COMPARATIVE ANALYSIS ON PERFORMANCE OF COMMUNICATION PROTOCOLS IN RESOURCE-CONSTRAINED IOT SCENARIO

*[1]R. Chawngsangpuii and [2]Dr. Prodipto Das

[1]*Department of Information Technology, Mizoram University, 796004, India*
[2]*Department of Computer Science, Assam University, Silchar, 788011, India*

***Abstract*— The term Internet of Things is a massive technological revolution which has revised and enhanced the present global Internet scenario to a concept of more advanced computing network in which all the interconnected physical objects in the world will be distinctively identifiable and connected to share information. All the objects or things in our walks of life such as cars, refrigerators, kitchen containers, clothes, etc can now be enabled to collect useful information by using sensors through IoT technology. With the advent of IoT, the number of connected devices for many applications has grown exponentially and expected to grow even more for providing better life in the world. Due to the ability of objects or things to communicate with the help of the technology, IoT has offered many applications for its users, which led to its immense popularity in the present scenario. IoT entails enormous autonomous flow of data to and fro the objects or things and automated smart actions can thus be taken based on the data collected. Nevertheless, the heterogeneous devices are usually resource-constrained in the IoT environment. As these devices have low communication and computation resources, specific communication protocols which are lightweight are required for providing machine-to-machine communication between large collections of diverse devices. There are several factors to be considered for implementing this technology such as the data and network protocols, and also the hardware to be used. Several communication protocols have been standardized for the IoT system to enable the various devices to communicate and make M2M possible between these devices. Selecting a suitable protocol can be a complicated task due to the constrained nature of the objects and the Wireless Sensor Networks. Since effective communication between devices is an important objective in any IoT system, a clear understanding of the protocols used for the same is essential for anyone utilizing the IoT applications. Furthermore, evaluating the performance of the application protocols is necessary for identifying which IoT protocol is best suited for what service. Hence, in this paper, we performed experimental comparative evaluation of CoAP, MQTT and AMQP based on different performance metrics.**

***Keywords*— Internet of Things, communication, CoAP, MQTT, AMQP.**

1. INTRODUCTION

The global increase in IoT devices led to the increase in connected devices to the internet and also for enhancing our day-to-day life as in fig. 1. The IoT has found its influence and potential in various development fields such as smart sensors, communication and RFID technologies. Many heterogeneous devices connected to IoT are able to share information via the internet forming a large distributed system. The sensors are sensing and collecting the data from the environment in the form of electrical signal, which are send to the microcontroller. On the other hand, the actuators transform the electrical signal to motion. Based on the value of the sensed data from the sensors, the actuators take suitable action. Microcontrollers such as arduino, raspberry pi, etc can be utilized as IoT devices to process the data from sensors as they have reasonably good computing power.

* Corresponding Author

Most of the devices used in IoT have constrained resources in energy power, processing power and storage [1]. The heterogeneous devices should be able to interoperate with each other to communicate and to share services [2]. These devices will not be able to communicate effectively without the help of communication protocols. As the traditional communication protocols are not suitable for resource-constrained devices, IETF (Internet Engineering Task Force) standardized lightweight communication protocols in support of IoT devices [3].

