Adaptive publish time and QoS level over MQTT protocol

Nuttakit Vatcharatiansakul and Panwit Tuwanut
Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang.
1 Chalongkrung Rd., Ladkrabang, Bangkok, Thailand 10520
E-mail: ggguuyy@gmail.com , panwit@it.kmitl.ac.th

Abstract: Several application protocols have been proposed for Internet of Things (IoT) solutions, but the most widely used are the Message Queuing Telemetry Transport (MQTT) and the Constrained Application Protocol (CoAP). To ensure that the messages are transmitted accurately and reliability, both MQTT and CoAP support Quality of Service (QoS). There are three QoS levels in MQTT and four levels in CoAP. However, the characteristic of a medium such as signal attenuation, propagation delays, bit error rates and others affect to this transmission. In this paper, we proposed that QoS level should be adaptively adjusted according to the consideration of link conditions. Moreover, the periodic period times to publish/post message also adaptively adjust. Based on experimental results show that the proposed scheme can ensure message reliability.

1. Introduction

The Internet of Things (loT) is a concept connecting any device to any other device through the Internet. Currently, the most widely used application protocols in the loT are MQTT[1,2], CoAP[3]. Message Queue Telemetry Transport (MQTT) is implementations of publish-subscribe model, and the other such as Data Distribution Service (DDS)[4]. The pub-sub model can provide flexibility and scalability, more than one publisher can publish messages to a topic, and more than one subscriber can consume messages from a topic. The main advantage of the publish/subscribe model is that it allows messages to be broadcast to multiple subscribers. [5]

In some application, such as healthcare application, the accurate and reliable data is very importance which impacts every decision made along the patient care continuum. Hence, the QoS Level 2 of MQTT, is suitable for healthcare application which broker passes the message through exactly once to the subscriber after the broker successfully received the message from publisher (patient's health record) by apply 4-way handshake. In addition, more network traffic density is generated primarily by higher QoS level.

Due to the flow control and congestion control in TCP, traffic may be delayed and burst which is not suitable for real-time monitoring. In 2012, W. Kang and et al proposed RDDS [6] which is a real-time data acquisition over a pub-sub model by integrating a control-theoretic feedback controller at the publishers and a queueing-theoretic predictor at the subscribers. With the modification and the aid of the broker, RDDS accomplish pub-sub flow control. However, the subscriber will keep the state data per sensor nodes, which result in scalability issue in large scale network.

The goal of this paper is to present that a QoS level and period time to publish a message to broker should be adaptively adjusted. The publisher will compute round trip time (RTT) between the publisher and broker, then exponential smoothing is used to predict next RTT. If a next RTT increase, it's shown that the network congestion occurs and bandwidth is insufficient, hence the period time to publish should be increased. Moreover, after increase a publish time, next RTT remains increase, thus QoS level is decreased in order to reduce network traffic.

The outline of this paper will be arranged as follows. In section II, the overviews of MQTT protocol and QoS level are discussed where the proposed algorithm and simulation results are given in section III. Finally the conclusion is presented in section IV.

2. MQTT overviews

MQTT is a lightweight messaging application protocol designed to be open, simple, lightweight and easy to implement. It is based on the pub-sub architecture. MQTT devices do not connect directly with each other, but via a broker. When a client publishes a message M with a specific topic T to the broker, next, the broker receives a publishing, it forwards the message to the subscriber which subscribed to the topic T. then all subscriber will receive the message M. The design of a MQTT system is shown in Fig. 1.

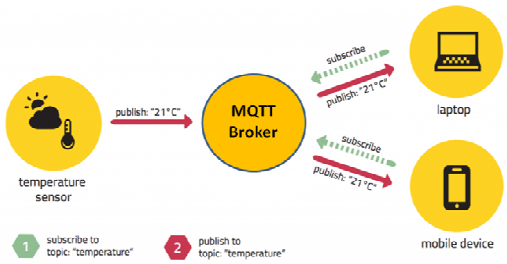

Figure 1. MQTT communication [7]

The reliability of messages in MQTT is taken by three Quality of Service (QoS) levels. In the QoS level 0, the sender only guarantees a message to be sent at most once. It guarantees a best effort delivery, no acknowledged by the receiver or stored and redelivered by the sender. Thus, the message can be lost while being delivered to the corresponding receiver. This is often called "fire and forget". The QoS level 1, it is guaranteed that a message will be delivered at least once to the receiver. A published message is stored in the publisher internal buffer until it receives the acknowledgment packet (PUBACK). When the acknowledgement is received, the message is discarded

from the buffer, and the delivery is complete. However, if the acknowledgment is lost, the message can be retransmitted multiple times.

