

A Set Notation Configuration CoAP Scheme for Internet of Things

Jinsuk Baek*, Munene W. Kanampiu*, Byeong Hyun Ko**

*Department of Computer Science, Winston-Salem State University, Winston-Salem, NC, USA

**Division of Computer and Electronics Systems Engineering, Hankuk University of Foreign Studies, Korea

baekj@wssu.edu, kanampiumw@wssu.edu, prisoner737@hufs.ac.kr

Abstract— The Constrained Application Protocol (CoAP) is a lightweight protocol specially designed to support constrained Internet of Things (IoT) devices and networks having limited bandwidth. The current growth of IoT market means more sensors in the networks and increased network traffic congestion. Since IoT clients prefer servers whose sensor values have been simultaneously updated it is of paramount importance that servers be highly efficient in synchronizing message updates from its sensors. Our research however, finds the legacy CoAP not optimally efficient in that it introduces network traffic and message duplication hence consuming extra bandwidth. We propose a new Set theory scheme that optimizes synchronized message delivery in IoT by reducing the number of duplicated messages. Our proposed scheme eliminates unnecessary duplicated messages, hence reducing network traffic. On comparing the number of messages sent by servers, our scheme outperformed the legacy CoAP by up to a 50% reduction in the number of messages sent.

Keywords— CoAP, IoT, Sensors, Set theory, Observe

I. INTRODUCTION

The Internet of Things (IoT) [1] is an ecosystem where computing devices and objects are interconnected. This system yields many amazing benefits in our life such as gaining access to the resources of the computing devices through transferring data over a network without the need for human intervention.

Since the IoT devices have limited capability in terms of memory, computational power, and battery, the traditional application protocols such as HTTP [2] are not suitable for IoT environment. As a result, there are lightweight protocols specially designed to support constrained IoT devices and networks having limited bandwidth.

The representative of the IoT protocols as defined in [3] is Constrained Application Protocol (CoAP). The main features of CoAP are very similar to HTTP. However, CoAP is not considered as a compact version of HTTP because its structure and operations are specially optimized for IoT environment. For example, while HTTP provides reliable, persistent, synchronous transport service over TCP, CoAP provides asynchronous message exchange over UDP with low overhead and simple parsing mechanism.

CoAP architecture [4, 5] involves three types of network entities, namely; sensors, server, and clients. Sensors register on a server with their specific topics and send their updated sensing values to the server in a periodic or real-time manner. The server manages a list of registered topics and their corresponding up-to-date values posted by sensors. A client interested in the status of a sensor's values makes the request to the server. That is, CoAP provides asynchronous transport services for clients to interact with a server to access a particular topic of a sensor.

As the IoT market becomes more prominent, the sensors participating in IoT networks have significantly increased both in their functional diversity and numbers. This growth has brought with it not only new IoT requirements but also higher levels of network traffic congestion.

An important observation is that although the topics covered by various sensors of a sensing area may individually differ, their individual topic values can produce a more complete and meaningful overall value when correlated and interpreted together as one entity. For example, in weather forecasting we need a combination of various topics to forecast a hurricane formation e.g. the atmospheric pressure, air temperature, humidity of the oceanic air, and presence of tropical winds. Individual updates of any one of these without the others would not be of much help to the forecasters.

Such a cumulative topics value has more real life meaning for clients, therefore, IoT clients are better served by servers whose sensor topic values have been simultaneously updated. This gives high importance to efficiency in simultaneous sensor topic-updating for servers in IoT. Our research finds the legacy CoAP is not optimal in terms of its existing simultaneously updated message requests in that it introduces network traffic as well as message duplication of some messages hence consuming extra bandwidth.

To improve the existing CoAP, in this paper, we propose incorporating into CoAP a new Set theory scheme that optimizes simultaneous sensor topic message flows in IoT by reducing the number of duplicated messages. Also, unlike with the legacy CoAP, this time, the server will not be allowed to send a message until all the values from all the sensors in the group have been updated.

Our proposed scheme eliminates unnecessary duplicated message transfers from sensors, hence reduces network traffic as well as bandwidth consumption. Furthermore, our proposed scheme does not require any extension of the existing CoAP since it can be implemented with an available option field of the CoAP message format.

The rest of this paper is organized as follows. Section 2 describes the request/response model and message format of the CoAP. In Section 3 we state our goals and explain our methodology while section 4 discusses the expected performance. We conclude the paper and suggest future work in Section 5.

