Performance Evaluation of Secure CoAP and MQTT Communication on Resource-Constrained Devices

Thomas Wiss and Stefan Forsström
Department of Information Systems and Technology
Mid Sweden University
Sundsvall, Sweden
Email: thwi1500@student.miun.se, stefan.forsstrom@miun.se

***Abstract*—One important consideration for the Internet of Things is protecting sensitive data communicated from resource-constrained devices. In this paper, we investigate the performance of the Constrained Application Protocol and the Message Queue Telemetry Transport protocol in combination with and without their respective transport layer security. The testbed consisted of Raspberry Pi devices as representatives of resource-constrained devices and the evaluation identified that the processing of certificates and the establishment of a secure connection represents a heavy burden to the devices. Furthermore, the results show varying performances between the protocols when evaluated under lossy network conditions.**

***Keywords*—Internet of Things, Security, CoAP, MQTT, DTLS, SSL, TLS, Raspberry Pi.**

I. Introduction

The Internet of Things (IoT) is a movement towards more connected devices and smart services which have launched a new era in the digitalized world [1]. Creating a value chain from the gathered sensor values of connected sensor systems, to data mining of information, and finally controlling of different actuators depending on the collected information. The scale of the IoT is expected to reach 50 billion connected devices by the end of 2020 [2]. At the same time, there exist numerous approaches, architectures, and protocols to enable this IoT [3]. One prominent architectural approach is to combine the almost infinitely available computing capacity of cloud systems with resource-constrained sensors deployed in various environments. For example, enabled by Representational State Transfer (REST) [4] technologies, the Constrained Application Protocol (CoAP) [5], or the Message Queue Telemetry Transport protocol (MQTT) [6]. Another architecture introduces a middle layer and is referred to as fog computing [7], in resemblance to a low-lying cloud system close to the ground and end user. The system in a fog computing environment is often built up by using single board computers which are somewhat limited in computational power and energy supply, but still more powerful than the deployed sensors. By doing so, the approach aims to relieve the cloud backbone of some of its responsibilities and traffic volume.

Regardless of the protocol or the architecture, one important consideration for the IoT is the protection of sensitive data communicated to and from resource-constrained devices [8]. For example, the gathered sensor data can be commercially valuable industrial data which must be protected, highly sensitive information of a person's whereabouts, or illicit controlling of actuators. To this end, there exist many different means for enabling secure communication. Most commonly for the IoT is Hypertext Transfer Protocol Secure (HTTPS) [9], Transport Layer Security (TLS) [10], Secure Sockets Layer (SSL) [11], and Datagram Transport Layer Security (DTLS) [12]. In many scenarios, the deployed sensor systems are fairly constrained devices with limited processing power and battery life, which makes a secure data transmission an expensive operation. Beyond that, end applications often require a rather low latency to ensure a high Quality of Service (QoS). The necessity of these features is important in numerous situations, such as transmitting business critical or user-oriented data. Furthermore, since most IoT devices are small and mobile they often communicate via wireless networks to avoid unnecessary wiring. However, this also makes them subject to poor wireless links and lossy networks which can drop data packets.

It is therefore important to evaluate the performance of the two most prominent protocols which are used for the IoT, namely MQTT and CoAP, especially in regard to their secure data transmission under different network conditions. Hence, the objective of this paper is to identify and highlight the additional expenses of a secure communication in comparison to an ordinary insecure data transmission. The authors perform this by studying the following research questions:

1) When comparing secure and insecure communication of IoT data, what is their respective consumed bandwidth and latency in an ideal network setup?
2) What is the performance and capacity of the IoT protocols in a scenario with an error prone and lossy wireless network?

The remainder of this paper is organized in the following way. Section II presents relevant related work and Section III presents our scenario and approach. Section IV presents our testbed and evaluation setup, and Section V presents the results from the evaluation. Finally, Section VI presents our conclusions and future work.

