Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 76 (2015) 400 – 405

Procedia Computer Science

2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015)

A Comparison of Lightweight Communication Protocols in Robotic Applications

Muhammad Harith Amaran*, Nazmin Arif Mohd Noh, Mohd Saufy Rohmad, Habibah Hashim

Faculty of Electrical Engineering, Universiti Teknologi MARA UiTM Shah Alam, 40450, Malaysia

Abstract

Robot communication is an essential element in robot operation. There are several protocols that serve robot communication applications, but the protocols used are not optimized for mobile and battery-operated robots. To operate in such condition, an optimized protocol for such environment must be used. This paper will evaluate Constrained Application Protocol (CoAP) and MQ Telemetry Transport for Sensor Nodes (MQTT-SN) which are designed for such devices. Result from experiment shows that MQTT-SN performs 30% faster than CoAP when transmitting the same payload.

Peer-review under responsibility of organizing committee of the 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015)

Keywords: Robotic; Robot; Communication Protocol; CoAP; MQTT-SN; Internet of Things

1. Introduction

Wireless mobile robot used multiple approaches to communicate. One example is implementation by Robot Operating System (ROS) which used a modified TCP/IP protocol stack called Transmission Control Protocol for Robot Operating System (TCPROS) to communicate[1]. As for the application protocol, ROS use XML-RPC (Extensible Markup Language-Remote Procedure Call) on top of HTTP to send data to and from the robot. HTTP is

* Corresponding author.
E-mail address: muhhammadharith@salam.uitm.edu.my

Peer-review under responsibility of organizing committee of the 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015)
doi:10.1016/j.procs.2015.12.318

slow[2] and the size of HTTP header is large which make it consume a lot of power. These elements make it unsuitable for application in mobile battery-operated robot.

There are several applications in robot which involve communication. One application is robot control. Remote controlled robot requires real-time response so that it can be operated properly. A delay in communication may result in an undesired outcome and can be dangerous if it involves heavy or fast moving robots. An example of controlled robot which depends on real time response is the *da Vinci*, a remote operated surgery robot[3]. The robot moves in accordance to the control of a surgeon that may be located hundreds of miles away. To achieve better response time, lightweight communication protocol must be used. This paper will discuss Constrained Application Protocol and MQ Telemetry Transport, which claim to be lightweight protocols.

2. Lightweight Communication Protocols

2.1 CoAP

Constrained Application Protocol (CoAP)[4] was co-developed by Internet Engineering Task Force (IETF) and ARM. It was standardized in June 2014 (RFC 7252). CoAP was developed for devices constrained in terms of processing capabilities and power. CoAP uses UDP as the underlying transport layer. This ensured minimum packet size as UDP only use 8 bytes as packet header. On top of that, CoAP only add 4 bytes as the application header. It is Internet Protocol (IP) based which means that it use the standard IP stack and is natively compatible with the internet. The RFC, however stated that it is intended to only support IPv6 and not IPv4. The RFC proposed that CoAP can be secured through the use of Datagram Transport Layer Security (DTLS) or the datagram counterpart of Transport Layer Security (TLS) which is widely used by web browsers to provide secure HTTP.

CoAP use Representational State Transfer (REST) which is a communication model shared with HTTP. Since CoAP is based on REST it operates similar to HTTP, which is by request and response. The same infrastructure made it easy for CoAP to interoperate with HTTP without the need of a complex translator.

In the standard installation, CoAP supports two modes, reliable and non-reliable. Reliability in CoAP is handled by using Confirmable (CON) messages. For every CON message sent to the server, this will be replied by Acknowledge (ACK) packages to the client. CoAP also support “Piggybacked Response” ACK message. This means that an ACK can be included in the response message of another message to further reduce communication.

2.2 MQTT

MQ Telemetry Transport (MQTT)[5] was formerly developed by IBM and then released to the open source community. The latest standard for MQTT as of writing is version 3.1.1 and standardized by OASIS. MQTT operates based on publish-subscribe mechanism where a subscriber subscribes to a topic published by a publisher to the broker.

To communicate, a publisher must first connect to the broker. After successful connection denoted by a reply from broker, the publisher must then register itself to the broker. After successful registration, the client publishes the topic and message to the broker. To end the session a disconnect packet is sent to the broker. If there are no disconnect packets or any packet received by the broker, the publisher will be regarded as disconnected after a predefined time. Registration is only done once. Subsequent connection only need a connect packet followed by the publish message.

Whenever a publish message to a topic was sent, message will be sent to all subscribers of that particular topic by the broker. Communication between publishers and subscribers of a specific topic only occur through the broker. Hence peer to peer communication between subscriber and publisher in MQTT is impossible.