II. CONSTRAINED APPLICATION PROTOCOL

From the abstract protocol layer, the CoAP is located between application layer and transport layer. The CoAP consists of two different layers, namely the message layer and the request/response layer. The message layer deals with UDP segment as the payload for asynchronous transport. The request/response layer manages interactions between the network entities for IoT networks. CoAP defines four different message types: 1) Confirmable (CON), 2) Non-confirmable (NON), 3) Acknowledgement (ACK), and 4) Reset (RST). In CoAP, a reliable transmission can be supported with CON message because a CON message requires the destination entity to acknowledge a successful message receipt by sending back an ACK message with the message ID of the received message. If the destination entity has trouble managing the incoming request, it sends back a RST message. In contrast, if a message does not contain any critical information, NON message, which does not require any ACK, is adopted.

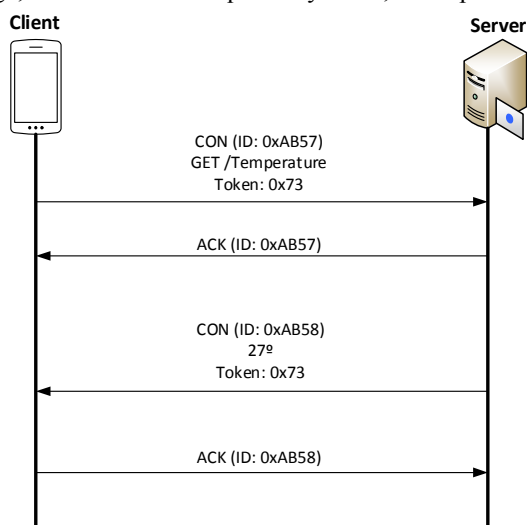


Figure 1. An example of the request/response model of CoAP

In the request/response layer, the request and response can be sent by using a CON or a NON message along with a command to access or post a specific value of a topic managed by the destination entity. Here, the messages can be differentiated with the message ID while a command for a specific topic is uniquely identified with a token ID. Figure 1

shows an example of a CON request and response message with a GET command to access temperature values.

The client sends a CON message to access topic “temperature” value with message ID and token ID. The message is acknowledged with the same message ID. The server now sends another CON message for the requested topic value along with the same token ID but different message ID. Finally, the CON message is confirmed with an ACK message with the same message ID.

Since the CoAP is designed for constrained environments, it defines a message format having a compact binary header as shown in Figure 2 [3].

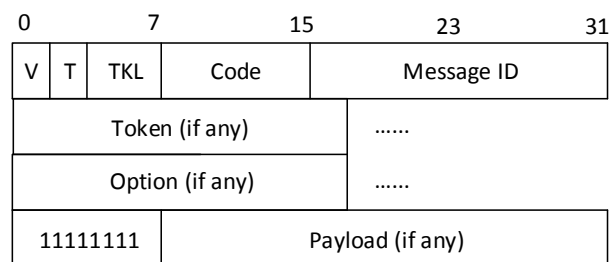


Figure 2. CoAP message format

In order to avoid a fragmented process, a message occupies the data section of a UDP segment. Each CoAP message has a fixed size 4byte binary header including 1) a 2-bit protocol version (V), 2) a 2-bit message type (T) to indicate message type, such as CON, NON, ACK, and RST, 3) a 4-bit token-size to represent token length (TKL), 4) a 8-bit command to represent command code such as GET, POST, PUT, and DELETE, and 5) a 16-bit unique message ID which facilitates in detecting message duplicates. In addition to the 4byte fixed binary header, the CoAP message has a token value field whose size depends on the TKL value, and the option field. Finally, there is a payload marker (0xFF) to indicate the starting position of the payload. The IoT entities recognize additional operations by referencing the option field. Since our proposed scheme requires the use of the option field, we discuss it more in detail in the next Section.

III. PROPOSED SCHEME

In order to be applied to IoT network, the three entities of the CoAP (sensor, server, and client) perform the registration process. At first, the server broadcasts an invite message in its network domain for local sensors to form sensor groups. Each sensor receiving the invite message responds with its identifier and topic information. The server collects these messages for a certain period of time and makes a list of the topics it will serve to its clients. After the configuration of a sensor group, the server advertises its topics by broadcasting the list of topics in its network domain. Clients who are interested in topics provided by the server initiate a request to the server by sending a CON-type GET message. The server responds with an ACK-type response message to allow the clients to join its service group. Based on the requests from the clients, the server maintains and updates its list of topics in a table form. The table consists of fields for each sensor including sensor

identifier, topic, a list of clients who want to get the topics values of the sensor, timestamp, etc. Once the groups are formed, each sensor updates its topic values list by sending a PUT message to the server in a periodic manner. Likewise, clients explicitly request topic values by sending a GET message to the server. Figure 3 shows a generalized CoAP model for humidity message flow.