II. Related Work

MQTT is a Transmission Control Protocol (TCP) based application layer protocol which applies a publish/subscribe pattern to provide subscribers of a specific topic with information about occurred events from a publisher. The setting is centered by a broker which coordinates the information and message exchange between a publisher and subscribers. The MQTT protocol allows a developer to set three QoS levels reaching from "at most once" (QoS 0), "at least once" (QoS

1) to ”exactly once” (QoS 2). To establish a secure connection between a broker and a publisher/subscriber the cryptographic protocol of TCP, SSL/TLS, can be applied.

CoAP is a User Datagram Protocol (UDP) based application layer protocol which applies a conventional client/server design. Due to the inherently unreliable nature of UDP, the reliability levels of CoAP consist only of two settings: The CoAP ”Non-Confirmable” (NON) message is often referred to as ”fire and forget” mechanism which doesn't require an acknowledgement, while the CoAP ”Confirmable” (CON) message requires exactly one return message of the type acknowledgement or the type reset. Secure connections can be established by adapting DTLS, a transport layer security for UDP-based messages.

Thangavel et al. [13] applied a self-designed and self-implemented middleware which acts as a gateway between the sensor nodes and the server, to evaluate the performance of MQTT and CoAP in regard to the end-to-end delay and the bandwidth consumption. To simulate a wide area network, an emulator was inserted in between the publisher/subscriber and the server/broker. The results of their evaluation indicate that the performance of the protocols vary depending on the network conditions. DeCaro et al. [14] compared the two IoT protocols MQTT and CoAP in a qualitative manner by deploying the sensing part of the structure on a smartphone. One conducted task performed the evaluation of the utilized bandwidth to establish a connection to the server/broker under different reliability/QoS settings. Another measurement was undertaken to detect the round-trip time for MQTT and CoAP transactions. Their findings show that CoAP performs better in case of the utilized bandwidth as well as requiring a shorter round-trip time. They also evaluated packet received ratio in a network that offers a random packet loss ratio of 20%, which results indicate that CoAP performs worse than MQTT in a notification interval of five seconds per message. A performance evaluation of MQTT, CoAP, Data Distribution Service (DDS), and a custom UDP-based protocol has been conducted by Chen et al. in [15]. It involves the transmission of collected health related data of a person to a server/broker traversing a simulated constrained wireless network. The evaluation covered measurements of the bandwidth consumption, the experienced latency and packet loss by varying the delay, packet loss rate, and offered bandwidth of the network. Their conclusions state that both MQTT and DDS perform excellent in lossy network conditions in terms of zero packet loss for the application. They further conclude that CoAP and the custom UDP-based protocols are better suited for the utilization in low bandwidth and low latency scenarios due to their smaller communication overhead.

To the best of our knowledge no previous study evaluated the two protocols MQTT and CoAP in regard of the consumed bandwidth and latency when their respective secure data transmission modes are applied. Our intention of this study is to eliminate this shortcoming, as well as to evaluate the suitability of a secure data transmission on resource-constrained devices in lossy wireless network environments.

III. Scenario and Approach

There exist various application areas for IoT, yet the focus of this paper lies on a smart environment in an industrial plant. In heavy industry surroundings, the environment to transmit data from sensors, or to control actuators, can be harsh and the available wireless network is most likely error prone. The occurrence of radio deflecting particles in the air, moving machine components, obstacles in the direct line of sight, and other obstructions can cause delay and packet loss in the network. To this end, we investigate a scenario consisting of a client publishing sensor values to a subscriber through a broker, where all communication is carried out in a lossy network environment. To further emphasis a fog computing environment, the experiment was solely executed on resource-constrained devices over wireless connections.

IV. Evaluation Setup

To simulate the lossy network condition the tool NetEM [16], which is available on Linux distributions such as the utilized Raspbian operation system, was applied. The consumed bandwidth has been measured with the packet analyzer Wireshark [17]. To ease the time recording for the round-trip measurement, the MQTT publisher and subscriber were deployed on the same device. This deployment option can have an application scenario where the fog node controls multiple sensors and actuators, latter acting according to subscribed events. To be able to make a fair comparison between the two protocols, the CoAP reliability was set to CoAP CON (the receiver acknowledges each message) and the MQTT QoS level was set to 1 (the sender stores the message until it gets acknowledged by the receiver, however, multiple message deliveries are possible) throughout the entire experiment. The payload of each request and response consisted in all cases of a single symbol in the form of an integer.

