ICOIN 2018
The 32nd International Conference on Information Networking (ICOIN 2018)
January 10-12, 2018, Chiang Mai, Thailand

Call for Papers  Committees  Author Guideline  Program  Registration
Venue and Travel Info

GENERAL INFORMATION
The 32nd International Conference on Information Networking (ICOIN) is organized by
KIIEEE and technically co-sponsored by IEEE Computer Society. For the past 31
years, computer communication and networking technologies have changed every
aspect of our lives and societies. While computer networks have contributed largely to
the current ICT advancement, it will play a key role in new ICT paradigms such as 5G,
IoT, SDN/NFV, and mobile cloud computing and will be applied to various areas of the
upcoming society including industry, business, politics, culture, e-health/medicine, etc.

ICOIN is the most comprehensive conference focused on the various aspects of
advances in computer communication and networking technologies. The main
purpose of ICOIN 2018 is to improve our research by achieving the highest capability
and encourage open discussions on recent advances in computer communication and
networking technologies.

Authors are invited to submit original unpublished manuscripts that demonstrate
recent advances in computer communications, wireless/mobile networks, and
converged networks in the theoretical and practical aspects. Accepted papers will be
published in the proceedings with an assigned ISBN number and submitted to IEEE
Xplore, SCOPUS, and EI Compendex..

IMPORTANT DATES
- Paper Submission Due (Final): October 13, 2017
- Acceptance Notification (New): November 6, 2017
- Camera-ready Papers Due: November 24, 2017
- Author Registration Due: November 24, 2017
- Conference Dates: January 10-12, 2018
P3-5: Occupancy-Based Adaptive Dimmable Lighting Energy Management Scheme Combined with Cyber Physical System, Changmin Lee and Duckhee Lee (Korea Railroad Research Institute, Korea); Jaiyong Lee (Yonsei University, Korea)

P3-6: Innovative and Technological Potential of the Region and Its Impact on the Social Sector Development, Djamiya Skripnik, Nikolay Didenko, Ksenia Kikkas and Vladislav Sevashkin (Peter the Great St. Petersburg Polytechnic University, Russia); Garif Romashkin (The Tyumen State University, Russia); Sergay Kulik (Peter the Great St. Petersburg Polytechnic University, Russia)

P3-7: Identifying Signal Source Using Channel State Information in Wireless LANs, Yonghwi Kim, Sanghyun An and Jungmin So (Hallym University, Korea)

P3-8: ISO as a Real Source of Funding. Pricing Issues, Julia Dubolazova (Peter the Great St. Petersburg Polytechnic University, Russia); Ekaterina Malevskai-Malevich (Peter the Great St. Petersburg Polytechnic University Russian Federation, Russia); Dnail Demidenko (Peter the Great St. Petersburg Polytechnic University, Russia)

P3-9: The Analysis of Convergence–Divergence in the Development of Innovative and Technological Processes in the Countries of the Arctic Council, Nikolay Didenko, Djamiya Skripnik, Ksenia Kikkas, Sergay Kulik and Victor Merkulov (Peter the Great St. Petersburg Polytechnic University, Russia); Garif Romashkin (The Tyumen State University, Russia)

P3-10: Energy Efficient K-means Clustering-based Routing Protocol for WSN Using Optimal Packet Size, Madhiha Razzaq, Devarani Devi Ningombam and Seokjoo Shin (Chosun University, Korea)

P3-11: Digitalization of Banking Services in Russia: Overview, Arseniy Bondarev, Tatiana Kudryavtseva and Angi Skhvediani (Peter the Great St. Petersburg Polytechnic University, Russia)

P3-12: A Fault Management System for NFV, Hanbum Lee (Kwangwoon University, Korea); Sang il Kim (Kwangwoon University, Korea); Hwasung Kim (Kwangwoon University, Korea)

P3-13: Behavior Recognition and Disaster Detection by the Abnormal Analysis Using SVM for ERESS, Shingo Nakajima, Toshiki Yamasaki, Koki Matsumoto, Kazuki Uemura and Tomotaka Wada (Kansai University, Japan); Kazuhiro Ohtsuki (Kobe University, Japan)

