MQTT

Introduction

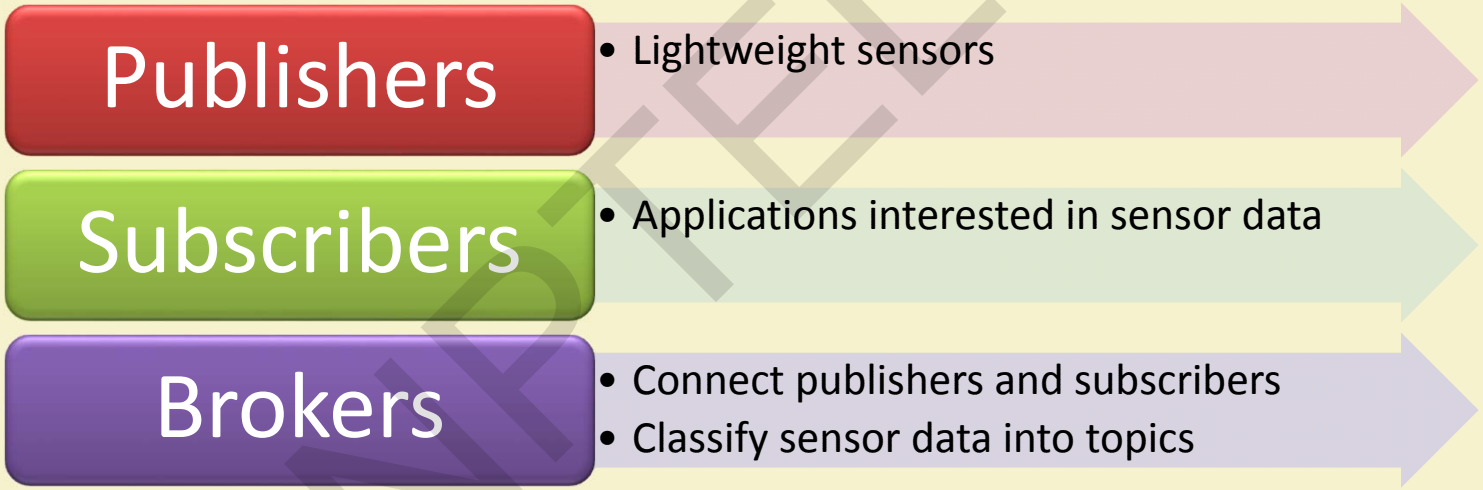
- ✓ **Message Queue Telemetry Transport.**
- ✓ ISO standard (ISO/IEC PRF 20922).
- ✓ It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
- ✓ MQTT was introduced by IBM in 1999 and standardized by OASIS in 2013.
- ✓ Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and networks and communications on the other side.

Source: "MQTT", Wikipedia (Online)

- ✓ 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

Source: "MQTT", Wikipedia (Online)

