# Comparative Analysis of IoT Communication Protocols

Burak H. Çorak[1], Feyza Y. Okay[1], Metehan Güzel[1], Şahin Murt[2], Suat Ozdemir[1]
[1]Department of Computer Engineering, Gazi University, Ankara, Turkey
[2]AIR Telekomünikasyon Çözümleri San. Tic. A.Ş. Ankara, Turkey

***Abstract*—With the proliferation of machine-to-machine communication, there are many communication protocols standardized for IoT applications. Performances of these protocols may significantly deviate from each other even under the same operating conditions. In this paper, we quantitatively compare the performances of a set of well-known IoT communication protocols, namely CoAP (Constrained Application Protocol), MQTT (Message Queuing Telemetry Transport) and XMPP (Extendible Message Persistent Protocol) in a real-world testbed. CoAP employs UDP packets for transmission while others use TCP. For this purpose, we design as small testbed that collects real-time environmental data. By designing such a system, we aim to reveal the differences among protocols in terms of packet creation time and packet transmission time. The obtained results show that XMPP is worse than other protocols in both metrics and MQTT and CoAP perform almost equally.**

***Index Terms*—Internet of Things (IoT), CoAP, MQTT, XMPP**

## I. INTRODUCTION

Today the tremendous amount of things that we use in our daily life become connectable to the internet. These things communicate with each other over a distributed network, by sharing and managing information stream for a specific task. The Internet of Things (IoT) is a relatively new concept which enables a high level of interaction among a large number of distributed things. Since the IoT concept incorporates the machine-to-machine (M2M) communication, they become a part of smarter devices systems [1]. The IoT includes embedded systems that measure environmental conditions via sensors. For example, collecting and analyzing sensor data such as temperature and air pollution provide substantial results like user behavior and the relationship between metrics and prediction for further analysis. In such a system, hundreds of different type of devices with thousands of sensors need to connect each other. Additionally, each of these devices is used for a different certain task requiring different data rates, packet sizes etc. Therefore, such communication network becomes highly heterogeneous [2]. Specific communication protocols are essentially required for connecting deployed IoT devices to the network in a distributed manner.

Many IoT devices generally have low cost and constrained-resources such as low network-connection, power, and processing limitation [3]. Communication protocols are very critical to organize data streams and determine how to IoT interact each other. The IoT requires new flexible protocols for the heterogeneous and constrained devices, unlike common communication protocols. More specifically, constrained devices are insufficient to meet the high demands of HTTP communication. To reduce the high requirements of complex protocols used for constrained devices, the Internet Engineering Task Force (IETF) standardizes protocols that enable lightweight communication [4].

In this paper, our aim is to compare differences of protocols that handle the communication between the server and limited IoT devices sending data from sensors over the Internet. In order to perform the IoT communication with high quality in a large-scale network for real-time data collection, protocols should have appropriate features such as low packet loss, high packet creation time and low packet response time etc. In accordance with this purpose, three protocols CoAP, MQTT, and XMPP are applied to examine communication metrics by employing embedded systems.

In this study, unlike existing studies, communication protocols are implemented and analyzed via embedded systems that produce and send real-time data through sensors. Thus, extensive comparative results of protocols are presented to give the readers the opportunity of demonstrating the weaknesses and strengths of communication protocols which are used between server and IoT devices.

The rest of the paper is organized as follows. In Section II, related works are presented. The structure, advantages, and disadvantages of the selected protocols are explained in Section III. In Section IV, experimental setup and performance evolution of protocols are presented. Section V concludes the paper by listing future directions in the area.

## II. RELATED WORKS

This section presents some previous works analyzing some communication protocols' performance in the literature.

Anusha et al. [6] handle the various data protocols XMPP, CoAP, AMQP, MQTT, DDS and MQTT-SN in the IoT concept. Authors aim to compare the functionality of each data protocol with the other data protocols according to performance metrics such as packet loss rate, message size, bandwidth consumption and latency. Each protocol's performance is evaluated depending on the application. Besides, XMPP has better performance results due to its XML stanza based transmitting for instant messaging applications over the internet.

Bandyapadhyay et al. [7] aim to determine which protocol is more suited for different application areas with constrained devices by comparing XMPP and CoAP. Android O/S and Intel X86 systems are used to perform protocol's evaluations. The

978-1-5386-3779-1/18/$31.00 ©2018 IEEE

software technologies for implementation are 'libcoap' library for CoAP and 'Mosquitto' project for MQTT. Also, Wireshark is used to analyze the network traffic. Protocols are compared in terms of energy consumption, bandwidth utilization, reliability. According to the results, CoAP is better than MQTT with regard to optimize energy usage.

