International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-6 Issue-5, June 2017

Comparison of Application Protocols for Resource Constrained Devices and WSN

Kamal Grover

Abstract: ***This paper explains the need for the application protocols needed for wireless sensor network. It explains and compares MQTT and CoAP based on different attributes like architecture, reliability, packet delay, security, quality of service and fragmentation. Finally, it's concluded that choice of protocol is dependent on type service they perform***

Index Terms*: *Application, Protocols, Wireless

I. INTRODUCTION

Ongoing innovations in the field of communication technology and digital environment have led to the deployment of the internet of things across the world. Device to device communication is coming up. These devices require wireless sensors which are used in almost every appliance today like cars, traffic lights, medical devices, and smartphones. However, the communication for these sensors must be considered wisely because they are resource constrained devices. Optimization of power and bandwidth usage is required for the ubiquitous environment. To accomplish this, two major lightweight application protocols which are MQTT (Message queue telemetry transport) and CoAP (Constrained Application Protocol) are used. These protocols give better performance in terms of bandwidth consumption, battery lifetime and communication latency.

Even the same scenario applies to the wireless sensor networks, it consists of sensors nodes, and gateways, these devices will be having limited resources, this paper sheds light on WSN's communication resources optimization with comparing different feature of MQTT and CoAP protocols and present the work conducted to understand existing issues and gaps in this domain.

Wireless sensors require resource constrained protocols to operate properly. This paper compares two popular protocols used in WSN. Section 2 explains the protocols briefly. Section 3 compares them based on different attributes. Finally, section 4 concludes the paper.

Manuscript published on 30 June 2017.
***** Correspondence Author (s)
Kamal Grover*, Department of Computer Science and Engineering, National Institute of Technology Jalandhar, India, E-mail: kamalgrover758@gmail.com

II. PROTOCOLS

A. MQTT

Message Queue Telemetry Transport was introduced by IBM in 1999. It is lightweight bi-directional communication. protocol that is intended to be used in an IoT context to enable the communication between a device and a server using broker [1]. It has limited memory capabilities and used to send data over low bandwidth networks. MQTT protocol uses TCP to ensure the reliability the packets transmission and requires a persistent connection between the server and the client. It has less overhead than HTTP.

It Uses of the publish/subscribe message architecture which provides one-to-many message distribution and decoupling of applications [2]. MQTT uses three QOS levels [3]. QoS level 0 means that a message is delivered at most once and no acknowledgment of reception is required. The message can be lost in this case. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after. QoS level 1 means that every message is delivered at least once and confirmation of message reception is required. In this case, message reception is confirmed but duplicates can occur. In QoS level 2, a four-way handshake mechanism is used for the delivery of a message exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied [3].

B. CoAP

Constrained Application Protocol is a specialized web trans-fer protocol for use with constrained nodes and constrained (e.g., low-power) networks [4]. It was developed by IETF CoRE Working Group in March 2010. It is based on Representational state transfer architecture and supports request-response model. It has asynchronous message exchange. CoAP provides its own reliability mechanism ie confirmable messages and non-confirmable messages. It has four types of messages Confirmable, Non-Confirmable, Acknowledgement and Reset (i.e. Nack) [5]. Since TCP has overhead in small transactions, CoAP is building on UDP, which has lower overhead. It uses the short fixed-length compact binary header of 4 bytes [6]. The message format is depicted in Figure II.1.

It ensures security by using it on top of Datagram Transport Layer Security (DTLS). It is expected that CoAP will deploy a wider variety of key management options for TLS [7].

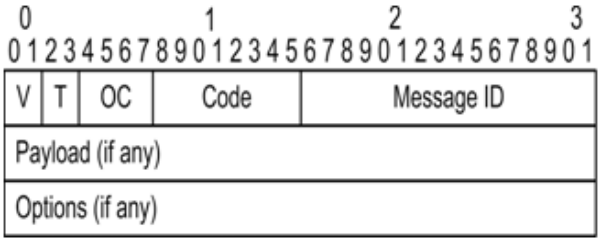

Fig. II.1. Message format of CoAP [6]

III. COMPARISON OF PROTOCOLS

A. Architecture

MQTT has Publish-Subscribe Architecture figure III.1. The principle of this communication model is that components which are interested in some information register. This process is called subscription, the interested party a subscriber. Components which want to produce certain information do so by publishing their information. They are publishers. The component which ensures that the data gets from the publishers to the subscribers is the broker. The broker coordinates subscriptions, and subscribers must contact the broker explicitly to subscribe [8]. Publishers are loosely coupled to subscribers. This provides the opportunity for better scalability than traditional client and server, through parallel operation, message caching, tree-based or network-based routing, etc. However, in certain types of tightly coupled, scalability for pub/sub products under high load is a research challenge. However, the most serious problems with pub/sub systems are a side-effect of their main advantage: the decoupling of the publisher from the subscriber.

