A Survey of Messaging Protocols for IoT Systems

Sagar P Jaikar [#], Dr. Kamatchi R. Iyer [*]

[#] *Research Scholar (CSE), Amity University Mumbai*

[*] *HOI, Amity School of Engineering and Technology (ASET), Amity University Mumbai*

***Abstract*— IoT is gaining popularity day by day; every object surrounding us is interacting among them. These objects can be your mobile phones, sealing fan, door bell or it can be anything you could imagine. The network of everything is growing rapidly with heterogeneous objects or devices. But the main concern is how they are communicating with each other? What protocols they are using? As we know that these devices are very constrained with respect to power, communication & processing potential, we cannot use standard communication protocols that we are using over internet. We need different set of protocols which could fit in the IoT scenario focusing on conserving power & having small payload size. As various IoT systems are different in nature with different messaging requirements, it is very challenging to choose appropriate messaging protocol. Different IoT protocols have emerged with everyone having their advantages & disadvantages but IoT industry is facing a selection conundrum as these protocols will not fit into each and every scenario. In this paper we are studying various messaging protocols, so that we can decide upon appropriate protocol based on application needs.**

***Keywords*— IoT, MQTT, CoAP, XMPP, DDS, AMQP**

I. Introduction

The IoT enables objects to interact with each other, to exchange information for decision making. The IoT transforms these objects from simple traditional objects to smart objects by exploiting its underlying technologies. But these objects are heterogeneous objects & depending upon application requirements they will use different communication protocols. So selecting messaging protocol is very tedious task. [12] To select efficient messaging protocol, first we need to understand messaging requirements of IoT system.

For Internet based communications, we mostly use HTTP as standard protocol, but that is not the case for IoT systems. We cannot use HTTP as a standard protocol for all IoT systems. [13] Considering this various protocols has been designed to meet different needs of IoT systems. To meet the need for reliable & fast communications, protocol like AMQP is designed. [3] [6]

For communication in constrained networks, MQTT & CoAP are proposed. [7][8][9] Few protocols are also designed to suit instant messaging needs. XMPP falls in this category. [3] Some protocols are designed for simple communication over internet which supports client/server type architecture. To name few HTTP, CoAP is among them. [7][10] So it is clear that various protocols are proposed for IoT systems which have different pros & cons. In this paper we are discussing few of them & in later section we have compared them for users to make an appropriate choice of protocol suiting their application needs.

II. IoT Messaging Protocols

A communication protocol is nothing but a language that is used by objects to interact among them. In simple words, a protocol is a set of rules that must be obeyed by the communicating objects. Communication Protocols are extremely essential in heterogeneous systems; where the objects interacting may be heterogeneous in nature, needing a common framework for them to interact.

A. MQTT (Message Queuing Telemetry Transport Protocol)

MQTT is a lightweight machine to machine communication protocol designed for connecting small devices to constrained networks. It follows publish/subscribe model for communication. It is very simple protocol which uses 2 byte header & in binary in nature. MQTT is reliable & connection oriented as it uses TCP. MQTT is generally employed in the scenario where we have large network of small devices which are maintained from single server. To provide QoS it has three different levels namely At most once, At least once & exactly once. [9]

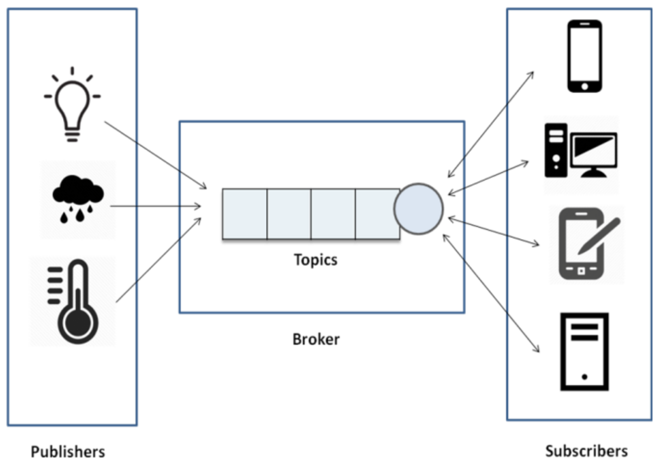

Fig. 1 Working of MQTT Protocol

These levels give the assurance about the message delivery. MQTT is not suitable for multicasting as well as device to device communications. MQTT was proposed in 1999 by Andy Stanford-Clark, IBM & Arlen Nipper, Arcom. Later on it was accepted as standard in 2013. [7] It is similar to blogs, where we can write something interesting that we would like with specific topic name. So anyone who is interested in our