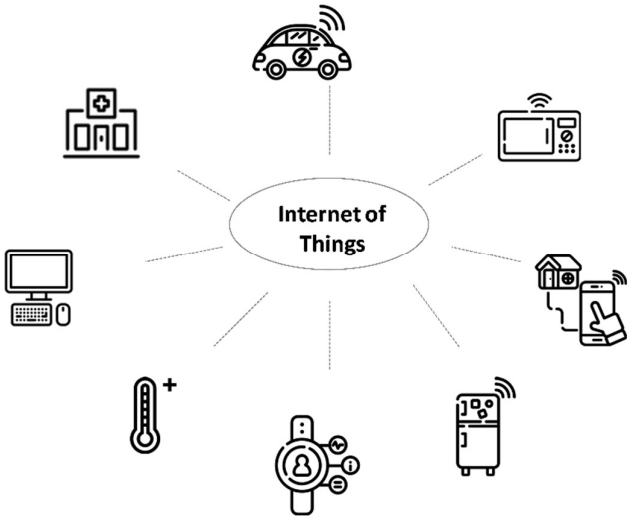

Fig. 1 The IoT System

Even though several IoT communication protocols have been proposed for the application layer, it is often cumbersome to choose the most suitable one for one's task due to the differences in performance with respect to the services offered by these protocols. It is therefore, necessary to provide a proper evaluation of the protocols for anyone trying to adopt IoT system to have a clear insight of the advantages and drawbacks for these protocols. In this work, we performed comparative evaluation of the IoT protocols, namely, CoAP (Constrained Application Protocol), MQTT (Message Queuing Telemetry Transport), and AMQP (Advanced Message Queuing Protocol) to gain clear understanding of the performances of these protocols in actual constrained-device environment.

2. RELATED WORKS

The authors in [4] designed a middleware for testing two protocols – CoAP and MQTT. They performed performance evaluation using an experiment to examine the bandwidth consumption and delay. It was found that the delay for MQTT was lower compared to CoAP when there was low packet loss. Also, MQTT experienced higher delay compared to CoAP when the loss rate is high. However, other performance metrics such as energy consumption, throughput values were not considered.

The authors in [1] surveyed on the IoT application layer protocols and described the various strengths and limitations. However, the work carried out lacked numerical comparison when drawing conclusions with respect to the performance of the protocols.

In [5], the authors carried out performance comparative of MQTT, CoAP, and WebSocket. The metrics such as efficiency of protocol and the average RTT (Round Trip Time) were considered by strictly adhering to certain overhead. As a result, the findings were not stable enough as it did not take into account for increased load.

The authors in [6] performed comparative evaluation of CoAP, XMPP, MQTT and WebSocket with the help of smart parking environment. The performance metric used in their work is the response time for each protocol with different loads. However, other metrics such as bandwidth consumption, latency and packet loss were not done.
The work by the authors in [7] carried out performance comparison of HTTP and MQTT and proposed enhancement for MQTT in order to provide better performance for the protocol. The evaluation was done on the bandwidth, payload size and round trips metrics. However, the comparative evaluation did not include other IoT application protocols.

3. OVERVIEW OF THE IOT PROTOCOLS

A brief outline of the three IoT protocols which are under study – CoAP, MQTT and AMQP are presented in this section.

3.1. CoAP

It is a stateless as well as sessionless protocol designed by the IETF and defined in RFC 7252 [8]. CoAP is mainly developed to be used for the resource-constrained devices and shared many similar characteristics of the HTTP. It is also based on REST (Representational State Transfer), popularly used in the present web applications. However, CoAP solved the problem of REST by having small overhead and message size. This made CoAP suitable for IoT devices but utilized UDP (User Datagram Protocol), a transport protocol which does not guarantee reliability. The CoAP protocol has two layers, which are request/response layer and messaging layer [9]. Error recovery is dealt with by the messaging layer and communication is handled by the request/response layer. An enhancement was proposed by the IETF which was defined in RFC 7641 and is observing resources in CoAP for making publish/subscribe mode possible [10]. The message sub-layer has support for four types of messages, which are CON (to denote Confirmable message), NON (to denote Non-confirmable message), RST (to denote Reset) and ACK (to denote Acknowledgement) [8].

