INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2021, VOL. 67, NO. 2, PP. 229-234
Manuscript received June 2, 2020; revised April, 2021 DOI: 10.24425/ijet.2021.135969

Stress Testing MQTT Server for Private IOT Networks

Ghofran Hijazi, Mohamed Hadi Habaebi, Ahmed Al-Haddad, and Alhareth Mohammed Zyoud

***Abstract*—The rapid development of the Internet of Things (IoT) and the wide area of application rise the IoT concept to be the future of the internet. Indeed, IoT environment has a special nature with a lot of constraints in term of resource consumption. Moreover, the data exchanged between things and the internet is big data. In order to achieve efficiency in IoT communication, many technologies and new protocols based on these technologies have been developed. This paper aims to study the performance of Message Queuing Telemetry Transport (MQTT) by implementing this protocol on test-bed network infrastructure and analyzing the performance properties such as delay, jitter, packet loss and throughput for real time and non-real time scenarios. Finally, future research issues in MQTT protocol are suggested.**

***Keywords*—IoT, protocols, MQTT, computer networks**

I. Introduction

SINCE long time ago we have dreamed of smart homes, smart cars, and smart clothes. We have always wanted to take control over things in our life, observe the environment changes all time and increase the efficiency in the industry. All this become true now by Internet of Things (IoT). IoT is the term that means to connect things with the internet and with other things. Things are usually sensors [1]. In IoT, sensors obtained the data and share it with the internet. The internet can be any interested client or application such as mobile applications. In dead, there are three main challenges in the communication between things and internet [2]. The challenges are constrained devices, big data and security. In order to meet the IoT requirements and achieve the efficiency in IoT communication, Message Queuing Telemetry Transport (MQTT) protocol has been developed [3].

This paper will implement and evaluate the performance of MQTT protocol in real-world network infrastructure. Z1 mote is used as constraint device and raspberry pi as a server. In addition, this paper gives a clear evaluation of the abilities and of raspberry pi and Z1 mote when they are used in IoT network. Also, it shows the effects of Quality of Service (QoS) levels on both real and unreal time connections. The result of this paper is so reasonable and realistic as it reflects real-world scenarios that can be implemented in a wide range of applications such as smart homes.

The main objectives of this paper are to.

- Develop a test-bed for the MQTT transport protocol.
- Evaluate the performance of the MQTT protocol in real-time and non-real time application.
- Study QoS performance of the MQTT protocol and evaluate the interaction response between test-bed components.

The rest of this paper is organized as follows: the literature review is presented in Section 2. Section 3 explains the experimental setup. The results are discussed in Section 4. In Section 5, future works are suggested. A conclusion is drawn in Section 6.

II. Literature Review

This section gives an overview about IoT as it is important to understand the environment of this study. Then the characteristic of MQTT is summarized followed by an explanation for publish/subscribe technology and QoS level as these are the core concept behind MQTT protocol.

A. The Internet of Things

Generally, IoT is the concept of connecting the physical things around us to the internet. This connection will allow humans to communicate with the things and the things to communicate with each other. There are two components in the term of IoT, the internet, and things. Internet part indicates the network-oriented view. The things part shows how objects can be integrated into one framework [2].

The idea of communicating with things has developed over decades. However, IoT term was first introducing by Kevin Ashton in 1999 [4]. The IoT has developed dramatically since that time especially after recent wireless technologies have been adopted, for instance enabling wireless technologies such as RFID tags, embedded sensor and actuator nodes. It is now concerned as the next revolutionary technology. It is really important to know that when we are talking about things we mean sensors and actuators where sensors and actuators are placed in the environment around us, and they share the information in order to build a common operating picture.

