parties within a system.
- MQTT has14 message types, that are used to connect/disconnect, to publish messages, to subscribe to topics, to maintain the connection or to ensure the QoS.
- An MQTT packet consists of a fixed header, an optional variable header and an optional payload:
- Fixed header, present in all MQTT Control Packets. It consist of the packet type, the flags and the remaining length.
- Variable header, present in some MQTT Control Packets. The content of it varies depending on the packet type.
- Payload, present in some MQTT Control Packets. The last part of a packet where application related information is carried.
- The smallest MQTT packet has a size of 2 bytes; the size of the fixed header.

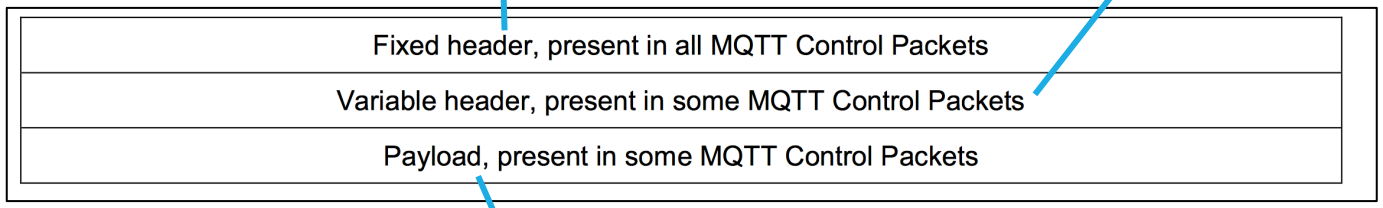
MQTT CONTROL MESSAGES

Type Name	Value	Description
CONNECT	1	CONNECT 1 Client request to connect to Server
CONNACK	2	Connect acknowledgement
PUBLISH	3	Publish message
PUBACK	4	Publish acknowledgement
PUBREC	5	Publish received (assured delivery part 1)
PUBREL	6	Publish release (assured delivery part 2)
PUBCOMP	7	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client subscribe request
SUBACK	9	Subscribe acknowledgement
UNSUBSCRIBE	10	Unsubscribe request
UNSUBACK	11	Unsubscribe acknowledgement
PINGREQ	12	PING request
PINGRESP	13	PING response
DISCONNECT		Client is disconnecting

MQTT MESSAGE STRUCTURE

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2…	Remaining Length							

Bit	7	6	5	4	3	2	1	0
byte 1	Packet Identifier MSB							
byte 2	Packet Identifier LSB							



Control Packet	Payload
CONNECT	Required
CONNACK	None
PUBLISH	Optional
PUBACK	None
PUBREC	None
PUBREL	None
PUBCOMP	None
SUBSCRIBE	Required
SUBACK	Required
UNSUBSCRIBE	Required
UNSUBACK	None
PINGREQ	None

Control Packet	Packet Identifier field
CONNECT	NO
CONNACK	NO
PUBLISH	YES (If QoS > 0)
PUBACK	YES
PUBREC	YES
PUBREL	YES
PUBCOMP	YES
SUBSCRIBE	YES
SUBACK	YES

MQTT QOS

- Three qualities of service for message delivery:
  - "At most once", where messages are delivered according to the best efforts of the underlying TCP/IP network. Message loss or duplication 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.

MQTT PUBLISH MESSAGE FIXED HEADER

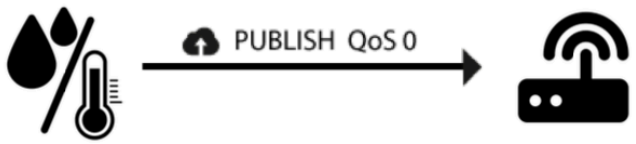
Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (3)				DUP flag	QoS level		RETAIN
	0	0	1	1	X	X	X	X
byte 2	Remaining Length							

- DUP flag is set to 0 to indicates that this is the first occasion that the Client or Server has attempted to send this MQTT PUBLISH Packet.
- If the DUP flag is set to 1, it indicates that this might be re-delivery of an earlier attempt to send the Packet. DUP is 0 for all QoS 0 messages.

- If the RETAIN flag is set to 1, in a PUBLISH Packet sent by a Client to a Server, the Server MUST store the Application Message and its QoS, so that it can be delivered to future subscribers whose subscriptions match its topic name

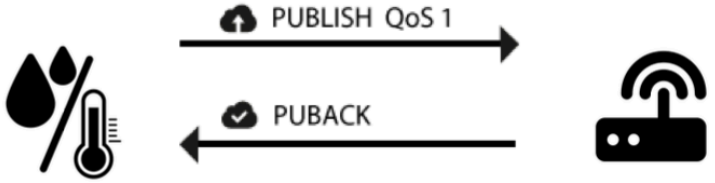
MQTT QOS

QoS 0 ✖



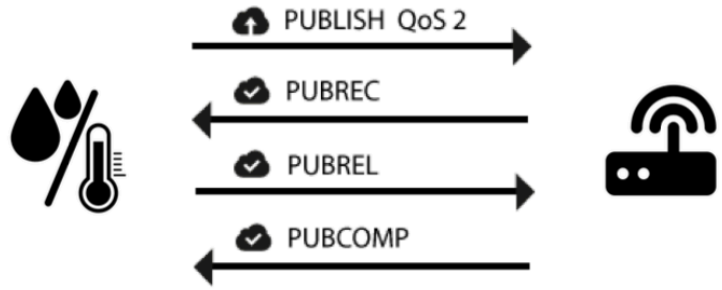
- Best-effort delivery.
- No guarantee of delivery.
- Recipient does not acknowledge receipt of the message and the message is not stored and re-transmitted by the sender

QoS 1 ✔



- It guarantees that a message is delivered at least one time to the receiver.
- The sender stores the message until it gets a packet from the receiver that acknowledges receipt of the message.
- Message can be sent or delivered multiple times.

