Implementation of CoAP-Lite Protocol over USN

Sun-Chul Jeong, So-Ra Yu, and Hoe-Kyung Jung, Member, KIMICS

Abstract—It is necessary for developing specific application protocol for mobile sensor end nodes and data collection devices that have constrained resources and computing powers. Because of these needs, IETF WG developed CoAP protocol and Internet draft was released, and processing to international standardization of this protocol. Because sensor nodes and information collection devices have restricted resources and computing powers, there are some limitations for applying to this proposed protocol directly. Thus in this paper we shows the implementation of full CoAP protocol for server, works on Linux and CoAP-Lite over telos which works on TinyOS for information collection devices. In this implementation we verified the main activities of CoAP protocol over USN by modifying some hardware with dependent algorithm.

Index Terms—CoAP, CoAP-Lite, USN, Sensor Network, TinyOS

I. INTRODUCTION

BECAUSE of small-sized and low-powered sensor network nodes with limited capabilities used for home networking, building management and remote information gathering such as temperature, humidity. It is necessary to develop the special purpose protocol for each application services. In sensor networks, program is developing inefficiencies in forms of hard-coding, because of absence of well-formed protocols. To improve it, a new type of protocol is needed to be developed, in the IETF (Internet Engineering Task Force) CoREWG (Working Group) has developed CoAP (Constrained Application Protocol) and the international standardization is in progress. Because of having limited resources and hardware characteristics, this proposed CoAP protocol is implemented in forms of CoAP-Lite. In this paper, we implemented CoAP protocol over sensor nodes and data collection devices, and show the Internet draft’s problems and propose the solutions. This paper consists as follows. Chapter 2 deals with related researches. Chapter 3 describes how to implement the CoAP protocol. Conclusions and future works are discussed in Chapter 4.

II. RELATED WORKS

2.1 Sensor Nodes
In recent years, there are many vendors that have made sensor nodes for USN (Ubiquitous Sensor Network). These sensor nodes can be divided into data collection devices and sensing nodes. Sensing nodes use RF (Radio Frequency) to communicate with each sensing nodes and data collection devices. In the other side, data collection devices use RF to communicate with remote sensing node, and use Ethernet to communicate with other Ethernet devices. The representative products of sensor nodes are MTM-CMx000 series [2] and MTI-ETH1000 [3] which manufactured by maxfor®.

![sensing node(CM5000) and Ethernet Device(ETH1000)](Fig. 1. MTM-CM5000 and MTI-ETH1000 for USN)

These sensor nodes are one of the wireless sensor network development platform that support 6LoWPAN based on TinyOS[4] 1.x and later version

2.2 CoAP Protocol
CoAP protocol is developing by IETF WG for sensor nodes and its managing servers. Compare with PCs, target sensor nodes have limited resources and computing powers, the main goal of this protocol is minimizing communication data size. Because the protocol is now Internet draft status and in progress to international standardization, there are some logical shortages. Though it has logical shortages, it is need to be implemented for uniformed-protocol that helps efficient managing sensor nodes.

2.3 6LoWPAN
6LoWPAN is one of the IETF WG and its main
protocol name that is in progress international standardization of architecture over Low-power Wireless Personal Area Network and sensor network. RFC 4944 defines header compression, fragmentation and assemble mechanisms for IEEE 802.15.4 frame transmissions [5]. Draft-ietf-6lowpan-nd-13 relates on “6LoWPAN Neighbor Discovery Optimization” and concentrates on functionalities of edge router [6]. A study on low-powered mobile sensor nodes deals with the way to sustaining services by controlling data sending and receiving. Continuous data sending and receiving consumes much battery and cannot sustain stable services. To solve these problems, the nodes are sleep state at normal, and nodes are active when neighbor nodes wake up.

III. COAP-LITE IMPLEMENTATION

3.1 Topology

Fig 2 shows the topology for CoAP-Lite protocol. There are three groups of network elements such as server that works on TinyOS, proxy and clients.

3.2 CoAP Protocol Message Flow

Fig 3 shows the simple example of CoAP protocol’s message flow. If there is no error, the messages flow by order (2) -> (3) -> (4) -> (5) -> (6) respectively if there is no retransmission. If client runs with retransmission module, timeout event is detected automatically. Proxy redirects the message from server to client. When request and response message is received, each network elements parse message, and if there are errors, the elements send a response message with proper response error code. In general, proxy server can have stateful and stateless mode, but CoAP protocol does not define each modes. Instead of definition of proxy mode, CoAP proxy uses URI_SCHEME information to determine destination of the received messages. If proxy received response message(6), it uses URI_SCHEME to determine destination of server too.