Topic, can read the blog. This is exactly how the MQTT works. In MQTT data blocks are sent by Publish message, while they are received by Subscribe message. Here data is identified by Topics. [10]

MQTT architecture contains three elements: Publishers, Broker, and Subscribers. Here Publishers are nothing but the small sensors that collect the data and transmit it to the Broker. However Subscribers are the applications which are interested in the gathered data. As we know, data is identified by Topics; Subscribers need to subscribe to particular Topic of interest to receive the concerned data. Brokers categorises data according to Topics & forwards it to subscribers interested in different Topics. [6] MQTT brokers may go for authentication from Subscribers to connect. To ensure privacy, the TCP connection may be encrypted with SSL/TLS.

A variant of MQTT is Secure MQTT (SMQTT). It uses attribute based encryption which allows broadcasting of encrypted message to multiple nodes. SMQTT follows symmetric encryption where Publishers & Subscribers need to register themselves with the broker to obtain shared secret key. Now Publishers will send encrypted data to broker which will forward it to interested Subscribers, where it will be decrypted with shared secret key. [6]

MQTT-SN is protocol specially designed for sensor networks. It doesn't use TCP as a transport protocol. As we are aware of sensor nodes, they have very limited power resources & very constrained processing capabilities. On the other hand TCP is very heavy transport protocol. MQTT-SN improves upon MQTT by using simple header, topic id's instead of topic names, error status & so on [6].

B. XMPP (Extensible Messaging and Presence Protocol)

Extensible messaging and presence protocol is designed to address heterogeneity issue in IoT networks. It is developed in 1999 named as Jabber which it later standardized by IETF. [8] Initially it was designed for messaging applications over Internet. XMPP is real time communication protocol which can be used for instant messaging, real time entertainment applications & tele-presence. Google talk, Facebook chat are examples where XMPP is used for real time messaging service. XMPP is IP based communication protocol with Extensible Mark up Language (XML) support. [8]

XMPP also uses Publish/Subscribe architecture like MQTT. It also has support for Request/Response architecture. So it gives flexibility to application developer to choose the proper architecture as per need. Unlike MQTT it doesn't provides any QoS guarantees, so it is not suitable for M2M communications. In XMPP, XML support addresses the heterogeneity issue but on contrary it adds additional overhead due to lot of tags & header formats which increases the power consumption in IoT devices. [6] This problem can be resolved by compressing XML streams using EXI. It supports sleeping sensor nodes which can go into sleep mode periodically leading to extended lifetime of IoT devices. It supports periodic data transmission as well as event driven transmission too. Openness, Scalability, Flexibility, Extensibility are some of the key advantages of XMPP. [8]

XMPP allows heterogeneous devices to interact with each other by sending instant messages over the Internet irrespective of the underlying operating systems. XMPP is very secure protocol which supports encryption, authentication, and access control for the addition of different new applications on top of the existing core protocols. [7]

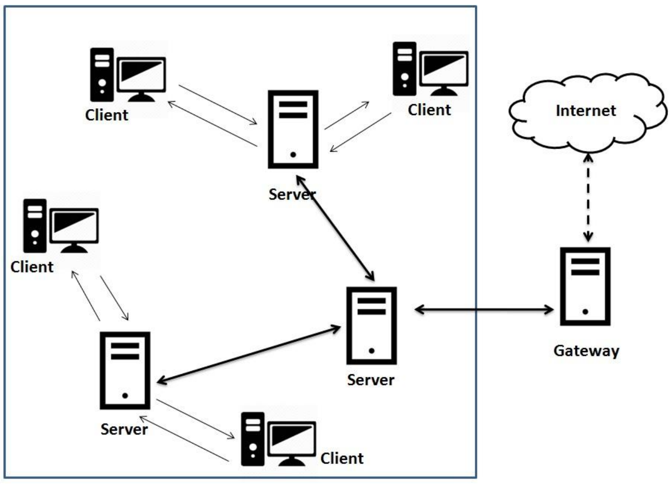

Fig. 2 Working of XMPP Protocol

XMPP uses XML stanzas for client to server communication. An XML stanza has three components: message, presence, and iq (info/query). Here source and destination addresses, types, and IDs are denoted by message component. The presence field notifies the clients about status updates. The iq field maps publishers & subscribers. [7]