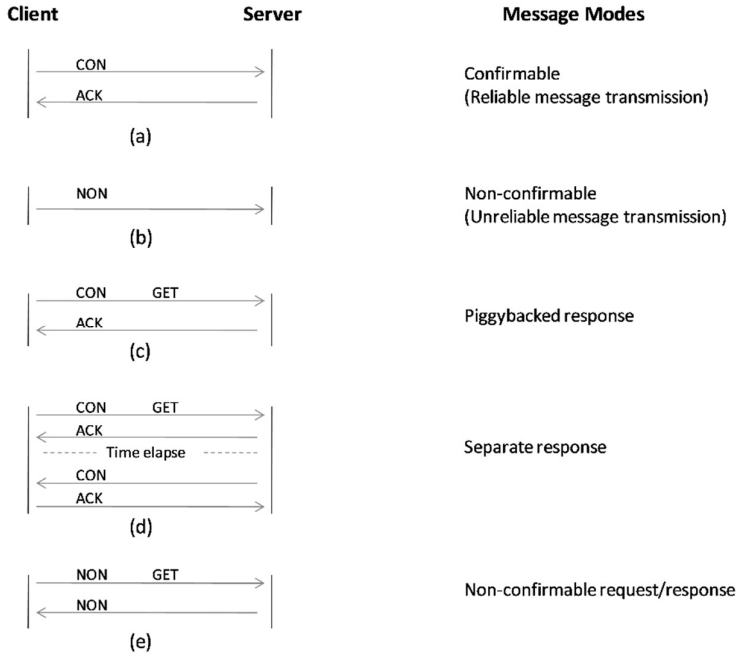

Fig. 2 Message layers of CoAP

From fig. 2, confirmable represents reliable transmission and non-confirmable represents unreliable message transmission. A CON message is sent to the server until an ACK message is received as in fig. 2 (a). If the CON message cannot be processed by the server, it replies by sending RST message as a replacement for ACK message. When reliable transmission is not required for a message, it is sent as

NON message as in fig. 2 (b). On the contrary, for the request/response sub-layer, the remaining messaging modes are used with the GET request. Piggybacked response as in fig. 2 (c) is carried in the ACK message when the server sends response to a client's request. When a server cannot respond to a client's request immediately, it just sends empty ACK message. After sometime, if the response is ready, it then sends separate response in CON message, which is acknowledged with ACK by the client as shown in fig. 2 (d). When the client transmits NON message with GET request, the server replies the response in a new NON message as shown in fig. 2 (e).

3.2. MQTT

This is publish/subscribe messaging lightweight protocol proposed in 1999 by IBM and Arcom, which eventually got standardized in 2014 by OASIS (Organization for the Advancement of Structured Information Standards) [11]. It is also widely used protocol for constrained devices which follow asynchronous message communication by using publish/subscribe architecture demonstrated in fig. 3.

MQTT has three components, namely publisher, broker and subscriber. The publisher can send any data or topic to the broker. If any client needs the data, it can subscribe to that topic through the broker. The broker in turn, will send the topic to the interested subscriber whenever new topic is available. Hence, the broker stores all data in the form of topic and topic status information. The topics in MQTT can have many levels and separated by slash. One such example is home/bedroom/temperature.

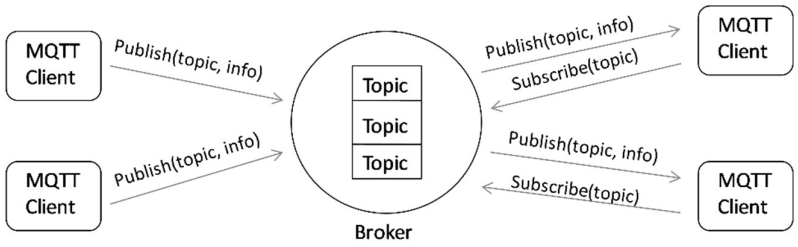

Fig. 3 MQTT Architecture

MQTT relies on TCP similar to HTTP but with a lower overhead, which makes it more suitable in a constrained environment. It also provides reliable message delivery even when there is unstable connection [12]. The three levels of QoS (Quality of Service) provided by MQTT are given as under [11].

- QoS 0: Message can be sent by the publisher without expecting ACK message back from the broker. This is used when data is not very critical and there is stable connection.
- Qos 1: Message is sent by the publisher until there is ACK message called PUBACK from broker. This is used when there is critical data and unstable connection.
- QoS 2: Message is sent by the publisher only once until there is PUBREC received message from broker. When this message is received, it removes any reference for the data and send back PUBREL released message to broker. This is done when there is critical data and unstable connection.

3.3. AMQP

AMQP is a messaging protocol developed in 2003 for the banking industry and standardized by OASIS in 2012 [13]. It relies on TCP like the MQTT protocol but can follow either request/response or publish/subscribe method. It was designed to make interoperability possible among diverse devices and applications [14]. In AMQP architecture, the broker has two components, namely exchange and topic queues as illustrated in fig. 4.

The publisher transmits message or topic to the exchange part of the broker, which then distributes to the various topic queues. The messages in the topic queues remain in the queue till the subscriber

receives them. If there are more subscribers wanting a specific message, the broker duplicates the copies to their own queues, subscribed by the subscribers.

