

Cluster-based CoAP for Message Queueing in Internet-of-Things Networks

Dong-Kyu Choi*, Joong-Hwa Jung*, Hyung-Woo Kang*, Seok-Joo Koh**

*School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea

**School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea

**Software Technology Research Center (SWRC), Kyungpook National University, Daegu, Korea
 supergint@gmail.com, godopu16@gmail.com, hwkang0621@gmail.com, sjkoh@knu.ac.kr

Abstract— The interest in Internet of Things (IoT) has been increased. The IoT can be used to exchange information between people and physical devices or between devices and devices. For constrained IoT network environments, the lightweight application protocols have recently been proposed, which include Constrained Application Protocol (CoAP) and Message Queueing Telemetry Transport (MQTT). However, those protocols still have the scalability problem in the network with a large number of sensors. In this paper, we propose a simple extension of CoAP using a clustering approach, in which a set of CoAP sensors are grouped into a cluster, and a cluster head is used for message aggregation and transmission for the sensors associated with the cluster. By implementation and experimentation, we can see that the proposed cluster-based CoAP provides the performance gains over the existing CoAP and MQTT protocols in terms of bandwidth consumption and transmission time.

Keywords— IoT, CoAP, MQTT, cluster, performance analysis

I. INTRODUCTION

Internet of Things (IoT) is the internetworking of physical devices, vehicles, home and other things embedded with electronics, sensors, actuators and network connectivity [1]-[3]. All of the IoT devices can be connected to Internet, and those devices can be sensed and remotely controlled. However, the IoT environment has the characteristics of low power battery, unstable network state, and lightweight program. For constrained IoT network environments, the lightweight application protocols have recently been proposed, which include Constrained Application Protocol (CoAP) and Message Queueing Telemetry Transport (MQTT) [4].

The CoAP is a lightweight application protocol that is used for low performance devices to allow them to communicate interactively over the Internet. CoAP was made with redesign of a subset of HTTP functions by taking into account the low processing power and energy consumption. The Internet Engineering Task Force (IETF) constrained RESTful (CoRE) environments working group has done the major standardization work for this protocol [5]-[6].

The MQTT is an ISO standard publish-subscribe based lightweight messaging protocol for use on top of the TCP/IP protocol. The publish-subscribe messaging pattern requires a broker which is responsible for distributing messages to the

interested clients based on the topic of a message and for matching publishers and subscribers [7]-[8].

With the growth of IoT, the number of sensor nodes are increasing steadily. It is important to receive data efficiently from the server in constrained environment. In MQTT, each subscriber must communicate with the broker even though a lot of subscribers have the same interests of topics. It may cause the traffic overhead problem and the energy wasting of constrained devices. In the meantime, CoAP is a request-response model that is also difficult to manage many devices. However, the CoAP has also the similar scalability problem with MQTT.

To overcome these problems, in this paper, we propose a simple extension of CoAP using a clustering approach, in which a set of CoAP sensors are grouped into a cluster, and a cluster head is used for message aggregation and transmission for the sensors associated with the cluster. Sensors in the proposed scheme are managed by a cluster head. The cluster head will manage many sensors. By using this clustering approach, it is easy to transfer messages containing topics. In the proposed scheme, the cluster head is used to reduce the amount of traffics concentrated on the server. The proposed scheme can also be used for CoAP-based many-to-many communication.

This paper organized as follow. Section II introduces the related works on the lightweight protocols for IoT environment. In Section III, we present the proposed cluster-based CoAP scheme. Section IV discusses the implementation and experimentation for performance analysis. Finally, Section V concludes this paper.

II. RELATED WORKS

A. MQTT

Message Queue Telemetry Transport (MQTT) is a lightweight Application layer protocol for IoT devices. MQTT is being standardized by the Organization for the Advancement of Structured Information Standards (OASIS). Publish-subscribe model is a messaging paradigm where the messages are published to a broker and the receivers can subscribe to the broker to receive messages. The publishers and subscribers do not need to know each other because the broker is employed. The publications and subscriptions are matched by a broker,

and the publications are delivered by the broker to the subscriber. The connection of MQTT is established by sending a CONEECT message which needs to be CONACK. The clients can publish and subscribe messages on a particular topic.

