Advances in Computer Science Research, volume 44
3rd International Conference on Wireless Communication and Sensor Network (WCSN 2016)

An Algorithm Managing Sleep-Nodes for CoAP Reliable Group Communications

Huan Guo, Geng-Yu Wei
School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
E-mail: 18811726089@163.com, weigengyu@bupt.com

***Abstract*-In constrained networks of IoT applications, Constrained nodes have sleep mechanism to reduce energy consumption, which causes an unreliable communication. CoAP has been defined as an application protocol for Constrained networks by IETF CoRE WG, with reliable unicast and unreliable group communication facility. Realizing the importance of reliable group communication capability for CoAP and IoT application to control vast amount of constrained nodes, this paper presents an algorithm managing sleep-nodes. The algorithm adopts the mechanism of a combination of synchronous and asynchronous communication and proxy cache to control sleep nodes in the group. Experimental results show that this scheme is of high efficiency and little data transmission latency.**

Keywords-component; CoAP; group communication; reliable; sleep nodes; proxy

I. INTRODUCTION

IoT(Internet of Things) is an extension and expansion of the Internet, which can achieve information exchange of M2M(machine to machine), thus facilitating goods identification, management and control[1]. The traditional HTTP protocol is heavy-weight applied in constrained environment imposed by the limit resources nodes, which have small memory space, limited computing power and energy. To cope with the restrictions, IETF(Internet Enginnering Task Force) Core(Constrained RESTful Environment) working group designed a RESTful protocol called CoAP(Constrained of Application Protocol)[2]. Constrained nodes have two features, one is the large number, another is the functional and regional relevance, which makes the group communication becomes an important way of communication[3](for example: control all lamps supported CoAP in one room by a switch).

CoAP is applied in constrained environment. In order to save energy, nodes will fall asleep period or aperiodic. The nodes will not receive and process any message when in the sleep cycle, which is called sleep nodes.

The current CoAP can only implements unreliable group communication based on IP multicasting. The unreliable CoAP group communication is not satisfied for lightweight M2M application requirements and architecture document proposed by OMA. In the application scenarios proposed by OMA, unreliable CoAP group communication will bring a great deal of risk.

Many factors can cause CoAP group communication unreliable, but the sleep node is one of the most important factor. Without considering retransmission, this paper proposes a novel approach based on proxy and cache mechanism. According to the nodes' sleep characteristics, the algorithm can control it synchronous and asynchronous. The proxy will control the nodes synchronously which have consistent sleep features, in contrast, it will control it asynchronous.

The rest of this paper are organized as following: Section 2 is about related works; section 3 gives the description of algorithm and its implementation; Section 4 presents the experimental results; and finally Section 5 is conclusion remark.

II. RELATED WORK

A. *Unicast-based mechanism*

Unicast-based mechanism is based on proxy to uncast CON message one by one to nodes, while CoAP group communication can only support NON(non-confirmation) type message[5]. The nodes must response a ACK(Acknowledgement) message when it receive a CON(Confirmation) request, which can make sure the communication reliable. A certain time later when the client not receive the response, it will retransmission the request util receive the response. Obviously, this method can make sure the reliability of group communication to a certain degree.

However, this method using unicast change the way of group communication totally. At the same time, such a large number of unicast messages will increase the network pressure, especially in constrained network environment, and the method not considering the nodes sleep feature at all.

B. *Message Wake-Up Mechanism*

This control mechanism requires nodes just enter a low-power state instead of completely asleep that the node has a low-power channel for receiving an external wake-up message sent by proxy. Upon receiving the wake-up message, these nodes will wake up into working state [6]. This method uses CoAP Resource Discovery(RD) server and proxy. Sleep nodes send a state change message to RD server, when RD server receive the message, it forward the message to proxy to notify it the nodes state changed. The proxy will inquire its database and select all the sleeping nodes and send a wake-up message to them before forwarding the message to the group.