C. CoAP (Constrained Application Protocol)

Constrained Application Protocol is synchronous Machine to Machine communication protocol which supports Publish/Subscribe as well as Resource/Observe architecture. CoAP is provides interoperability for RESTful web & HTTP. CoAP doesn't use topics; it uses Uniform Resource Identifier (URI).

In CoAP Publisher publishes data to the URI and subscriber has to subscribes it from corresponding URI. Whenever new data is published to the URI, all the subscribers are notified about it. Similar to MQTT, CoAP is also a simple binary protocol with fixed 4 byte header. As a transport protocol CoAP uses UDP which means it is connectionless with less reliability. Despite of this it uses two different QoS levels to provide reliability namely "Confirmable" & "Non-Confirmable". [9]

The strong security capabilities of CoAP make it a desired choice among the available IoT protocols. For providing security CoAP relies on Datagram Transport Layer Security (DTLS). CoAP is more suitable for the IoT as it uses UDP. However, CoAP works in presence of lossy and noisy links by modifying some of the HTTP functionalities to meet the IoT requirements such as low power consumption.

CoAP has two sub-layers, namely: the messaging sub-layer and the request/response sub-layer. The task of detecting duplications is done by messaging sub-layer. On other hand the request/response sub-layer is responsible for handling REST communications. As in HTTP, CoAP uses methods such as GET, PUT, POST and DELETE to perform Create, Retrieve, Update and Delete operations. [7]

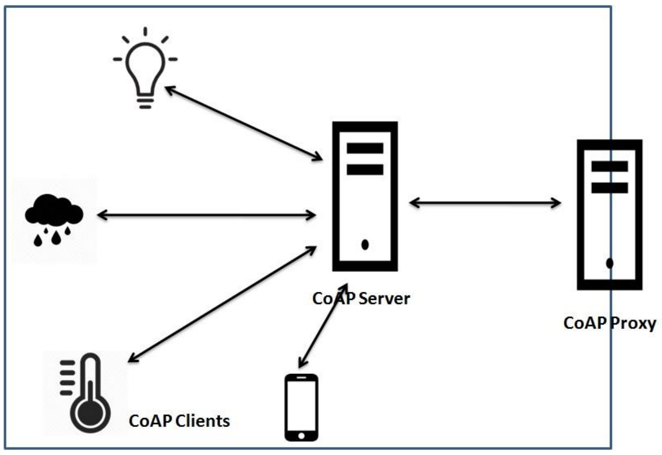

Fig. 3 Working of CoAP Protocol

D. HTTP (Hyper Text Transfer Protocol)

HTTP is specially designed for Internet. It was developed by Tim Berners Lee & later standardized by IETF in 1997. Though HTTP uses Request/Response architecture, it doesn't use topics. HTTP is based on Representational State Transfer (REST), an architectural style that makes information available as resources identified by URIs [18]. HTTP is simple text based protocol where no fixed header size is defined. It has features on persistent & non persistent connections. By default TCP is used as HTTP's transport protocol, but HTTP doesn't have any QoS support. [9]

HTTP is very powerful protocol, but it's relatively expensive in implementation and network resource usage. This makes it difficult to adopt HTTP as it is for IoT networks. HTTP transfers a large number of small packets over web but overhead of HTTP causes many problems, such as consumption of network resources and large delays. [10]

E. DDS (Data Distribution Service)

DDS is developed by Object Management Group for real-time Machine to Machine communications. Like most other M2M protocols DDS also supports Publisher/Subscriber architecture. But unlike others it doesn't uses broker based architecture. Instead of broker, it uses multicasting to provide QoS guarantees. The advantage of DDS is an excellent quality of service levels and reliability guarantees. DDS architecture defines two layers: Data-Centric Publish- Subscribe (DCPS) which disseminates information to subscribers and Data-Local Reconstruction Layer (DLRL) which is an optional and is an interface to the DCPS functionalities. It shares data among distributed objects. [7]

DCPS layer performs data distribution. Here data writer communicates with the publishers about the data and changes to be sent to the subscribers. While data readers read the published data and deliver it to the subscribers.