For reliability, MQTT offered three types of modes which are called Quality of Service (QoS). QoS 0 mode sends a packet only one time without requiring confirmation messages or ACK. QoS 1 mode ensure that the message is delivered at least once by requiring an ACK. QoS 2 mode guarantee that the message is delivered exactly once. For MQTT, TCP is used for the transport protocol. This means that even though QoS 0 is used which does not require MQTT ACK responses, TCP still provide TCP ACK for every package sent.

2.1 MQQT-SN

MQ Telemetry Transport for Sensor Nodes (MQTT-SN)[6] is not only a variant of MQTT but a protocol focused on constrained devices. Formerly known as MQTT-S prior to December 2013, the name was changed due to confusion that MQTT-S meant secure MQTT. According to the specification the protocol was based on MQTT but modified for lossy wireless network, sleeping device, low power, low bandwidth, and constrained devices.

Unlike MQTT, MQTT-SN can use UDP as the transport protocol although the specification does not limit the protocol to be used only on UDP. The advantage of using UDP is removing the handshakes of TCP and made the communication connectionless thus making it more lightweight.

Apart from using a different transport layer, MQTT-SN also introduces MQTT-SN gateway (GW) and forwarder. The gateway function as a translator which transforms MQTT-SN packets to the standard MQTT packets. The gateway then connects to the broker using TCP connection if it is located on a different device or location. There are two types of connection between gateway and the broker. The first is transparent and the second is aggregating. Transparent gateway means every single MQTT-SN will have their own TCP connection to the broker. Aggregating gateway however only have a single TCP connection to MQTT broker. All the MQTT-SN connections will then be tunneled using the single connection.

3. Related Works

There are some papers that have already discussed the difference between CoAP and MQTT and compare the performance between both of them. As been discussed earlier, CoAP and MQTT use different transport layer (UDP vs TCP) therefore the result is uncertain whether it is because the protocols themselves or from the usage of different transport layers. Here we show some of the findings from other papers.

An experiment which checks the delay caused by loss in the network shows that when the loss rate is at 20% or lower, MQTT performs better than CoAP with delay of less than one second for every message[7]. But at 25% MQTT performs very bad with delay of more than 10 seconds for a successful communication. At 25% loss, CoAP shows delay of 2.8 seconds which is far better than MQTT[7].

Another factor that has been studied is power consumption. A study[8] shows that CoAP only require 40% of power normally used compared to HTTP. This is true when the message is sent once for every hour. Another study[9] shows MQTT power consumption is 10% less compared to HTTP per hour of operation. However, the messages sent by MQTT in one hour is ten times more than HTTP. This means that MQTT not only reduce power consumption, but also much faster than HTTP. This shows that using both MQTT and CoAP is more power efficient than HTTP. Another paper[10] shows that CoAP is indeed the more power efficient compared to MQTT.

Bandwidth consumption is also one of the factors discussed. An experiment[11] shows the bandwidth used for sending one message with reliability for both protocols. CoAP leads the result when using only 140 bytes total (including link, network and transport layer header). MQTT with QoS-1 is second with 162 bytes while MQTT with QoS-2 requires 318 bytes total. Generally, result shows that CoAP and MQTT consumes low bandwidth.

It is presumed that the advantages in result of CoAP is contributed by the use of UDP as transport layer. In this sense CoAP should be compared to a protocol using similar transport layer, such as MQTT-SN. However, there are hardly any work that compare CoAP to MQTT-SN. Therefore, we suggest that more experiment should be done to compare CoAP and MQTT-SN, not MQTT.

4. Protocol Comparison

CoAP and MQTT-SN both can use UDP as the underlying protocol. Using the same transport layer, any differences in performance between both protocols thus come from the design of protocol. In the section below we analyze the differences between both protocols.

4.1 Header Size

The main difference and functions of a communication protocols reside in the structure of header size. One of the reason CoAP and MQTT-SN claims to be a lightweight protocol is because of the size of the header size they need

for every transmission. CoAP claims it needed of 4 bytes for a packet header while MQTT-SN needed only 2 bytes for the header. Lightweight protocols enable shorter header by using binary as options instead of character strings converted to binary such as in HTTP. The following are the headers for CoAP and MQTT-SN.

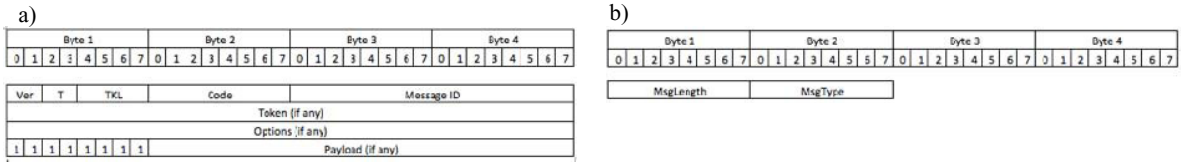

Fig. 1: (a) CoAP Header; (b) MQTT-SN Header

Figure 1(a) shows the CoAP header is started by 2 bits showing the version of the CoAP Protocol. The next 2 bits shows the type of message whether it is CON, NON, ACK or RST denoted by 00,01,10, or 11. Following that are Token Length. Code and Message ID fields. Message ID is unique for every message. Message ID can be used for unique identification and also same Message ID for ACK. The last field of the header is the Options field and finally the header and payload is separated by 0xFF. The specific usage of the fields will be discussed in the next section.