The new version of AMQP protocol even allows the implementation to be used without the broker, thereby making communication between different topologies possible. There are three levels of QoS provided by AMQP, which are similar to with the MQTT. The store-and-forward feature provides an advantage of reliability even in times of disruption in the network [15]. However, since message delivery is given high importance in AMQP, the overhead is higher than the MQTT.

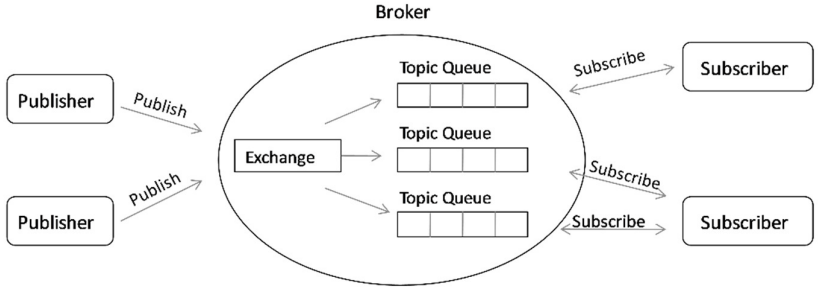

Fig. 4 AMQP Architecture

3.3. CONCEPTUAL COMPARISON OF THE PROTOCOLS

With the outline given above, the protocols are compared on their features as given in Table I. The comparison based on security scheme used by the protocols and their QoS details are further provided in Table II.

Table I. Feature Comparison

Protocol	Header Size	Architecture	Transport	Standard	Port No
CoAP	4 bytes	Req/Res Pub/Sub	UDP	IETF	5683 (UDP) 5684 (DTLS)
MQTT	2 bytes	Pub/Sub	TCP	OASIS	1883 / 8883 (TLS/SSL)
AMQP	8 bytes	Pub/Sub Req/Res	TCP	OASIS	5671 / 5672 (TLS/SSL)

Table II. Security and QoS Comparison

Protocol	Security	QoS Level	QoS Description
CoAP	DTLS	2	Confirmable message, Non-confirmable
MQTT	TLS/SSL	3	Level 1 : At most once (i.e., QoS0) Level 2 : At least once (i.e., QoS1) Level 3 : Exactly once (i.e., QoS2)
AMQP	TLS/SSL, SASL	3	Level 1 : At most once (i.e., QoS0) Level 2 : At least once (i.e., QoS1) Level 3 : Exactly once (i.e., QoS2)

4. EXPERIMENT SETUP

The hardware and software used in the experiment are discussed in this section. To have a realistic IoT environment, hardware devices are chosen which are low-cost and constrained. An affordable, small computer board, Raspberry Pi 3 [16] and a Lenovo Laptop with a specification of Intel Core i3 processor, 64-bit Windows 10 having 4 GB RAM were used. With a 64-bit ARMv7 QC processor together with integrated Wi-Fi and a capability of faster Ethernet, the Raspberry Pi is a suitable IoT

device. The test bed for the experiment is provided in fig. 5. The Pi acts as the client and the laptop acts as the server for each of the protocols. The devices communicated using a LAN (Local area network).

Raspbian OS [17] was used for the Raspberry Pi, which is freely available and open-source primarily made for small single-board devices. For the CoAP protocol, a well-known implementation of CoAP called libcoap [18] was used.

Mosquitto [19], which is suitable for constrained devices was implemented for the MQTT protocol. For the AMQP protocol, most widely utilized open-source implementation RabbitMQ [20] was employed as it supports many features.

Finally, a highly-approved packet analyzer tool called Wireshark [21] was used for capturing the traffic and analyzing the packets based on different performance metrics.