In IoT technology, we need to achieve reliable connectivity for three types of communication. The first type is communication between two devices (D&D). The second type is communication between device and server (D&S) to send the data that have been collected to the server. The third type is the communication between two servers (S&S) to share the information with the internet [5]. This meaningful connectivity is one of the most critical challenges in IoT due to the nature of the devices and the sensor. Sensors have low power consumption and supply (battery-powered devices), limited resources and the huge number of the connected devices that

This work was conducted in IoT and Wireless Communication Protocols Lab and is partially funded by international Islamic University Malaysia Publication Research Initiative Grant Scheme no. P-RIGS18-003-0003.

Ghofran Hijazi, Mohamed Habaebi, and Ahmed Al-Haddad are with Department of Electrical and Computer Engineering, Faculty of Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia. (e-mail: ghofran.abd.hijazi@gmail.com, habaebi@iium.edu.my, itshaddad@gmail.com).

Alhareth Zyoud is with Department of Electrical and Computer Engineering, Faculty of Engineering and Technology, Birzeit University, Birzeit, Ramallah, Palestine. (e-mail: alhmtz@gmail.com).

cause an address problem. Therefore, we need to develop new protocols that meet the needs of IoT. This requirement can be surmised as following key points [6].

- Deliver data from one to many.
- Deduct the changes whenever they may happen.
- Share small packets of data in a massive amount.
- Coast of transmitted data.
- Power consumption (battery-powered devices).
- High response time (real-time).
- Security and privacy.
- Scalability.

IoT will affect our daily life substantially. From normal user prospective, this effect will appear clearly in assisted living, smart homes, and offices e-health and enhanced learning. These applications are only a few examples of the wide application of IoT. From the point of view of the business user, it will enhance the automation and industrial manufacturing, logistics, business process management, intelligent transportation of people and goods [2]. Figure 1 shows the wide range of IoT application listed based on popularity.

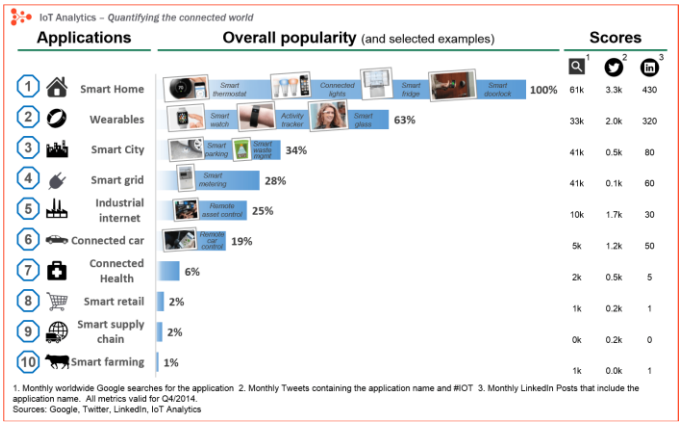

Fig.1. Applications of the internet of things [7]

B. *MQTT*

MQTT was invented by Dr. Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech), in 1999 [8]. The main function of MQTT is remote monitoring by collecting data from the large network with a large number of small devices, and then sending it to the internet. It also monitors the small devices in the network that need to be controlled from the cloud. MQTT is suitable for the application where we need.

The main characteristic of MQTT is listed as following:-

- It works on top of IP/TCP transport protocol.
- It uses a data-centric communication publish/subscribe technology. Data-centric communication is the technique that focuses on the data itself. The main unit is the data object value, not the message. The infrastructure conceder successful when all nods get a correct understanding of the data value.
- It is extremely simple.
- It is a lightweight messaging protocol. It has less payload and small overhead.
- It is designed for constrained devices that have a severe limitation on power, memory, and processing resources.

It is suitable for low-bandwidth

- It is suitable for high-latency or unreliable networks
- It ensures reliability by using some degree of assurance of delivery. MQTT support three levels of quality of service (QoS) that will describe in details later in Section 2.4.
- It has a tool to alert interested devices to an unexpected disconnection of a client by using the Will message and Testament feature.

