The Embedded Systems for IoT Devices

Neven Nikolov
Faculty of Computer Systems and Technologies
Technical University of Sofia, 8 Kl. Ohridski Blvd
Sofia 1000, Bulgaria
{n.nikolov@tu-sofia.bg}

ABSTRACT: *We have studied the embedded systems and the difference in the communication protocols used in IoT devices. We then detailed the characteristics of each feature contained in it. The advantages and disadvantages and the features such as security, power consumption and usage are also addressed in this work.*

Keywords: Embedded Systems, IoT, Protocol, Cloud

Received: 9 November 2020, Revised 24 February 2021, Accepted 10 March 2021

DOI: 10.6025/jcl/2021/12/2/52-58

1. Introduction

There are exists so many standards and protocols for IoT devices. IoT is used any were, like at in industry, smart homes, military and every were. IoT can read sensors, control motors, machines, relays and everything. IoT devices must communicate between them, and IoT devices must send and collect data to the IoT Server/Cloud. There are exist various IoT protocols. Some of them and most used are described on this article.

2. IoT Protocols

Higher-level protocols for the Internet of Things (IoT) offer various features that make them suitable for a broad range of applications. IoT protocols have various features and offer different capabilities. Most of these protocols were developed by specific vendors, and these vendors typically promote their own protocol choices.

2.1 MQTT

This is Message Queuing Telemetry Transport. MQTT is a publish/subscribe messaging protocol designed for lightweight M2M communications Figure 1. It was originally developed by IBM and is now an open standard. Architecture of MQTT has a client/server model, where every sensor is a client and connects to a server. The server is known as a broker over TCP. MQTT is message oriented, and every message is a discrete chunk of data. Every message is published to an address. That is known as a topic.

Architecture of MQTT has a client/server model, where every sensor is a client and connects to a server. The server is known as a broker over TCP. MQTT is message oriented, and every message is a discrete chunk of data. Every message is published to an address. That is known as a topic.

Clients may subscribe to multiple topics. Every client subscribed to a topic receives every message published to the topic. The MQTT protocol overview is shown on Figure 2. There is showing Client A, B, C and Broker. For a later time, Client A publishes a value. The broker forwards the message to all subscribed clients.

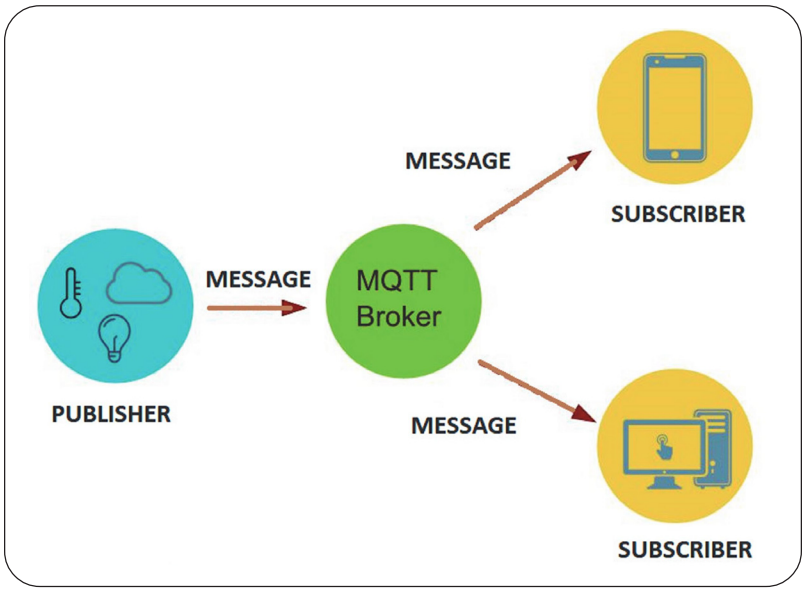

Figure 1. MQTT Publish/ subscribe messaging protocol

The publisher subscriber model allows MQTT clients to communicate one-to-one, one-to-many and many-to-one. In MQTT topics are hierarchical, like a filing system. Wildcards are allowed when registering a subscription allowing whole hierarchies to be observed by clients.

MQTT supports three quality of service levels. There was “Fire and forget”, “delivered at least once” and “delivered exactly once”. MQTT clients can register a custom “last will and testament” message to be sent by the broker if they disconnect.