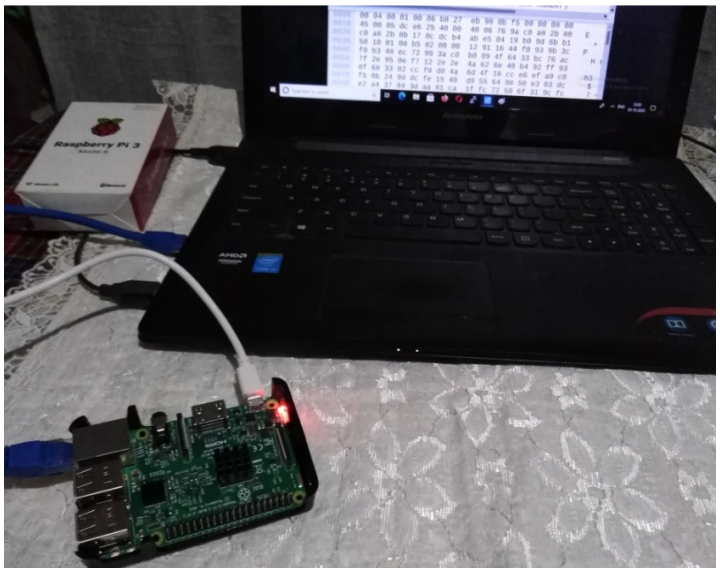

Fig. 5 Test Bed for the Experiment

5. PERFORMANCE RESULTS

The screenshots for the three protocols under study captured using Wireshark tool are presented in fig. 6, fig. 7 and fig. 8, in that order.