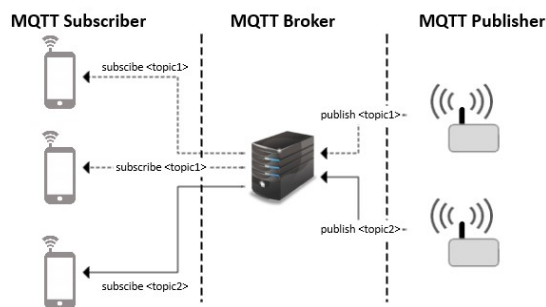


Figure 1. MQTT System Architecture

Figure 1 shows the architecture of MQTT. Subscribers interact with a broker, and act as either a publisher or a subscriber. A publisher can update the state to the broker, and the subscribers who are interested in the topic receive the state from the broker.

The broker is a MQTT server that provides an interface to the clients. The broker needs to be reachable by all clients and also has sufficient resources, such as storage, bandwidth, etc. A client of MQTT interacts with the broker through the MQTT interface. The clients initiate all interactions with the broker. A client of MQTT can make use of both publish and subscribe. The clients and broker use the topics to identify a particular resource or object in the publish-subscribe system. Topics are conventionally formed as a hierarchy, such as. "/sensors/weather/barometer/pressure". The topics may alternatively be a collection of one or more sub-topics, consisting of links to the sub-topic URIs.

The reliability of messages in MQTT is divided into the three Quality of Service (QoS) levels. QoS level 0 sends a message at most once and no acknowledgement of reception is required. QoS level 1 sends a message at least once and acknowledgement of reception is required. QoS level 2 uses the 4-way handshake mechanism for the delivery of a message exactly once.

B. CoAP

CoAP is a web transfer protocol to use with nodes in the constrained environments with low power devices, low performance and lossy networks. The CoRE working group of the IETF has done the standardization of CoAP.

CoAP is a model that redesigned the REST architecture into a suitable form for the constrained nodes and networks. The CoAP is similar to HTTP [9], which is based on a client and server model. The CoAP manages resources by using the HTTP methods like GET, PUT, POST and DELETE.

CoAP defines the four types of messages: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK),

and Reset. CoAP uses CON and ACK messages for reliability of messages. When client sends message with CON type, receiver needs to send ACK message to sender. Contrary to CON, NON messages do not provide reliability. However, NON messages has less overhead and bandwidth usage than CON messages.

A client of CoAP that is interested in the state of resource will initiate a request to the server. The server then sends the response messages which include the state of the resource at the time of the request. CoAP was recently extended for a CoAP client to observe a resource on a CoAP server [10]. The client retrieves a state of the resource and requests the notification when the state is updated by other clients. The topic is a resource in the namespace of a CoAP server. The state of the resource can change when other clients update the state of the resource. An observer is a CoAP client that is interested in particular topic. A client needs to register its interest of a topic by using the extended GET CoAP method to the server. The server adds the client to the list of observers when the request arrived. The server notifies each client in the list of observers when the state of a resource changes. Notifications are additional CoAP responses in reply to the extended GET method.

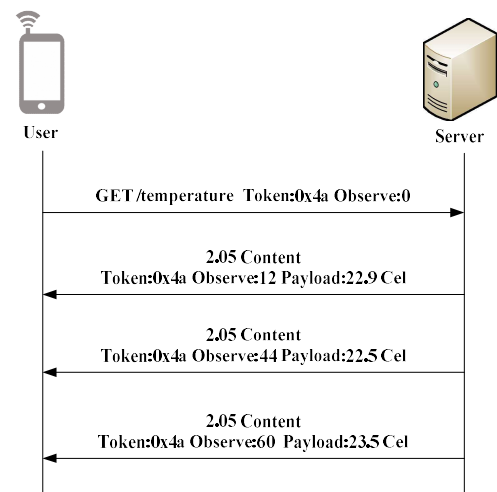


Figure 2. Example of Observe in CoAP

Figure 2 shows an example of CoAP observe mechanism. The client sends messages to the server so as to register its interested resources. After this registration step, the server sends a response message to the client. The server sends messages with the state of resource. Whenever the state of resource is changed, server notifies clients about resource.

