IOT Protocols and Introduction to the ThingsBoard Cloud

WIJETUNGE ARACHCHIGE Dona Dilini Manushi

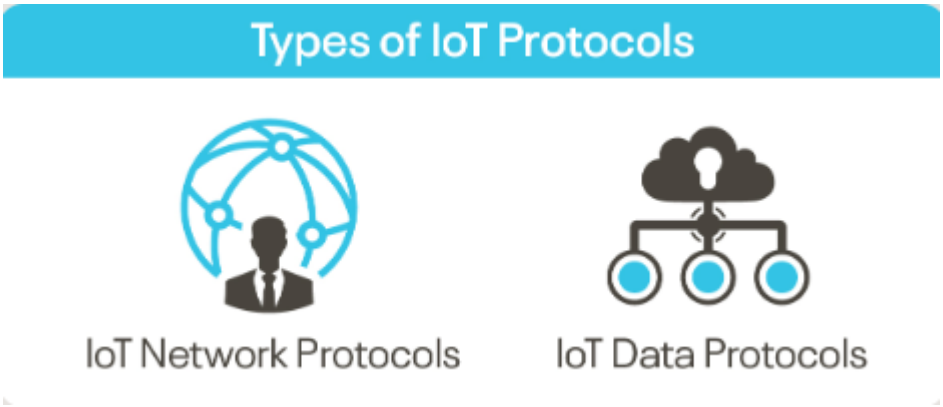
The low power development board, that was initially created for the water meter device, could be used in several devices that involve scenarios that are like the water meter and as the output data must be transmitted and displayed to the user, we had to find a way to do that.

IoT communication protocols are modes of communication that protect and ensure optimum security to the data being exchanged between connected devices.

The IoT Devices are typically connected to the Internet via an IP (Internet Protocol) network. However, devices such as Bluetooth and RFID allow IoT devices to connect locally. In these cases, there's a difference in power, range, and memory used. Connection through IP networks are comparatively complex, requires increased memory and power from the IoT devices while the range is not a problem. On the other hand, non-IP networks demand comparatively less power and memory but have a range limitation.

As far as the IoT communication protocols or technologies are concerned, a mix of both IP and non-IP networks can be considered depending on usage.

Internet of Things Protocols and Standards can be broadly classified into two separate categories:



1. IoT Network Protocols

IoT network protocols are used to connect devices over the network. These are the set of communication protocols typically used over the Internet. Using IoT Protocols, end-to-end data communication within the scope of the network is allowed.

2. IoT Data Protocols

IoT data protocols are used to connect low power IoT devices. These IoT Protocols Provide point-to-point communication with the hardware at the user side without any Internet connection. Connectivity in IoT data protocols is through a wired or a cellular network.

The low power development board can only be used to create low power IoT devices, and we are also planning to use an external sim to gain access to a cellular network, so we decided to choose an IoT data protocol.

Message Queue Telemetry Transport (MQTT)

One of the most preferred protocols for IoT devices, MQTT collects data from various electronic devices and supports remote device monitoring. It is a subscribe/publish protocol that runs over Transmission Control Protocol (TCP), which means it supports event-driven message exchange through wireless networks. MQTT is mainly used in

devices which are economical and requires less power and memory. For instance, fire detectors, car sensors, smart watches, and apps for text-based messaging.

Constrained Application Protocol (CoAP)

CoAP is an internet-utility protocol for restricted gadgets. Using this protocol, the client can send a request to the server and the server can send back the response to the client in HTTP. For light-weight implementation, it makes use of UDP (User Datagram Protocol) and reduces space usage. The protocol uses binary data format EXL (Efficient XML Interchanges). CoAP protocol is used mainly in automation, mobiles, and microcontrollers. The protocol sends a request to the application endpoints such as appliances at homes and sends back the response of services and resources in the application.

Advanced Message Queuing Protocol (AMQP)

AMQP is a software layer protocol for message-oriented middleware environment that provides routing and queuing. It is used for reliable point-to-point connection and supports the seamless and secure exchange of data between the connected devices and the cloud. AMQP consists of three separate components namely Exchange, Message Queue, and Binding. All these three components ensure a secure and successful exchange and storage of messages. It also helps in establishing the relationship of one message with the other. AMQP protocol is mainly used in the banking industry. Whenever a message is sent by a server, the protocol tracks the message until each message is delivered to the intended users/destinations without failure.

Extensible Messaging and Presence Protocol (XMPP)

The XMPP is uniquely designed. It uses a push mechanism to exchange messages in real-time. XMPP is flexible and can integrate with the changes seamlessly. Developed using open XML (Extensible Markup Language), XMPP works as a presence indicator showing the availability status of the servers or devices transmitting or receiving messages. Other than the instant messaging apps such as Google Talk and

WhatsApp, XMPP is also used in online gaming, news websites, and Voice over Internet Protocol (VoIP)

Data Distribution Service (DDS)

It is a middleware IoT Data Protocol and API standard typically used for transmitting data in real-time systems that are running in a distributed environment. DDS works on a UDP-based protocol and is architecturally based on the publish-subscribe design pattern. Defense, aerospace industrial internet of things, healthcare and automotive are top areas where DDS IoT Data Protocol are best implemented to ensure fast, efficient, and predictable data communication. The major benefits of using Data Distribution Service are low-latency data connectivity, extreme reliability, and a scalable architecture.