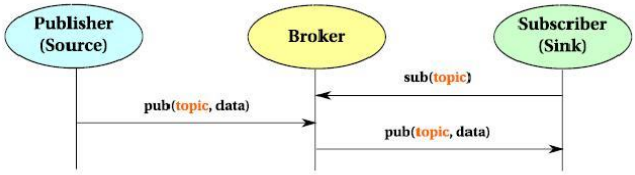

Fig. III.1. Pub/Sub Communication Model

CoAP uses the REST architectural style Figure III.2. It is based on UDP and unencumbered by historical baggage, how-ever, CoAP aims to achieve its modest goals with considerably less complexity [10]. This twofold nature of CoAP makes it a more flexible solution for application developers. It has RESTful architecture and is using a subset of the HTTP verbs to manipulate resources, CoAP enables a seamless integration with most web applications through HTTP-CoAP proxies [7].

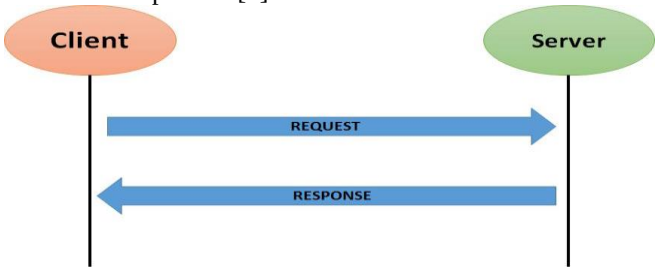

Fig. III.2. Request Response model in CoAP

B. Reliability

To ensure the reliability of messaging, MQTT supports 3 levels of Quality of Services(QoS) [8]. Figure III.3 represents mechanisms taken place in these 3 QoS levels. QoS Level 0 sends a message only once and does not check whether the message arrived at its destination or not. Therefore, in case, it is possible that the message will be lost in the way. QoS Level 1 sends the message at least once and checks the delivery status of the message by PUBACK. However, when PUBACK is lost, it is possible that the server will send the same message twice since it has no confirmation of the message being delivered in return this may lead to congestion. QoS Level 2 passes the message through exactly once. It is not possible to have a message loss in this level, but due to the complicated process of 4-way handshake, it is possible to have relatively longer end-to-end delays and congestions [11]. Higher QoS level would mean many exchange of packets and these complicated processes will increase the end-to-end delay and congestion in the network. However, this will ensure reliability but it has some trade-offs too. Graph III.4 represents bandwidth in different levels of QoS.

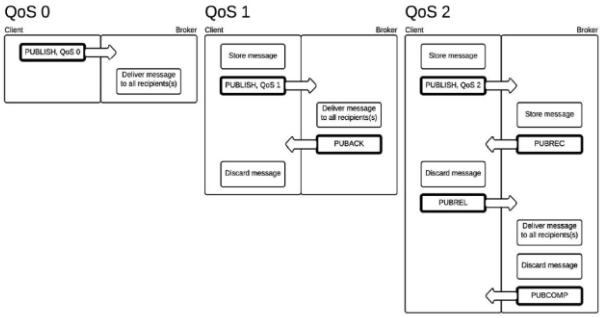

Fig. III.3. Sequence for publishing at all QoS levels [1]

CoAP ensures reliability by confirmable messages, non-confirmable messages, acknowledgments, and retransmissions [3]. A message can be labeled as Confirmable (CON), which implies that it is retransmitted up to a maximum number of times by using a default response timeout and to schedule retransmissions until the recipient sends either an Acknowledgement (ACK) message. In another case, a message is Non-Confirmable (NON) where no Acknowledgement message is sent back to the transmitter Figure III.4.