C. *Publish- Subscribe*

Publish-Subscribe (pub/sub) technology is highly needed for the flowing reasons. First of all, because The Wireless Sensor Network (WSN) dynamic and temporal nature using network address as a communication system between Sensor/Actuator devices (SA devices) are complicated and not efficient. SA devices change their addresses in unpredictable time. They may stop working, so they need to be replaced. The wireless link itself may fail. Moreover, as Hunkeler et al. [9] noticed during their experiment that some network protocols that use to connect between SA devices such as ZigBee change the address of the devices. Thus, using network address is troubled.

Second, in the most cases, the applications do not need to know the actual address of the devices. They require for the information and the data that has been collected by SA devices. For example, GPS application does not need to know the address of a moving car. They need more to know the geographical location at the specific instant of time. In addition, many applications may request for the same sensor data for different intent or objective. From communication means, SA needs to deal with the different application in parallel. This will go beyond the limited resources and the low-coast of SA devices. Therefore, the network address communication approach need to be replaced by another one.

To overcome the problem described above, we use data-centric approach where the delivery off the message depends on the interest, not on the network address. Publish/subscribe message system is one of the most common examples of the data-centric system, see Fig. 2.

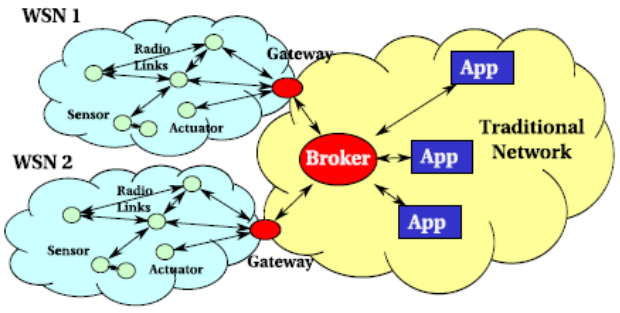

Fig. 2. Integrated wireless sensor networks with pub/sub communication [9]

The general concept of pub/sub method is that the consumer who is interested to resave type of information will register its interest and it will be called the subscriber. The device that produces the same information will publish its information and it will be called the publisher. The element which responsible to ensures that the data is sent from the publishers to the subscribers is called the broker. The broker is the server that connects the gateway to the cloud.

There are three basic types of pub/sub system. First, one is the type-based system. It is not widely used. Second is the content-based system. It is the most adaptable system where the subscriber uses a tiny DB and SQL query to define the content of the message that they would like to restive. The third one is the topic-based system where the subscription and publishing can be done only on the certain list of topics that has been determined in the design stage. It is the simplest system and the most applicable for the wireless sensor network that based on the hardware. Figure 3 shows how pub/sub messaging system works. The subscriber will send sub (topic) message to the broker. The publisher will send pub (topic, data) to the broker. The broker will look for matching topics. If it finds any matching topics between publisher and subscriber, the pub (topic, data) will be forwarded to the subscriber.

The main advantages of sub/pub system are that the application will not be affected if a failure accrues in SA side, it will get its information when the SA device replaced. Therefore, the application does not need to know about the SA failure. In the same way, the SA does not need to be aware of what application needs its data or how many applications. Simply the SA will send the data to the broker and the broker will be speared the data to the subscriber application. In addition, sub/pub system covers the complexity of the underlying networks. That makes it easier for the developers and makes them focus only on the application [9].

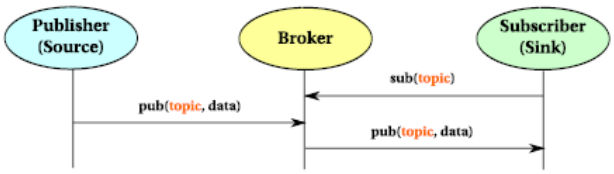

Fig. 3 Topic based pub/sup [9]

D. Quality of Services

MQTT supports reliability using three levels of QoS. The application chose the suitable level for itself depending on how much reliability is needed in the message delivered to the destination. The first level is level 0. In this level, there is no acknowledgement and no retransmission. Therefore, the message may be delivered once or not delivered. Second is level 1, the delivery of a message in this level is ensured and acknowledge message will be sent but the message may deliver more than once due to retransmission. Lastly, level 2 ensure that the message will be delivered only once by using four steps handshake. The nature of application determines what level of QoS should be used. Figure 4 shows the three levels of QoS [10].