III. PROPOSED CLUSTER-BASED COAP SCHEME

A. Overview

In this paper, we propose a lightweight protocol extension to CoAP, named Cluster-based CoAP for Message (Cluster CoAP MQ). The proposed scheme uses the CoAP observe function which is suitable for less periodic messages. We use a cluster head to manage a cluster of constrained devices and

interact with the server. In the proposed scheme, the cluster head uses the topics representing the attributes of sensors. For this reason, user or server can communicate with sensors through the cluster head. Each node will register to the cluster head with an interested topic. After configuration of a cluster, the cluster head receives messages from the server and sends messages to sensors on a particular topic by using multicast. To maintain connectivity between the cluster head and the server, server sends appropriate control messages to the cluster head.

B. Architecture

A cluster head interacts with the server using particular topics. User interacts with a server to transfer messages on a particular topic to sensors. The sensors and users don't need to communicate directly with each other. Figure 3 shows the architecture of the proposed scheme. When the cluster head receives messages on a particular topic, it sends messages to the sensor nodes in the cluster by using multicast. The cluster head also manages the members of cluster by using CoAP POST, PUT, and DELETE methods. The server collects and stores temporarily the messages from the sender until these messages are transmitted to the cluster head.

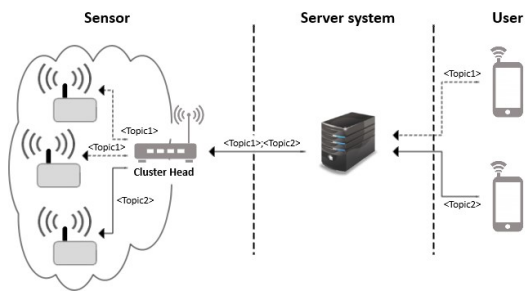


Figure 3. Cluster CoAP MQ System Architecture

C. Cluster Configuration

Cluster head advertises its own ID using by broadcasting. For registration, CoAP sensors will send respond with the response message about to the cluster head on a particular topic by using CoAP POST method. The cluster head collects the register messages of sensors. After that, the cluster head registers the information of interested topics to the server by using POST messages. When the cluster head receives an ACK message, the cluster configuration is completed. The cluster head repeats periodically these transactions. Figure 4 shows how to make a cluster of sensors and register to the cluster.

D. Cluster CoAP MQ Data Flow

Figure 5 shows the data flow of the Cluster CoAP MQ scheme. A cluster head registers to the server by using a CoAP extended GET method that contains the topic that the cluster head wants to receive. The server collects messages on particular topics and sends a message to the cluster head. In this case, the server sends a message containing states of resources in reply to the previous GET message. The cluster

head transfers the states of resources to sensors in the cluster by using multicast.

