VOL 1 (2017) NO 4

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

A Sleep-Awake Scheme Based on CoAP for Energy-Efficiency in Internet of Things

Wenquan Jin [#], DoHyeun Kim [#]

[#] *Computer Engineering Department, Jeju National University, South Korea*
E-mail: wenquan.jin@jejunu.ac.kr, kimdh@jejunu.ac.kr

***Abstract*—Internet Engineering Task Force (IETF) have developed Constrained Application Protocol (CoAP) to enable communication between sensor or actuator nodes in constrained environments, such as small amount of memory, and low power. IETF CoAP and HTTP are used to monitor or control environments in Internet of Things (IoT) and Machine-to-Machine (M2M). In this paper, we present a sleep-awake scheme based on CoAP for energy efficiency in Internet of Things. This scheme supports to increase energy efficiency of IoT nodes using CoAP protocol. We have slightly modified the IoT middleware to improve CoAP protocol to conserve energy in the IoT nodes. Also, the IoT middleware includes some functionality of the CoRE Resource Directory (RD) and the Message Queue (MQ) broker with IoT nodes to synchronize sleepy status.**

***Keywords*— Sleeping scheme, RD, CoAP, IoT.**

I. INTRODUCTION

Internet of Things is large network of connected devices including various sensors and actuators to achieve desired objective of an individual or organization. It is composed of various components, communication technologies, data storage facilities and decision support systems.

IETF CoRE (Constrained RESTful Environment) WG (Working Group) had proposed CoAP for constrained networks [1]. CoAP provides a request/response interaction model between application endpoints using REST architectural style based on UDP. A size of CoAP message is small than HTTP and CoAP is more efficient than HTTP [2, 3]. IETF CoRE WG begin the standard track from 2010 and it had been published RFC 7252 in 2014. For CoAP, there are several implementations are available, and several IoT or M2M platforms use CoAP as communication protocol. IETF CoAP provides numerous potentially beneficial features for implementing wireless sensor networks with embed applications [4,5]. Therefore, CoAP will become a part of the new dominant design for IoT applications [6].

In this paper, we present a sleepy-awake scheme for energy efficiency of IoT nodes using CoAP protocol. Sleepy approaches are necessary for constrained devices. We have designed and implemented a sleepy mechanism for supporting a sleepy approach for IoT nodes. The proposed network includes IoT node, IoT middleware and Web Client. The proposed scheme uses HTTP and CoAP for communication between each element in the network. For this scheme we have used CoAP libraries to implement communication that are not only used for sleepy mechanism but also used for delivering data to the web client application.

Rest of this paper is structured as follows; Section 2 introduces proposed sleepy scheme and section 3 illustrates the usage of the scheme for IoT nodes. Section 4 illustrates experiment of the sleepy scheme using IoT nodes and section 5 describes the performance. Finally, we conclude our paper in section 5.

II. PROPOSED SLEEPY-AWAKE SCHEME

The CoAP is a protocol intended towards devices which are constrained in terms of memory, processing and power i.e. small low power sensors, switches and valves etc. The CoAP allows such devices to interactively communicate over the Internet. The CoAP is a specialized web transfer protocol for constrained devices. It is expected that in CoAP networks, there will be a certain portion of nodes that temporarily suspend CoAP protocol communication to conserve energy.

Sleepy feature is necessary for constrained environment. CoAP nodes work on constrained environment, therefore, direct discovery of nodes is not practical due to sleepy nodes. There have been several IETF drafts for sleepy nodes in constrained environments. Our implementation design is based on the sleepy mechanism presented in [7].

CoRE RD is an entity which hosts descriptions of CoAP nodes held on a server. The server should be based on some

power supply (not batteries), which can always allow lookups for retrieving registered information [8]. In CoAP based communication network, a CoAP client can search a CoAP node from RD and access it. Before the searching process, the CoAP node needs to register its information to the RD [9].

An extension of the CoRE RD called MQ broker which is published as a draft in IETF [9]. Functionalities for publish-subscribe communication are incorporated using the broker which enables to store and forward message from CoAP nodes. A CoAP node can send a data to the MQ broker and can switch into sleep mode. Then a CoAP client can request to MQ broker for getting the data which is sent by the CoAP node before sleep. The data is temporal, that can be updated or removed.