This mechanism relieve the network pressure compared to the unicast-based mechanism, but it require nodes have a low-power channel open to listen wake-up message arrived when they in sleep state which increase the energy consumption.

III. IMPLEMENTATION OF SLEEP NODES MANAGING ALGORITHM

In IoT, Servers are often resource limited, and sender are often telephone or other terminal devices that have weak process capacity, so the mechanisms proposed by this paper are all based on GC proxy(Group Communication proxy). Fig.1 shows an overview of the involved components.

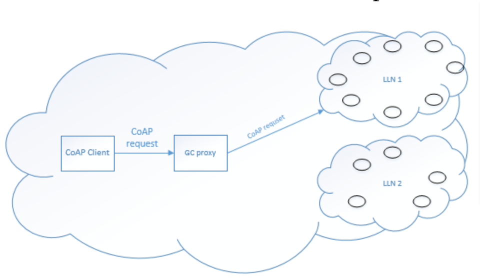

Figure 1. Overview system.

Node sleep behavior may be random or periodic, so according to the sleep characteristics of the group of nodes that can be controlled in different ways. This paper discusses three solutions for three situations. The three mechanisms will be shown in the following paper.

A. *Synchronous Contorl Mechanism*

When members of group nodes' sleep features are periodical, predictable, and all consistent, the synchronous control mechanism can be used.

1) *Design of synchronous control algorithm*

The key of the synchronous control algorithm is that group and GC proxy should keep clock synchronous. The flow diagram of synchronous control algorithm as follows.

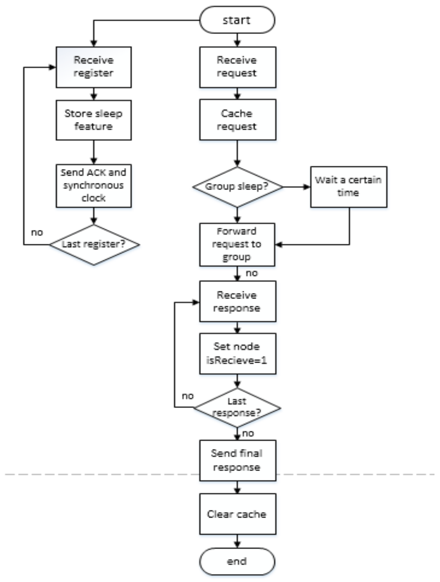

Figure 2. Flow diagram of synchronous control algorithm.

2) *Implementation of synchronous control mechanism*

The key of synchronous control mechanism is that proxy need to save group nodes' sleep feature and keep clock synchronized. The specific implementation is as follo The key of synchronous control mechanism is that proxy need to save group nodes' sleep feature and keep clock synchronized. The specific implementation is as follows.

- Register. Group nodes send register request to GC proxy, which include the group information registered and its sleep feature. The register request is a CON message that proxy must send a ACK response when receives nodes' register request. The ACK is a piggybacking that proxy can put its clock information and the next time nodes sleep in it. Meanwhile, GC proxy maintains a clock to record group nodes' current state by saving the nodes' sleep feature.
- Synchronization. After receive the ACK response from GC proxy, group nodes synchronize its clock and fall sleep in a time according to the response. And then fall sleep and wake up in cycle.
- Send of requests. Client unicasts a request to GC proxy. Upon receiving the request, GC proxy caches it and then inquires the group nodes' sleep state. If the nodes are waking, GC proxy forwards the request immediately, otherwise, it will wait a certain time that nodes are all waking and then forward the request.
- Response. All nodes in the group will unicast response to GC proxy when they received the request forwarded by GC proxy. GC proxy makes sure all the nodes send back the response, aggregates all responses to a final response and sends the final response to client.
- When the client receive the final response from GC proxy, it represents one communication is end, and all nodes in group have received the request send by client.