Chen et al. [8] perform a comprehensive survey to compare performances of CoAP, MQTT, DDS and a custom UDP-based protocol in medical monitoring application by addressing the bandwidth consumption, latency and packet loss metrics on a real-time data which is collected from patients. Also, the authors clarify how protocols perform their functions under a constrained, low quality wireless networks. The hardware technologies are Raspberry Pi model 2, Arduino Uno revision 3 and Windows laptop ASUS Zenbook. The software technologies for implementation are "Californium CoAP" for CoAP server and client), "HiveMQ" for MQTT server implementation, "Mosquitto" for Broker and MQTT clients (both subscriber and publisher), "OpenDDS" for DDS server and client. Performances of protocols are analyzed with "TBF", "NetEM" and "Wireshark" tools. Performance results show that both TCP-based protocols (DDS and MQTT) are more reliable than UDP-based protocols (Custom-UDP and CoAP) in low quality wireless networks. However, TCP-based protocols have more latency than UDP-based protocols in the same network condition. In addition, DDS performs better than MQTT in poor network conditions.

Thangavel et al. [9] evaluate and compare the performances of MQTT and CoAP protocols in terms of packet-loss, retransmitting messages delay, data transferred per message. The authors especially focus on the data transmission between sensors at the gateway node to the back-end server for CoAP or broker for MQTT. A laptop as a server, a BeagleBoard-xM for middleware implementation and a netbook for Wide Area Network (WAN) emulator are used as the hardware-technologies. The software technologies are "Wanem" (the wide area network emulator) to transfer messages, "Mosquitto" project for MQTT Broker, "libcoap" library for CoAP and "Wireshark" to measure metrics. Results show that MQTT messages have lower delays than CoAP for lower packet loss. On the other hand, MQTT has higher delays than CoAP for higher packet loss. Also, CoAP has less traffic when message size is smaller and packet loss rate is less.

TABLE I: COMPARISON OF RELATED WORKS

| Features / Authors | Protocols | Hardware Technologies | Software Technologies | Metrics | Purpose(s) | Result(s) |
|---|---|---|---|---|---|---|
| **Anusha et al. [6]** | XMPP, CoAP, AMQP, MQTT, DDS, MQTT-SN | - | - | Network packet loss rate, message size, bandwidth-consumption, latency | Compares functionality of each data protocol | The performance of each protocol depends on the application |
| **Bandyapadhyay et al. [7]** | MQTT, CoAP | Android O/S, Intel X86 systems | libcoap, Mosquitto | Energy consumption, bandwidth utilization, reliability | Highlights which protocol is mapped to best suited for constrained application areas | CoAP is better than MQTT for energy usage |
| **Chen et al. [8]** | CoAP, MQTT, DDS, custom UDP | Raspberry Pi model 2, Arduino Uno revision 3, Windows laptop ASUS Zenbook | Californium, HiveMQ, Mosquitto, OpenDDS | Bandwidth consumption, latency and packet loss | Clarify how protocols function under a constrained, low quality wireless networks | TCP-based protocols are more reliable and have more latency than UDP-based protocols |
| **Thangavel et al. [9]** | MQTT, CoAP | a BeagleBoard-xM, Laptop, Netbook | libcoap, Mosquitto | Packet-loss, retransmitting messages delay, data transferred per message | Focuses on the data transmission between sensors at the gateway node to the server | MQTT messages have lower delays than CoAP for lower packet loss. In higher packet loss, this situation is reversed |
| **Kayal et al. [10]** | CoAP, MQTT, XMPP, WebSocket | - | libcoap, Mosquitto, HiveMQ, Openfire Paho Python | Response time by varying the traffic load | Measures performances of protocols in constrained devices to ensure efficiency | In lower server utilization, CoAP performs better than other protocols. XMPP performs better than others at lower server utilization |

Kayal et al. [10] handle communication protocols CoAP, MQTT, XMPP, and WebSocket. The main aim of the paper is to measure performances of protocols in constrained devices to provide effective communication. In accordance with this, a smart parking scenario is implemented to compare protocols' response time by varying the traffic loads. As software technologies, "libcoap" library for CoAP, "Mosquitto" project for MQTT Broker, "Openfire" project for XMPP Server, "Smack Client" for XMPP Client, "HiveMQ" for Websocket Broker and "Paho Python" library for WebSocket client are used. According to the results, CoAP performs better than other queue-based protocols at lower server utilization. However, XMPP performs better than others at lower server utilization when the application can support multi-threading. Further, the average response time of protocols increases except WebSocket when Server utilization is increased. Also, XMPP serves the processes in the order of most available first, whereas other protocols use FIFO scheduling. Table I. shows the comparison of related works in the literature.