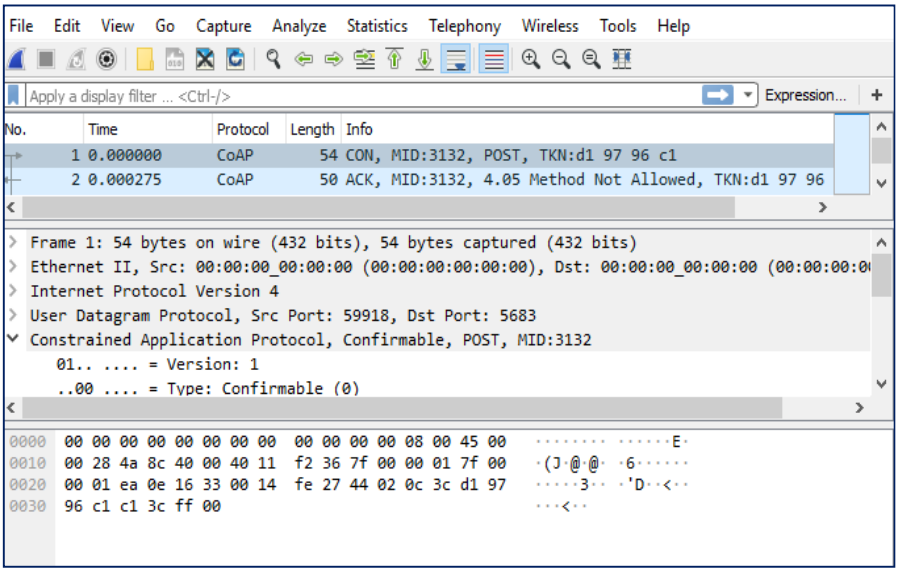

Fig. 6 Captured CoAP packets

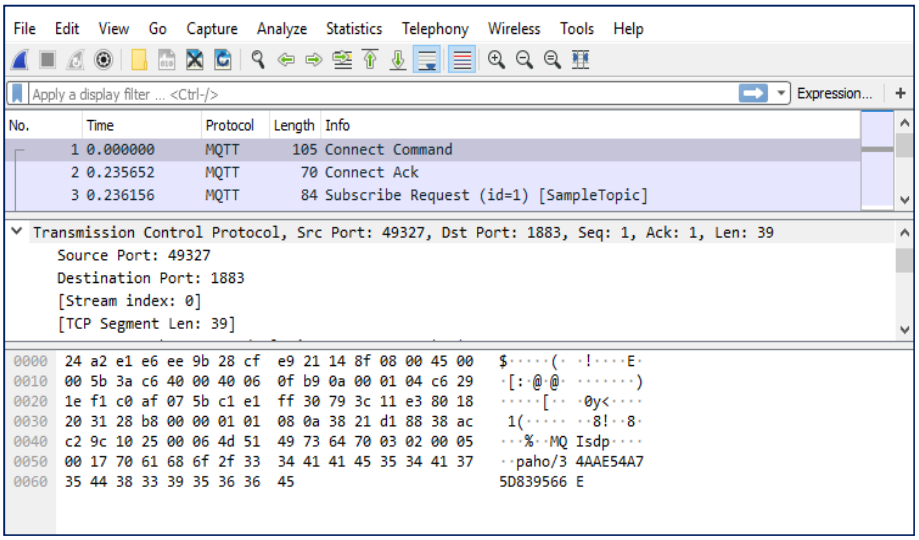

Fig. 7 Captured MQTT packets

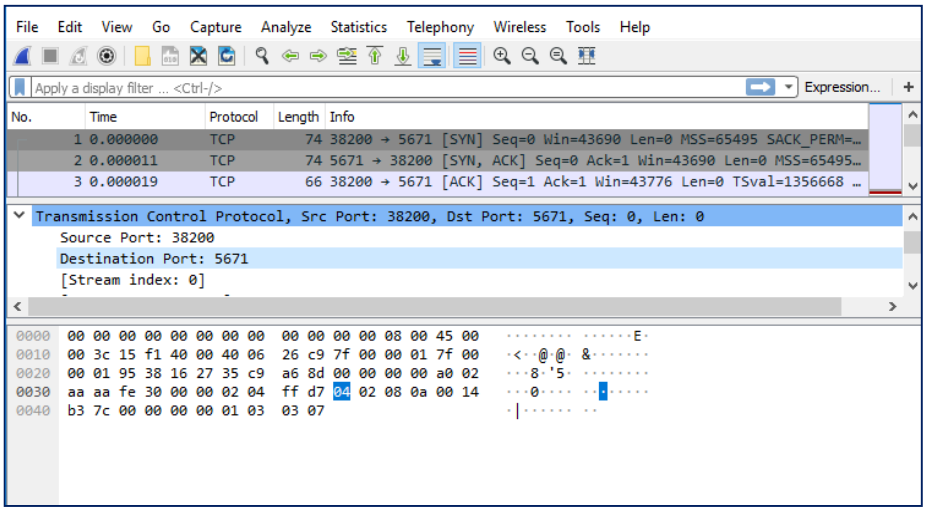

Fig. 8 Captured AMQP packets

5.1. LATENCY

The indicators for network latency are connection time and the round-trip time. In TCP, a 3-way handshake is followed for ensuring reliable connection by use of SYN (synchronize), SYN/ACK and ACK (acknowledge) messages or flags. A SYN is sent by a source device wanting to establish connection to destination device. SYN/ACK is sent by destination device if the SYN is received. ACK is sent back to destination to acknowledge the received data. The time usually in milliseconds, which measures the duration of packets travelling from source device to the destination device and again back to the source is called round-trip time. The maximum payload of the message for each of the protocols was 64 kilobytes.

The low latency of CoAP in fig. 9 is due to the protocol using UDP, which does not require the sender and the receiver to first establish a handshake before sending packets. Hence, it has experienced 0.519 ms as the latency. On the other hand, MQTT and AMQP require TCP handshakes which accounts for the overheads involved in sending packets. Therefore, the average latency for both are 0.884 ms and 0.748 ms, respectively.

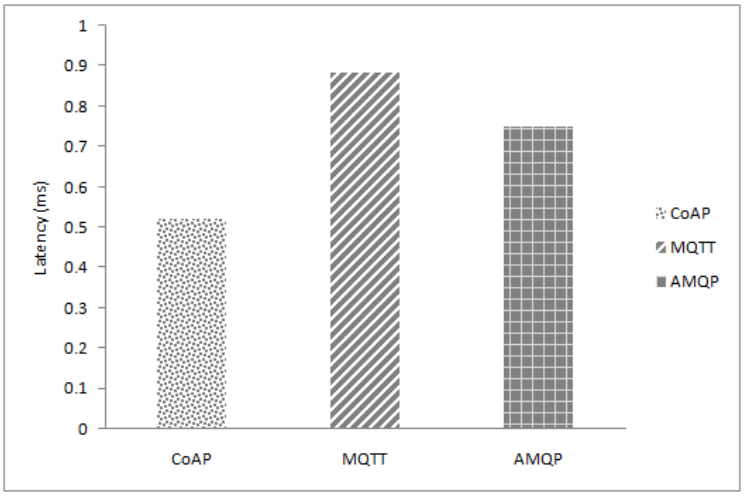

Fig. 9 Average Latency

5.2. PACKET SIZE

Since the size of packet has an impact on the throughput of the network [22], it was considered to be used as performance metric, when comparing different protocols. The table III presents the different average packet size for the protocols used in our work.