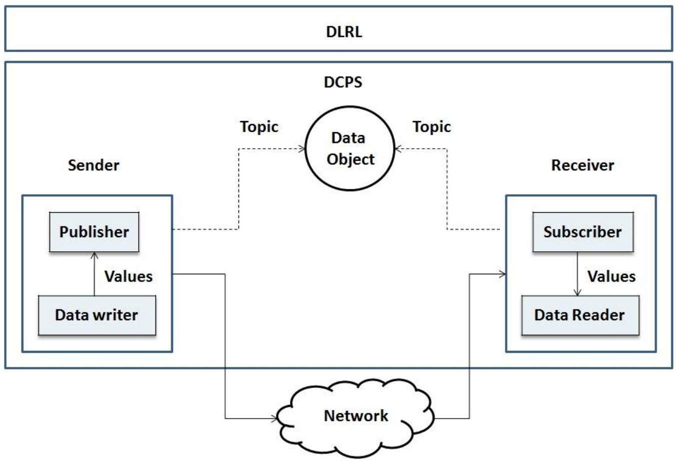

Fig. 4 Working of DDS Protocol

Subscribers are the nothing but the receivers of the data to be delivered to the IoT application. So the tasks of broker are handled by data writers and data reader to support broker-less architecture.

F. AMQP (Advanced Message Queuing Protocol)

AMQP is application layer protocol designed for message oriented networks & it has a support for Publisher/Subscriber architecture. It uses TCP as a transport protocol to provide reliable communication. Besides this, to provide QoS guarantees it has three levels of delivery namely at least once, at most once & exactly once. Along with Publishers & Subscribers it has two more components, Exchanges & Message queues. Exchanges perform the routing functionality by forwarding messages to appropriate message queues. These messages can be stored into message queues before forwarding it to Subscribers. AMQP uses two different message types. First, bare messages which are used by Publishers and second, annotated messages which are used by Subscribers. [7]

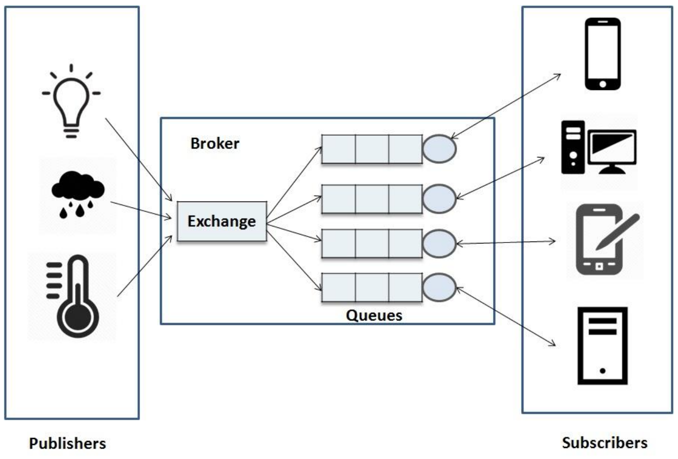

Fig. 5 Working of AMQP Protocol

TABLE I
COMPARATIVE ANALYSIS OF IOT MESSAGING PROTOCOLS

Sr. No	Criteria	MQTT	XMPP	CoAP	HTTP	DDS	AMQP
1	Year	1999	1999	2010	1997	2004	2003
2	RESTful	No	No	Yes	Yes	No	No
3	Transport Protocol	TCP	TCP	UDP	TCP	TCP/UDP	TCP
4	Publish / Subscribe Model	Yes	Yes	Yes	No	Yes	Yes
5	Request / Response Model	No	Yes	Yes	Yes	No	No
6	Security	SSL	SSL	DTLS	SSL	SSL / DTLS	SSL
7	QoS	Yes	No	Yes	No	Yes	Yes
8	Header Size	2	-	4	-	-	8
9	XML Support	No	Yes	No	Yes	No	No
10	Encoding Format	Binary	Character	Binary	Text	Binary	Binary
11	Default Port	1883/8883	5222/5223	5683/5684	80/443	7400/7401	5671/5672
12	Proxy Support	Partial	Yes	Yes	Yes	Yes	Yes

IoT has several messaging protocols which we discussed in previous section. The choice among these protocols is dependent on applications as they are very specific to applications. MQTT stands tall among all the protocols & is the most widely used in IoT due to various benefits such as extremely low overhead and very less power consumption. However, selection of protocols are highly application sensitive. For example, if an application has been built with XML and can accept a bit of overhead in its headers, XMPP might be the best option to choose among all protocols. On the other hand, if the application is really overhead and power sensitive, then choosing MQTT would be the best option, however, that comes with the additional broker implementation. If the application requires REST functionality as it will be HTTP based, then CoAP would be the best option if not the only one.