Figure 1(b) shows the basic MQTT-SN header is only 2-bytes long. The first byte contains Message Length information and the second byte contains the information about the type of message. This is the most basic header for MQTT-SN. We can see later that the header structure for each type of message is different for each type of MQTT-SN message. Below is an example of a publish packet header.

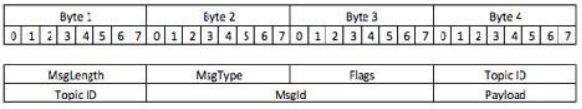

Fig. 2: MQTT-SN PUBLISH Header

We can see from Figure 2 that the real size of a publish header is bigger than two bytes. The size of the PUBLISH header is effectively seven bytes. Following a two byte basic MQTT-SN header is the Flags option. After that is Topic ID and Message Id which each takes two bytes of space. After the Message ID comes the Payload.

4.2 Reliability

Both CoAP and MQTT-SN both support reliable and unreliable mode of transport. For CoAP, reliable transport is accomplished using confirmable (CON) flag. The confirmable message sent by client require an ACK reply by the server. MQTT-SN support reliable transmission through the use of QoS 1 and 2. QoS 1 operates in a similar manner to CoAP CON message by requiring an ACK for message sent. QoS 2 offers reliability in a similar manner to QoS 1 but with an added feature. A QoS 2 message will make sure that a message will only be delivered once.

4.3 Duplication

To ensure message is not duplicated CoAP implements message ID for every message. Message ID is a random 2-byte length number given unique to every message save for ACK messages. ACK will have similar Message ID similar to the message it is intended to acknowledge. For MQTT, duplication is negated by using Message ID and DUP flag. Every QoS 1 and 2 will contain a 2-byte message ID, but QoS 2 feature DUP flag that will indicate if the message is being sent for the first time.

4.4 Resource Locator

Resources are managed by different conventions in MQTT-SN and CoAP. MQTT-SN manage resources by using conventions named topics and messages. Topics can have subtopics and they are case-sensitive. All topics are stored on the original device and the broker. CoAP on the other hand stores resources on the device itself. Resources on CoAP are similarly managed to HTTP URI. A resource on a CoAP devices can be located by using the format: *"coap://<ip.address>:<port>/location/to/resources"*. Similar to HTTP, CoAP URI is case insensitive. Although they use different conventions, both MQTT-SN and CoAP support similar multiple hierarchy resource location.

4.5 Communication Mode

MQTT-SN communicates based on Publish-Subscribe model. A publisher pushes data related to a topic through the gateway to the broker. The broker saves the topic and send data to any subscriber to the topic. Before publishing the data, the broker requires the subscriber to connect to broker and register itself. Registered devices will then only need to send connect packet before sending the actual publish packet.

CoAP communicates based on REST architecture and support four modes of REST which are POST, PUT, DELETE and GET. A client sends a request with any one of the four modes and the server will response with a reply in accordance to the request.

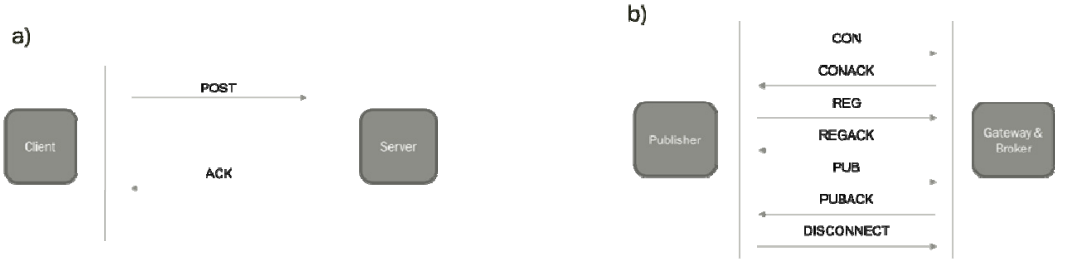

Fig. 3: (a) CoAP Communication; (b)MQTT-SN Communication

5. Experimental Setup

The devices used in the experiment are a Raspberry Pi B (ARMv6 processor) connected to the network through Local Area Network Ethernet cable. This device has been configured as a client device sending data to the server. On the other end, an Intel Xeon powered computer serve as the server connected to the network through the Ethernet. The CoAP test was performed using *libcoap* c-library as client and *microcoap* c-library for the server. In the case of MQTT-SN, the test was implemented using Really Small Message Broker (RSMB) for both gateway and broker. For the publisher and subscriber, a Basic MQTT-SN client library by Nicholas Humfrey was used. To emulate similar environment for CoAP that only involve client and server, the client (Raspberry Pi) was set as the publisher, while the server contains all Gateway, Broker, and Subscriber in the same device.