Table III. Packet Size analysis

Protocol Name	Size in Bytes
CoAP	55
MQTT	78
AMQP	162

As CoAP uses a simple transport protocol, it also minimizes the message size and overhead as shown in fig. 10. MQTT employs stream-oriented method for writing frames but message fragmentation is not permitted. Hence, the average size for CoAP and MQTT packets are 55 bytes and 78 bytes respectively. The high size in AMQP is due to the buffer-based approach utilized using the message queues, adopted by the protocol.

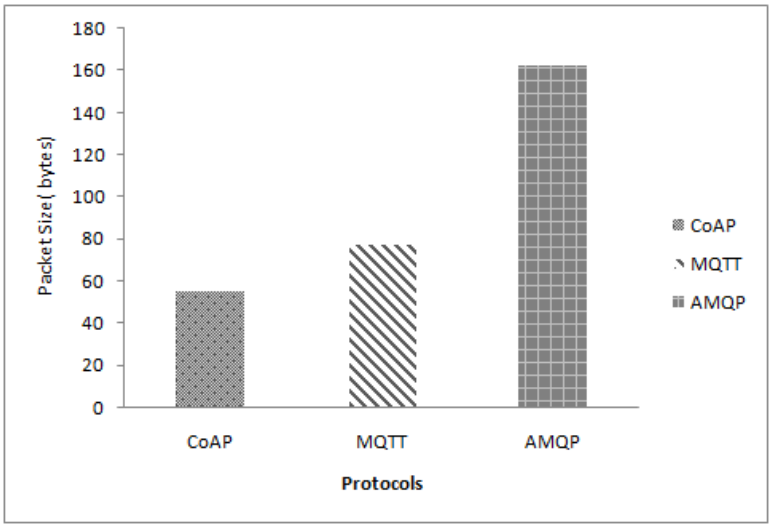

Fig. 10 Average Packet Size

5.3. THROUGHPUT

Throughput is another performance metric carried out for determining actual network performance with respect to packet delivery. It is often measured in bits/sec and can be in other units also. It is calculated using the formula given in equation (1) given below.

$$Throughput = \frac{Successful\ delivered\ packets\ (bits)}{Time\ (s)} \quad (1)$$

The low throughput of MQTT as shown in fig. 11, is due to the latency experienced by the protocol since TCP ensures message delivery from source to destination and also due to retransmission. CoAP has moderate throughput since it is not affected by network latency as it uses UDP. The throughput in AMQP is high as the packet is high compared to the other protocols.

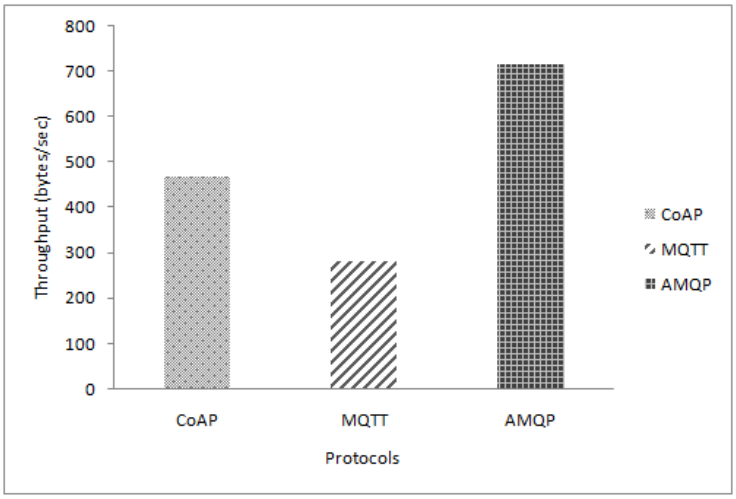

Fig. 11 Average Throughput

6. CONCLUSION

In this work, we studied three IoT application protocols and performed comparative evaluation of CoAP, MQTT and AMQP protocols respectively. The evaluation was done first with the conceptual comparison between the different protocols based on the theoretical concepts. The experimental comparative evaluation was carried out next with affordable IoT devices to emulate realistic environment. The work carried out provides important insight into the performance of the protocols with respect to latency, packet size and throughput. Even though MQTT may not perform well in latency and throughput comparison, it is still a good candidate for network communication due to its reliability and wide device-support capability. AMQP, on the other hand, has more security feature than others through TLS extensions such as authenticating clients and encrypting data with the help of SASL (Simple Authentication and Security Layer).