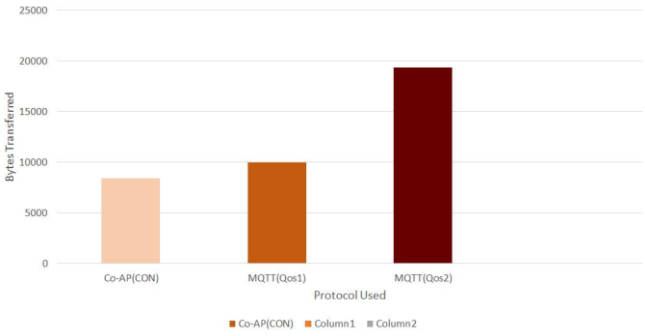

Fig. III.4. Bandwidth Usage different reliability scenarios [10]

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

C. Packet delay

The packet loss in any network leads to retransmission of messages and thus leading to longer delays in the message reception [3]. This happens in MQTT gateway QoS 1 and confirmable messages of CoAP. These two settings are similar in terms of the transmission and acknowledgment of a mes-sage. Graph III.5 plots the delay as cumulative distribution function (CDF). It can be seen from the graph that messages experienced a lower delay in MQTT for lower values of packet loss. But as the packet loss increased, CoAP performed better than MQTT in terms of delay. This happens because of the greater TCP overheads involved in the retransmission of messages compared to the smaller UDP overheads in CoAP.

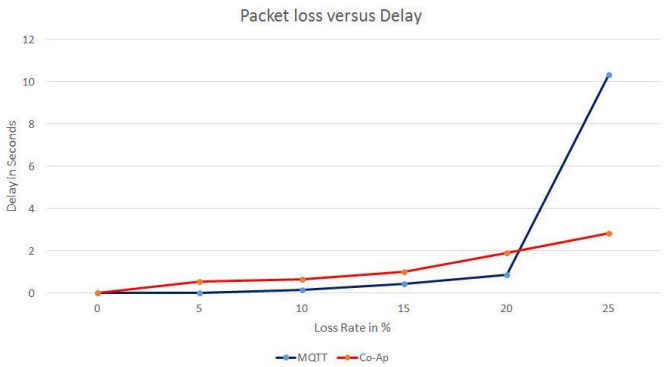

Fig. III.5. Packet Loss rate versus delay [3]

D. Underlining protocol

MQTT relies on TCP, whereas CoAP relies on UDP. MQTT and CoAP inherit from TCP and UDP their advantages and disadvantages in terms of functionality and performance. MQTT becomes more persistent and reliable because of TCP whereas there are more chances of losing message quickly and overload in CoAp.

E. Security

To enhance security in MQTT, SMQTT is used. SMQTT is a secure protocol. As shown in figure III.6 there are three entities are used in it: (i) Publisher device which publishes the data on the given topic. (ii) Subscriber device receives the data under the same topic through a Broker. (iii) PKG or broker is the trusted third party. There are four phases in the protocol. In the setup phase, registration of publisher and subscriber driver is done and PKG generates Master Secret key. During encrypt phase, data is encrypted using public parameters and in publishing phase, Publisher publishes encrypted data and sends it to the broker which in return forwards the message to all Subscribers. In decrypt phase, data is decrypted by subscribed devices using private attribute keys. This phase ensures security [12].

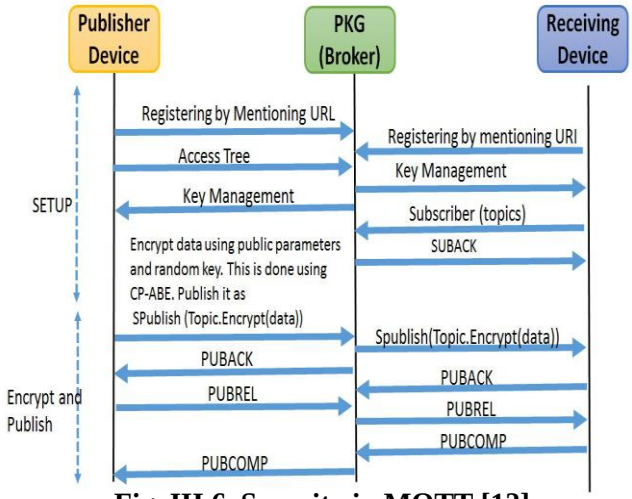

Fig. III.6. Security in MQTT [12]

In the case of CoAp, DTLS is incorporated in its architecture to minimize communication overhead and ROM occupation. The Datagram Transport Layer Security (DTLS), is the UDP-based version of TLS. It is designed to provide security. Figure III.7 represents the mechanism taken place in DTLS. The client starts the DTLS handshake with the server by sending Hello message. The server takes the decision of which cryptographic algorithms to use and correspondingly sends it response. The handshake then proceeds with an exchange of information, such as Certificates and Key Exchanges, required establishing a common secret. A long message, Change Cipher Spec, then informs the other party to switch to the encrypted mode using the algorithms. Finally, the negotiation process ends verifying the previous messages exchanged which contains a message authentication code (MAC) computed based on all the previously sent and received handshake messages as seen by each peer. The subsequently [13] exchanged data are encrypted using the session key. CoAP proposes to use DTLS protocol to provide end-to-end security, requiring the support of No Sec mode when DTLS is disabled.

