A Review Paper on IoT Communication Protocol

S.S.VELUMANI
Field Support Engineer, Centre for Development of Telematics (C-DOT),
Telecom Technology Centre of Government of India, Bengaluru, Karnataka, India
(E-mail: velu@cdot.in)

Abstract**—**Internet of Things (IoT) covers a wide range of industries and use cases that scale from one constrained device to massive cross-platform deployments of embedded technologies and cloud systems. This paper describes the favored IoT communication protocols MQTT, AMQP, HTTP and CoAP. The IoT system will operate and transfer data between nodes once the devices are connected to the communication network. This paper will facilitate the reader to understand the various IoT protocols in brief and help them select a suitable one for their projects.

Keywords**—***Internet of Things, MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol), HTTP (Hyper Text Transfer Protocol), AMQP (Advanced Message Queuing Protocol), REST (Representational State Transfer)*

I. INTRODUCTION

A huge scheme of connected devices, named Internet of Things (IoT) is a rising technology and is the leading communication paradigm into the future. IoT will enable ‘sensors’ and ‘actuating devices’ to communicate across numerous platforms through a unified framework. The IoT standards and protocols are being developed and new ones are continuously evolving to be a part of the ever-expanding communication network.

The primary objective of this paper is to explain in brief the various IoT protocols namely MQTT, AMQP, HTTP and CoAP.

A. *Internet of Things (IoT)*

The IoT is actually a set of physical device(s) embedded with electronics, software, sensors, actuators and network connectivity that equip the device(s) to collect and exchange data between them and with others interested in their data.

The first device’s connected to the worldwide web in 1982 was by Coca-Cola coin machine that monitored the temperature of the machine and kept track of the quantity of bottles in it. During the early period of the 21st century, fast changes and development happened in the world of IoT. Currently IoT has grown to a level that not all existing protocols are able to satisfy its requirements and provide seamless communication. This is the reason why it is necessary to form specialized IoT communication protocols and standards.

II. MQTT (MESSAGE QUEUE TELEMETRY TRANSPORT)

MQTT is a simple and light-weight Publish/Subscribe messaging protocol that uses a Broker. This was originally developed by IBM and is now an open standard. It had been designed to attach sensor nodes over communication networks that have (a) low bandwidth and high-latency or (b) is unreliable or (c) is both. It runs on top of the TCP transport protocol, which ensures its reliability. Due to its simplicity, and a really small message header in comparison with other messaging protocols, it is the communication solution of choice in IoT.

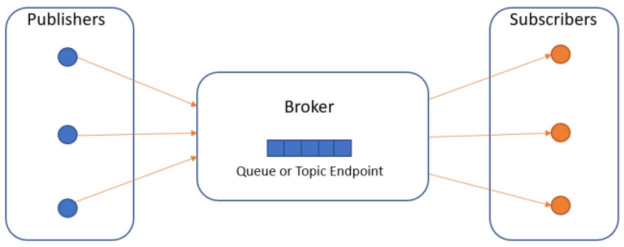


FIG 1: MQTT ARCHITECTURE

It is a Publish/Subscribe architecture as shown in Fig 1, where the system consists of three main components namely publishers, subscribers and a broker. In IoT point of view, the publishers are sensors that hook up with broker to send their data (whenever possible) and return to sleep mode. Subscribers are applications that have an interest in a certain topic/queue, of sensory data. For this, the subscribers hook up with brokers to be informed whenever new data is published by sensors. Broker is a central component that accepts messages from publisher’s and with the assistance of topic filtering, delivers them to the interested subscriber’s.

The message structure of MQTT is shown in Fig 2. MQTT offers the highest level of quality of service. It defines three levels of QoS

1. QoS 0: At most once delivery (Only TCP guarantee)

2. QoS 1: At least once delivery (guarantee with confirmation).

3. QoS 2: Exactly once delivery (guarantee with handshake).

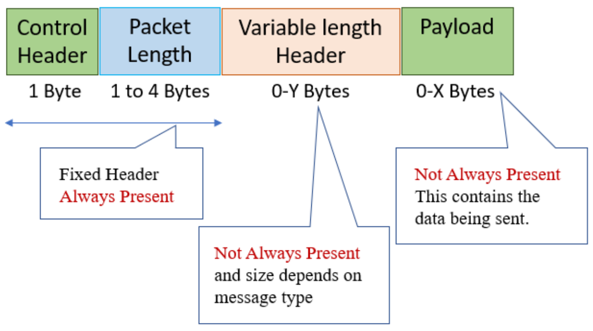

Control Header
Packet Length
Variable length Header
Payload
1 Byte
1 to 4 Bytes
0-Y Bytes
0-X Bytes
Fixed Header Always Present
Not Always Present and size depends on message type
Not Always Present This contains the data being sent.