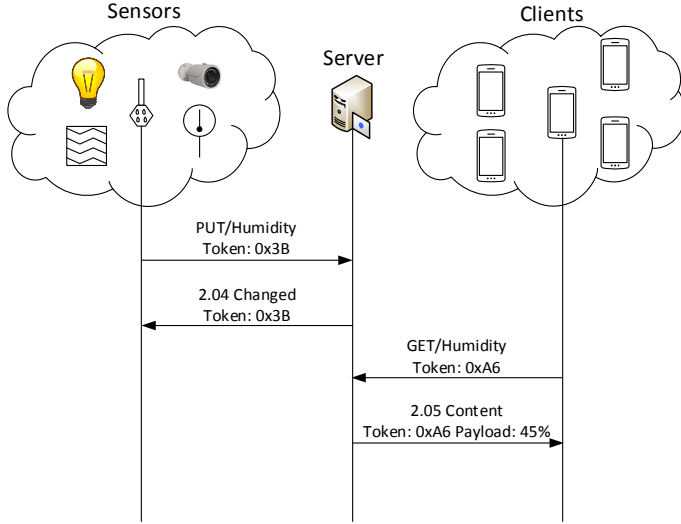


Figure 3. A generalized CoAP model message flow for humidity

Clients with the CoAP-observe option [6] can subscribe for topic values change. Since our aim is to alleviate the communication overhead observed in the legacy CoAP, we need to consider the client requests that are sensitive to the simultaneous changes in the multiple sensor values environment.

In this paper, we focus on how the proposed scheme can efficiently handle the requests of such clients by extending the functionality of the legacy CoAP. In order to simplify our discussion, let us consider two different sensors, A , and B , having different topics. Let us also consider existence of 10 clients, C_1 through C_{10} who want to access the sensor values from these sensors through an interaction with their server. We consider these 10 clients are grouped into four categories based on their interests as follows:

- (1) Clients who only want to access values from sensor A when the value is updated: C_1, C_2, C_3
- (2) Clients who only want to access values from sensor B when the value is updated: C_4, C_5, C_6
- (3) Clients who want to access values from sensor A and B when either value is updated: C_7, C_8
- (4) Clients who want to access values from sensor A and B when both values are updated: C_9, C_{10}

The legacy CoAP has no efficient way to differentiate between the clients in category (3) and (4). In both cases, the server sends individual messages with topic values of sensor A or sensor B .

Let us suppose S_A and S_B is a collection of clients who want to access topic values from sensor A and B , respectively. It maintains the two separate sets as $S_A = \{C_1, C_2, C_3, C_7, C_8, C_9, C_{10}\}$, and $S_B = \{C_4, C_5, C_6, C_7, C_8, C_9, C_{10}\}$. In addition, it can have $S_A \cap S_B = \{C_7, C_8, C_9, C_{10}\}$, as depicted in Figure 4.

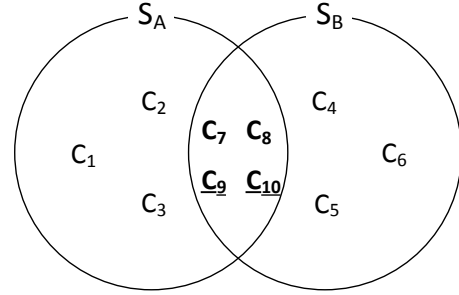


Figure 4. Venn diagram for set S_A and S_B for the legacy CoAP

However, in the case of the clients in category (4), we claim that the received topic values for sensor A or sensor B are meaningless. Normally if the topic value of sensor A is changed, the server sends a message accordingly to all clients in set S_A upon request. Similarly, the server will send a message to all clients in set S_B when the value of sensor B is changed. Therefore, clients C_9 and C_{10} will each receive duplicated messages (each message twice) especially if the two events are intermittent.

In order to handle requests for clients in category (4), the proposed scheme requires the server to maintain an additional intersection set for S_A and S_B , which in our case we define as $S_A \bullet S_B$. Since the clients in category (3) also belong to both sets S_A and S_B , we end up with newly defined sets $nS_A = \{C_1, C_2, C_3, C_7, C_8\}$, $nS_B = \{C_4, C_5, C_6, C_7, C_8\}$, $nS_A \cap nS_B = \{C_7, C_8\}$, and $nS_A \bullet nS_B = \{C_9, C_{10}\}$, as depicted in the Venn diagram in Figure 5.