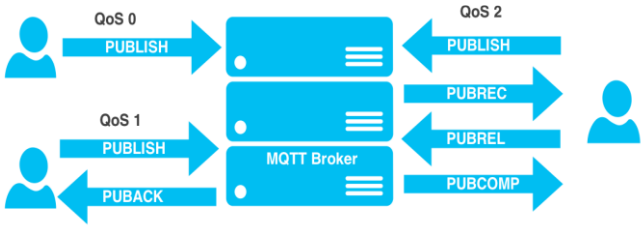

Fig. 4 QoS levels [10]

III. Experimental Setup

This section tackles the software and hardware that had been used in this research. It also talks about the performance elements under study. Then, it explains different suggested scenarios. Finally, the steps of the followed experiment procedure are listed.

A. Software/Hardware

There are several parts that are going to be used in this study. At the beginning, a private network is going to be constructed and MQTT protocol will be used as the main messaging protocol. The hardware parts that are going to be used are Raspberry Pi microprocessor to host the MQTT broker and Z1 sensors. For the software used the study, Contiki Operating System will be used to connect the sensors to the network. To use MQTT on Raspberry Pi then Mosquitto broker needs to be downloaded and installed. Also, MySQL database is to be used to hold on the data fetched from the sensors.

- Hardware:
  - Raspberry Pi
  - Z1 sensors
- Software:
  - Contiki
  - Mosquitto broker
  - MySQL

B. Performance Parameters

Average of total delay: can be simply defined as the time difference between sending publish request message from the client and resaving the response message that carries the data from the sensor. It can be divided into four types: Transmission delay, propagation delay, queuing delay and processing delay.

Delay variation (jitter): simply is the delay difference between packets that are sent from the same source to the same destination. In this case, one packet will have less delay than expected and another packet will get more delay than expected, this difference is called jitter.

Packet loss due to error or congestion: If an error happens or congestion in any point of the network packet may drop and if does not send again it will be lost.

Throughput or transmission rate: is the number of bits passing through the point of the network per second.

C. Real-time Case

This case studies when the user needs to know the information for this moment. So when he sends request he connects with the corresponding sensing mote and gets the data. So, it will be dealing with the MQTT domain and the ZigBee domain.

Unreal-time Case

This is when the user needs to get old information from the database. In this case, it needs to be connected only with the broker. And no need to be connected with the gateway or Z motes.

E. Experiment Procedures

The procedure for this experiment will be as following:

1. Set QoS level.
2. Send a different number of the messages starting from the small number of the messages, around two messages.

3. Increase the number of messages gradually.
4. Continue increasing until it reaches the crash system level.
5. Calculate the total average delay, jitter, packet loss and throughput.
6. Record the result for further analyses.
7. Repeat the steps from 2 to 6 for all QoS levels.

IV. ESULTS AND DISCUSSION

A. Network Design

Figure 5 illustrates the design elements, physical connection and logical connection.

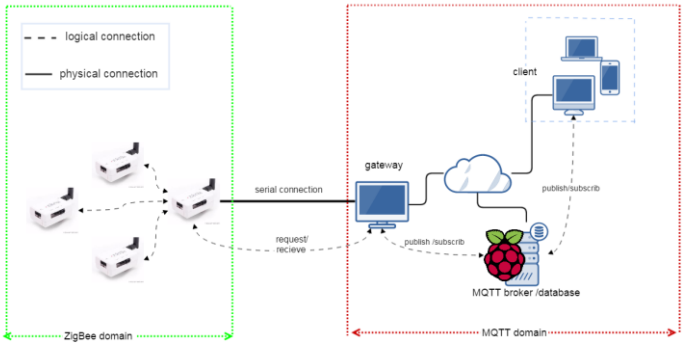

Fig. 5 Network design

B. Message Sequence

Figure 6 is the sequence diagram that shows the mechanism of forwarding messages in the network.