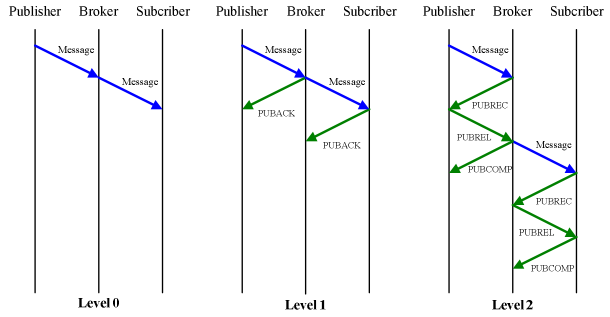

Figure 2. MQTT QoS levels

In the QoS level 2, the protocol guarantees that a published message will be delivered "exactly once". Neither loss nor duplication of messages are acceptable. A four-way handshake mechanism is used by exchanging control messages (PUBREC, PUBREL, and PUBCOMP). The problem associated with this level is the increased overhead, since the transmission of one message involves the interchange of four messages. This level is the safest and also the slowest quality of service level.

3. Proposed Scheme

In order to provide the accurate and reliable data in some IoT application, it prefers to use QoS level 2, hence, there is a trade off with a network traffic. The proposed algorithm depicts on Algorithm 1, The publisher will compute round trip time (RTT) between the publisher and broker (line 2-6), then exponential smoothing is used to predict next RTT (line 7). If a predict next RTT increase, it indicates that the network congestion occurs and bandwidth is insufficient, hence the period time to publish should be increased(line 8). Next, if a publish time increase, then a predicted next RTT still increases, thus QoS levels is decreased (line 12-18) in order to reduce network traffic.

Algorithm1 : Proposed algorithm on publisher

1: *period* ⇐ Initial , *QoS* ⇐ 2, *old_RTT*⇐ 0
2: *pub_time* ⇐ get_timestamp()
3: publish(message)
4: wait_PUBCOMP()
5: *ack_time* ⇐ get_timestamp()
6: *RTT* ⇐ ack_time - pub_time
7: *Predict_RTT* ⇐ smooting_exponential(RTT, old_RTT)
8: **if** *Predict_RTT* > *RTT* **then**
    period ⇐*period* × 2
9: **else**
    period ⇐*period* / 2
10: **end if**
11: *old_RTT* ⇐*RTT*

12: *pub_time* ⇐ get_timestamp()
13: publish(message)
14: wait_PUBCOMP()
15: *ack_time* ⇐ get_timestamp()
16: *RTT1* ⇐*ack_time - pub_time*
17: *Predict_RTT1* ⇐ smooting_exponential(RTT1, RTT)
18: **if** *Predict_RTT1* > *RTT1* **then**
    QoS =2 ⇐*QoS =1*
    wait_PUBCOMP()⇐wait_PUBACK()
19: **else**
    Qos = 2
20: **end if**
21: *RTT* ⇐*RTT1*

4. Experimental Work

4.1 Experimental set up

In this section, the experiment was set up to verify the relation between the network traffic and a packet loss in any QoS level. Next, we implement an algorithm to verify the proposed scheme. We installed Raspbian and Moquitto on Raspberry Pi 2 Model B which has a specification as a 900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM and Ethernet port. As shown in Fig. 3, the experimental test-bed is composed of four Raspberry Pi that are connected using a hub with 10Mbps speed. The first one act for a publisher, the next represent to a subscriber and MQTT broker and the last one stand for background traffic generator.

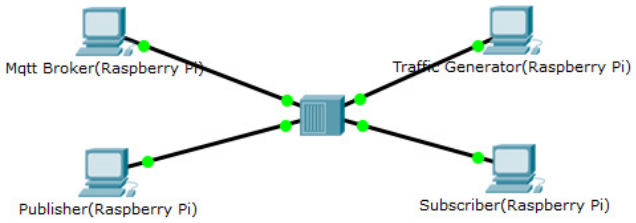

Figure 3. Experimental test-bed

4.2 Experimental result and analysis

In the experiment, we start examining the effect of traffic from no background traffic to 25%, 50% and 75% bacgroud traffic of available bandwidth, then the effect began to appear as display in Fig 4. In Fig 4, the end to end delay without background traffic are illustrated with the difference QoS level. From this figure, we can see that the high level of QoS has more delay than the lower level. For the real time application such as voice over IP, the standard values are displayed on Fig 5. [8] The acceptable of end to end delay is less than 100 ms.