FIG 2: MQTT MESSAGE STRUCTURE

MQTT has low level of security with minimal authentication feature. It assumes that the Publishers/Subscribers are online and because of this the messages are short lived.

The Highlights of MQTT:

a) This protocol is suitable for monitoring small sensor networks

b) This is ideal for networks that give data in small bursts occasionally. (i.e., in Non-Transaction mode, no fragmentation of messages)

c) There is no scope for sharing data among multiple subscribers (i.e., no multi-tenancy, since the messages are deleted as soon it is consumed by a subscriber)

III. AMQP (ADVANCED MESSAGE QUEUING PROTOCOL)

AMQP is an open standard protocol that follows the publish/subscribe design runs over transmission control protocol. It's designed to enable interoperability between a range of assorted applications and systems, in spite of their internal design. Originally it had been developed for business messaging with the thought of providing an answer which is able to manage an oversized amount of message exchanges may happen within short period of time. It's another session layer protocol that was designed for financial transaction purpose. The AMQP architecture is shown in Fig 3.

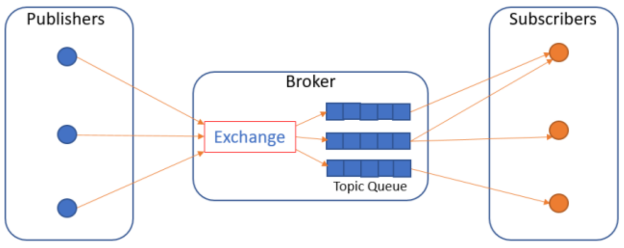


FIG 3: AMQP ARCHITECTURE

The AMQP architecture is analogous to MQTT, the distinction is that the broker is split into two main components, exchange and queues as shown in Fig 3. The exchange represents a part of the broker that is used to direct the messages received from publishers. After processing the messages received are routed into one or more appropriate queues. Queues essentially represent the topics that are signed up by subscribers which may get the device information whenever they are offered. It provides three completely different levels of QoS, same as MQTT. Additionally, the AMQP protocol provides complementary security mechanisms. For TLS protocol encryption for protecting data. and for authentication, it uses SASL (Simple Authentication and Security Layer).

There are four main message exchange types: a) direct b) topic c) fanout and d) header.

a) Direct Exchange: Message will be delivered based on exact matches between routing key (topic) and binding key (subscription). Example: Distribute jobs between workers

b) Topic Exchange: Message will be delivered based on match between routing key's (topic's) with binding key (subscription) using a wildcard. Example: Stock Prize update

c) Fanout Exchange: Broadcast message unconditionally to all queues. Example: Sports site update live match scores.

d) Header Exchange: Messages will be routed based on Header value. The header will have multiple attributes. Example: Transfer of work in a multistage workflow project.

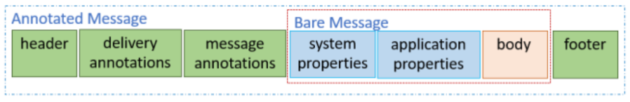


FIG 4: AMQP MESSAGE STRUCTURE

The AMQP message structure is shown in Fig 4.

The message is composed of two parts 1) Annotated Message 2) Bare Message.

Annotated Message: mutable message between sender and receiver. This mutable message may optionally have an immutable part too.

Bare Message: immutable message between sender and receiver.

This messaging protocol is used in the AADHAR project of Government of India.

The Highlights of AMQP:

a) this protocol is suitable for usage in monitoring large sensor networks

b) The data published can be shared among multiple subscribers (multi-tenancy is feasible)

c) The message structure allows support for transaction mode of communications.

IV. HTTP (Hyper Text Transper Protocol)

HTTP is a text protocol that allows for sending documents back and forth on the Web. Representational State Transfer (REST) is the architecture that uses HTTP protocol to provide functionalities depending on the requirements of software applications or Web pages in a very light-weight, simple and effective way. There is no specific size limitation on the message length.