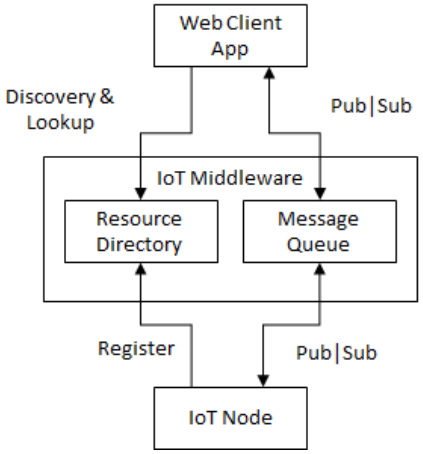

Fig. 1 Functionality of RD and MQ in IoT Middleware for sleep mode

The IoT middleware is used for in constrained networks for several reasons such as to improve performance and sleeping device scheduling. In CoAP networks, there will be a certain portion of devices that are "sleepy" and which may occasionally go into a sleep mode and temporarily suspend CoAP protocol communication. We present a mechanism for looking up sleepy nodes through interaction with IoT Middleware in the IoT. The functionality of RD and MQ are incorporated as part of the IoT middleware as shown in Figure 1.

IoT middleware includes RD functionality to manage information of IoT node. RD supports HTTP service to Web Client application for discovering and looking up information of IoT node which are registered by IoT node through CoAP service using RD. Functionality of MQ is used for performs store-and-forward messaging [10]. MQ enables IoT node publishes context data to the middleware to be subscribed by web client application. In the same way, web client application publishes a command to the IoT middleware and forward to IoT node when the node is available.

Models can explicitly explain processes and visualize a flow in a view [11,12]. Business process model and notation (BPMN) is a used for representing processes. BPMN involves flowchart which is better than UML diagrams for non-complex programs [13,14]. Figure 2 shows sleepy scenario using business process model. The process begins from web client application part. It sends message to RD to get node's data. RD check node's sleep state by node ID and RD returns sleep information to web client application. Web client application gets node's sleep delay data and synchronize the sleep state information with IoT middleware. When web client application knows node is awake, web client application can get the contextual data of node.