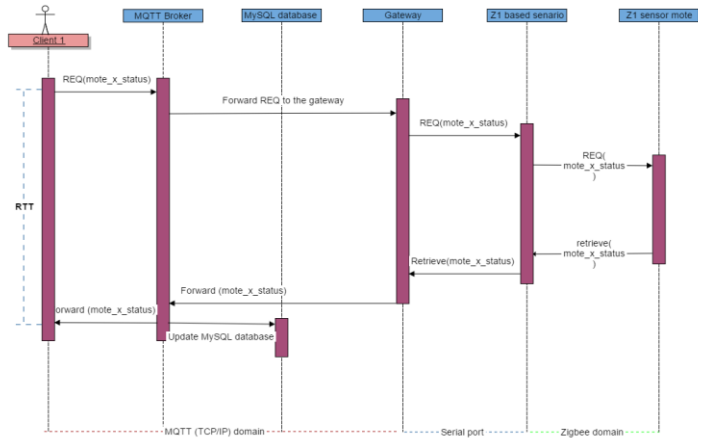

Fig. 6 Message sequence.

C. Exchanged Messages Size

- Publish request message sent by client to broker: the size of the message is 14-16 byte depending on identifier value.
- Publish data message from the gateway to broker and from broker to client: the size is between 54-56 depending on the identifier value.
- Connect message: the size is 39 byte.
- Acknowledgment message that sends for QoS1 and QoS2 only and connects acknowledgment. The length is 4 bytes.
- Disconnect message: the size is only 2 bytes.

D. Discussion

1) Delay

Figure 7 shows the average delay for non-real time scenario for the three levels of QoS. Overall, the delay increases dramatically when the number of messages increases. However, the delay in QoS2 is the highest because of the four-hand check that applied in QoS2 which increases the load on the network. Another reason why QoS2 has a higher average delay is that the server needs to wait for the acknowledgment. It is important to clarify that the delay of received messages only was calculated. The delay in QoS1 comes in second place because it waits only for one acknowledgment and the load in the network is less than QoS2. Third place is QoS0, it shows the best result for the delay because the load on the network is less and no need to wait for an acknowledgment. The lost messages were not included. Therefore, the delay drop is noticed when the number of messages gets over 5000 messages for QoS0 and QoS1, and 2500 for QoS2. If the delay of lost messages is considered, the trend will rise up to infinity. From Fig. 8 it can be seen that the QoS level has no impact on the delay. All QoS follow the same trend. This result is reasonable because in real-time, the network is working under the constraints of Z1 mote and ZigBee that limit the performance of MQTT. This could be understood more clearly when looking at the maximum number of messages that can be sent in both scenarios.