MQTT Components

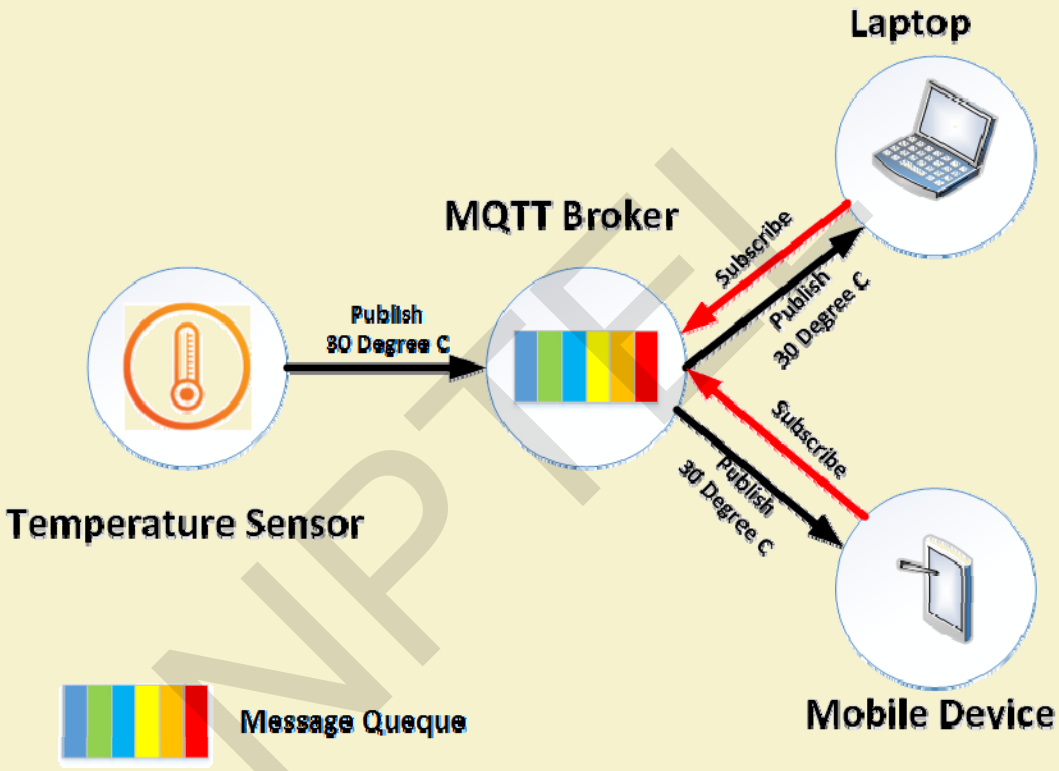


Source: "MQTT", Wikipedia (Online)

MQTT Methods

- Connect
- Disconnect
- Subscribe
- Unsubscribe
- Publish

Source: "MQTT", Wikipedia (Online)



Source: "MQTT 101 – How to Get Started with the lightweight IoT Protocol", HiveMQ (Online)

Communication

- ✓ The protocol uses a **publish/subscribe** architecture (HTTP uses a request/response paradigm).
- ✓ Publish/subscribe is **event-driven** and enables messages to be pushed to clients.
- ✓ The central **communication point is the MQTT broker**, which is in charge of dispatching all messages between the senders and the rightful receivers.
- ✓ Each client that publishes a message to the broker, includes a **topic** into the message. The **topic is the routing information for the broker.**

Source: "MQTT 101 – How to Get Started with the lightweight IoT Protocol", HiveMQ (Online)

- ✓ Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client.
- ✓ Therefore the clients don't have to know each other. They only communicate over the topic.
- ✓ This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

Source: "MQTT 101 – How to Get Started with the lightweight IoT Protocol", HiveMQ (Online)

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*.
- ✓ On one hand the client (e.g. mobile device) can subscribe to the exact topic or on the other hand, it can use a **wildcard**.

Source: "MQTT 101 – How to Get Started with the lightweight IoT Protocol", HiveMQ (Online)

- ✓ 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** (*#*).
- ✓ It allows to subscribe to all underlying hierarchy levels.
- ✓ For example *house/#* is subscribing to all topics beginning with *house*.

Source: "MQTT 101 – How to Get Started with the lightweight IoT Protocol", HiveMQ (Online)

Applications

- ✓ **Facebook Messenger** uses MQTT for online chat.
- ✓ **Amazon Web Services** use Amazon IoT with MQTT.
- ✓ **Microsoft Azure** IoT Hub uses MQTT as its main protocol for telemetry messages.
- ✓ The **EVRYTHNG IoT platform** uses MQTT as an M2M protocol for millions of connected products.
- ✓ **Adafruit** launched a free MQTT cloud service for IoT experimenters called Adafruit IO.

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.

Source: M. Singh, M. Rajan, V. Shivraj, and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," in Fifth International Conference on Communication Systems and Network Technologies (CSNT 2015), April 2015, pp. 746-751

- ✓ 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.

Source: M. Singh, M. Rajan, V. Shivraj, and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," in Fifth International Conference on Communication Systems and Network Technologies (CSNT 2015), April 2015, pp. 746-751

Thank You!!