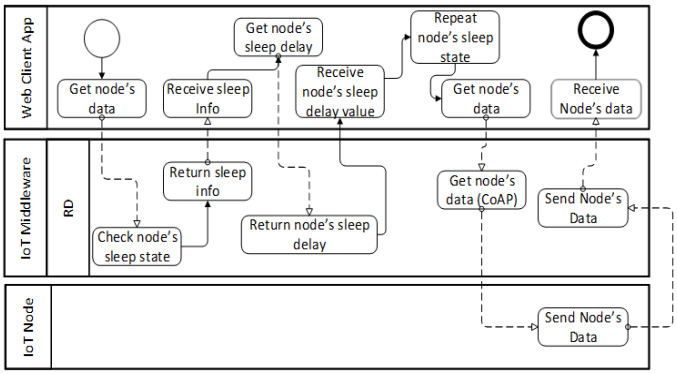

Fig. 2 Business process model for sleepy scenario

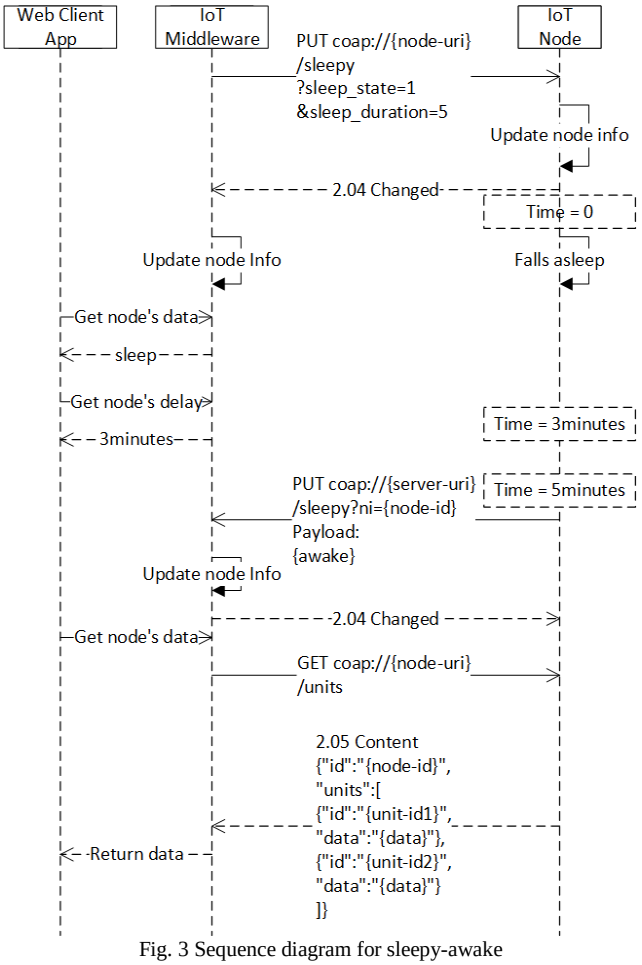

Fig. 3 Sequence diagram for sleepy-awake

III. Sleepy Mode for IoT Node

The sleepy information is updated in middleware and IoT node at the same time. Sleep state indicates whether the node is currently in sleep mode or not. Sleep duration indicates the maximum duration of time that the node stays in sleep mode. There is notify process which includes sleepy mode.

We design the mechanism using functionalities of RD and MQ for web client application to learn about sleepy state information of IoT node. Synchronous process is used by web client application to synchronize the sleep time of the IoT node.

Figure 3 shows IoT node receives a PUT request from middleware for change sleepy state. When web client application requests for get node's data, then IoT middleware sends a respond message with sleep information. Web client application gets node's delay information from IoT middleware to synchronize the sleep state information.

IV. Experiment and Results

Working of sleepy scheme is shown in Figure 4. Web client application sends sleep command request to IoT node through IoT middleware through HTTP protocol using node ID, sleep_state and sleep_duration as parameters. IoT middleware retrieve the IP address of IoT node via the node ID, and forward the sleep properties to IoT node through CoAP. IoT node invokes do_sleep() function for falling asleep.

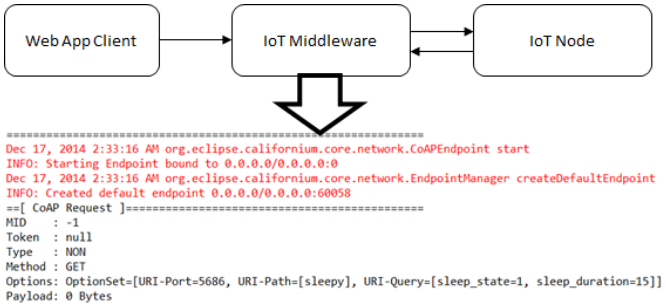

Fig. 4 Implementation of request command for sleep mode

We have used Java framework to implement IoT middleware and C library to implement IoT node [15][16]. IoT node is implemented in Linux C compile environment because of IoT node working in the constrained environment. IoT middleware is implemented in java compile environment. The IoT middleware is a web application which support web link to be accessed by user such as web client and HTTP client. In the service layer, our service provider supports SOAP web service which uses HTTP client to request the IoT middleware.

We implement the CoAP protocol in the proposed environment, and verify the interoperability. Figure 5 shows environment for testing CoAP client and CoAP server program. The CoAP client was written by Java, so it was executed in JRE environment. The CoAP server uses Linux C library and can be executed where the Linux that GCC compiler is exist.

The CoAP client slide uses Californium version which is based on the Java language, and the CoAP server uses Californium library which is based on the C language. The Californium library is ETSI IoT CoAP protocol and it uses "3-clause BSD" license [15]. The Libcoap protocol is implemented by using Linux C library, and it uses "BSD" license.

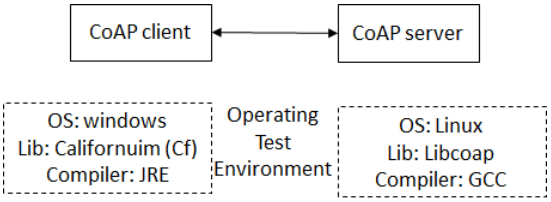

Fig. 5 Experiment environment for CoAP protocol

When the client sends a message to the server, the server sends the response message. The CoAP protocol is based on UDP protocol, and when the response message is not reached to the server in the pre-defined time, it sends message again. Repeat this again 5 times and if it is not received response message, this means that the communication is failed.