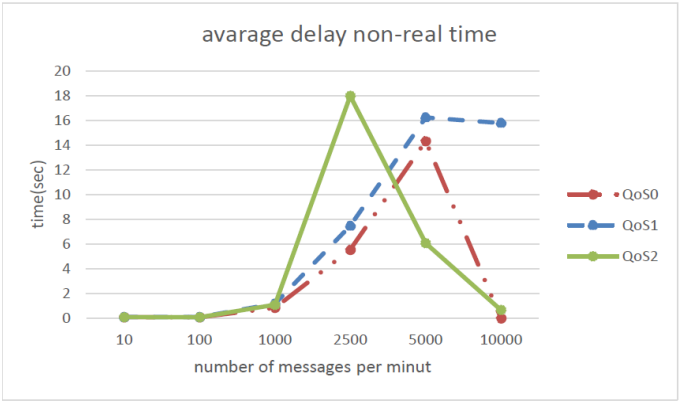

Fig. 7 Average delay non-real time

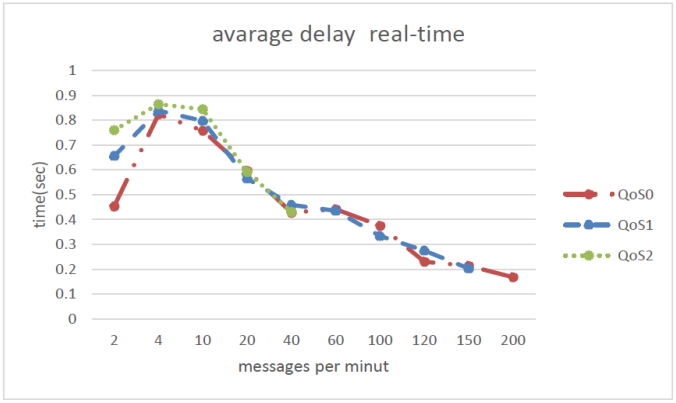

Fig. 8 Average delay real-time

2) Jitter

As a propagation jitter is not expected, this jitter is the processing jitter. The jitter is quit high for a large number of messages in non-real time case as shown in Fig. 9. But it fluctuates in the real-time case as shown in Fig. 10 with much smaller value than non-real time.

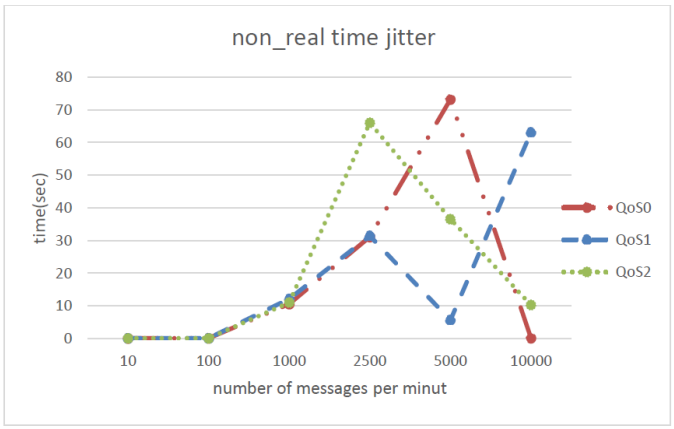

Fig. 9. Jitter non-real time

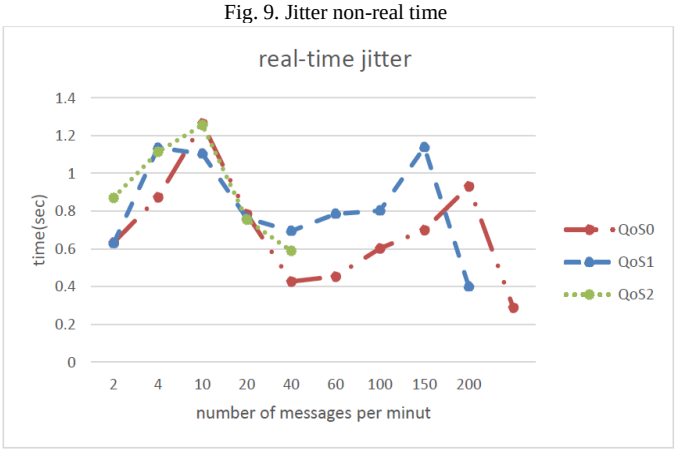

Fig. 10. Jitter real time

3) Throughput and Packet Loss

The packet loss diagram is the opposite of the throughput diagram. So, same analysis will provide for them. For non-real time analysis, as shown in Fig. 11 and Fig. 12, at a small number of messages (0-1000) the throughput is maximum at 100% while the packet loss ratio is equal to 0. After 1000 messages per minute, the packet loss ratio increases until the packet lost rich the pick at around 80%. While the throughput decreases until the packet lost rich the bottom at around 20%. In addition, QoS2 has the best throughput and less packet lost followed by the Qos1 and last QoS0. That is because of the reliability that ensured by the acknowledgment messages of QoS2 and QoS1.