P3-14: Enhanced Cluster-based CoAP in Internet-of-Things Networks, Dong-Kyu Choi and Joong-Hwa Jung (Kyungpook National University, Korea); Seok-Joo Koh (Kyungpook National University & College of IT Engineering, Korea)

P3-15: Distributed Pub/Sub Model in CoAP-based Internet-of-Things Networks, Joong-Iwa Jung and Dong-Kyu Choi (Kyungpook National University, Korea); Seok-Joo Koh (Kyungpook National University & College of IT Engineering, Korea)

P3-16: Identification of TCP Congestion Control Algorithms with Convolution Neural Networks, Takahiro Nagiwa and Kouji Hirata (Kansai University, Japan)

P3-17: Formal Verification for Wireless Sensor Network in Consideration of Communication Errors, Akihiro Ikeda, Naoki Akiyama and Toshiaki Miyazaki (The University of Aizu, Japan)

P3-18: Low-Power Wind Generator, Albina Gazizulina (Peter the Great St. Petersburg Polytechnic University, Russia); Y Makarichev, A Anufriev and Yury Ivannikov (Samara State Technical University, Russia); Nikolay Didenko (Peter the Great St. Petersburg Polytechnic University, Russia)

P3-19: Use of Nomadic Computing Devices for Storage Synchronization, Anuradha Wickramarachchi, Dulaj Randima Atapattu, Pamoda Wimalasiri and Ravidu Mallawa Arachchi (University Of Moratuwa, Sri Lanka); Gihan Dias (University of Moratuwa, Sri Lanka)

P3-20: Cooperative Between V2C and V2V Charging: Less Range Anxiety and More Charged EVs, Oanh Tran Thi Kim, Nguyen H. Tran, VanDung Nguyen and Choong Seon Hong (Kyung Hee University, Korea)
Distributed Pub/Sub Model in CoAP-based Internet-of-Things Networks

Joong-Hwa Jung  
School of Computer Science and Engineering, Kyungpook National University  
Daegu, Korea  
godopu16@gmail.com

Dong-Kyu Choi  
School of Computer Science and Engineering, Kyungpook National University  
Daegu, Korea  
supergint@gmail.com

Seok-Joo Koh  
School of Computer Science and Engineering, Kyungpook National University  
Daegu, Korea  
sjkoh@knu.ac.kr

Abstract— As the number of IoT devices with high performance has increased, so many streaming services using IoT devices have been developed. One of the major challenges in the realization of the services is delay. MQTT and CoAP are usually used for such services because these protocols can support the pub/sub architecture. However, those protocols use only one broker or server. Thus, they may give poor performance, when the number of the topics associated with the IoT streaming service gets larger. In this paper, we propose a new scheme for distributed, interoperable architecture among servers, in which two or more servers can create a group and a client can view the group as a powerful server. By implementation and experimentation, we can see that the proposed scheme provides the performance gains over the existing MQTT and CoAP schemes in terms of total delay by approximately 45% where the number of topics is 500 in the IoT streaming services.

Keywords—Internet of Things, CoAP, MQTT, distributed, brokers

I. INTRODUCTION

The Internet of Things (IoT) is where everything, including people, is connected to the Internet and information about everything is created, collected, shared and utilized. IoT technologies will be utilized while overcoming technological and economic limitations such as popularizing smart sensors, lowering prices of communication modules, spreading smart devices, expanding wireless communication networks, and changing to the future Internet. It is anticipated that various sensors and IoT market will grow in earnest in line with the rapid spread of smart phones and the realization of wearable computing [1, 2].

The IoT environment has characteristics such as limited bandwidth, low power energy, and storage capacity. The general protocol used on the Internet like HTTP is not suitable for IoT environment [9]. The Internet Engineering Task Force (IETF) constrained RESTful (CoRE) working group enacted the CoAP (Constrained Application Protocol) which is a lightweight protocol for IoT environment. Like CoAP, the MQTT is also an ISO standard lightweight messaging protocol over TCP/IP protocol for IoT environments [3].