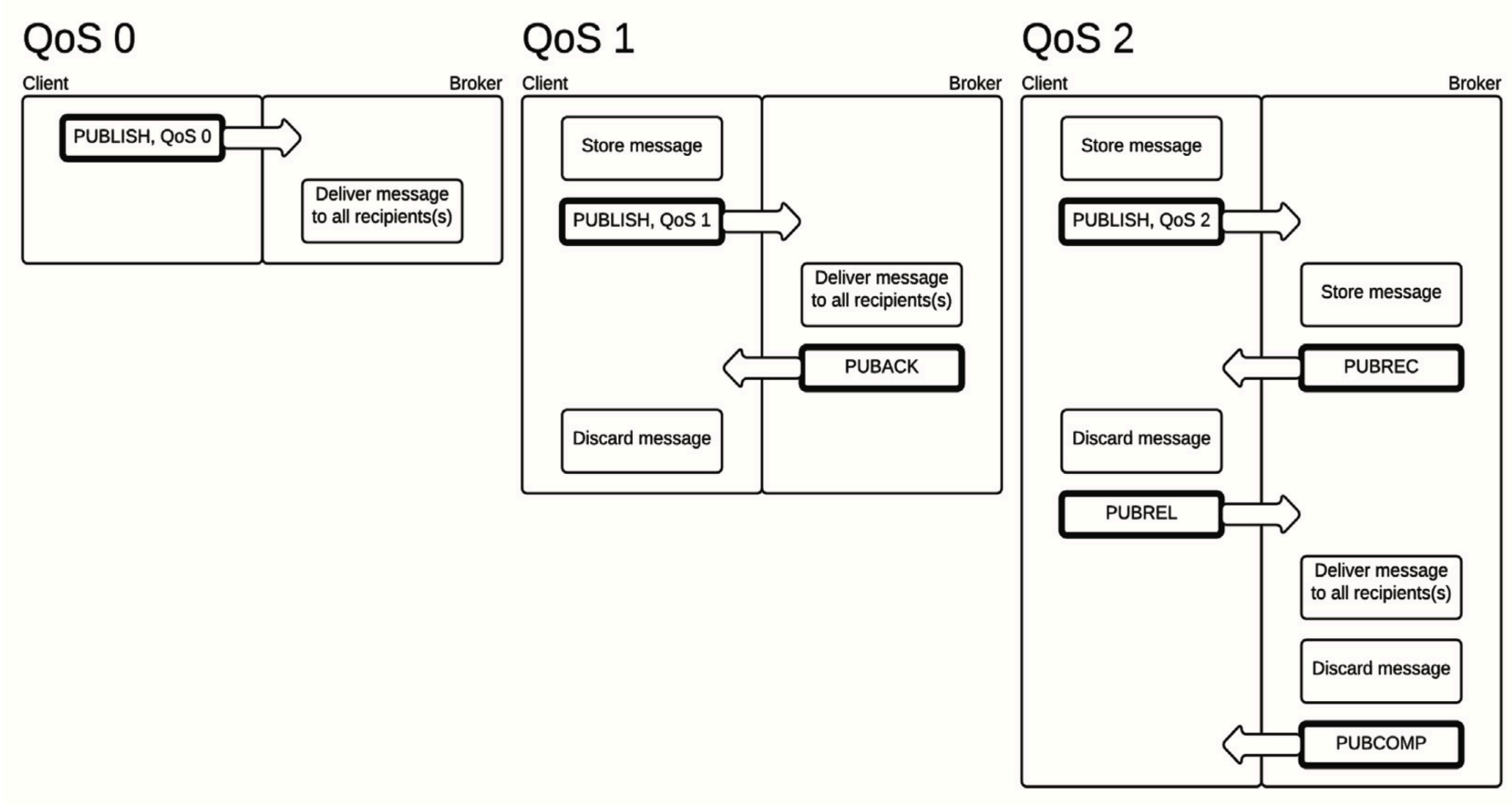
QoS 2 ✔



- It guarantees that each message is received only once by the intended recipients.
- Safest and slowest QoS.
- Guarantee is provided by at least two request/response flows (a four-part handshake) between the sender and the receiver

https://www.researchgate.net/publication/328051666_Service_Provisioning_in_Vehicular_Networks_Through_Edge_and_Cloud_An_Empirical_Analysis

MQTT MESSAGE EXCHANGE (QOS 0,1,2)



MQTT ADVANTAGES

- Scalability
  - MQTT's "pub/sub" model scales well and can be power efficient. Brokers and nodes publish information and others subscribe according to the message content, type, or subject. Generally, the broker subscribes to all messages and then manages information flow to its nodes.

- Space decoupling
  - While the node and the broker need to have each other's IP address, nodes can publish information and subscribe to other nodes' published information without any knowledge of each other, since everything goes through the central broker. This reduces overhead that can accompany TCP sessions and ports, and allows the end nodes to operate independently of one another.

MQTT ADVANTAGES

- Time decoupling
  - A node can publish its information regardless of other nodes' states. Other nodes can then receive the published information from the broker when they are active. This allows nodes to remain in sleepy states even when other nodes are publishing messages directly relevant to them.
- Synchronization decoupling
  - A node that's in the midst of one operation is not interrupted to receive a published message to which it's subscribed. The message is queued by the broker until the receiving node is finished with its existing operation. This reduces repeated operations by avoiding interruptions of ongoing operations or sleepy states.

MQTT IMPLEMENTATION

- Open source implementations:
  - Paho
  - Mosquitto
  - MQTT.js