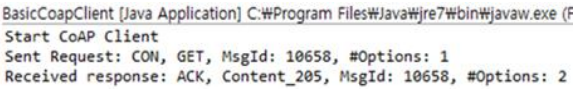

Fig. 6 Snapshot of CoAP Client execution

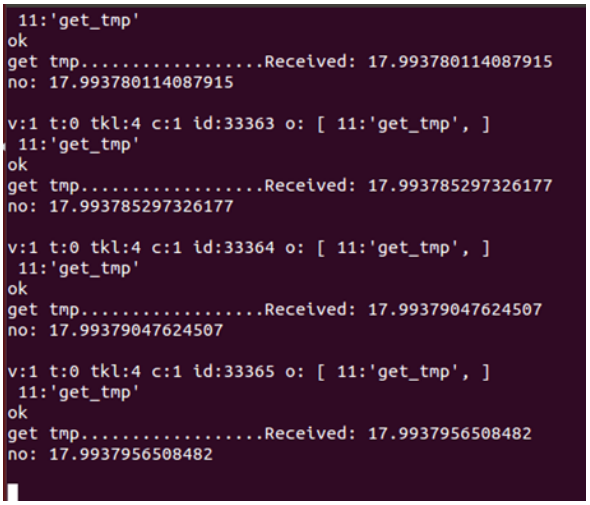

Fig. 7 Snapshot of CoAP server execution

We have developed testing applications for both client and server side. Figure 6 and Figure 7 presents sample snapshot of execution of client and server application respectively. When the CoAP client sends request message to the CoAP server via the CoAP protocol, the CoAP server processes it and sends reply message back to the CoAP client. These figures show exchange of request and response messages. The value "CON" is the T (Type) value of the CoAP protocol message format, and the "GET" is the method type of the CoAP protocol, and the IETF defines 4 method types. MsgId (i.e. 10658) means ID of message, and this value can be created automatically or developer can define it. The "#Options: 1" means the number of message attributes included in CoAP protocol message, and basically it is "MsgId" attribute.

Figure 7 shows experiment screenshot of node emulator in CoAP server. This program is executed with Console in Linux environment, and receiving message from a CoAP protocol node and collecting current temperature is shown on Console screen.

```
1 11.4541666666667,Wed Dec 17 02:32:55 2014
2 11.4538888888889
3 11.4536111111111
4 11.4533333333333
5 11.4530555555556
6 11.4527777777778,Wed Dec 17 02:33:00 2014
7 11.4525
8 11.4522222222222
9 11.4519444444444
10 11.4516666666667
11 11.4511111111111,Wed Dec 17 02:33:06 2014
12 11.4508333333333
13 11.4505555555556
14 11.4502777777778
15 11.45
16 11.4497222222222,Wed Dec 17 02:33:12 2014
17 11.4494444444444
18 11.4491666666667
19 11.4488888888889
20 11.4483333333333
21 11.4413888888889,Wed Dec 17 02:33:41 2014
22 11.4411111111111
23 11.4408333333333
24 11.4405555555556
25 11.4402777777778
```

Fig. 8 Context data record list in the sleep mode of IoT node

Figure 8 shows a context data record list when the IoT node falls asleep after receiving a sleep command from IoT middleware. IoT node received a CoAP message with a URI-query specifying sleep duration is 15 seconds as shown in Figure 7. In this period, the IoT node will stop unit's functions. The record list given in Figure 8 shows that in the period from "02:33:12" to "02:33:41", the IoT node had slept 15 seconds and waked up.

V. Performance Evaluation

Using our test applications for the proposed IoT system, we evaluated the performance of the message interaction for sleepy schemes. In this experiment, when an IoT node wakes up from sleep, then a web client application will get context data of IoT node.

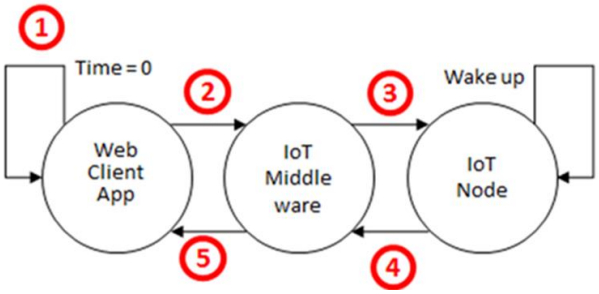

Fig. 9 Message transfer process for sleep-awake scheme

Figure 9 shows message transfer process for sleep scheme. In the synchronous scheme for sleepy IoT node, the state information of sleepy time is known by the web client application. When the time of sleep is up in the web client application, then it sends a request to the IoT middleware for acquiring context data of IoT node. Then the IoT middleware sends a CoAP request message to the IoT node to get context data. Finally, IoT middleware responds the context data to the web client application.

Figure 10 shows testing results of sleep scheme. This screenshot shows the results of context data of IoT node and time estimation for the process. The unit of time is millisecond. From the result, the timestamp of the process is 26 ms.