With the growth of IoT market, the number of sensor nodes and smart devices has increased exponentially. For this reason, the number of topics that the server needs to manage increases, which results in a heavy load on the processing time of the server in the IoT environments.

To overcome these problems, in this paper, we propose a simple extension of CoAP using a distributed approach. In this approach, several servers operate such as one powerful server. The servers independently manage the subscribers who want to receive notification for its topics. But a server brings some topics from another server in the same group easily. By using this approach, we can reduce the delay to transfer a message from publisher to subscriber and increase the safety of the system. The proposed scheme can be used for CoAP-based streaming service.

This paper organized as follow. Section II introduces the related works on the MQTT and CoAP protocols for IoT environment. In Section III, we describe the proposed scheme with a distributed pub/sub architecture. Section IV presents the implementation and performance comparison with the existing schemes. Finally, Section V describes the conclusion of this paper.

II. RELATED WORKS

A. MQTT

The Message Queue Telemetry Transport (MQTT) is a lightweight message transmission protocol based on push technology, which is optimized for a limited bandwidth communication environment such as Machine-to-Machine (M2M) and Internet of Things (IoT). The MQTT protocol uses a message broker to send a specific message and a recipient to subscribe to the message, instead of the Client/Server method commonly used in push technology [4].

MQTT is a protocol based on a publish-subscribe model rather than on a client/server basis. This model works with topics. Topics can be organized hierarchically using forward slashes (/) to manage efficiently large numbers of sensor devices. When a broker or a client publishes the messages, it does not check whether the message was sent successfully. The subscriber who subscribed to a specific topic will be able to receive the message, when the message is published to the broker. In the pub-sub model, publishers and subscribers do not interact directly. They can only communicate via a broker. The communication procedure is illustrated in Figure 1. Figure 1 is the architecture of MQTT. Both of the subscriber and the publisher are the clients of the broker. The publisher sends a message containing the status to the broker, when the new status is updated. When a broker receives a message, it sends a message to the subscriber who is interested in the particular topic the message belongs to [5].
MQTT provides Quality of Service (QoS) for reliable message transmission. QoS refers to the level of guaranteed latency and data loss rate in the network for the quality of the communication service. The MQTT defines 3-levels of QoS. QoS 0 transmits a message only one time unconditionally, and QoS 1 receives a confirmation message at least once. QoS 2 delivers exactly once through a 4-way-handshaking. In QoS 1, when PUBLISH is sent, acknowledgment is possible by PUBACK message.

B. CoAP and CoAP Observe

1) CoAP
CoAP is a standardized protocol in the Constrained RESTful Environment (CoRE) working group of the IETF for the communication of devices with limited computing capability, such as IoT. Reliable synchronous transport over TCP and HTTP is not suitable for Internet environment where there are many resource constraints [8]. CoAP is a communication protocol including the concept that compensates the shortcoming of UDP over UDP of asynchronous transport method [6].

2) CoAP Observe
The CoAP Observe is a simple extension of CoAP. It is appropriate when a client is interested in a status of resources over a period time. In this protocol, "observer" will register with a specific "subject", for which the observer wants to be notified about the status of the subject. Figure 3 shows CoAP observe model.

The subject is a resource in the namespace of a CoAP server. By assigning a space called "subject" to the CoAP server, the client interested in the subject accesses the server through the subject. The subject manages a list of registered observers. If an observer is interested in multiple subjects, it should register separately on each subject. When a client makes an Observe request on a specific subject, it will receive notifications from the server without a separate request message whenever the subject changes. This communication procedure is very similar with MQTT rather than CoAP [7].

3) CoAP Message Type
Figure 4 shows the CoAP message format. The first line of the figure depicts the CoAP header. In the figure, V means the CoAP Version and T means the message type such that CON (0), NON (1), ACK (2) and RESET (3). TKL indicates the length of the Token field. The Code field indicates the method such as GET, DELETE, POST and PUT in the request, which implies the response code such as 2.02, 4.04 in the response message. The header is followed by the Token value. A message may include a list of one or more options or not.