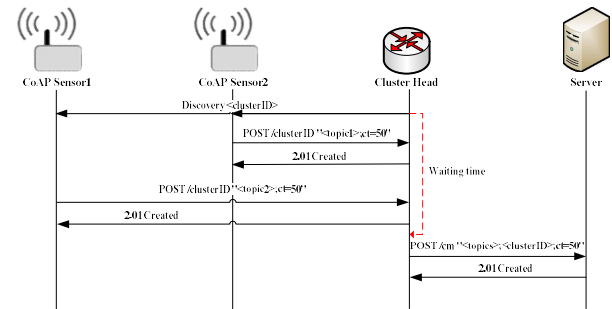


Figure 4. Cluster Configuration Process

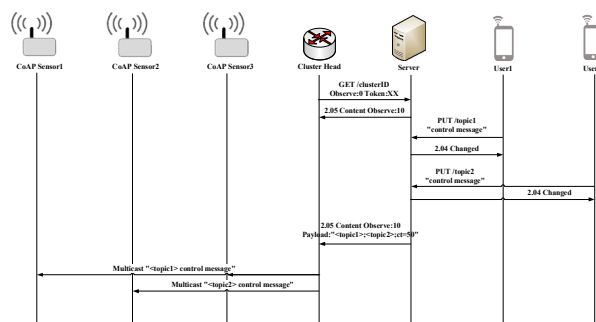


Figure 5. Data Flow of Cluster CoAP MQ

E. Condition of Message Transmission in Server

CoAP sender sends a message to server on a topic. The server needs to send the message to a cluster head in time. For this purpose, we define the three types of conditions in which the server sends to cluster head. In the first condition, the sender can adjust the message timeout by using the CoAP option field. The server sends periodically messages to the cluster head. The messages may have no content on a particular topic. In this case, the messages act as messages of connection maintenance. In the second condition, messages are delivered continuously on a same topic. For example, the messages are delivered in the order of {"topic1", "topic2", "topic1"}. There are two messages on a topic1. In this case, the server sends the messages in a message queue to the cluster head except the latest message on a topic1. The last condition is the case of a message with a direct option. When a message with direct option arrives, the server sends all messages in a message queue to the cluster head.

Table 1 shows the message type, flags, and description. We use the 12th and 13th bits of CoAP option field to classify these messages types.

Table 1. Message Type

Option	Flag	Description
NORMAL	00	Normal condition of server
DIRECT	01	Send this packet immediately
LIMIT	10	Has deadline of stored time

IV. IMPLEMENTATION AND PERFORMANCE ANALYSIS

A. Testbed Implementation

We implemented the testbed of the proposed scheme by using Californium and Paho libraries based on JAVA. Figure 6 shows a class diagram of Cluster CoAP MQ. The ClusterManager class forms and deletes resources of cluster head. The Cluster class confirms the updating states of particular topics by using the TopicObserver class. The TimeoutTask class manages the time about each cluster.

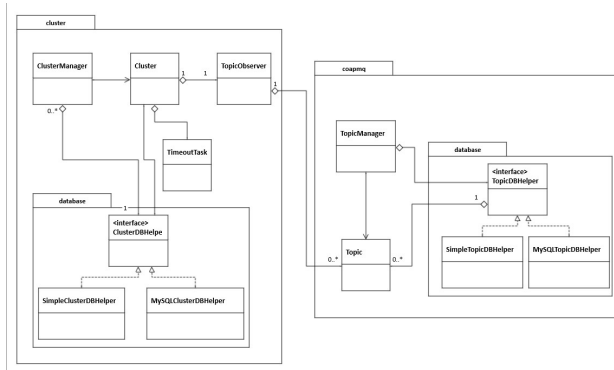


Figure 6. Class diagram of Cluster CoAP MQ

B. Experimental setup

For the performance analysis, we made up the experimental model with five sensors, five users, a cluster head, and a server. We implemented the testbed in wireless networks using CoAP californium. There are the best case and the worst case in the proposed scheme. We assume that the arriving time of packet follows an exponential distribution to consider the general case and that no packet losses occur. In addition, the cluster with 5 topics is generated. Each user sends 100 messages of a 10-byte size. Wireshark was used to measure the channel occupancy time, bandwidth usage, connection maintenance overhead throughout the experiments [11].

C. Channel Occupancy Time

We first compared the channel occupancy times for the existing MQTT and CoAP schemes and Cluster CoAP MQ scheme. Message transmission process from users to server is assumed to be equal in the all schemes. In the CoAP scheme, the channel occupancy time represents the time elapsed from the transmission of a CON message to the reception of the related ACK. Cluster CoAP MQ has the additional elapsed time by multicast. Similarly, in the MQTT case we consider the time for the SUBSCRIBE and SUBACK messages to be exchanged.