In the literature, [10] is the closest work to our paper. Our main difference from this paper is that we focus on environmental data rather than smart parking data. Each sensor sense and collect environmental data such as temperature, humidity and light and send them to the related server through different protocols in real-time. Furthermore, in [10] performance metrics are measured by GET, PUT and DELETE message types which are provided by CoAP. Unlike these message types, our metrics are compared by sending the actual values from the sensors to their own message type.

## III. Preliminaries

In this section, preliminaries of CoAP, MQTT, and XMPP protocols are given.

### *A. CoAP (Constrained Application Protocol)*

CoAP is a transfer protocol based on UDP layer which is customized for constrained nodes and constrained networks. Because CoAP is inspired by HTTP when it is first designed, REST architectural style is used [11]. Then, it is standardized by an independent group called the Internet Engineering Task Force (IETF). In addition, the communication between Server and Client is peer-to-peer. However, Server or Client can response unicast and multicast requests. The CoAP has four different message types to request resources from server: GET, PUT, POST and DELETE [12]. Fig. 1 shows CoAP's messaging architecture.

Advantages of CoAP:

- Operates fast communication by sending small packets with UDP layer.
- Asynchronous communication is provided.
- Peer-to-peer communication doesn't need to the intermediate server between clients. Also, many-to-many communication is supported.
- Datagram Transport Layer Security (DTLS) provides integrity, security, and privacy by authorizing encrypting and securing.
- Good option for constrained devices.

Disadvantages of CoAP:

- Messages are unreliable. Therefore, ACK (acknowledgement) packets are sent to confirm the message arrives. However, it does not show clearly whether these messages are decoded correctly or completely.
- It is still standardizing. It is selected the most unstandardized protocol among other protocols.

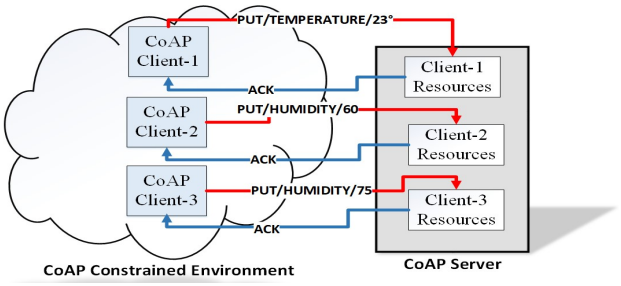


Fig. 1. CoAP architecture between constrained nodes and server

### *B. MQTT (Message Queuing Telemetry Transport)*

MQTT is a lightweight M2M communication protocol for constrained devices and unreliable networks. It has publisher/subscriber client which runs over TCP/IP [13]. Also, TCP provides message reliability and bidirectional connections between nodes. The nodes can publish messages with some specific topics on Broker, and accordingly, different nodes can subscribe these topics to receive messages. The Broker, which must control the traffic of this message, takes part in communication. In fact, Broker is a server where clients can publish/subscribe topics and the message traffic runs through. Moreover, clients can be authorized for a broker by accessing with username and password. MQTT supports three QoS levels to determine the sending quality of the message. Security of message is encrypted by SSL/TLS [14]. Fig. 2 shows MQTT's messaging architecture based on publisher and subscriber structure.

Advantages of MQTT:

- Provides reliability of messages by supporting QoS levels.
- Effectively utilizes bandwidth through packet agnostic. The data may contain binary or text.
- Publish/Subscribe mechanism has capabilities such as one-to-one, many-to-many or one-to-none. Also, this mechanism provides bi-directional communication.
- Utilizes simple methods for communication.

- The communications among nodes are asynchronous. Messages can publish/subscribe anytime.

Disadvantages of MQTT:

- It uses TCP/IP and TCP requires more communication capabilities unlike UDP.
- Broker has limited capacity for communication.
- All nodes are connected to Broker. Therefore, the communication collapses when the broker is a failure.