CoAP

Constrained Application Protocol
(Web Protocol for IoT)

Aniruddha Chakrabarti
Associate Vice President and Chief Architect, Digital Practice, Mphasis
ani.c@outlook.com | Linkedin.com/in/aniruddhac | slideshare.net/aniruddha.chakrabarti/ | Twitter - anchakra

Agenda

- What is CoAP
- IoT (Internet of Things) protocol stack
- CoAP - Similarity and differences with HTTP
- Comparison with other IoT protocols
- CoAP Libraries
- CoAP – sample client and server

What is CoAP

"The Constrained Application Protocol (CoAP) is a specialized **web transfer protocol** for use with **constrained** nodes and constrained networks in the **Internet of Things.**

The protocol is designed for machine-to-machine (M2M) and IoT applications such as smart energy and building automation."

- CoAP is an application layer protocol (similar as HTTP) and follows the request-response pattern used by HTTP – CoAP has a transparent mapping to HTTP
- CoAP uses familiar HTTP stuff like Methods (Get, Post, Put, Delete), Status Codes, URIs, content type / MIME
- Think CoAP as HTTP REST for Constrained environment (low memory, low bandwidth, high rate of packet failure, low power)
- CoAP core protocol spec is specified in RFC 7252
- Like HTTP, CoAP can carry different types of payloads, and can identify which payload type is being used. CoAP integrates with XML, JSON, CBOR, or any data format of your choice.

http://coap.technology/

IoT protocol stack (or protocol soup?)

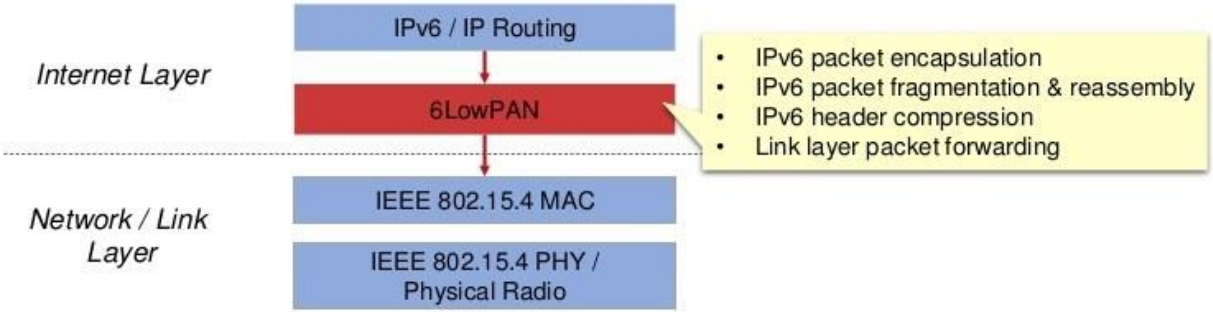
TCP/IP Model	**IoT stack**	**Internet / Web App stack**
	IoT Applications; Device Management	Web Applications
Data Format	Binary, JSON, CBOR	HTML, XML, JSON
Application Layer	**CoAP**, MQTT, XMPP, AMQP	HTTP, DHCP, DNS, TLS/SSL
Transport Layer	UDP, DTLS	TCP, UDP
Internet Layer	IPv6 / IP Routing; 6LowPAN	IPv6, IPv4, IPSec
Network / Link Layer	IEEE 802.15.4 MAC; IEEE 802.15.4 PHY / Physical Radio	Ethernet (IEEE 802.3), DSL, ISDN, Wireless LAN (IEEE 802.11), Wi Fi

IEEE 802.15.4 (Network / Link Layer)

- Standard for wireless communication that defines the Physical Layer (PHY) and Media Access Control (MAC) layers.
- Standardized by the IEEE (Institute for Electrical and Electronics Engineers) – similar to IEEE 802.3 for Ethernet, IEEE 802.11 is for wireless LANs (WLANs) or Wi Fi.
- 802.15 group of standards specifies a variety of wireless personal area networks (WPANs) for different applications - For instance, 802.15.1 is Bluetooth.
- IEEE 802.15.4 focusses on communication between devices in constrained environment with low memory, low power and low bandwidth.