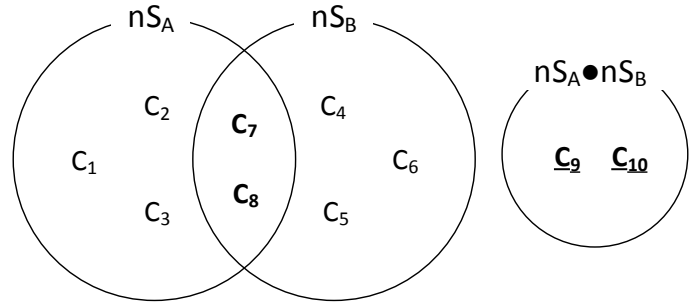


Figure 5. Venn diagram for set nS_A and nS_B for the proposed scheme

For the same intermittent events for sensor A and sensor B , the server now sends a message to all clients in set S_A and set S_B , respectively. Since it refrains from sending a $2n(S_A \bullet S_B)$ unnecessary messages, the number of messages to be sent by the server can be represented by $n(S_A) + n(S_B) - 2n(S_A \bullet S_B)$. On the other hand, if the two events are simultaneous, the proposed scheme will send only $n(S_A \bullet S_B)$ messages while the legacy CoAP would send $n(S_A) + n(S_B)$ messages, which is 14, while introducing duplicated messages to C_9 and C_{10} .

The proposed scheme can be implemented by requiring a client in category (4) to specify the option number in its request message. Figure 6 shows the CoAP option format.

0	1	2	3	4	5	6	7
Option Delta				Option Length			
Option Delta (Extended)							
Option Length (Extended)							
Option Value							

Figure 6. CoAP Option field format

The 4-bit Option Delta can have 16 different values to specify option types, which are represented by numbers. In more detail, the first 13 values, starting from $(0000_{(2)} = 0)$ to $(1100_{(2)} = 12)$ represent the corresponding option type. The $(1101_{(2)} = 13)$, indicates that the protocol uses 1-byte extended Option Delta. Accordingly, the corresponding option type can be obtained by adding 13 to the value of the 1-byte extended Option Delta.

The maximum option type number for a 1-byte extended Option Delta can be 268 which is $(1111111_{(2)} = 255)$ plus $(1101_{(2)} = 13)$. Similarly, the $(1110_{(2)} = 14)$, indicates that the message uses a 2-byte extended Option Delta. Therefore, the option type can be obtained by adding 269 to the values of the 2-byte extended Option Delta. Finally, the $(1111_{(2)} = 15)$ represents the payload marker.

Option Length presents the length of the associated option value. Since the CoAP already defines serval options such as Location-Query, Uri-Path, Proxy-Uri, and so on, we recommend to using a 1 or 2 byte extended Option Delta to support our proposed scheme.

IV. EXPECTED PERFORMANCE

We discuss the expected performance of the proposed scheme with a numerical analysis for a generalized multiple sets environment. We first consider three different sets, S_A , S_B , and S_C with the legacy CoAP as shown in Figure 7.

The clients can be grouped into the following 7 categories: (1) S_A , (2) S_B , (3) S_C , (4) $S_A \cap S_B$, (5) $S_B \cap S_C$, (6) $S_C \cap S_A$, and (7) $S_A \cap S_B \cap S_C$. If we define m as the number of sets, then the server has $\sum_{i=1}^m mC_i$ (or $2^m - 1$) categories.

Unlike in the legacy CoAP in Figure 7, the proposed scheme introduces separate sets for those clients who want to receive a message only when the events involved are simultaneous.

Therefore, as seen in Figure 8, the scheme introduces an extra category for each case where a client's request comes from multiple sets. As such Figure 8 produces $mC_1 + 2\sum_{i=2}^m mC_i$ (or $2^{m+1} - 2 - m$) total categories.