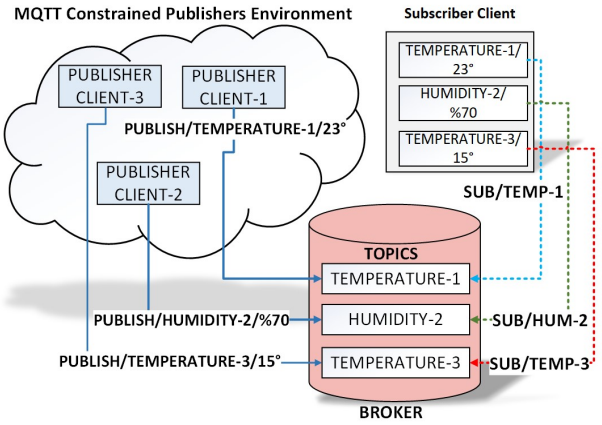


Fig. 2. MQTT architecture between constrained nodes and server (SUB: subscribe, HUM: humidity, TEMP: temperature)

### C. XMPP (Extensible Messaging and Presence Protocol)

XMPP is a protocol to provide communication and file-transfer among nodes in a distributed network using XML technology [15]. Communications including instant messaging, group-chat, collaboration, presence and transferring other types of data such as audio and video are based on TCP. Also, XML stanzas support real-time communication. XMPP allows the server to authorize unique clients in order to access the server and enables messaging among these clients using XML stanzas. Moreover, XMPP assigns the client presence indicator such as online, offline or busy [16]. Thus, the client tells the server whether it is suitable for messaging. Fig. 3 describes that XMPP's messaging architecture.

Advantages of XMPP:

- It is extendable and adaptable.
- Multiple servers provide uninterrupted communication.
- Supports communication between client-client, client-server and server-server.
- The presence indicator offers more options to the messaging.
- It has TCP protocol based on XML-Stanzas and this provides more reliability communication.

Disadvantages of XMPP:

- The server has limited capacity for communication.
- The authorization takes much time while clients request access to the server.
- Using XML Stanzas in communication causes delays.

## IV. Experimental Results and Performance Evaluations

This section presents, experimental setup and the performance evaluation results of CoAP, MQTT and XMPP. The experiments are performed using an Intel Galileo Gen 2 board and ASUS Laptop (Intel Core i7-6700HQ CPU 2.60Ghz, 4 GB RAM and Windows 64bit O/S). The board is used to implement protocols' client and the laptop is used to implement protocol's

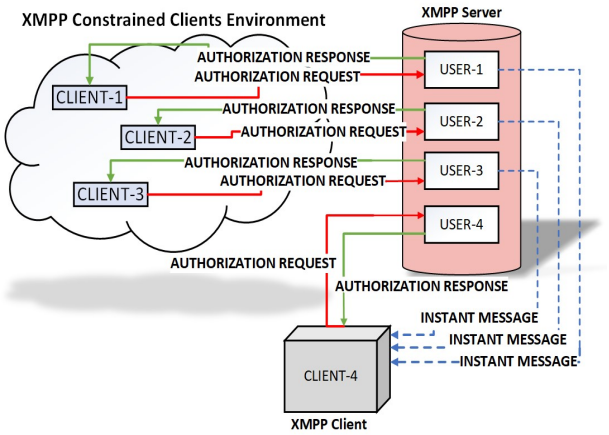


Fig. 3. XMPP architecture between constrained nodes and the server

server and the data collection parts of clients such as MQTT and XMPP. These devices communicate with each other over a local area network. The clients of these protocols are embedded the same board in order to compare the protocols under the same conditions. Clients send messages sequentially to the servers over the same board. "CoAP-simple-library" [17], "Pubsubclient" [18], "XMPPArduino" [19] are used to apply the client of CoAP, the publisher client of MQTT and sending messages from the board using XMPP protocol, respectively. Also, "Californium" [20], "Paho" [21], "Smack" [22] projects are used for implementation of the server of CoAP, the subscriber client of MQTT and the message listener client of XMPP, respectively. Lastly, "Mosquitto" [23] and "Openfire" [24] projects are used for the implementation of the MQTT Broker and the server of XMPP, respectively. Fig. 4 and Fig.5 show the hardware setup. To ensure fair comparison, presented protocols send real-time environmental data such as temperature, humidity and light continuously using these libraries.