3) *Sleep feature of group nodes*

The feature of sleep nodes include all the sleep relevant information, mainly four parameters in the following:

- Sleep duration: time the nodes sleep in a cycle.
- Work duration: time the nodes work in a cycle.
- Next sleep/work time: the time before next sleep or work. If the node is in sleep state, the time is how long before next work, otherwise, it is how long before next sleep.
- Current state: node current state, work or sleep.

Nodes sleep feature can be sent by the register message, or be sent alone by a message. Proxy must save the nodes' sleep feature in order to forward request purposely.

4) *Limitations of synchronous control mechanism*

The advantages of synchronous control mechanism is simple to implement, convenient to control, low cost; however, the disadvantages is that it only applied to the environment that all nodes have the same sleep feature, and frequent clock synchronization when the group changes frequently. Frequent clock synchronization increases complexity in some degree, and clock synchronization can leads to clock drift problem.

Synchronous control mechanism is only applied in the environment that all the nodes have the consistent sleep feature, but there are many situations that the nodes have different sleep feature. Obviously, this mechanism is not suitable for this environment, so, this paper propose the asynchronous control mechanism which can be applied boarder and without considering the clock drift problem.

B. *Asynchronous contorl mechanism*

When nodes' sleep features are all random and unpredictable, the asynchronous control mechanism can be used.

1) *Design of synchronous control algorithm*

The key of asynchronous control algorithm is nodes send wake up message to GC proxy. And then GC proxy forwards request to the node. The flow diagram is shown blow.

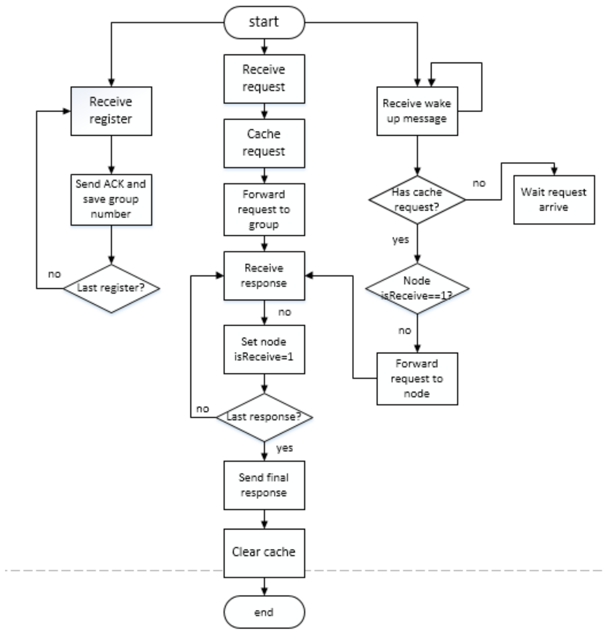

Figure 3. Flow diagram of asynchronous control algorithm.

2) *Implementation of asynchronous control mechanism*

In the asynchronous control mechanism, nodes send a wake up message to notify the GC proxy that its state is now in wake. In this situation, GC proxy does not need to know nodes' sleep feature, nodes and GC proxy don't need to keep clock synchronized.

a) *Group creation*

Group creation can be the nodes' active or passive behavior, the active behavior is the process that nodes apply for joining in group, while the passive behavior is the process that nodes is be set to be a member of a group. The active behavior is shown:

- Nodes send a register message to GC proxy to apply for joining the group.
- Once GC proxy receives the register message, it records the information of the sender node and the group it applied for, and the sender information include the node' IP address and group ID.
- To achieve reliable group communication, GC proxy need to assign a Boolean variable for every node in group which initial value is false to record the whether the node has received the request for every node in the group. When receives the response from a node, GC proxy will change the variable to true, presenting the node has received the request.