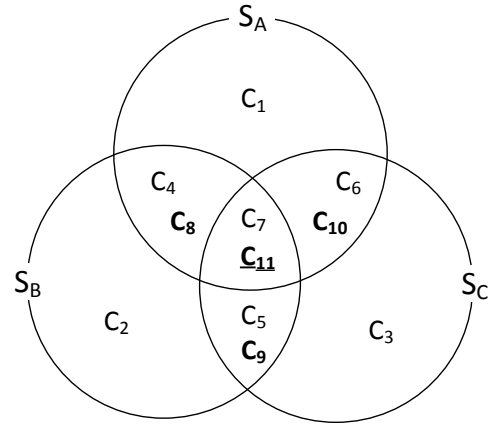


Figure 7. Venn diagram for set S_A , S_B , and S_C for the legacy CoAP

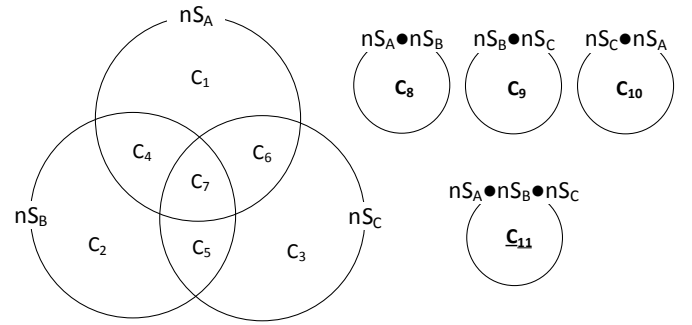


Figure 8. Venn diagram for set nS_A , nS_B , and nS_C for the proposed scheme

For simplification, if we assume each category has one client as in Figure 8, then the number of clients in an individual set S_A , S_B , and S_C for the legacy CoAP can be computed as:

$$n(S_A) = n(S_B) = n(S_C) = mC_1 + 2\sum_{i=2}^m mC_i \quad (1)$$

For example, let us consider $n(S_A)$ of Figure 7 when m is equal to 3. One client (C_1) belongs to $S_A \cap (S_B)^c \cap (S_C)^c$, four clients, in pairs of two, ((C_4, C_8) , (C_6, C_{10})) belong to $S_A \cap S_B \cap (S_C)^c$ and $S_A \cap (S_B)^c \cap S_C$, and two clients (C_7, C_{11}) in $S_A \cap S_B \cap S_C$. Therefore, the set has 7 ($=1+4+2$) clients. This number proportionally increases as a function of the number of clients per each category.

With the legacy CoAP, the server has to send m times of $n(S_A)$ messages when it gets an updated topic value from the m sensors regardless of the property of the events. For the proposed scheme, the number of messages to be sent for intermittent event with m sensors is

$$n(S_A) + n(S_B) + n(S_C) - m\sum_{i=2}^m mC_i \quad (2)$$

Figure 9 shows the number of messages sent by the server for both schemes as a function of the number of sets involved in the given IoT network when the events for the corresponding sensors happen in an intermittent pattern. The difference between the two schemes is shown in Figure 10.