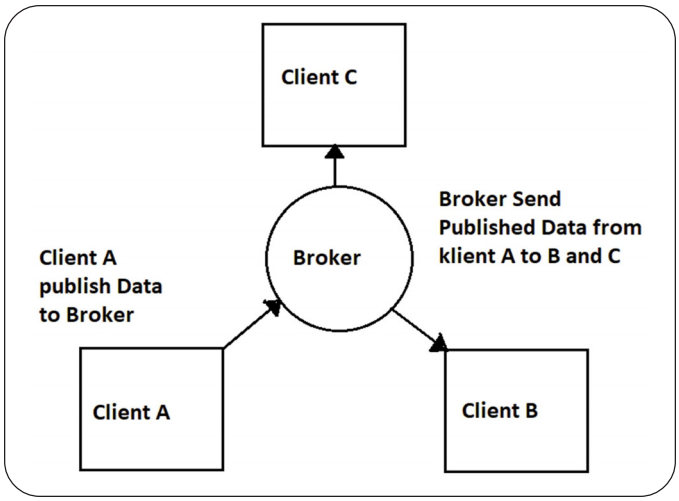

Figure 2. The publisher subscriber model

These messages can be used to signal to subscribers when a device disconnects. MQTT has support for persistent messages stored on the broker. When publishing messages, clients may request that the broker persists the message. Only the most recent persistent message is stored. When a client subscribes to a topic, any persisted message will be sent to the client. Unlike a message queue, MQTT brokers do not allow persisted messages to back up inside the server.

For security MQTT brokers may require username and password authentication from clients to connect. To ensure privacy, the TCP connection may be encrypted with SSL/TLS.

2.2 CoAP

This is Constrained Application Protocol. Like HTTP, CoAP is a document transfer protocol Figure 3. Unlike HTTP, CoAP is designed for the needs of constrained devices.

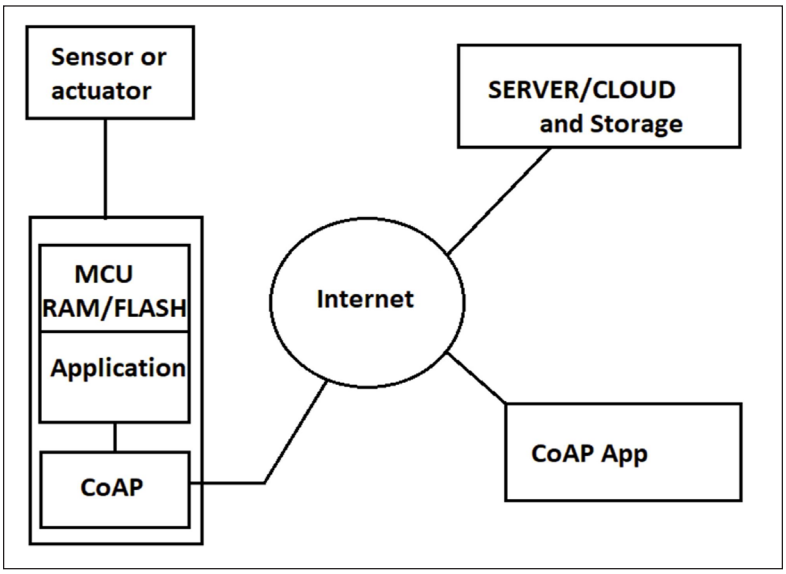

Figure 3. CoAP (Constrained Application Protocol)

CoAP packets are much smaller than HTTP TCP flows. Bitfields and mappings from strings to integers are used extensively to save space. Packets are simple to generate and can be parsed in place without consuming extra RAM in constrained devices. CoAP runs over UDP, not TCP. Clients and servers communicate through connectionless datagrams. Retries and

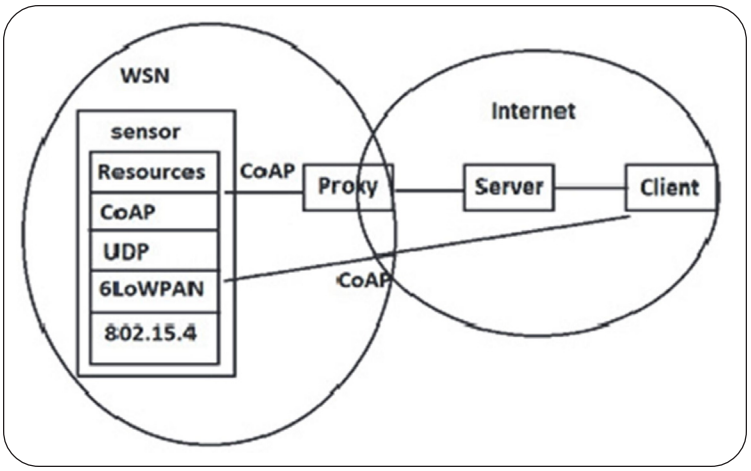

Figure 4. CoAP and RESTful

reordering are implemented in the application stack. Removing the need for TCP may allow full IP networking in small microcontrollers. CoAP allows UDP broadcast and multicast to be used for addressing. CoAP follows a client/server model. Clients make requests to servers, servers send back responses. Clients may GET, PUT, POST and DELETE resources. CoAP is designed to interoperate with HTTP and the RESTful web at large through simple proxies Figure 4.