Each option has specific functionality. We will define some options to help the proposed pub/sub model. For this purpose, let us see the CoAP option field. Figure 5 shows the CoAP option format. Option Delta indicates Option type. A number between 0 and 12 indicates the option type, in which three values are reserved for special constructs. The number 13 means that the 1-byte option delta extended field is used. The value of the option type number minus 13 is stored in the option delta extended field. Similarly, the number 14 implies that 2-byte option delta extended field is used. This field has
the value of the option type number minus 269. With this principle, we have to fill out the option delta field with 13 and the option delta extended field with 37 in order to represent the option type number 50. The Option length indicates the length of the option value. The Option length and the Option length extended fields will be used in the similar way with the option delta and option delta extended fields.

### Table 1. Option Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REDIRECT</td>
<td>Indicates that this packet includes another server address who manage the target resource.</td>
</tr>
<tr>
<td>JOIN</td>
<td>It is used for a server to participate in a group.</td>
</tr>
<tr>
<td>QUERY</td>
<td>It is used to retrieve the number of topics in the receiver.</td>
</tr>
<tr>
<td>SHIFT</td>
<td>It is used to bring some topics from the receiver.</td>
</tr>
</tbody>
</table>

Each option uses CoAP option fields. Let us take a closer look. First, the number between 300 to 304 is used to define the four options type. Option Extended Delta field is used in order to express the corresponding options type. For example, if a packet has the REDIRECT option only, the Option Delta Field of the packet has 14 and the Option Delta Extended Field has 31 that represent REDIRECT option type 300. Option value fields may or may not be used, depending on options. Table 2 shows the option type and option value of each option.

### Table 2. The Option Type and Value of Each Option Field

<table>
<thead>
<tr>
<th>Option name</th>
<th>Message type</th>
<th>Option Type</th>
<th>Option Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>REDIRECT</td>
<td>Response</td>
<td>300</td>
<td>Data type (0 or 1)</td>
</tr>
<tr>
<td>JOIN</td>
<td>Request</td>
<td>301</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>301</td>
<td>Data type (0 or 1)</td>
</tr>
<tr>
<td>QUERY</td>
<td>Request</td>
<td>302</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>302</td>
<td>The number of topics</td>
</tr>
<tr>
<td>SHIFT</td>
<td>Request</td>
<td>303</td>
<td>Data type (0 or 1)</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>303</td>
<td>Data type (0 or 1)</td>
</tr>
</tbody>
</table>

REDIRECT option is used only in a response message. The data types 0 and 1, used in Table 2, mean that the data type in the payload of the packet is JSON and XML.

### C. Dataflow

The four message options for the proposed model were defined. In this section, we will discuss the details about the following three scenarios.

1) **JOIN Process**

Figure 7 shows that the dataflow of a procedure for participating a group. In this figure, Server 5 is the one who wants to join a group. Server 5 sends the GET request with the JOIN option to Server 2, which belongs to the group, in order to find out the IP address of the head server. Server 2 responds with the head server address. In this point, the head server is one who starts to make a group and the head server manages too many topics compared to other servers. To solve these problems, we define four options. Table 1 shows each option name and description.
the information of servers that belongs to the group by using a circular queue. After receiving the head server's IP address, Server 5 sends the PUT request containing JOIN option to the head server, and the head server responds with the addresses of Server 3 and Server 4. In this scenario, Server 5 is located between Server 3 and Server 4 in the circular queue. Then, Server 5 sends messages with JOIN and SHIFT options to Server 3 and Server 4 in order to get some topics managed by them. Server 3 and Server 4 send the notification with REDIRECT to clients who are observing to the specific topic from them.

![Dataflow of JOIN Process](image)

Fig. 7. Dataflow of JOIN Process

2) Subscription Register Process
A client who is interested in a topic in the group can issue a GET request with an OBSERVE option to head server of the group. The head server, who receives the request, will check whether there is a server who provides the notification for the target topic. If not, the head server returns with a message containing the OBSERVE and REDIRECT options.

If not, the head server sends a POST message to a server in order to make the server provide the notification for the topic. After that, the head server responds with a message containing the OBSERVE and REDIRECT options. The packet with REDIRECT option must include other server’s IP address. This scenario is illustrated in Figure 8.