A. Hardware Setup

The testbed, displayed in Figure 1, includes the following devices: a wireless router type ASUS RT-AC66U, a Raspberry Pi Model B+ with a Comfast wireless network dongle to host the server/broker application, a second Raspberry Pi Model B+ with a Comfast wireless network dongle to host the client respectively the publisher/subscriber, and a Samsung Series 9 notebook to initiate and coordinate the measurements. All the above-mentioned devices are connected through the same local network provided by the router which offers a IEEE 802.11ac 2.4 GHz wireless network.

B. Software Setup

Both Raspberry Pi devices were running the latest version of the Raspbian operating system, Jessie, and had a complete Java execution environment. The software implementation of the CoAP protocol stack was provided by the Java-based Californium [18] library, and was applied for both the client and server application. Furthermore, the secure data transmission with CoAP was enabled by the Java-based Scandium [19] library which included certificates for demonstration applications. The implementations of the MQTT publisher and subscriber were developed on top of the Java-based Paho [20] library of the Eclipse Foundation. The broker in the MQTT approach was provided by Mosquitto [21], an open source MQTT broker published by the Eclipse Foundation. The Mosquitto broker allows common insecure connections as well as secure connections through SSL/TLS. To establish a secure

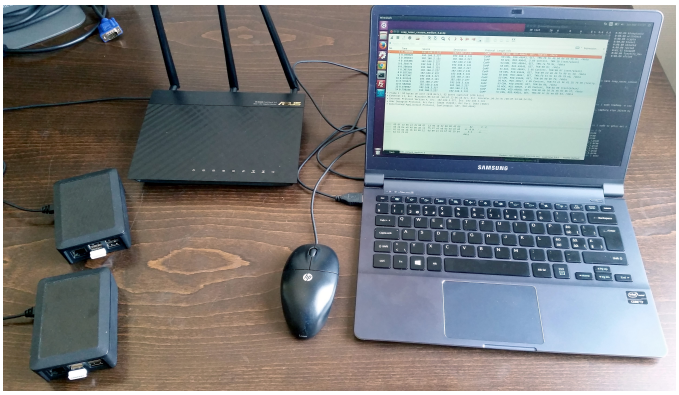

Fig. 1. The testbed with two Raspberry Pi Model B+ acting either as server/broker or as publisher/subscriber. The network emulator NetEM was executed on the server side and the notebook was managing the measurements.

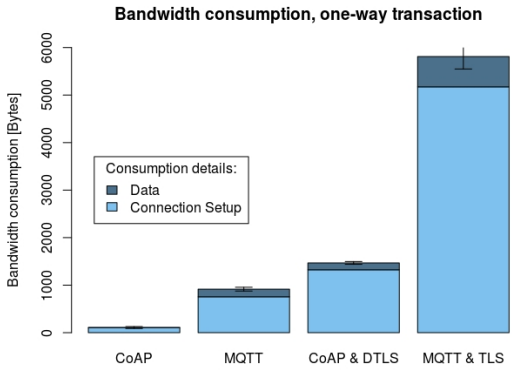

Fig. 2. The bandwidth consumption of a one-way transaction, including a potential session establishment, between a publisher/broker or a client/server measured in bytes.

connection to the broker, the MQTT publisher/subscriber applied the open source Java cryptographic API Bouncy Castle [22]. The certificates to encrypt the SSL/TLS transactions are self-created and self-signed with OpenSSL on Linux.

V. Results

Our results are divided into two parts: The first part presents our measurements on the influence of a secure data transmission in terms of latency and consumed bandwidth, while the second part highlights the influence of a lossy wireless network in terms of consumed bandwidth and packet arrival rate.

