overview

MQTT stands for MQ Telemetry Transport or Message Queuing Telemetry Transport and is an ISO standard (ISO/IEC 20922) publish/subscribe, lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements while ensuring reliability and assurance of delivery.

These design principles make MQTT ideal for the machine-to-machine (M2M) or Internet of Things (IOT) world of connected devices. The protocol usually runs over TCP/IP although any network protocol that provides ordered, lossless, bi-directional connections can support MQTT.

MQTT utilises a “hub and spoke” structure with event-driven messaging and "small code footprint". The publish-subscribe model is configured with a loose coupling between senders and receivers, and transports messages to control outputs, read and publish data, and exchange data between devices. MQTT Clients send messages to an MQTT Broker only when needed and receive messages if they subscribe to certain topics: clients subscribe to a narrow selection of topics and only receive the information they are looking for.

MQTT is fast becoming one of the main protocols for IOT deployments and could probably be summarised as a more modern and cloud-centric method of integration than BACnet.

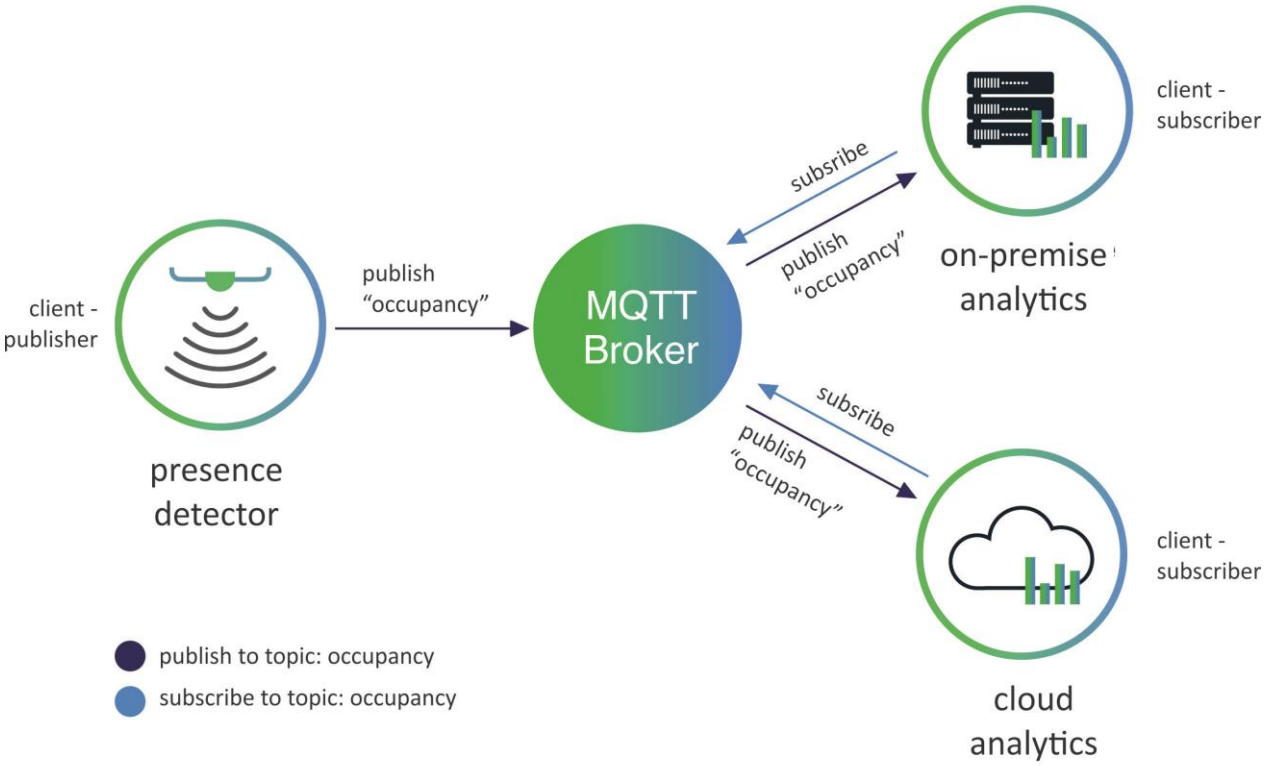
origins

MQTT was originally developed to monitor oil and gas pipelines running through the desert.

In many of the applications, monitoring devices were in remote locations where landline, wired, or radio transmission connection to monitoring servers was difficult or impractical. The only option, at the time, was satellite communication which was very expensive and charged based upon the amount of transmitted data. With installations comprising thousands of field sensors, the challenge was to develop a form of communication that could provide reliable data using minimal bandwidth.

communication and architecture

MQTT uses a **publish - subscribe** model by which a device (**client**) can **publish** a **message** on a **topic** or can **subscribe** to a particular **topic** to receive messages on that topic.



When a **client** (**publisher**) has a new item of data to distribute (publish), it sends a control message with the data to the connected broker. The **broker** then distributes the information to any **client** (**subscriber**) that has subscribed to that topic. The publisher does not need to know the number or locations of subscribers, and subscribers, in turn, do not have to be configured with any data about the publishers.