In this figure, Client 1 first issues a GET request with an observe option to head server. But, the notification is not provided from any server. So head server sends POST message to server 1 in order to make server 1 provide the service. After that, the head server returns a message with REDIRECT option and the address of Server 1 to client 1. Client 1 can notice the correct location of the topic with the response and send a request message to Server 1.

![Dataflow for Subscribe Process](image)

Fig. 8. Dataflow for Subscribe Process

3) Topic-Balance Process
It severely reduces the performance of the proposed scheme for a server to have too many topics compared with other servers. For this reason, we need some functionality for load-balancing. In this paper, the functionality for load-balancing is called Topic-Balance. Figure 9 shows that the data flow of Topic-Balance process.

![Dataflow for Topic-Balance Process](image)

Fig. 9. Dataflow for Topic-Balance Process

For Topic-Balance process, a periodic interaction between two servers is needed. In Figure 9, Server 1 periodically sends a message with QUERY option and Server 2 reply a packet which has the number of topics. If Server 1 find out that Server 2 has many topics compared with one’s topics, Server 1 issues a GET request with SHIFT option in order to bring some topics from Server 2. After that, server 1 sends a PUT request to head server and Server 2 sends a REDIRECT message to the client 1 in order to notify the movement of the topics.
IV. IMPLEMENTATION AND PERFORMANCE ANALYSIS

A. Testbed Implementation

We implement the testbed of the proposed scheme by using Californium and PAHO libraries based on JAVA [10]. Figure 10 shows a testbed scenario.

![Figure 10. Experimental Structure](image)

We assume that each topic has 1 publisher and 20 subscribers in this scenario. We analyze the register processing time and the total delay from a publisher to a subscriber by comparing the scheme with existing CoAP and MQTT scheme. Table 3 briefly shows the parameter information used for the experiment.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_t$</td>
<td>The number of topics</td>
<td>variable</td>
</tr>
<tr>
<td>$N_s$</td>
<td>The number of subscribers</td>
<td>1 subscriber per a topic</td>
</tr>
<tr>
<td>$N_p$</td>
<td>The number of publishers</td>
<td>1 publisher per a topic</td>
</tr>
<tr>
<td>$T_d$</td>
<td>Total Delay</td>
<td>obtained by experimental</td>
</tr>
<tr>
<td>$T_r$</td>
<td>Register Processing Time</td>
<td>obtained by experimental</td>
</tr>
</tbody>
</table>

B. Register Processing Time

We first compare the register processing time for the existing MQTT and CoAP schemes and the proposed distributed pub/sub scheme. The three schemes commonly have the register processing time in which a client represents its interest in a topic. Figure 11 shows the comparison of register processing time for the candidate schemes when the number of topics increases. The proposed scheme has higher processing time that the existing MQTT and CoAP schemes. It is because that a client receives a response containing the REDIRECT option when the server, whom the client issues the GET request to, do not manages the target topic. In this case, it seems to give worse performance to use this proposed scheme when registering and canceling a subscription occur frequently.

![Figure 11. Register Processing Time](image)

C. Total Delay

We compare total delay ($T_d$) what is one of the best important components of their performance in the publish/subscribe scheme. $T_d$ is the sum of transmission time from a publisher to a subscriber via server and processing time of the server. Each publisher sends a message to the server in the interval following Exponential distribution. Figure 12 shows that the figure of total delay based on the number of topics. We can see that the proposed scheme provides the best performance, as the number of topics increases. This is because that the processing time is reduced by distributing the topics into many brokers/servers.

![Figure 12. Total Delay](image)

V. CONCLUSION

In this paper, we proposed a distributed pub/sub scheme based on the CoAP observe model. We have configured a testbed for
the distributed pub/sub scheme and then analyzed the performance.

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 distributing topics into multiple servers. The proposed distributed pub/sub scheme can reduce total delay by approximately 45%, compared to the existing CoAP scheme when the number of topics is 500.

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), and by the National Program for Excellence in SW (2015-0-00912) supervised by the IITP (Institute for Information & Communications Technology Promotion), and also by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2017R1D1A3B03032156).

REFERENCES

[8] IETF RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, June, 1999