Next, end to end delay, jitter and packet loss with and without a background traffic is illustrated in Table1-3, and Fig 6. and Fig 7., consequently.

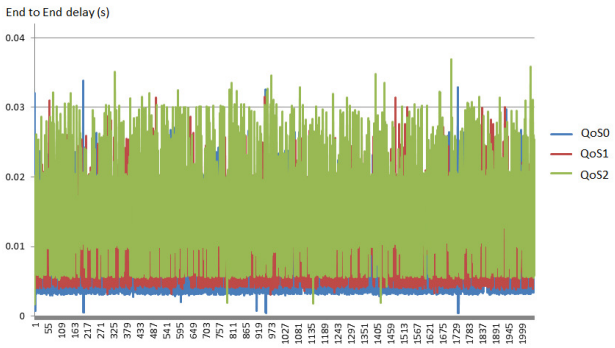

Figure 4. Delay without background traffic

Traffic Class	Technology Attributes	QoS Metrics						
		Timeliness			Preciseness			Accuracy
		Response time Expected by Users	Delay (ms)	Jitter (ms)	Data Rate (bps)	Required Bandwidth (bps)	Loss Rate	Error Rate
Voiceover IP	Real Time and Symmetric	Phone to Phone Delay: <150 ms	<100	<400			< 1%	< 1%
Coding Standard								
G.711					64 K	80 K		
G.726					40~16 K	50~22 K		
G.728					16 K	22 K		
G.729					8 K	11 K		
G.723.1					6.3/5.3 K	9 / 8 K		
GSM FR					13 K	18 K		
GSM EFR					12.2 K	17 K		
Network Capacity								
Link: Refer to Appendix I					Router: Refer to Appendix II			

Figure 5. A QoS matric for VoIP [8]

QoS level	Average end to end delay (ms)			
	No background traffic	Background traffic 25%	Background traffic 50%	Background traffic 75%
QoS0	8.933025	10.924081	11.21839396	19.70615905
QoS1	11.448357	125.1881596	192.09104	263.611665
QoS2	15.465627	243.6281809	-	-

Table 1. Average end to end delay

QoS level	Jitter (ms)			
	No background traffic	Background traffic 25%	Background traffic 50%	Background traffic 75%
QoS0	0.0358	31.44795	92.34556993	135.8020646
QoS1	0.0387	1820.373202	2049.110944	5436.319233
QoS2	0.0515	2923.538348	-	-

Table 2. Jitter

QoS level	Packet loss (%)			
	No background traffic	Background traffic 25%	Background traffic 50%	Background traffic 75%
QoS0	0	0	0	0
QoS1	0	8.42	24.30	35.33
QoS2	0	0	100	100

Table 3. Packet loss

From the results in table 1-3., they indicate that when the background traffic increase, it is the most affect to the QoS in end to end delay, jitter and packet loss. Due to the ACK packet loss between the publisher and subscriber, while in the case of Qos level 0 which doesn't have ACK packet, the payload directly sends to the subscriber, background traffic affect a small delay but no packet loss.