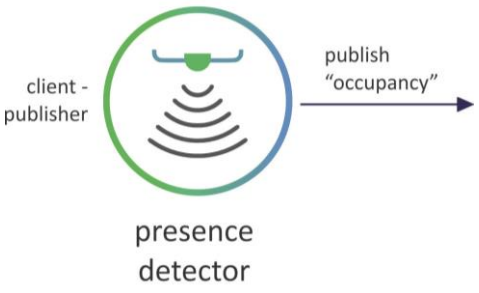
MQTT is an **event-driven** protocol. There is no periodic or ongoing data transmission. This keeps transmission to a minimum. A client only publishes when there is information to be sent, and a broker only sends out information to subscribers when new data arrives.

communication and architecture

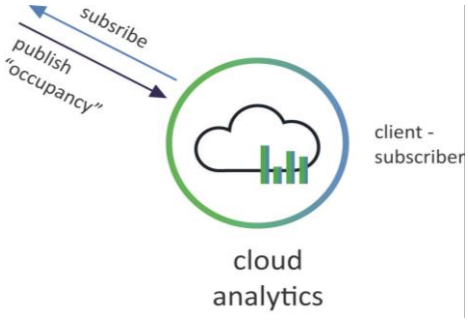
Within the MQTT protocol there are two types of network entities: a **client** and a **broker.**

- An MQTT **client** is a device that runs an MQTT library and connects to an MQTT broker over a network: a client can **publish** or **subscribe** to messages.

a client – **publishing** messages

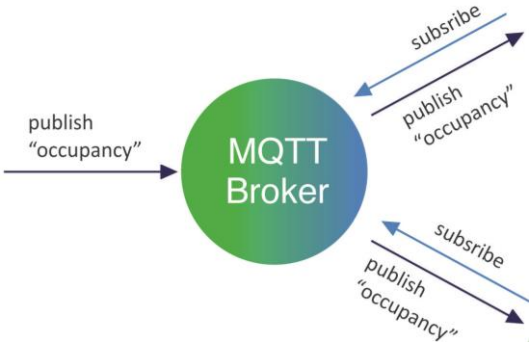


a client – **subscribed** to messages



- An MQTT **broker** is a server that receives **messages** from the clients and then routes the messages to the appropriate destination clients. A **MQTT broker** is the server through which clients communicate. The broker receives communications from clients and sends those communications on to other clients. Clients do not communicate directly with each other, but rather connect to the broker. Each client may be either a publisher, a subscriber, or both.

  The broker is primarily responsible for **receiving** all messages, **filtering** the messages, **decide** who is interested in them and then **publishing** the message to all subscribed clients. Clients only interact with a broker, but a system may contain several broker servers that exchange data based on their current subscribers' topics.



communication and architecture

- **Messages** are the information exchanged between devices such as commands or data.

  A minimal MQTT control message can be as little as two bytes of data although a control message can carry nearly 256 Mb of data if required. There are fourteen defined message types within MQTT that are used to connect and disconnect client from brokers, to publish data, to acknowledge receipt of data, and to supervise connections between clients and servers.

  If a broker receives a message on a topic for which there are no current subscribers, the broker discards the message unless the publisher of the message designated the message as a retained message. The broker stores the last retained message and the corresponding QoS (Quality of Service) for the selected topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic: this approach ensures that new subscribers to a topic receive the most current value rather than having to wait for the next update from a publisher.

- **Topics** are the means of registering interest for incoming messages and specifying where messages should be published: information is organized in a hierarchy of topics.

message format (payload)

The payload (message format) within MQTT is typically UDMI (Universal Device Management Interface).

UDMI schema provides a high-level specification for the management and operation of physical IoT systems. Data is typically exchanged with a cloud entity that can maintain a "digital twin" or "shadow device" in the cloud. Nominally meant for use with **Google's Cloud IoT Core**, as a schema it can be applied to any set of data or hosting setup: additionally, the schema has provisions for basic telemetry ingestion, such as datapoint streaming from an IoT device.

The UDMI schema is intended to be:

- **U**niversal: applicable to all subsystems in a building, not a singular vertical solution.
- **D**evice: operating on an IoT device, ie. a managed entity in a physical space.
- **M**anagement: focusing on device management rather than command and control.
- **I**nterface: defining an interface specification rather than client-library or RPC (remote procedure call) mechanism.

To provide for M2M IOT automation, UDMI strives for the following principles:

- **secure and authenticated:** a secure and authenticated channel from device to managing infrastructure.
- **declarative specification:** describing the desired state of the system and relying on the underlying mechanisms to match actual state with desired state.
- **minimal elegant design:** initially underspecified, with the aim of making it easy to add new capabilities in the future.
- **reduced choices:** too much choice results in more work to implement and more ambiguity so UDMI aims to only one way of doing each thing.
- **structure and clarity:** not a "compressed" format and not designed for very large structures or high-bandwidth streams.
- **property names:** uses *snake_case* convention for property names.