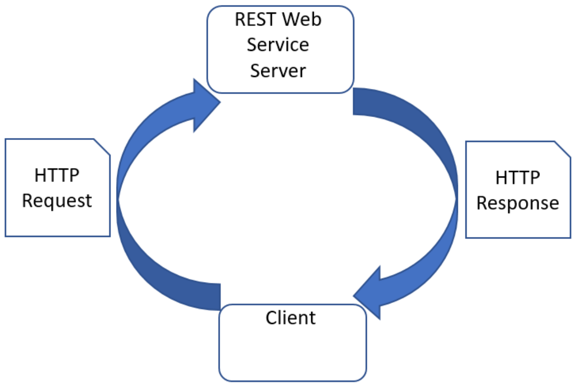


Fig 5: HTTP REST Architecture

The HTTP Rest Architecture is shown in Fig 5. HTTP communication between a client and a server happens via request/response messaging, with client sending an HTTP request message and server then returning a response message, containing the resource that was requested in case the request was accepted. The mixture of HTTP protocol and REST, the devices can build their state information easily. This state information is obtainable, due to the regular operations of create, read, update, and delete (CRUD database operations). For performing operations like creating, updating, reading and deleting resources, this protocol offers methods namely HTTP POST, GET, PUT and DELETE.

The Highlights of HTTP:

a. HTTP is a heavy protocol it needs more power and memory.

b. HTTP does not have Quality of Service, So, it relies on TCP to guarantee the message delivery.

c. It uses TLS/SSL to authentication and encrypt the messages.

V. CoAP (Constrained Application Protocol)

The CoAP is a session layer protocol designed by IETF to supply lightweight RESTful (HTTP) interface. REST is the standard interface between HTTP client and servers. However, for lightweight applications like IoT, REST could be burdened with significant overhead and power consumption. CoAP is developed to enable low-power sensors to use RESTful services while meeting their power constrains. It's designed over UDP, instead of TCP which is commonly used in HTTP and features a light mechanism to supply reliability. The CoAP architecture is shown in Fig 6.

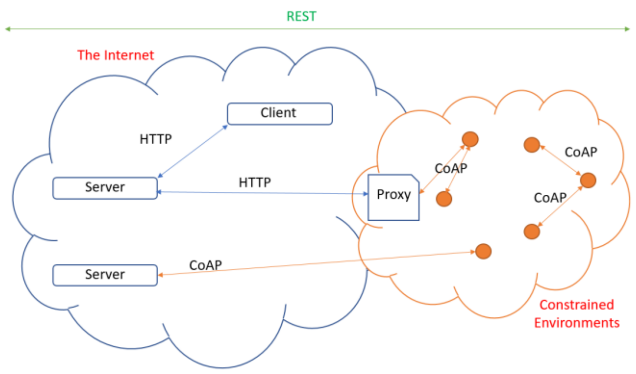
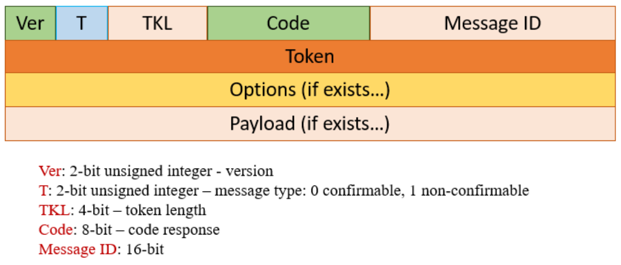


Fig 6: CoAP Architecture

This protocol depends on a structure that is divided into two logically different layers. The primary one known as request/response GET, PUT, POST or DELETE layer, implements RESTful paradigm and allows for CoAP clients to use the HTTP-like methods for sending requests. The CoAP support CRUD operation with HTTP methods namely GET, PUT, POST or DELETE that use known URI's. The sensor makes its data available to the network through specific URI methods. Similar to HTTP Request/Response mechanism, when a client needs to fetch any sensors data, it will initiate the GET method with the relevant server URL and the sensor will respond to this request with the data in its response packet. In the same way the sensor may use relevant URL to POST its current data to update a client or tool. In CoAP, the request and responses are matched through a token that is unique for each communication. All communications among the clients/sensors happen necessarily through tokens. The token brings reliability to the communication.



Ver	T	TKL	Code	Message ID
Token				
Options (if exists...)				
Payload (if exists...)				

Ver: 2-bit unsigned integer - version
T: 2-bit unsigned integer – message type: 0 confirmable, 1 non-confirmable
TKL: 4-bit – token length
Code: 8-bit – code response
Message ID: 16-bit

Fig 7: CoAP Message Structure

CoAP has the following messaging modes a) confirmable, b) non-confirmable, c) acknowledgement and d) separate.

a) In confirmable mode the message elicits exactly one return message of type acknowledgement.

b) In non-confirmable mode the message doesn't need an acknowledgement.

c) In acknowledgement mode the client/server involve in direct communication where for each request an acknowledgement message that is sent to confirm receipt of the request. This is followed by for the response to the request.

d) In The separate mode, for each request between the server/client the acknowledgment is sent as part of the message response itself to avoid delays due to re-transmission.