With respect to these protocols, we first measure the packet creation time of each protocol. The packet creation time denotes the speed of message generation. For example, PUT message type is used only for CoAP so that real-time received data could be transferred to the main server (For TCP-based protocols, data is collected in the client board).

Fig. 4. Sensors used in the testbed (temperature, humidity and light)

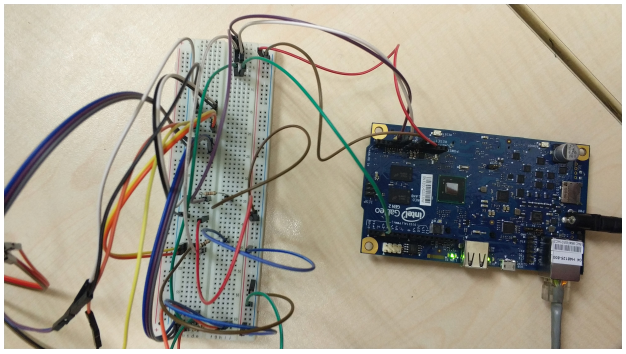

Fig. 5. Processing unit of the testbed

To give the results accurately, 100 messages are created for 25 times. The results of average creation times are 419 microseconds for CoAP, 119 microseconds for MQTT and 12855 microseconds for XMPP. According to these measurements, MQTT creates packets faster than other protocols. While the CoAP and MQTT packet creation times are approximately same, the time for XMPP is quite slow compared to other protocols. Because the XMPP uses XML Stanza to create packets, other protocols have simpler packet structures. However, the difference between the CoAP and the MQTT is dependent on the optimization of the libraries. Fig. 6 demonstrates the average creation times for each protocol.

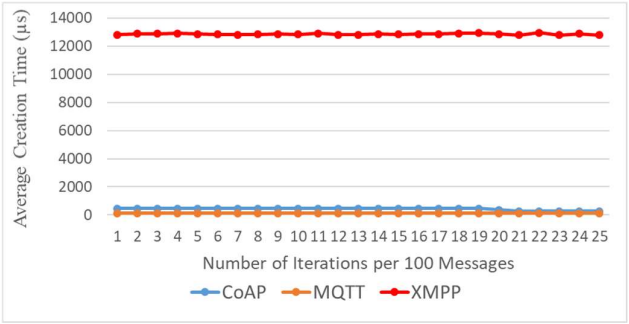


Fig. 6. Packet creation time for protocols (µs)

Secondly, the transmission times for the packets from the board to the main server are measured. More specifically, the transmission time for CoAP is measured the time between a client on the board and the server on the main server. For MQTT and XMPP, the times between the client on the board and the client on the main server are measured. Also, these measurements do not contain ACK messages for MQTT and XMPP. They are calculated by considering the delivery times of the protocols under the same conditions. Total 100 messages are used for this measurement. Accordingly results, MQTT has an average arrival time of about 589 microseconds each message while CoAP has 821 microseconds and XMPP has 41383 microseconds. The reason of why XMPP is slower than other protocols is that it reads the sent messages according to the XML format. It is expected that CoAP has better results than MQTT; because, CoAP uses UDP, whereas MQTT uses TCP for messaging. In contrary, MQTT has no delay for messaging to wait for ACK. Publisher sends messages to Broker and then subscriber receives those data from Broker. In addition, all protocols use small packet size for messaging. Moreover, MQTT is a more optimized and standardized protocol than CoAP. For these reasons, COAP messages are delivered to the main server two times slower than the MQTT. Fig. 7 shows the average transmission times for each protocol separately.

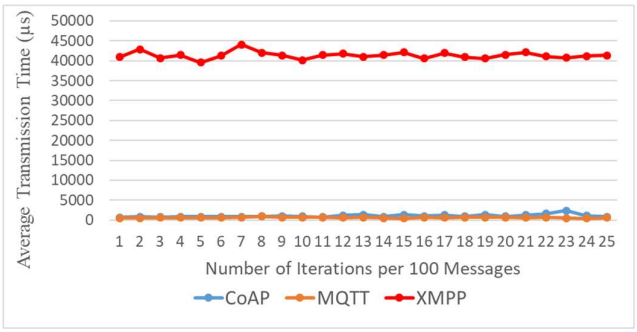


Fig. 7. Packet transmission time for protocols (µs)