A. Influence of a Secure Data Transmission

The evaluation of the bandwidth consumption and the latency for all four protocol settings have been conducted under ideal network conditions with no constraints. For the common data transmission, no remarkable observations were made. During the execution, it could however be observed that the duration of loading, processing and verifying of the certificates for the SSL/TLS encryption required at least 40 seconds on the Raspberry Pi. Further shall be noted that the completion of the initial round-trip of a secure CoAP transmission with DTLS, in all cases, took more than 20 seconds.

To evaluate the bandwidth consumption of the protocols a single one-way transaction was conducted and executed repeatedly to verify the measurement. This transaction includes the session establishment and a single message transmission from the publisher to the broker respectively from the client to the server. Figure 2 shows the results of this evaluation. It shall be noted that the bandwidth consumption for the connection establishment was measured separately for the connection oriented MQTT protocol and for CoAP with DTLS enabled, to illustrate the proportions.

A second approach measured the round-trip time for the two IoT protocols in a common and a secure manner. For each situation, a total of 2000 round-trips were conducted and timed. The round-trip time was measured after the connection was established. The result, a box-and-whisker plot displayed in Figure 3, states that MQTT has more widespread whiskers in both the common and secure data transmission. MQTT

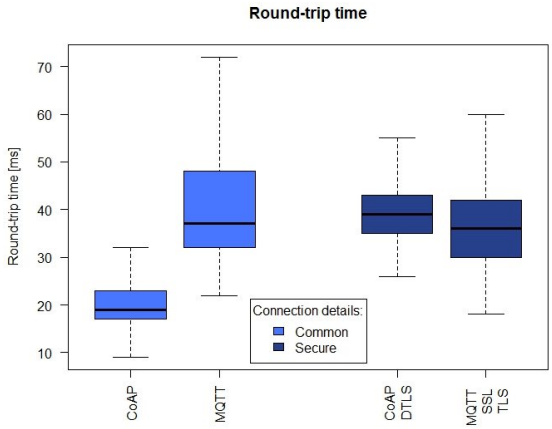

Fig. 3. A plot of the measured round-trip time in milliseconds for the common and the secure version of the two IoT protocols.

with SSL/TLS shows only a slight increase in the measured round-trip time compared to the common MQTT since the same number of packets respectively processing overhead, to complete one round-trip, is needed.

B. Influence of a Lossy Wireless Network

Various rounds of measurements have been performed with three differing wireless network settings. The network packet loss ratio was varied with the NetEM tool and reaches from ideal with 0%, medium with 25%, to high with 50% packet loss rate. There were no additional network delays added, nor was a bandwidth cap applied. The network emulator NetEM was attached to the interface on the Raspberry Pi which executed the server/broker.

In all three network scenarios, a measurement for each protocol has been performed to find the consumed bandwidth for the transmission of 100 round-trip messages. The connection establishment is included in the measurements. The client and the publisher transmitted repeatedly a message every 200 milliseconds. Every measurement was repeated at least six times for every network setting and each protocol. An average of the consumed bandwidth to transmit all 100 messages has

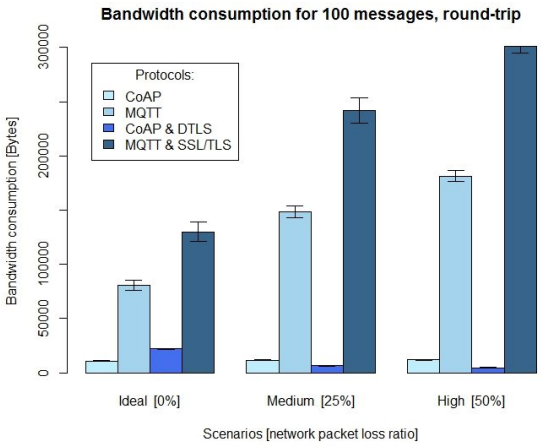

Fig. 4. Bandwidth consumption summary in bytes for 100 messages conducted in a round-trip mode including a one-time connection establishment. The groups display the three wireless network settings with a network packet loss ratio ranging from 0% to 50%.