Simple Text Oriented Messaging Protocol (STOMP)

STOMP is a text-based protocol. It is developed to work with message-oriented middleware, and it uses interoperable wire format. This format enables clients to communicate with message broker to enable easy and widespread messaging interoperability. This communication is irrespective of languages, platforms and brokers used in the architecture. STOMP protocol provides message header with properties and frame body like AMQP. It is a simple and light weight with wide range of language bindings. Some transactional semantics are also provided by it.

Since MQTT is mainly used in devices which are economical and requires less power and memory, we decided to choose MQTT as the IOT data protocol for the low power development board.

Benefits of MQTT

The MQTT protocol has become a standard for IoT data transmission because it delivers the following benefits:

Lightweight and efficient

MQTT implementation on the IoT device requires minimal resources, so it can even be used on small microcontrollers. For example, a minimal MQTT control message can be as little as two data bytes. MQTT message headers are also small so that you can optimize network bandwidth.

Scalable

MQTT implementation requires a minimal amount of code that consumes very little power in operations. The protocol also has built-in features to support communication with many IoT devices. Hence, you can implement the MQTT protocol to connect with millions of these devices.

Reliable

Many IoT devices connect over unreliable cellular networks with low bandwidth and high latency. MQTT has built-in features that reduce the time the IoT device takes to reconnect with the cloud. It also defines three different quality-of-service levels to ensure reliability for IoT use cases— at most once (0), at least once (1), and exactly once (2).

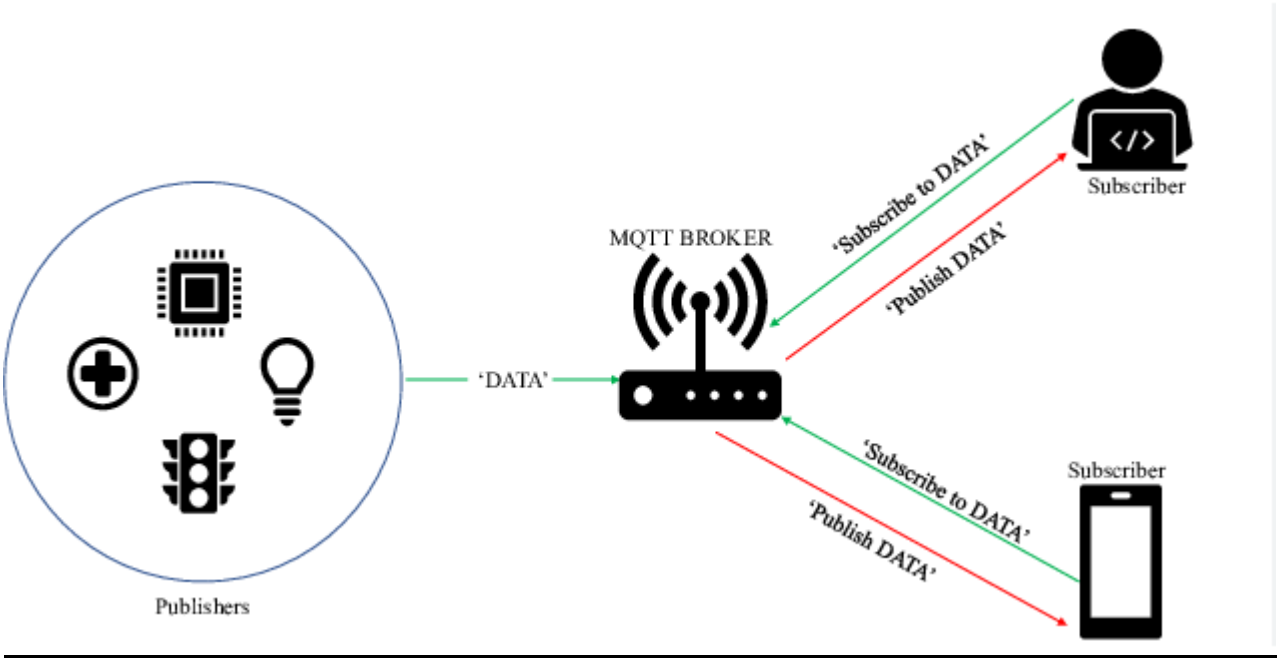
Secure

MQTT makes it easy for developers to encrypt messages and authenticate devices and users using modern authentication protocols, such as OAuth, TLS1.3, Customer Managed Certificates, and more.

Well-supported

Several languages like Python have extensive support for MQTT protocol implementation. Hence, developers can quickly implement it with minimal coding in any type of application.

How does MQTT work?