## V. Conclusion and Future Directions

In this paper, we prove that protocols are essentially required for the IoT devices to collect environmental data in real-time. According to the requirements, we examine the performance metrics of CoAP, MQTT, and XMPP and by comparing these performance metrics, we aim to reveal the differences of these protocols in a real-time communication environment. Protocols are compared by packet creation time, packet delivery speed metrics to determine the delay differences in the real-time communication environment. As a result, MQTT has better packet creation and transmission time than other protocols, although CoAP is UDP-based protocol. Moreover, the MQTT delivers its packets two times faster than the CoAP. MQTT is better than other protocols due to several reasons such as the network has wide bandwidth, the packets being transfer are at a lower size and the COAP is less standardized. When XMPP is examined in such a network, it has a slowing structure like XML stanza, which causes extra latency when compared to the other protocols.

Our future plan is to measure presented protocols for different conditions such as low bandwidth, high collision rate and expand the environment to collect environmental continuous data from different locations.

## REFERENCES


[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Communication Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376, Jun. 2015.

[3] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, 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.

[4] Z. Sheng, S. Yang, Y. YU, A. V. Vasilakos, J. A. Mccann and K. 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, Dec. 2015.

[5] J. Gubbia, R. Buyyab, S. Marusica and M. Palaniswamia. "Internet of Things (IoT): A vision, architectural elements, and future directions," Future generation computer systems, vol. 29, no. 7, pp. 1645-1660, Sep. 2013.

[6] M. Anusha, E. S. Babu, L. S. M. Reddy, A. V. Krishna and B. Bhagyasree, "Performance Analysis of Data Protocols of Internet of Things: Qualitative Review," International Journal of Pure and Applied Mathematics, vol. 115, no. 6, pp. 37-47, 2017.

[7] S. Bandyopadhyay and A. Bhattacharyya, "Lightweight Internet Protocols for Web Enablement of Sensors using Constrained Gateway Devices," IEEE 2013 International Conference on Computing, Networking and Communications (ICNC), pp. 334-340, Jan. 2013.

[8] Y. Chen and T. Kunz, "Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network," IEEE 2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT), Apr. 2016, pp. 1-7.

[9] D. Thangavel, X. Ma, A. Valera, H. X. Tan and C. K. Y. Tan, "Performance Evaluation of MQTT and CoAP via a Common Middleware," 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 21–24 Apr. 2014, pp. 1-6.

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

[11] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," IEEE Internet Computing, vol. 16, no. 2, pp. 62-67, Mar. 2012.

[12] Z. Shelby, K. Hartke and C. Bormann, "The Constrained Application Protocol," Internet Engineering Task Force (IETF), Request for Comments: 7252, Jun. 2014.

[13] Y. F. Gomes, D. F. Santos, H. O. Almeida and A. Perkusich, "Integrating MQTT and ISO/IEEE 11073 for health information sharing in the Internet of Things," IEEE 2015 International Conference on Consumer Electronics (ICCE), pp. 200-201, Jan. 2015.

[14] S. Lee, H. Kim, D. K. Hong and H. Ju, "Correlation analysis of MQTT loss and delay according to QoS level," 2013 IEEE International Conference on Information Networking (ICOIN), pp. 714-717, Jan. 2013.

[15] M. Kirsche and R. Klauck, "Unify to bridge gaps: Bringing XMPP into the Internet of Things," 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 455-458, Mar. 2012.

[16] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," Internet Engineering Task Force (IETF), Request for Comments: 6120, Mar. 2011.

[17] CoAP-simple-library, Jan. 30, 2018. Available: https://github.com/hirotakaster/CoAP-simple-library

[18] Pubsubclient, Jan. 30, 2018. Available: https://github.com/knolleary/pubsubclient

[19] XMPPArduino, Jan. 30, 2018. Available: https://github.com/adamvr/XMPPArduino

[20] Californium Project, Jan. 30, 2018. Available: https://www.eclipse.org/californium/

[21] Paho Project, Jan. 30, 2018. Available: https://www.eclipse.org/paho/

[22] Smack Project, Jan. 30, 2018. Available: https://github.com/igniterealtime/Smack

[23] Mosquitto Project, Jan. 30, 2018. Available: https://mosquitto.org/

[24] Openfire Project, Jan. 30, 2018. Available: https://www.igniterealtime.org/projects/openfire/