6. Results & Discussion

An experiment comparing CoAP and MQTT-SN was performed by sending 10,000 consecutive messages with reliability (CON and QoS 1). CoAP POST was use to send a text data and the similar text was also sent using MQTT-SN publish method.

It was found that the average time for message transmission using MQTT-SN is 4.38 millisecond. The first message, however, took 14.3 millisecond. This value is relatively higher than the average time. The higher value is probably due to the registration process done the first time a publisher connects to the broker. As shown before in Figure 3, MQTT-SN similar to MQTT need multiple steps to publish message reliably. After successful registration the subsequent messages will only involve connect, publish and disconnect messages thus resulting in faster transmission time. On the other hand, it was found that the average time for message transmission using CoAP is 6.35 millisecond. The first message transmitted took only 5.9 milliseconds which is not far from the average time. Result shows that there is no significant difference in time for every message sent by CoAP.

The experiment also shows that MQTT-SN performs 30% better than CoAP when it comes to average time. Figure 4 shows the compared result side by side. The result may relate to how these protocols were designed. Further experiments must be conducted to compare other segments of communication such as throughput and power consumption. This initial result shows that MQTT-SN works as well as MQTT even without using TCP. Without TCP, it is foreseen that MQTT-SN will not inherit the same problems that occurred in MQTT such as sever delay in lossy networks.

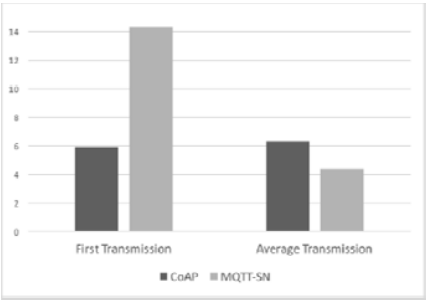

Fig. 4: CoAP vs MQTT-SN Transmission Time

7. Conclusion

CoAP and MQTT-SN both shows promising result in terms of performance as a communication protocol. Both protocols have been shown to be viable alternatives as replacement protocols for mobile and energy-aware robotic communication. Between the two, MQTT-SN shows a 30% better transmission time over CoAP. Further tests have to be done on MQTT-SN to investigate other variables of communication and in different types of robotic applications. As future work, it is proposed to secure the protocol by integrating the MQTT-SN with security utilizing existing Connect and Register method.

Acknowledgements

This project was made possible by funds from Malaysian National Research Grant Scheme (NRGS) 600-RMI/NRGS 5/3 (5/2013) from the Ministry of Education, Malaysia.

References

1. Keppmann, Felix Leif, Maria Maleshkova, and Andreas Harth. Building REST APIs for the Robot Operating System–Mapping Concepts and Interaction, 2015.
2. Zhang, Jingsong, and Robert D. McLeod. A UDP-based file transfer protocol with flow control using fuzzy logic approach. Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on. Vol. 2. IEEE, 2003.
3. Bodner, J., et al. First experiences with the da Vinci™ operating robot in thoracic surgery. European Journal of Cardio-thoracic surgery 25.5 2004: 844-851.
4. Shelby, Zach, Klaus Hartke, and Carsten Bormann. Constrained Application Protocol. RFC 7252. URL: https://datatracker.ietf.org/doc/rfc7252. 2014.
5. MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. 2014.
6. Stanford-Clark, A., & Truong, H.. MQTT for sensor networks (MQTT-SN) protocol specification. http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf. 2013.
7. Ma, X., Valera, A., Tan, H., & Tan, C. K. Performance Evaluation of MQTT and CoAP via a Common Middleware, 2014, 21–24.
8. Levä, T. Comparing the cost-efficiency of CoAP and HTTP in Web of Things applications, 2014. 1–41.
9. N. Stephen Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android. http://stephendnicholas.com/archives/1217. 2015
10. Caro, N. De, Colitti, W., & Steenhaut, K. Comparison of two lightweight protocols for smartphone-based sensing, 0–5. 2013.
11. Bandyopadhyay, S., & Bhattacharyya, A. Lightweight Internet protocols for web enablement of sensors using constrained gateway devices. 2013 International Conference on Computing, Networking and Communications (ICNC), 334–340. 2013.