MQTT clients publish (Publisher) messages to a topic (e.g., 'DATA') through the MQTT server (Broker) over TCP. (Transmission Control Protocol (TCP) is a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks.) Topics are subscribed by other clients (Subscribers) as illustrated in the above figure. Then all the clients (Subscribers) subscribed to that topic will receive the message. Publishers and subscribers do not know each other's addresses and every message contains a topic(subject) where clients can subscribe to more than one topic and receive published messages. The MQTT broker plays a critical role as a receiver or as a server. The MQTT broker receives messages from the publisher and forwards these messages to the subscribers. The MQTT broker uses the list of topics to filter the MQTT clients that will receive the message. In essence, the topic creates a virtual channel between the publisher to its subscriber. MQTT is an asynchronous protocol (Asynchronous Messaging is a communication method where participants on both sides of the conversation have the freedom to start, pause, and resume conversational messaging on their own terms, eliminating the need to wait for a

direct live connection (aka synchronous messages).); it does not block the client while it waits for the message.

What is ThingsBoard?

ThingsBoard is an open-source IoT platform that enables rapid development, management, and scaling of IoT projects. It enables server-side infrastructure for IoT applications.

Uses of ThingsBoard

Provision devices, assets, and customers, and define relations between them.

Collect and visualize data from devices and assets.

Analyze incoming telemetry and trigger alarms with complex event processing.

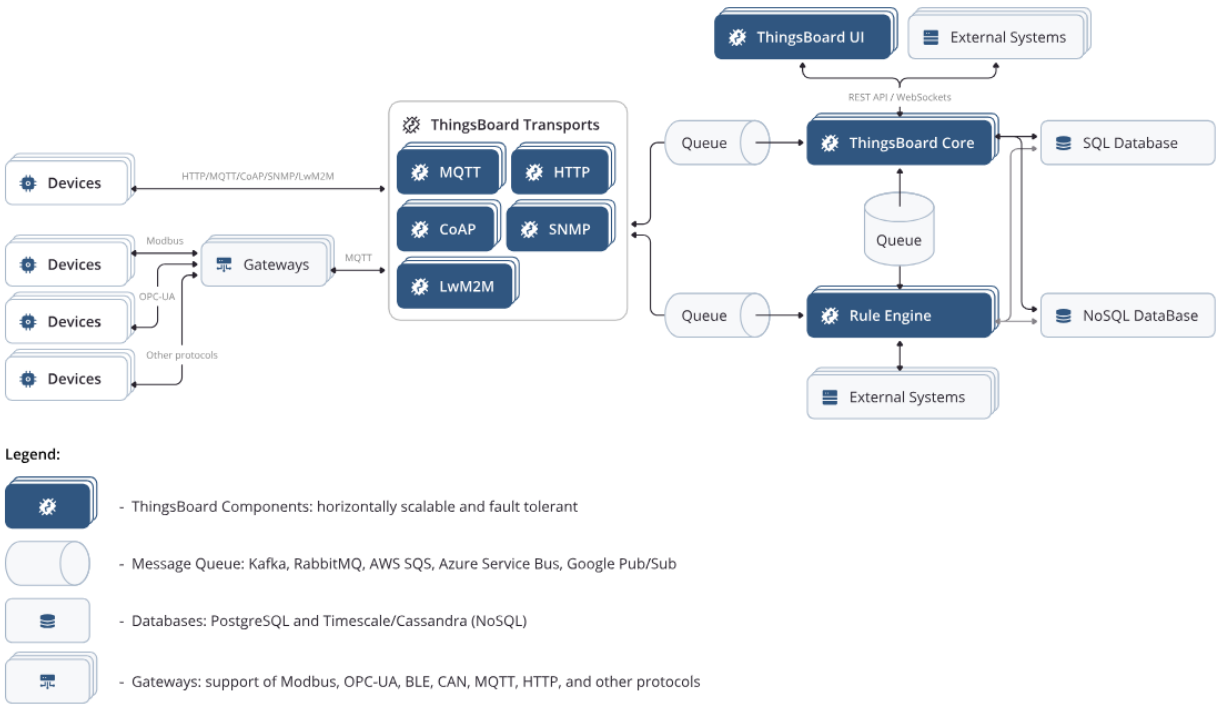
Control your devices using remote procedure calls (RPC).

Build workflows based on a device life-cycle event, REST API event, RPC request, etc.

Design dynamic and responsive dashboards and present device or asset telemetry and insights to your customers.

Enable use-case specific features using customizable rule chains.

Push device data to other systems.



ThingsBoard is designed to be:

scalable: horizontally scalable platform, build using leading open-source technologies.

fault-tolerant: no single-point-of-failure, every node in the cluster is identical.

robust and efficient: single server node can handle tens or even hundreds of thousands of devices depending on use-case. ThingsBoard cluster can handle millions of devices.

durable: never lose your data. ThingsBoard supports various queue implementations to provide extremely high message durability.

customizable: adding new functionality is easy with customizable widgets and rule engine nodes.

I was given the task of handling the scenario where ThingsBoard cloud is used as a way of displaying data to the users of the devices which are created by the digital lab. You can refer to the separate report which is only about the ThingsBoard cloud.