The performance results can be utilized by network designers opting for IoT system for choosing the protocol most suited for their respective needs. For future work, we plan to compare other IoT protocols and include other performance metrics to achieve more knowledge on the protocols.

REFERENCES

[1] 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.

[2] J. Kiljander, A. D’Elia, F. Morandi, P. Hyttinen, J. Mattila, A. Ylisaukko-Oja, et al., “Semantic Interoperability Architecture for Pervasive Computing and Internet of Things,” IEEE Access, vol. 2, pp. 856-873, 2014.

[3] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. Mccann, and K. Leung, “A survey on the IETF protocol suite for the Internet of Things: Standards, Challenges, and Opportunities,” IEEE Wireless Communications, vol. 20, no. 6, pp. 91–98, 2013.

[4] 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 Intern. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), April 2014, pp.1-6.

[5] S. Mijovic, E. Shehu, and C. Buratti, "Comparing application layer protocols for the Internet of Things via experimentation," in Proc. IEEE 2nd Intern. Forum in Research & Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI), Bologna, Italy, Sep. 2016, pp. 1-5.

[6] P. Kayal and H. Perros, "A Comparison of IoT application layer protocols through a smart parking implementation," 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), May 2017, pp. 331-336.

[7] T. Yokotani and Y. Sasaki, "Comparison with HTTP and MQTT on Required Network Resources for IoT," Intern. Conf. on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016, pp. 1-6.

[8] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," Internet Engineering Task Force (IETF), RFC: 7252, Tech. Report, pp. 1-112, June 2014.

[9] T. Salman, R. Jain, "A Survey of Protocols and Standards for Internet of Things," Advanced Computing and Communications, vol. 1, no. 1, pp. 1-20, 2017.

[10] K. Hartke, "Observing Resources in the Constrained Application Protocol (CoAP)," Internet Engineering Task Force (IETF), RFC: 7641, Tech. Report, pp. 1-30, Sept. 2015.

[11] MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. Oct. 2014. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[12] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate and P. Manzoni, "Improving MQTT Data Delivery in Mobile Scenarios: Results from a Realistic Testbed," Mobile Information Systems, Vol. 2016, pp. 1-11, 2016.

[13] OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0 Part 3: Messaging. Oct. 2012. OASIS Standard. http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html.

[14] [14] Z. B. Babovic, J. Protic, and V. Milutinovic, "Web performance evaluation for Internet of Things applications," IEEE Access, vol. 4, pp. 6974-6992, 2016.

[15] [15] F. T. Johnsen, T. H. Bloebaum, M. Avlesen, S. Spjelkavik and B. Vik, "Evaluation of transport protocols for web services," in Proc. Military Communications and Information Systems Conference (MCC), 7-9 Oct. 2013, pp. 1-6.

[16] Raspberrypi.org. 2016. Raspberry Pi 3 Model B. [online] Available at: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/> [Accessed 8 Dec 2016].

[17] Raspbian.org. 2015. Frontpage - Raspbian. [online] Available at: <https://www.raspbian.org/> [Accessed 8 Dec 2016].

[18] Libcoap.net. 2015. Implementation of Coap. [online] Available at: <https://libcoap.net/> [Accessed 9 December 2016].

[19] Eclipse Mosquitto. 2015. Eclipse Mosquitto. [online] Available at: <https://mosquitto.org/> [Accessed 9 December 2016].

[20] Rabbitmq.com. 2014. Messaging That Just Works - Rabbitmq. [online] Available at: <https://www.rabbitmq.com/> [Accessed 9 December 2016].

[21] Wireshark.org. 2016. Wireshark - Go Deep.. [online] Available at: <https://www.wireshark.org/> [Accessed 10 December 2016].

[22] B. Zhang, Y. Li and Y. Liang, "Impact of Packet Size on Performance of TCP traffic with Small Router buffers," Proc. Intern. Conf. on Electronic Information Technology and Computer Engineering (EITCE), MATEC Web of Conferences, 2017, vol. 128, pp. 1-4.