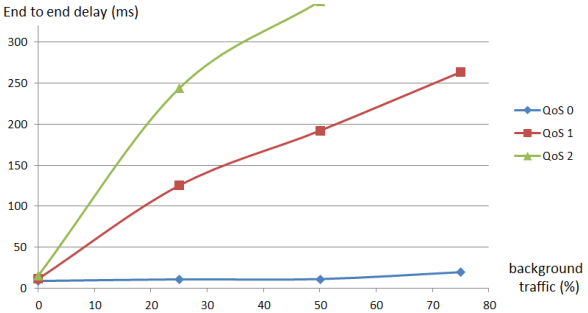

Figure 6. Average end to end delay at a difference QoS level

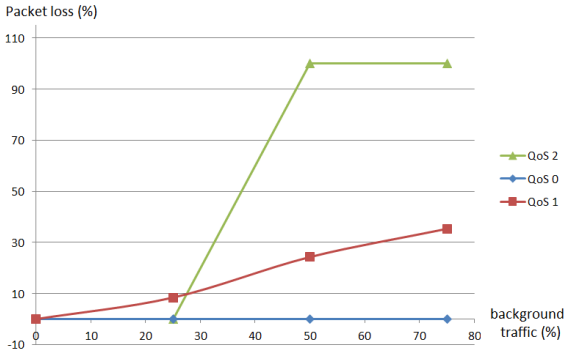

Figure 7. Packet loss at a difference QoS level

Next, round trip time is used to calculate before publishing a payload to a subscriber, as shown in algorithm 1. The result of this algorithm is displayed in Fig 8. and Fig 9. In the Fig 8., is the end to end delay of QoS 2 when the background traffic about 25% without apply a proposed algorithm, but in the Fig 9., is the end to end delay when apply the proposed algorithm.

From the experimental result, we found that if background traffic enlarge, it results to increase round trip time, then QoS level change to the lower level and effect to reduce end to end delay and reduce a packet loss. With this algorithm, the end to end daly down from 200ms to 8 ms and packet loss down from 10% to 0%.

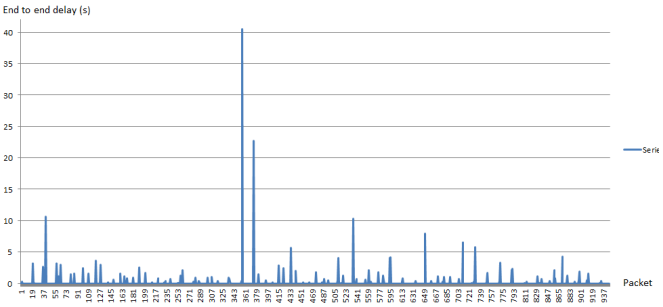

Figure 8. End to end delay on QoS level 2., in normal state.

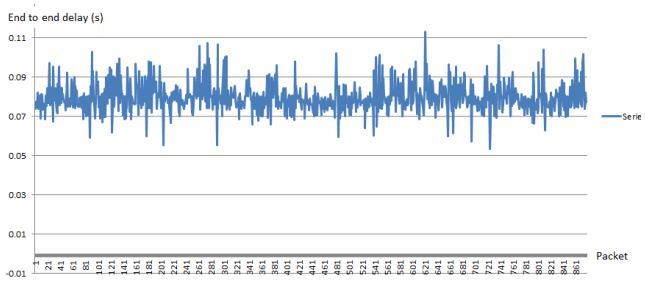

Figure 9. End to end delay when apply an algorithm.

The future work , we will study real time protocol such as DDS protocol and compare the performance to each others. Moreover, we will apply to use an adaptation the Qos level over MQTT protocol over a personal area network such as Zigbee or Bluetooth.

5. Conclusion

In this paper, we proposed that the QoS level and period time to publish a message to broker should be adaptively adjusted, based on a current network traffic which obtained by the round trip time (RTT). This proposed algorithm no need to modify the broker and a subscriber won't keep the state of the publisher. Only the publisher will adapt a period to publish a message and QoS level to reduce network traffic, hence it eases to implement in a real-time IoT application with the accurate and reliable data.

References

[1] D. Locke, "MQ Telemetry Transport (MQTT) V3.1 Protocol Specification," 2010.

[2] OASIS Standard, MQTT version 3.1.1, Oct. 2014.

[3] Z. Shelby, Sensinode, K. Hartke, C. Bormann, and U. B. TZI, "Constrained application protocol (coap) draft-ietf-core-coap- 17," http://tools.ietf.org/html/draft-ietf-core-coap-17, 2013.

[4] G. Pardo-Castellote, OMG data-distribution service: architectural overview, In Proc. of ICDCS Workshops, 2003.

[5] https://docs.oracle.com/cd/E19587-01/821-0028/6nl41ccqd/index.html

[6] W. Kang, K. Kapitanova, and S. H. Son, RDDS: A Real-Time Data Distribution Service for Cyber-Physical Systems, IEEE Transactions on Industrial Informatics, 8(2):393-405, May 2012.

[7] Images Fig 1. Available at: www.hivemq.com

[8] Yan Chen, Toni Farley and Nong Ye, "QoS Requirements of Network Applications on the Internet", Journal Information-Knowledge-Systems Management. Volume 4 Issue 1, January 2004 Pages 55 - 76