For security CoAP is built on top of UDP not TCP, SSL/TLS are not available to provide security. DTLS, Datagram Transport Layer Security provides the same assurances as TLS but for transfers of data over UDP. Typically, DTLS capable CoAP devices will support RSA and AES or ECC and AES.

2.3 XMPP

This is Extensible Messaging and Presence Protocol for message-oriented middleware based on XML (Extensible Markup Language). This is open technology for real-time communication, which powers a wide range of applications including instant messaging, presence, multiparty chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data Figure 5.

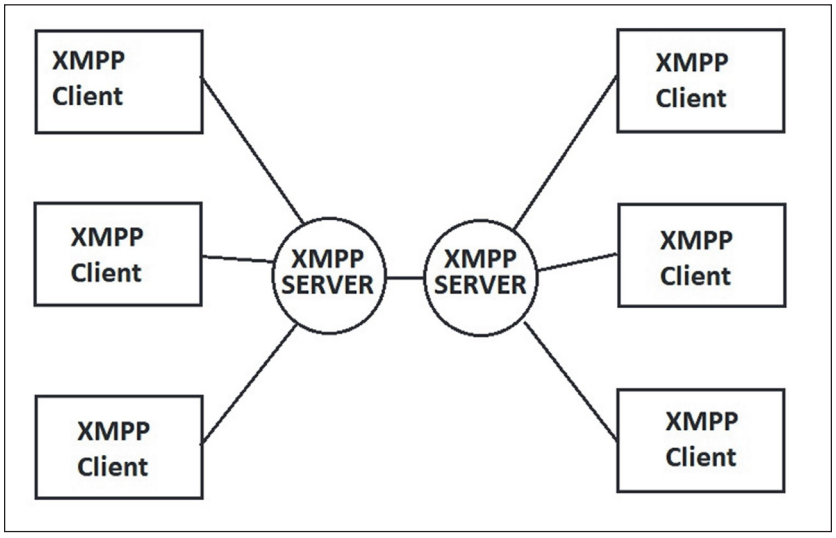

Figure 5. XMPP architecture

2.4 AMQP

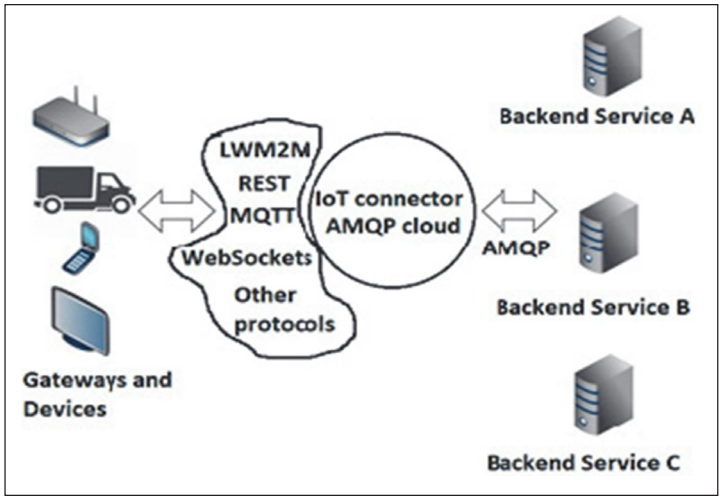

Figure 6. AMQP

This is Advanced Message Queuing protocol. AMQP is open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security. Devices connected to the IoT system have to connect to a kind of centralized hub that allows them to exchange their data with the other devices and backend services. The device that can't be properly connected to the rest of the application ecosystem, is useless from the IoT point of view Figure 6.

2.5 HTTP

This is Hypertext Transfer Protocol. HTTP and web sockets are common existing standards, which can be used to deliver XML or JavaScript Object Notation (JSON) in the payload. JSON provides an abstraction layer for Web developers to create a state full Web application with a persistent connection to a Web server. HTTP is the foundation of the client-server model used for the Web. The more secure method to implement HTTP is to include only a client in your IoT device, not a server. In other words, it is safer to build an IoT device that can only initiate connections, not receive. After all, you do not want to allow outside access to your local network. HTTP is defined the GET , POST, PUT, DELETE and more methods.