For real-time analysis shown in Fig. 13 and Fig. 14, it is seen that the resulting change. QoS2 is not the best now. It derives the system to failure quickly because of the huge load on the network that cannot be taken by the Z1 mote. QoS1 shows the best performance relative to packet loss and throughput.

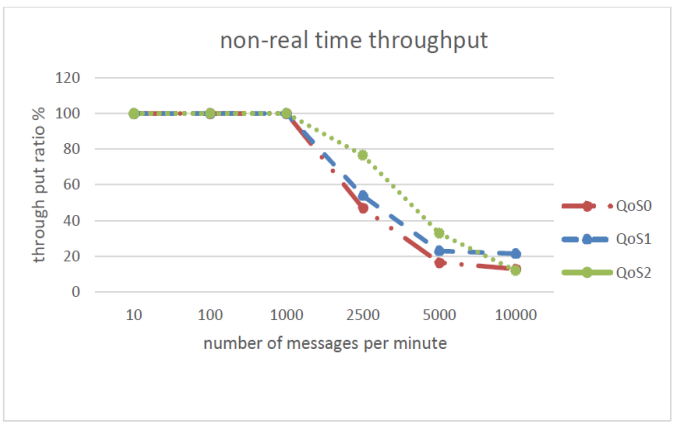

Fig. 11 Non-real time throughput

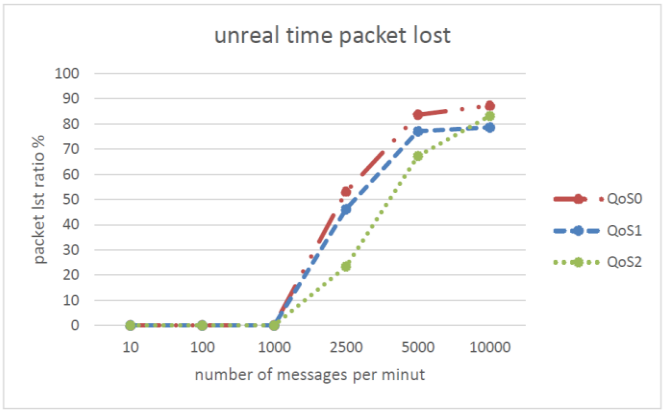

Fig. 12. Non-real time packet lost

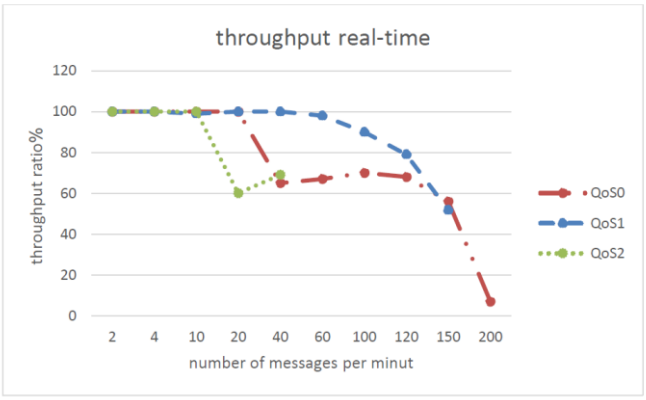

Fig. 13. Real time throughput

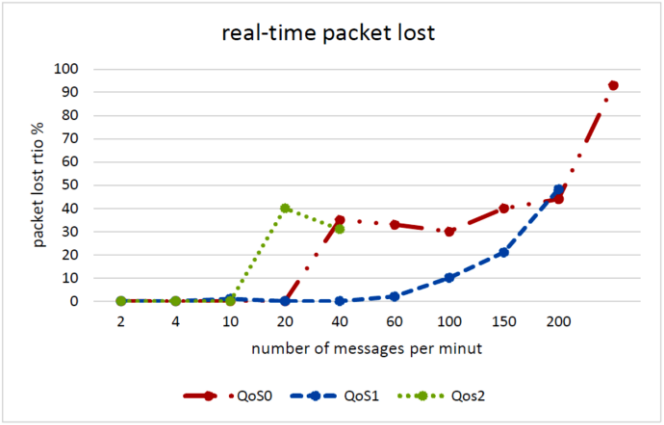

Fig. 14. Real time packet lost

Comparing with non-real time scenario, the number of messages that can be sent in real-time scenario is small. In non-real time, up to 1000 message could be sent before the packet lost start increasing while in real time scenario; only 60 messages could be sent for QoS0, 20 for QoS1 and 10 messages for QoS2. This because of the limitation of Z1 mote and the bottleneck occurs at Z1 base mote. Moreover, the system will fall down at 40 messages for QoS2 and 150 messages for Qos1. It gives better result in term of system fall for QoS0 at 200 messages. This gives the readers clear understanding of the limitation of Z1.

V. Future Work

For future work, it is planned to work on the recent network and codes to develop it in order to get better performance. Ones this is done, the best implementation design to be conducted with MQTT will be suggested. Moreover, it is planned to implement another protocol such as CoAP in the same scenario and compare the performance of all protocols. This will help the developers to define the strength area of application for each protocol.

This will be a considerable contribution in IoT implementation and performance.

Conclusion

From the results, it was shown that MQTT can work perfectly as a reliable real-time protocol for a small number of messages. The protocol loses its efficiency when the load of the network (number of messages) goes extremely high. Four different performance properties were analyzed. For real time scenarios, it was found that the QoS level has no impact on the delay. However, for non-real time scenarios, the delay is dramatically increased as the number of messages increased. QoS0 showed the best results of delay. Moreover, the processing jitter is much smaller for real time scenarios than non-real time scenarios. In terms of throughput and packet loss, QoS2 showed the best performance for non-real time scenarios, while for real time scenarios the QoS1 was the best.