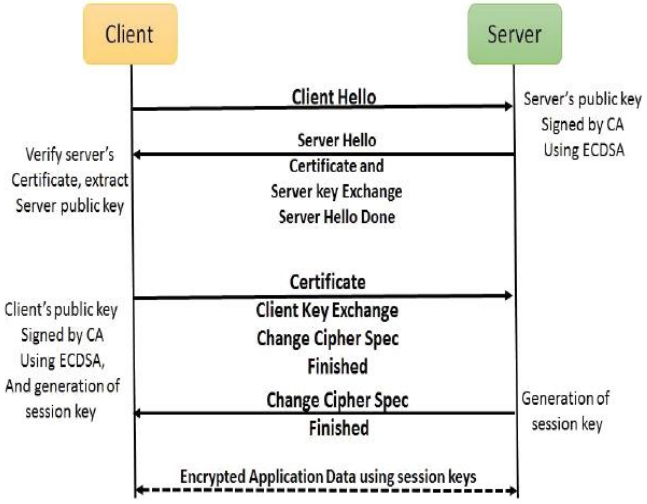

Fig. III.7. Security on CoAP [13]

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

F. Fragmentation

Fragmentation is driving the large messages into smaller ones so that these messages can be sent through network easily. MQTT does not allow fragmentation. This could Hamper's transmission of large messages. CoAp has a block-wise mechanism. It Transfers the larger data in a block-wise fashion.

TABLE I. Comparison of MQTT and COAP

Attribute	MQTT	CoAP
Architecture	Publish Subscribe	Request-Respose
Reliability	Three level QoS	CON and NON messages
Underlining protocol	TCP	UDP
Security	SMQTT	Datagram Transport Layer Security
Fragmentation	Does not allow message fragmentation	Block wise mechanism

IV. CONCLUSION

It seems hard to decide which protocol is the best among MQTT and CoAP as it depends on the application requirements. If the application is sending multiple small messages very frequency and requires the interaction with web services CoAP should be used. If the application needs the best QOS, sends big messages and there is no web interaction, MQTT is best suitable. In future works Adaptively changing protocol can be de-veloped. In this case, the protocols can be changed based on network conditions. The gateway can detect the network condition and then choose the protocol which can be used.

REFERENCES

1. Olivier Deschambault*, Abdelouahed Gherbi*, and Christian Lgar, "Efficient implementation of the mqtt protocol for embedded systems," Journal of Information Processing Systems, vol. 13, pp. 28–39, Feburary 2017.
2. Mqtt version 3.1.1. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf
3. D. Thangavel, X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan, "Performance evaluation of mqtt and coap via a common middleware," in 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), April 2014, pp.1–6.
4. Mqtt version 3.1.1. [Online]. Available: https://tools.ietf.org/html/rfc7252
5. B. C. Villaverde, D. Pesch, R. D. P. Alberola, S. Fedor, and M. Boubekeur, "Constrained application protocol for low power em-bedded networks: A survey," in 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, July 2012, pp. 702–707.
6. W. Colitti, K. Steenhaut, N. D. Caro, B. Buta, and V. Dobrota, "Evaluation of constrained application protocol for wireless sensor networks," in 2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN), Oct 2011, pp. 1–6.
7. C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," IEEE Internet Computing, vol. 16, no. 2, pp. 62–67, March 2012.
8. Behnel, L. Fiege, and G. Muhl, "On quality-of-service and publish-subscribe," in 26th IEEE International Conference on Distributed Com-puting Systems Workshops (ICDCSW'06), July 2006, pp. 20–20.
9. U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s x2014; a publish/subscribe protocol for wireless sensor networks," in Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on, Jan 2008, pp. 791– 798.
10. N. D. Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," in 2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT), Nov 2013, pp. 1–6.
11. S. Lee, H. Kim, D. k. Hong, and H. Ju, "Correlation analysis of mqtt loss and delay according to qos level," in The International Conference on Information Networking 2013 (ICOIN), Jan 2013, pp. 714–717.
12. M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar, "Secure mqtt for internet of things (iot)," in 2015 Fifth International Conference on Communication Systems and Network Technologies, April 2015, pp. 746–751.
13. Capossele, V. Cervo, G. D. Cicco, and C. Petrioli, "Security as a coap resource: An optimized dtls implementation for the iot," in 2015 IEEE International Conference on Communications (ICC), June 2015, pp. 549–554.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)