Fig. 10 Result of context data and time estimation for sleep scheme

VI. Conclusion

IETF CoRE WG presented CoAP for constrained environment, and there are several extensions for CoAP. In this study, we have presented a sleepy scheduling scheme to build on IoT middleware for energy conservation in IoT elements. We applied CoAP and CoAP extensions for interaction of these elements. IoT node is based on CoAP protocol, which works in constrained environment using low power and limited RAM and ROM. We designed and implemented the IoT node to fit the requirement. IoT middleware works with IoT node via CoAP in the Constrained RESTful Environment. We proposed an enhanced mechanism for management of sleepy nodes using CoRE RD and CoAP MQ. This mechanism is tested with IoT node and IoT middleware based on CoAP. In this paper, we have only presented our design and initial working implementation. In future, we will perform experiments to extensively analyse our design and resultant energy savings.

Acknowledgment

This research was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (2012-0-00265, R0101-17-0129 , Development of high performance IoT device and Open Platform with Intelligent Software) and this research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2017-2014-0-00743) supervised by the IITP(Institute for Information & communications Technology Promotion)" Any correspondence related to this paper should be addressed to DoHyeun Kim; kimdh@jejunu.ac.kr.

References

[1] Z. Shelby, B. Frank, D. Sturek, "Constrained Application Protocol (CoAP)", RFC 7252, June, 2014.

[2] C. Bormann, A. P. Castellani, Z. Shelby, CoAP: An Application Protocol for Billions of Tiny Internet Nodes, IEEE Internet Computing, Vol. 16, No. 2, pp 62-67, 2012.

[3] Tapio Levä, Oleksiy Mazhelisb, Henna Suomia, "Comparing the cost-efficiency of CoAP and HTTP in Web of Things applications", Decision Support Systems, Vol. 63, pp. 23–38, July 2014.

[4] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, V. Dobrota, "Evaluation of constrained application protocol for wireless sensor networks", Proc. of 18th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN), 2011.

[5] Z. Shelby, "Embedded web services", IEEE Wireless Communications, 17 (6), 2010.
[6] M.L. Tushman, J.P. Murmann, "Dominant designs, technology cycles, and organizational outcomes", R. Garud, A. Kumaraswamy, R.N. Langlois (Eds.), Managing in the Modular Age: Architectures, Networks, and Organizations, Blackwell Publishers, Oxford, 2003.
[7] A. Rahman, "Enhanced Sleepy Node Support for CoAP", Internet-Draft, draftrahman-core-sleepy-05, February 2014.
[8] Jin Wen-Quan, Kim Do-Hyeun, "Implementation and Experiment of CoAP Protocol Based on IoT for Verification of Interoperability", The Journal of the Institute of Webcasting, Internet and Telecommunication, Vol. 14, Iss. 4, pp. 7-12, 2014
[9] Z. Shelby, M. Koster, C. Bormann, P. van der Stok, "CoRE Resource Directory", draft-ietf-coreresource- directory-05, October 16, 2015
[10] M. Koster, A. Keranen, J. Jimenez, "Message Queueing in the Constrained Application Protocol (CoAP)", Internet-Draft, draft-koster-core-coapmq-00, July 2014.
[11] Kinam Park, Heuiseok Lim, "A computational model explaining language phenomena on Korean visual word recognition", Cognitive Systems Research, Volume 27, Pages 11-24, March 2014.
[12] Chang, J-K., Seungteak Ryoo, and Heuiseok Lim, "Real-time vehicle tracking mechanism with license plate recognition from road images." The Journal of Supercomputing, 2013.
[13] Danial Hooshyar, Rodina Binti Ahmad, Moslem Yousefi, Moein Fathi, Shi-Jinn Horng, Heuiseok Lim, "Applying an online game-based formative assessment in a flowchart-based intelligent tutoring system for improving problem-solving skills", Computers & Education, Volume 94, Pages 18-36, March 2016.
[14] Danial Hooshyar, Rodina Binti Ahmad, Moslem Yousefi, Moein Fathi, Shi-Jinn Horng, Heuiseok Lim, "Applying an online game-based formative assessment in a flowchart-based intelligent tutoring system for improving problem-solving skills", Computers & Education, 2016.
[15] Californium (Cf) CoAP framework in Java, http://people.inf.ethz.ch/mkovatsc/californium .php
[16] libcoap: C-Implementation of CoAP, http://libcoap.sourceforge.net/