References

[1] I. T. S. Sector, Recommendation ITU-T Y. 2060, “Overview of the Internet of things. Series Y: Global information infrastructure, internet protocol aspects and next-generation networks-Frameworks and functional architecture models,” 2012. Retrieved from https://www. itu. int/rec/T-REC-Y, 2060-201206.

[2] D. Bandyopadhyay, and J. Sen, “Internet of Things: Applications and Challenges in Technology and Standardization,” *Wireless Pers Commun*, vol. 58(1), pp. 49-69, 2011. http://dx.doi.org/10.1007/s11277-011-0288-5

[3] MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

[4] Postscapes.com, Internet of Things Examples – Postscapes, 2020. Retrieved 20 April 2020, from http://postscapes.com/internet-of-things-examples/

[5] S. Schneider, Understanding The Protocols Behind The Internet Of Things. Electronicdesign.com, 2013. Retrieved 20 April 2020, from http://electronicdesign.com/iot/understanding-protocols-behind-internet-things

[6] C. Karasiewicz, Why HTTP is not enough for the Internet of Things (The Mobile Frontier). Ibm.com, 2013. Retrieved 20 April 2020, from https://www.ibm.com/developerworks/community/blogs/mobileblog/date/201309?lang=en

[7] IoT Analytics - Market Insights for the Internet Of Things, The 10 most popular Internet of Things applications right now, 2015. Retrieved 20 April 2020, from http://iot-analytics.com/10-internet-of-things-applications/

[8] Mqtt.org, FAQ - Frequently Asked Questions | MQTT, 2020. Retrieved 20 April 2020, from http://mqtt.org/faq

[9] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks,” *In 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops* (COMSWARE'08), Jan. 2008, pp. 791-798, IEEE. https://doi.org/10.1109/COMSWA.2008.4554519

[10] R. Webb, A Brief, but Practical Introduction to the MQTT Protocol and its Application to IoT | Zoetrope, 2016. Retrieved 20 April 2020, from https://zoetrope.io/tech-blog/brief-practical-introduction-mqtt-protocol-and-its-application-iot