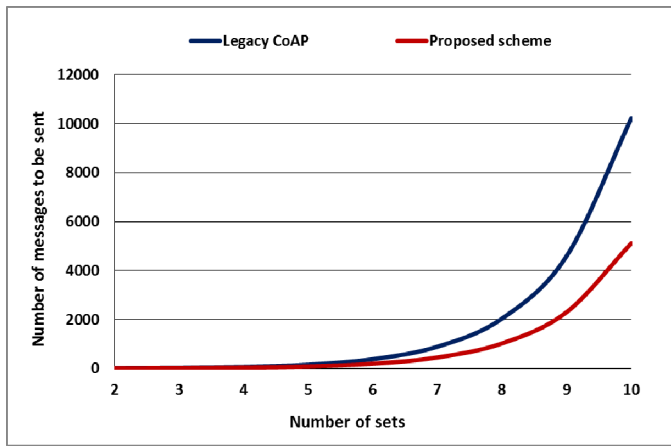


Figure 9. A comparison of the number of messages sent by server

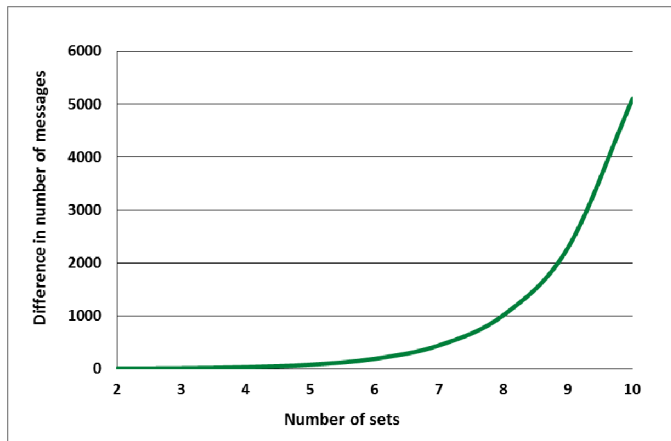


Figure 10. The difference in the number of messages sent by both schemes

In our analysis, we set the minimum number of sets to two and the maximum to ten. When the number of sets is two, (S_A , S_B), the legacy CoAP requires the server to send 6 messages while the proposed scheme requires 4 messages, as the result of excluding the client in set $n_{S_A} \cap n_{S_B}$ two times. When the number of sets is ten, 10,230 messages are sent for the legacy CoAP compared to 5,120 for the proposed scheme, almost 50% reduction in the number of messages sent. We need to note that this result represents the minimum difference for both schemes since we assumed only one client per set. If we considered multiple clients for each set group, the difference would increase proportional to the number of clients. For example, when there are five sets each with only one client, the difference is 75 messages (155 vs. 80), while the difference becomes 150 messages with two clients in each set, and 225 messages with three clients in each set. The difference for both schemes is gets bigger as the number of sets increases, which indicates that the proposed scheme is more appropriate for the growing large scale IoT networks with increasing number of clients and requirements.

With infinite number of clients, this difference theoretically converges to 50%, which indicates the maximum efficiency of the proposed scheme is 50% of the legacy CoAP in terms of the number of messages sent by server and associated amount of bandwidth consumption.

We also evaluate how much bandwidth the proposed scheme can save when the number of clients is unevenly distributed over sets. With the arbitrary number of clients, we consider each set has at least one client. We then distribute all of the remaining clients only to non-simultaneous sets. It means 0% of remaining clients for simultaneous sets, which results in minimum difference for bandwidth consumption between both schemes. After that, we increase the proportion up to the 100% to the simultaneous sets to observe the maximum difference of the bandwidth consumption. We could observe the proposed scheme can save 5% bandwidth whenever we increase this proportion 10%.

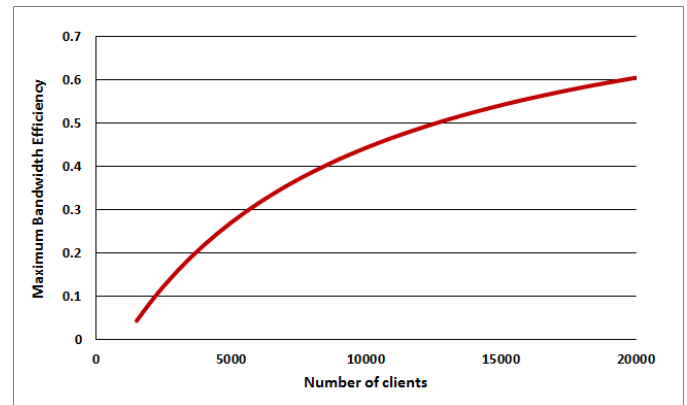


Figure 11. A comparison of the number of messages sent by server

Figure 11 shows the difference for both schemes. In order to show feasibility of the experiment, we consider the number of clients always greater than the number of sets. Therefore, with 1023 sets, we start with 1500 clients and increase it up to 20000 clients. As we can see, the proposed scheme can save 60.54% of bandwidth compared to the legacy CoAP with 20000 clients.

V. CONCLUSION

For a solution for simultaneously updated messages delivery, we proposed a new Set theory scheme that optimizes simultaneous sensor topic message flows in IoT by reducing the number of duplicated messages. Our numerical experiment results show the proposed scheme outperformed the legacy CoAP by up to a 50% reduction in the number of messages sent, indicating that the proposed scheme is more appropriate for the growing large scale IoT networks with increasing number of clients and requirements.

REFERENCES

- [1] S. Chaudhary, et al., "CRAIoT: Concept, Review and Applications of IoT," *Proc. of IoT*, April 2019.
- [2] RFC 2616, "Hypertext Transfer Protocol – HTTP/1.1, June, 1999.
- [3] RFC 7252, "The Constrained Application Protocol (CoAP), June 2014.
- [4] H. Hejazi, et al., "Survey of platforms for massive IoT," *Proc. of IEEE International Conference on Future IoT Technologies*, Jan. 2018.
- [5] B. Dhanalaxmi, and G. A. Naidu, "A survey on design and analysis of robust IoT architecture," *Proc. of International Conference on Innovative Mechanisms for Industry Applications*, Feb. 2017.
- [6] RFC 7641, "Observing Resources in the Constrained Application Protocol (CoAP), Sept. 2015.