3.3 CoAP-Lite Message Flow

Though CoAP proxy uses URI_SCHEME for getting destination URI, end nodes can use very short message, so URI_SCHEME information is too long to process at the end of the nodes relatively. For this reason, we modified data message flow for the verification of CoAP protocol over the sensor nodes. Fig 4 shows the CoAP-Lite message flow we used.

![Fig. 3. CoAP Protocol Message Flows with Retransmit](image)

![Fig. 4. CoAP-Lite Message Flows](image)

3.4 CoAP-Lite Data Processing

CoAP-Lite protocol is highly related on hardware characteristics such as telos and UART(Universal Asynchronous Receiver Transmitter). There are several threads for implementation of CoAP-Lite protocol and circular queue for producer-consumer algorithm.

![Fig. 5. CoAP-Lite Threads and Activities](image)

Fig 5 shows the threads and circular queue for UART data buffer and its logics. Because different threads share one circular queue, it is needed to solve the racing condition problem. To solve the problem producer-consumer algorithms are used. “Send Thread” makes CoAP request message and add this message to telos message’s data fields. In telos, CRC (Cyclic Redundancy Check) is necessary to verify the original data and received messages. This CRC value is in inner side of telos message. If the correct telos message is made, “Send
Thread” sends this telos message to proxy. “Producer Thread” inserts the received messages to UART buffer. Because of sensor nodes’ characteristics, sensor nodes and data collecting devices send the arbitrary bytes of message. In experimental environments, almost all the data is fragmented. Other side, “Consumer Thread” gets telos message from UART buffer, and, must remove telos message. Received UART data is exchanging ‘0x7E’ to ‘0x7D5E’, ‘0x7D’ to ‘7D5D’ to avoid conflicting with telos specific symbol (‘0x7E’), so these special characters must be changed correctly.

Fig 6 shows server-side message dump screen shots. In Fig 6(a), 14th byte (0x41) is a start of real CoAP message, and 29th byte (0x65) is the last of CoAP message. 30th ~ 31th bytes (0x5E09) indicate CRC and 1st, 32th byte is a symbol of telos message. Payload (19th ~ 29th bytes) indicates “temperature”. Fig 6(b) shows received 28 bytes message dump that is fragmented to 1 byte and 27 bytes. This fragmented data is stored into UART buffer with assembled telos format. “Consumer Thread” uses this data showed at fig 6(c). Fig 6(d) shows data dump after deleting telos symbol and special character and fig 6(e) is a pure CoAP message that its response code is “200 OK” and the payload indicates “32.63 oC”.

In fig 6(a), dump data from 14th to 28th byte is “0x41 0x51 0x3A 0xBE 0x9B 0x74 0x65 0x6D 0x70 0x65 0x72 0x61 0x74 0x75 0x72 0x65”. Table 1 shows the brief meaning of CoAP message format.

3.5 Sensor Node Activities
In fig 4, client sends gathering information periodically. But in general, ideal protocol topology use protocol between all other network elements shown in fig 7. Though, there is an ideal topology, we didn’t use this topology because of our policy the verification must be performed based on Internet draft [1]. In Internet draft, this protocol works based on UDP (User Datagram Protocol) binding. Unfortunately, the sensor nodes we have cannot get IP (Internet Protocol) address just because it communicates with not Ethernet but RF. To communicate between sensor nodes or between base station and sensor nodes, RF is a unique way. Without IP address UDP binding is meaningless. Moreover when it communicate between server and sensor nodes (2 CoAP shown in fig 7), it is impossible without having RF device at server-side because of the same reason. In addition, sensor nodes have limited resources, sensing data that have forms of oscilloscope values depend on some data such as inner voltage, temperature, humidity and so on.

<table>
<thead>
<tr>
<th>nth</th>
<th>Hexa</th>
<th>Binary</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41</td>
<td>0100 0001</td>
<td>01(ver),00(confirmable req) 0001(option count)</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td>0000 0001</td>
<td>0000 0001(code GET)</td>
</tr>
<tr>
<td>3-4</td>
<td>3ABE</td>
<td>0011 1011</td>
<td>Transaction ID (Little Endian)</td>
</tr>
<tr>
<td>5</td>
<td>9B</td>
<td>1001 1011</td>
<td>1001(option delta) 1011(length)</td>
</tr>
<tr>
<td>6-16</td>
<td>74~65</td>
<td>0111 0100~0110 0101</td>
<td>temperature(payload)</td>
</tr>
</tbody>
</table>