The CoAP message structure is shown in Fig 7.

The Highlights of CoAP:

1) The CoAP is ideal for communication among resource-limited devices.

2) It is efficient with usage ofpower efficiency and consumes less memory.

3) CoAP addresses, security i.e encrption and authentication through DTLS and IPsec protocols respectively.

4) It has low message overhead, hence fragmentation of message has to be avoided.

5) CoAP by default depends on client system driven congestion control mechanism.

VI. SUMMARY

In this paper, we have provided a comprehensive outlook of MQTT, AMQP, HTTP and CoAP protocols used for IoT. Many such protocols are developed by IETF, IEEE, ITU, and other organizations and lots more are in development.

MQTT is a light binary protocol and works smoothly in low bandwidth. Based on CPU capacity MQTT can be scalable to connect thousands or even millions of devices. This protocol is widely used in consumer electronics, automotive and industrial applications. The minimum overhead feature makes MQTT very attractive.

AMQP is also a light binary protocol with low overhead. It will emphasize guaranty of message delivery, till then it holds the message on queue. The architecture is publish/ subscribe and supports two levels of QoS.

HTTP is a heavy text protocol with high overhead. The architecture is based on Request/ Response and the message size depends on server capacity. This does not provide QoS as a feature and relies on TCP to ensure message delivery.

TABLE I. TABLE STYLES

Protocols	UDP/ TCP	Architecture	Security	QoS	License	Protocol Type	Memory/ Power Consumption
MQTT	TCP	Publish/ Subscribe	TLS/SSL	3	Open Standard	Binary	Low
AMQP	TCP	Publish/ Subscribe	TLS SNI/ SASL	2	Open Standard	Binary	Low
HTTP	TCP	Request/ Response	TLS/SSL	-	Open Standard	Text	High
CoAP	UDP	Publish/ Subscribe and Request/ Response	DTLS/ IPsec	-	Open Standard	Binary	Low

CoAP is a light binary protocol with low overhead. The architecture supports two modes such as Publish/ Subscribe and Request/ Response. This does not support QoS as a feature. It has a small message size and ensures reliable communication using tokens.

In the Table 1 consolidated details of various protocols discussed is listed. Because of the different architectural possibilities, each protocol performs differently in different segments and is thus suited for different types of applications. Implementation stability is a key factor for choosing an appropriate protocol choice for system developers. The aim of this paper is to give an insight to developers and service providers of IoT.

VII. GLOSSARY

MQTT - Message Queue Telemetry Transport

CoAP - Constrained Application Protocol

HTTP - Hyper Text Transfer Protocol

AMQP - Advanced Message Queuing Protocol

REST - Representational State Transfer

TCP – Transmission Control Protocol

TLS – Transport Layer Security

SSL – Secured Sockets Layer

SNI – Server Name Indication

DTLS – Datagram Transport Layer Security

IPsec – Internet Protocol Security

UDP – User Datagram Protocol

IETF – Internet Engineering Task Force

IEEE – Institute of Electrical and Electronics Engineers

ITU – International Telecommunication Union

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

SASL - Simple Authentication and Security Layer

REFERENCES

[1] D. Locke, "MQ telemetry transport (MQTT) v3. 1 protocol specification," IBM Developer Works Technical Library, 2010, http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html

[2] "OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0". OASIS, 2012, http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols and applications," IEEE Communications Surveys Tutorials, vol. PP, no. 99, 2015, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=7123563.

[4] J. Granjal, E. Monteiro, and J. Sa Silva, "Security for the internet of things: A survey of existing protocols and open research issues," IEEE Communications Surveys Tutorials, vol. 17, no. 3, pp. 1294-1312, 2015, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7005393.

[5] https://www.w3.org/Protocols/

[6] Constrained Application Protocol (CoAP) "draft-ietf-core-coap-18", IETF, June 28, 2013, https://tools.ietf.org/html/draft-ietf-core-coap-18.

[7] Leonard Richardson and Sam Ruby. RESTful web services. O'Reilly Media, May 2007.

[8] W. Drytkiewicz, I. Radusch, S. Arbanowski, and R. Popescu-Zeletin. pREST: a REST-based protocol for pervasive systems. In Proc. of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pages 340–348. IEEE, 2004.

I am S.S. Velumani working as Field Support Engineer in Validation and Field Support Group of Centre for Development of Telematics (C-DOT), Telecom Technology Centre of Government of India, Bengaluru, Karnataka, India. I have an expertise in Complete Software Development Life Cycle, Web Development, Database Administration, System Administration.