TABLE I. AVERAGE PACKET RECEIVED RATIO [%]

Average Packet Received Ratio [%]				
		ideal	medium	high
Common	CoAP	100	95	74
	MQTT	100	100	100
Secure	CoAP	100	6	0
	MQTT	100	100	100

been used to create the graph. Figure 4 shows the bandwidth consumption in graph form while Table I states the average packet received ratio for each protocol in percent for the three network settings. It was observed that in case of CoAP with DTLS in the high packet loss network condition, not a single message in all performed measurements could complete a full round-trip. The reason lies in the inability of exchanging the necessary packets to establish the DTLS session due to the high network packet loss. This further explains the low amount of consumed bandwidth in Figure 4 for the medium and high network condition.

VI. CONCLUSIONS

We can conclude that processing certificates and establishing a secure connection on resource-constrained devices demands a high amount of time and processing power. The encryption technologies utilized in this experiment are unlikely to be feasible for weaker devices than the Raspberry Pi. We can also conclude that secure data communication with both CoAP and MQTT on resource-constrained devices perform well in a stable and non-lossy network setting. However, in case of an error prone and lossy network, we strongly recommend the utilization of MQTT with SSL/TLS. We aim to continue this work by further conducting performance measurements for different ciphers, algorithms and key lengths for the secure communication of both protocols. We will also continue our research in QoS aspects for the area of fog computing and its related distributed systems technologies.

ACKNOWLEDGEMENT

This research was supported by grant 20150363, 20140319, and 20140321 of the Swedish Knowledge Foundation.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] R. T. Fielding, "Architectural styles and the design of network-based software architectures, Chapter 5: Representational state transfer (REST)," Ph.D. dissertation, University of California, Irvine, 2000.

[5] C. Bormann. Specification CoAP. [Online]. Available: http://coap.technology/spec.html

[6] A. Banks and R. Gaptu. MQTT Version 3.1.1 OASIS Standard. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[8] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the internet of things: a review," in *Computer Science and Electronics Engineering (ICCSEE), 2012 international conference on*, vol. 3. IEEE, 2012, pp. 648–651.

[9] E. Rescorla. HTTP Over TLS. [Online]. Available: https://tools.ietf.org/html/rfc2818

[10] T. Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. [Online]. Available: https://tools.ietf.org/html/rfc5246

[11] A. Freier and P. Karlton. The Secure Sockets Layer (SSL) Protocol Version 3.0. [Online]. Available: https://tools.ietf.org/html/rfc6101

[12] E. Rescorla. Datagram Transport Layer Security Version 1.2. [Online]. Available: https://tools.ietf.org/html/rfc6347

[13] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*. IEEE, 2014, pp. 1–6.

[14] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," in *Communications and Vehicular Technology in the Benelux (SCVT), 2013 IEEE 20th Symposium on*. IEEE, 2013, pp. 1–6.

[15] Y. Chen and T. Kunz, "Performance evaluation of IoT protocols under a constrained wireless access network," in *Selected Topics in Mobile & Wireless Networking (MoWNeT), 2016 International Conference on*. IEEE, 2016, pp. 1–7.

[16] The Linux Foundation. NetEM - Network emulator. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[17] Wireshark community. Wireshark packet analyzer. [Online]. Available: https://www.wireshark.org/

[18] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: Scalable cloud services for the internet of things with CoAP," in *Internet of Things (IOT), 2014 International Conference on the*. IEEE, 2014, pp. 1–6.

[19] Eclipse Foundation. Scandium (Sc) - Security for Californium. [Online]. Available: https://github.com/eclipse/californium/tree/master/scandium-core

[20] ——. Eclipse Paho Java Client. [Online]. Available: https://eclipse.org/paho/clients/java/

[21] ——. Mosquitto MQTT broker. [Online]. Available: https://mosquitto.org/

[22] Legion of the Bouncy Castle Inc. Bouncy Castle Crypto API. [Online]. Available: https://www.bouncycastle.org/java.html