3.5 Sense Node Activities
In fig 4, client sends gathering information periodically. But in general, ideal protocol topology use protocol between all other network elements shown in fig 7. Though, there is an ideal topology, we didn’t use this topology because of our policy the verification must be performed based on Internet draft [1]. In Internet draft, this protocol works based on UDP (User Datagram Protocol) binding. Unfortunately, the sensor nodes we have cannot get IP (Internet Protocol) address just because it communicates with not Ethernet but RF. To communicate between sensor nodes or between base station and sensor nodes, RF is a unique way. Without IP address UDP binding is meaningless. Moreover when it communicate between server and sensor nodes (2 CoAP shown in fig 7), it is impossible without having RF device at server-side because of the same reason. In addition, sensor nodes have limited resources, sensing data that have forms of oscilloscope values depend on some data such as inner voltage, temperature, humidity and so on.

Fig. 6. Message Data Dump
3.6 Developed Environments

As shown in fig 2, there are three types of network elements. Server, proxy, sensor nodes are implemented by followed environments.

3.6.1 Server

Server works on TinyOS version 1.5. Because the server is highly related on administrators, we implemented server over Windows. It can easy to control, understand and use. CoAP-Lite protocol stack that is modified by CoAP protocol full stack is used to control UART buffer. For example, it is added eliminating telos specific processing and producer-consumer algorithm.

3.6.2 Proxy

Proxy consists of two devices to communicate with RF and Ethernet. Ethernet is used to communicate between proxy and server, and RF is used to communicate between proxy and clients. Proxy is implemented by modifying TOSBase[7] module that is developed by NesC[8] supported by TinyOS. To communicate with RF and Ethernet, we used RF side devices as MTM-CM2000, and Ethernet side devices as MTI-ETH1000 manufactured by maxfor®. To develop proxy module, we must use TinyOS and porting to devices. Because proxy must have IP address, we should configure MTI-ETH1000 by EG-SR-7100A Configuration Tool as shown in fig 8.

3.6.3 Client

In USN, client is the remote sensor nodes that collect temperature, humidity and so on, and transmit gathering information periodically to data collecting nodes with RF. To implement a client, we use Oscilloscope module developed by NesC based on TinyOS.

To manage the sensor nodes efficiently, standardized protocol is needed. In this needs IETF WG is developing CoAP protocol and in progress to international standardization to unify managing protocol. But this protocol is under development and it can contain some shortages. Therefore, it is essential to verify of this CoAP protocol over USN, so we implemented this protocol to the server and data collection devices. These devices should not have CoAP protocol full stack because of having limited resources and hardware dependent characteristics. So we implemented modified CoAP-Lite typed protocol stack and keep hardware dependent characteristics. To maintain these properties, some parts of CoAP protocol full stack have been changed. For example, when we send request message from server to client side, we make CoAP message, and include this to data field of telos. On the other side, when we receive response data, we eliminatetelos message excluding CoAP message. Basically telos has a short data length, we cannot send long message over telos. For this reason, we don’t use URI_SCHEME logics. In addition, sensor devices send arbitrary bytes of data at one time, producer-consumer algorithm and circular queue are necessary.

Though Internet draft proposed the UDP binding to communicate between network elements, UDP binding is meaningful in case works on IP address. To verify the CoAP protocol logic and its behaviour, we use our topology instead ideal protocol topology.

In this implementation, we verified the CoAP-Lite protocol over USN by modifying some algorithms for hardware dependent characteristics.

As described before, to verify CoAP protocol stack quickly over USN, we didn’t consider full CoAP protocol stack supposed at Internet draft, and concentrated on the activities and its algorithm.

To save battery consumption or power of sensor node, the client must transmit when request is received instead periodic transmission. For this, it is necessary to develop an appropriate algorithm.
REFERENCES


Sun-Chul Jeong He received the B.S. degrees from Korea Maritime University and the M.S. degrees in Mechanical Engineering from ChungNam National University, in 1990 and 1999, respectively. From 1995 to 2010, he worked as a researcher in Nuclear Power Lab., KEPRI. Since 2011, he has worked as a senior researcher of Operation Technology Center. Since 2011 he is a Ph.D student in Department of Computer Engineering at PaiChai University. His current research interests include USN, Web Services.

So-Ra Yu She received the B.S. degrees in Computer Engineering from PaiChai University, in 2011. She is a master course students of department of computer of PaiChai University. Her current research interests include XML, Web Services, Semantic Web, USN.

Hoe-Kyung Jung He received the B.S., M.S. and Ph.D. degrees in computer engineering from Kwangwoon University, in 1985, 1987 and 1993, respectively. Since 1994, he has worked as a professor in Department of Computer Engineering at PaiChai University. His current research interests include XML, Web Services, semantic web, MPEG-21, Mobile RFID.