3. Advantages and Disadvantages of IoT Protocols

Protocol	Advantages	Disadvantages
MQTT	- Good for low battery consumption. - Lightweight API requires minimal processing on a device - Message header can be as small as two bytes. This makes it very bandwidth efficient, ideal for spotty coverage or limited networks - Supports the major IoT message patterns: publish/subscribe and request/reply - MQTT-SN supports topic ID instead of topic name and UDP, ZigBee, Bluetooth and other wireless protocol	- No message queue support (i.e., only the most recent message is stored in a message broker). - No support for such header fields as TTL (time-to-live), reply To and user properties. - It has no section for message properties.
CoAP	- Has the same strengths as REST except for TCP. - Very fast device-to-device communication in UDP.	- Has the same weaknesses as REST except for quality of service levels. - Offers "confirmable" and "non-confirmable" quality of service. - Supports only request-reply message exchange pattern.
XMPP	- XMPP stands for Extensible Messaging and Presence Protocol - uses the XML text format as its native type, making person-toperson communications natural - runs over TCP, or perhaps over	
AMQP	- Support for most message exchange patterns including publish-subscribe, requestreply and message queue. - Support for all classes of service. - Support for detailed header fields such as TTL, replyTo and user properties - Enables portable encoding of messages. - Supports both TCP and UDP.	- Power, processing and memory requirements for a device are relatively high. - Its required header fields are rather long.
HTTP	- Does not require a client library on the device. - Simplifies the architecture if device data loss is acceptable - Provides "lowest common denominator" connectivity, since most devices can use HTTP POST or GET.	- Its header fields are relatively long (if network bandwidth matters) - No support for quality of service levels. - No support for varied message patterns. - The application needs to handle all reliability.

Table 1. Advantages and disadvantages IoT protocols

MQTT, CoAP, XMPP, AMQP and HTTP are useful as IoT protocols, but they have fundamental differences. There are used for connection between IoT device to IoT device, IoT device to Server/Cloud. Each of them is made for specific purpose and they have advantages and disadvantages. IoT protocols are compared in Table 1.

Protocol	Architecture	Usage	Resources	Transport
MQTT	Tree	IoT msging	10Ks/RAM flash	TCP
CoAP	Tree	utility field area	10Ks/RAM flash	UDP
XMPP	Client Server	high Manditory	10Ks/RAM flash	TCP
AMQP				
HTTP	Client Server	Smart Energy	10Ks/RAM flash	TCP

Protocol	Messaging	2G,3G,4G	Low Power	Security
MQTT	Pub/Subsrb	Excellent	Good	Medium
CoAP	Rqst/Rspnse	Excellent	Excellent	Medium
XMPP	Pub/Subsrb	Excellent	Fair	High
AMQP				
HTTP	Rqst/Rspnse	Excellent	Fair	Low

Table 2. Features of iot protocols

4. Conclusion

The Internet of Things covers a huge range of industries and use cases that scale from a single constrained device up to massive cross-platform deployments of embedded technologies and cloud systems connecting in real-time. Tying it all together are numerous legacy and emerging communication protocols that allow devices and servers to talk to each other in new, more interconnected ways.

References

[1] Rowe, K. (2014). *Internet of Things Requirements and Protocols*, www.embedded-computing.com, (February, 21).

[2] Kwon, Y. (2017). *Understanding IoT Protocol – Matching your Requirements to the Right Option*, www.solace.com, (January 25).

[3] Schneider, S. (2013). *Understanding the Protocols behind the Internet of Things*, www.beta.electronicdesign.com, (October 09).

[4] Components, R. S. (2015). *11 Internet of Things (IoT) Protocols You Need to Know About*, www.rs-online.com, (April 20).

[5] Jaffey, T. (2014). *MQTT and CoAP, IoT Protocols*, www.eclipse.org, (February).

[6] IoT Stantards and Protocols, www.postcapes.com.

[7] Kang, B., Choo, H. (2016). An Experimental Study of a Reliable IoT Gateway, (September 13).

[8] Amina, R., Kumara, N., Biswasb, G. P., Iqbalc, R., Changd, V. (2016). *A* light Weight Authentication Protocol for IoT-Enabled Devices *in Distributed Cloud Computing Environment*, (November 18).

[9] Atzori, L., Iera, A., Morabito, G. (2009). *The Internet of Things: A Survey*, December 10, 2009.

[10] Fernandes (Jr.), R. F., Brandao, D. (2016). *Proposal of Receiver* Initiated MAC Protocol for WSN in Urban Environment using *IoT*, (December 16).

[11] Woo, M. W., Lee, J. W., Park, K. H. (2017). *A Reliable IoT System for Personal Healthcare Devices*, (April 12).

[12] Ray, P. P. (2016). *A Survey of IoT Cloud Platforms*, (December).