III. CONCLUSIONS

In this paper we have analyzed & compared few of the Messaging protocols for IoT systems but still there is no perfect evaluation of all these protocols together. We have discussed some of the well know IoT messaging protocols. We started our discussion with MQTT & then we discussed XMPP, CoAP, HTTP, DDS, and AMQP. Each of these protocols has their different pros & cons. They are designed for particular scenarios. So it is very difficult to prescribe any single protocol for diverse IoT applications. Also we performed comparative analysis of this protocol which will help us to choose appropriate messaging protocol depending upon application requirements. But as we know that IoT is rapidly gaining popularity, the current scenarios might not stand as it is in future. The significant changes in underlying technology will make it interesting to use & evaluate these protocols practically in near future. We believe that this will be significant area of research in future.

REFERENCES

[1] D. Thangavel, X. Ma, A. Valera, H. Tan, and C. K. Tan, “Performance evaluation of MQTT and CoAP via a common middleware,” in Proc. IEEE 9th Int. Conf. ISSNIP, 2014, pp. 1–6.

[2] L. Tan and N. Wang, “Future Internet: The Internet of Things,” in Proc.3rd ICACTE, 2010, pp. V5-376–V5-380.

[3] P. Lopez, D. Fernandez, A. J. Jara, and A. F. Skarmeta, “Survey of Internet of Things technologies for clinical environments,” in Proc. 27th Int. Conf WAINA, 2013, pp. 1349–1354.

[4] J. Gubbi, R. Buyya, S.Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[5] M. R. Palattella et al., "Standardized protocol stack for the Internet of (important) things," IEEE Commun. Surveys Tuts., vol. 15, no. 3, pp. 1389–1406, 3rd Quart. 2013.

[6] Hwaiyu Geng, Tara Salman, Raj Jain, "Networking Protocols and Standards For Internet Of Things", 2016

[7] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, Moussa Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications", IEEE Communication Surveys & Tutorials, Vol. 17, No. 4, Fourth Quarter 2015

[8] Heng Wang, Daijin Xiong, Ping Wang, And Yuqiang Liu, "A Lightweight XMPP Publish/Subscribe Scheme for Resource-Constrained IoT Devices" IEEE ACCESS, 2017

[9] N. N. Naik, "Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP", in IEEE International Systems Engineering Symposium (ISSE), 2017

[10] Tetsuya Yokotani, Yuya Sasaki,"Comparison with HTTP and MQTT on Required Network Resources for IoT",International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016

[11] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," Transaction on IoT and Cloud Computing, vol. 3, no. 1, pp. 11–17, 2015.

[12] R. S. Cohn, "A comparison of AMQP and MQTT," 2011.

[13] T. Jaffey. (2014, February) MQTT and CoAP, IoT protocols.[Online].Available: https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php

[14] Naik, P. Jenkins, P. Davies, and D. Newell, "Native web communication protocols and their effects on the performance of web services and systems," in 16th IEEE International Conference on Computer and Information Technology (CIT). IEEE, 2016, pp. 219–225.

[15] N. N. Naik and P. Jenkins, "Web protocols and challenges of web latency in the web of things," in 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN). IEEE, 2016, pp. 845–850.

[16] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future Internet: The Internet of Things architectre, possible applications and key challenges," in Proc. 10th Int. Conf. FIT, 2012, pp. 257–260.

[17] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol HTTP/1.0", IETF RFC 1945, 1996

[18] IBM, "MQTT V3.1 Protocol Specification", 2012, http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqttv3r1.html

[19] T. Fujita, Y. Goto, A. Koike, “M2M architecture trends and technical issues”, The Journal of IEICE, Vol.96, pp.305 － 312, 2013