Figure 7 shows the comparison of channel occupancy times for the candidate schemes in the experiment. The proposed scheme has less channel occupancy time than the existing MQTT and CoAP schemes.

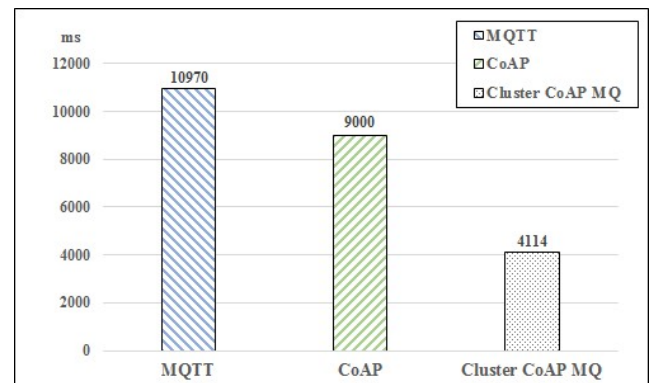


Figure 7. Channel Occupancy Time

D. Bandwidth Usage

We now compare the three candidate schemes with reliability (MQTT QoS 1 and CoAP CON). We consider the bandwidth usage of messages from the server, and we assume that bandwidth usage of messages from users is the same. The bandwidth usage is measured in unit of bytes transferred. An MQTT QoS 1 transaction consists of the two messages: SUBSCRIBE and SUBACK. A CoAP CON transaction also consists of the two messages: CON and ACK. In the meantime, a Cluster CoAP MQ transaction has the three messages: CON, ACK, and MULTICAST.

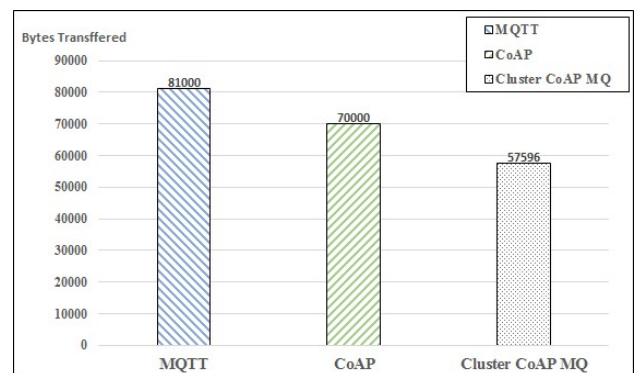


Figure 8. Bandwidth Usage

Figure 8 shows bandwidth usages for the candidate schemes. From the figure we can see that the proposed scheme uses a slightly smaller bandwidth than the existing MQTT and CoAP schemes.

E. Connection Maintenance Overhead

CoAP sensors and MQTT sensors send messages periodically to the server to recognize the connectivity in the network. However, in the proposed scheme, the server sends the messages for connection maintenance to a cluster head. We measured the messages used for connection maintenance messages by using Wireshark, as shown in Figure 9.

From the figure we can see that the Cluster CoAP MQ scheme gives less connection maintenance overhead than the MQTT and CoAP schemes. A connection maintenance message of MQTT is a size of 116 bytes. The message in

CoAP is 125 bytes, and the message of Cluster CoAP MQ is 227 bytes. In MQTT and CoAP, each sensor sends the message to broker. However, in the proposed scheme, only a single cluster head sends the message to the server. Overall, we can see that the proposed Cluster CoAP MQ scheme provides the best performance in the viewpoint of connection maintenance overhead.

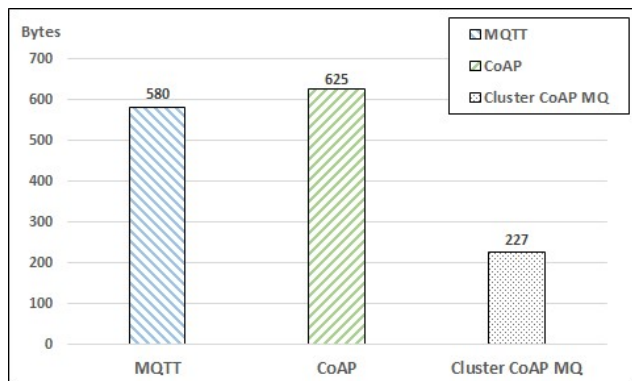


Figure 9. Connection maintenance Overhead

V. CONCLUSIONS

In this paper, we proposed the Cluster CoAP scheme for Message Queuing as an extension to CoAP. We have implemented a testbed of Cluster CoAP MQ, and the performance analysis was made with a testbed implementation.