6LowPAN (Internet Layer)

- Secret sauce that allows larger IPv6 packets to flow over 802.15.4 links that support much smaller packet size.
- Acronym of IPv6 over Low power Wireless Personal Area Networks - an adaptation layer that allows to transport IPv6 packets over 802.15.4 links. Without 6LowPAN IPv6 and internet protocols would not work in these Low power Wireless Personal Area Networks that uses IEEE 802.15.4.
- 6LoWPAN is an open standard defined in RFC 6282 by the Internet Engineering Task Force (IETF), the standards body that defines many of the open standards used on the Internet such as UDP, TCP and HTTP to name a few.
- As mentioned previously, an IPv6 packet is too large to fit into a single 802.15.4 frame. What 6LowPAN does to fit an IPv6 packet in 802.15.4 frame is -
  - **Fragmentation and Reassembly -** Fragments the IPv6 packet and send it through multiple smaller size packets that can fit in a 802.15.4 frame. On the other end it reassembles the fragmented packets to re-create the IPv6 packet.
  - **Header Compression** – Additionally it also compresses the IPv6 packet header to reduce the packet size.



CoAP – Similarities with HTTP

- Inspired from HTTP – similar to HTTP CoAP also uses a request response model.
- Can be transparently mapped to HTTP - However, CoAP also provides features that go beyond HTTP such as native push notifications and group communication.
- From a developer point of view, CoAP feels very much like HTTP. Obtaining a value from a sensor is not much different from obtaining a value from a Web API.
- **REST model for small devices** - It implements the REST architectural style
- **Made for billions of nodes with very less memory -** The Internet of Things will need billions of nodes, many of which will need to be inexpensive. CoAP has been designed to work on microcontrollers with as low as 10 KiB of RAM and 100 KiB of code space.
- Designed to use minimal resources, both on the device and on the network. Instead of a complex transport stack, it gets by with UDP on IP. A 4-byte fixed header and a compact encoding of options enables small messages that cause no or little fragmentation on the link layer.
- Like HTTP, CoAP can carry different types of payloads, and can identify which payload type is being used. CoAP integrates with XML, JSON, CBOR, or any data format of your choice.
- **Discovery integrated** - CoAP resource directory provides a way to discover the properties of the nodes (devices/things in IoT) on your network.

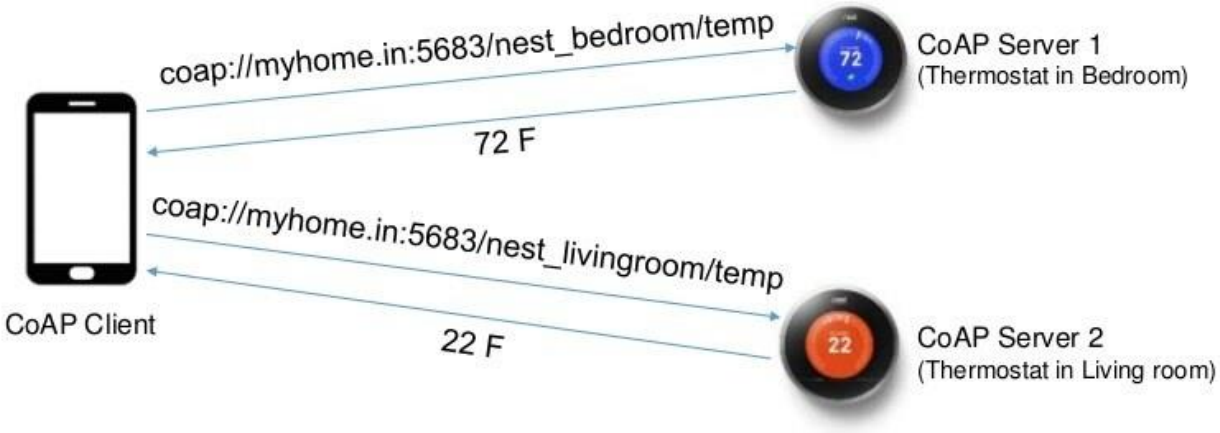
1 KiB (Kibibyte) = 1024 bytes. Kibibyte is established to replace the kilobyte in those computer science contexts in which the term kilobyte is used to mean 1024 bytes. Typically Killo represents 1000, and so causes confusion.