From the experimental results, we can see that the proposed scheme provides better performance than the existing MQTT and CoAP schemes. The performance gains of the proposed scheme come from the configuration of a cluster for many sensors. The major difference between CoAP and MQTT is that former runs on top of User Datagram Protocol (UDP) and the latter runs on top of TCP. For this difference, CoAP has less bandwidth usage than MQTT. In the meantime, the proposed cluster-based CoAP scheme can reduce about 18% bandwidth usage, compared to CoAP.

ACKNOWLEDGMENT

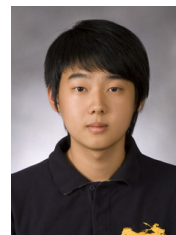
This study was supported by the BK21 Plus project (SW Human Resource Development Program for Supporting Smart Life) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (21A20131600005).

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the National Program for Excellence in Software (R2215-16-1004) supervised by the IITP (Institute for Information & communications Technology Promotion).

REFERENCES

- [1] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, pp. 1645-1660, Sept. 2013
- [2] Lu Tan, and Neng Wang, "Future internet: The Internet of Things," 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Aug. 2010

- [3] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 19., pp. 1497-1516, Sept. 2012
- [4] Dinesh Thangavel, et al., "Performance Evaluation of MQTT and CoAP via a Common Middleware," 2014 IEEE ISSNIP Symposium on sensor networks, April. 2014
- [5] IETF RFC 7252, *The Constrained Application Protocol (CoAP)*, June, 2014
- [6] IETF RFC 7390, *Group Communication for the Constrained Application Protocol (CoAP)*, Oct, 2014
- [7] MQTT version 3.1.1 Plus Errata 01, OASIS Standard, <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>, December, 2015
- [8] Mosquitto.org, "Mosquitto," An Open Source MQTT v3.1/v3.1.1 Broker, <http://www.mosquitto.org>
- [9] IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, June, 1999
- [10] IETF RFC 7641, *Observing Resources in the Constrained Application Protocol (CoAP)*, Sept, 2015
- [11] Wireshark.org, "Wireshark," <http://www.wireshark.org>, September, 2016



Dong-Kyu Choi received B.S degree in Computer Science and Engineering from Kyungpook National University in 2015. Since March 2015, he enters to the M.S program. His current research interests include Internet of Things, CoAP, MQTT, and ZeroMQ.



Joong-Hwa Jung received B.S degree in Computer Science and Engineering from Kyungpook National University in 2016. Since March 2016, he enters to the M.S program. His current research interests include Internet of Things, Message Queuing, CoAP, and MQTT.



Hyung-Woo Kang received B.S and M.S degrees in Computer Science and Engineering from Kyungpook National University in 2011 and 2013. He is now the Doctor's course in school of Computer Science and Engineering from Kyungpook National University. His current research interests include Internet of Things, CoAP, ZeroMQ, and Machine Learning.



Seok-Joo Koh received B.S. and M.S. degrees in Management Science from KAIST in 1992 and 1994, respectively. He also received Ph.D. degree in Industrial Engineering from KAIST in 1998. From August 1998 to February 2004, he worked for Protocol Engineering Center in ETRI. Since March 2004, he has been with the school of Electrical Engineering and Computer Science in the Kyungpook National University as an Associate Professor. His current research interests include mobility control in Future Internet, mobile SCTP, and mobile multicasting. He has also participated in the International Standardization as an editor in the ITU-T SG13 and ISO/IEC JTC1/SC6.