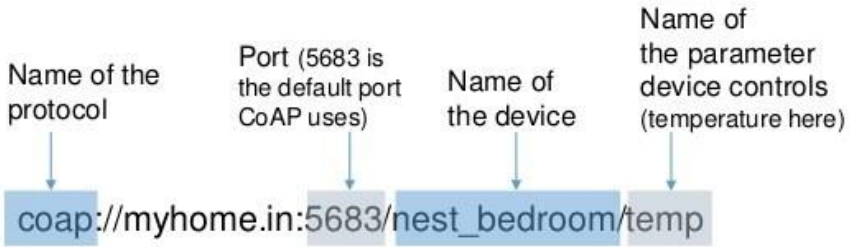
CoAP – How it's different from HTTP

- CoAP runs over UDP and not TCP (HTTP typically uses TCP, though it can use UDP also)
- CoAP replaces the text headers used in HTTPU (HTTP Unicast) with more compact binary headers
- It reduces the number of options available in the header
- CoAP also reduces the set of methods that can be used; it allows
  - GET
  - POST
  - PUT, and
  - DELETE
- Method calls can be made using confirmable & nonconfirmable message services
  - When a confirmable message is received, receiver always returns an acknowledgement. The sender resends messages if an acknowledgement is not returned within a given time.
  - When a nonconfirmable message is received, receiver does not return an acknowledgement.
- No of response code has also been reduced (to make implementation simpler)
- CoAP also broke away from the Internet Media Type scheme used in HTTP and other protocols and replaced this with a reduced set of Content-Formats.

http://www.iana.org/assignments/core-parameters/

CoAP – Request Response





CoAP Methods

- GET
- POST
- PUT
- DELETE
- OBSERVE (Not present in Http, New in CoAP)
- Method calls can be made using confirmable & nonconfirmable message services
  - When a confirmable message is received, receiver always returns an acknowledgement. The sender resends messages if an acknowledgement is not returned within a given time.
  - When a nonconfirmable message is received, receiver does not return an acknowledgement.
- No of response code has also been reduced (to make implementation simpler)
- CoAP also broke away from the Internet Media Type scheme used in HTTP and other protocols and replaced this with a reduced set of Content-Formats.

CoAP Message Types

CON / Confirmable message

A confirmable message requires a response, either a positive acknowledgement or a negative acknowledgement. In case acknowledgement is not received, retransmissions are made until all attempts are exhausted.

NON / Non-confirmable message

A non-confirmable request is used for unreliable transmission (like a request for a sensor measurement made in periodic basis. Even if one value is missed, there is not too much impact). Such a message is not generally acknowledged.

ACK / Acknowledgement

Sent to acknowledge a confirmable (CON) message.

RST / Reset

This represents a negative acknowledgement and means “Reset”. It generally indicates, some kind of failure (like unable to parse received data)

CoAP Status Codes

CoAP Status Code	Description
2.01	Created
2.02	Deleted
2.03	Valid
2.04	Changed
2.05	Content
2.31	Continue
4.00	Bad Request
4.01	Unauthorized
4.02	Bad Option
4.03	Forbidden
4.04	Not Found
4.05	Method Not Allowed
4.06	Not Acceptable
4.08	Request Entity Incomplete
4.12	Precondition Failed
4.13	Request Entity Too Large
4.15	Unsupported Content-Format
5.00	Internal Server Error
5.01	Not Implemented
5.02	Bad Gateway
5.03	Service Unavailable
5.04	Gateway Timeout
5.05	Proxying Not Supported

HTTP Status Code	Description
1xx	Informational
2xx	Successful 200 – OK 201 – Created 202 – Accepted 204 – No Content
3xx	Redirection 301 - Moved Permanently 305 - Use Proxy 307 - Temporary Redirect
4xx	Client Error 400 – Bad Request 401 – Unauthorized 403 – Forbidden 404 - Not Found 405 – Method Not Found 408 – Request Timeout
5xx	500 – Internal Server Error 501 – Not Implemented 503 – Service Unavailable 504 - Gateway Timeout

Only mostly used HTTP Status Codes are listed here