Group members are dynamic, nodes can join in or exist in any time. GC proxy updates the group information in time according to the nodes' behavior.

b) *Process of request and response*

Due to the presence of GC proxy, client send request to the GC proxy instead of sending to the group directly. The detail process as follows:

- Client unicasts the CoAP request to the proxy.
- Upon receiving the request, GC proxy parses the message to get the real group address and caches the request.
- GC proxy encapsulates the parsed message and then forwards the request to the group by multicasting.
- Group members send responses to the GC proxy when receive the request forward by proxy.
- GC proxy receives the response and then set the sender's corresponding variable to true.(The variable represent the node whether receives the request)
- For current awaking nodes, they have received the request in the (4), (5) steps. However, for current sleeping nodes, they can't receive the request in previous steps. GC proxy must wait the node wake, receive a wake up message from the node, and then forward the request to the node by unicasting. Repeating the forth and fifth processes until all nodes receive the request.
- GC proxy aggregates all responses received from nodes to a final response, and then send it to the client.
- The GC proxy receives the ACK from the client to make sure the final response has be sent to the client. After receiving the ACK message, the GC proxy clears its caches and set the nodes' corresponding Boolean variable to be false.

c) *Function of GC proxy*

GC proxy has many functions, but the main functions are showed as follows:

- Receiving nodes' register messages, achieving group creation, saving the information of group and its members.
- Forwarding the request message and caching it.
- Receiving sleep nodes' wake up messages, and forwarding the cached request to it.
- Receiving group nodes' responses.
- Making sure receiving all the nodes' responses, aggregating all the responses to a final response, and then sending it to the client.

3) *Limitaions of asynchronous control mechanism*

In asynchronous control mechanism, sleep nodes send wake up message when its state is changed from sleep to awake. The sleep duration is a very short time, so the number of wake up messages is very large which can increase the pressure of network, especially in the constrained network. Otherwise, when only a small part of nodes' sleep feature is unpredictable, this mechanism is not more optimal compared with synchronous control mechanism.

So in this situation, this paper proposes a combination of synchronous and asynchronous control mechanism.

C. *Combination Of Synchronous And Asynchronous Contorl Mechanism*

When a part of group nodes' sleep feature is the same, and other part of group nodes' sleep feature is random, combination of synchronous and asynchronous is a suitable approach to control the sleep nodes.

1) *Implementations of combination of synchronous and asychronous control mechanism*

- Nodes send a register message to GC proxy to apply for joining the group.
- GC proxy receives the nodes' register messages, parses them and get the nodes' sleep feature, and store it in its database. When GC proxy receives a new node register message, it compares the new node's sleep feature with the group previous nodes'. If there have the same sleep feature nodes, mark them and record its numbers. And chooses a group which has the largest number of nodes as the sleep group in the all group.
- When GC proxy forwards request, it inquires whether the group has sleep group in first, if exists, GC proxy judges whether delay a certain time to forward the request according the sleep group's state, if not exists, it forwards the request by asynchronous control mechanism totally.

The combination of synchronous and asynchronous control mechanism solves the disadvantages of using of synchronous control mechanism or asynchronous control mechanism alone, expands the applicable scope and improves the flexibility. GC proxy judges to use which mechanism according to the group nodes' sleep feature, which improve effective.

2) *Limitations of combination of synchronous and asychronous control mechanism*

Combination of synchronous and asynchronous control mechanism is based on synchronous control mechanism and asynchronous control mechanism, which is more comprehensive and effective. In this mechanism, GC proxy can determine to use which mechanism according to the group nodes' sleep feature automatically. When all nodes are in same sleep feature, it uses synchronous control mechanism alone. When all nodes are in different sleep feature, it users asynchronous control mechanism alone. Other situations, it uses the combination of synchronous and asynchronous control mechanism.

However, the percentage of nodes that have the same sleep feature is an important factor that can affect the effective of the algorithm. Meanwhile, the three mechanisms only applied in single-hop network. For multi-hop network, cache mechanism will increase network latency.

IV. EXPERIMENTAL RESULTS

After the introduction above, combination of synchronous and asynchronous control mechanism is the most effective.

In order to verify the effectiveness of the control algorithms, we used Java to simulate experiments based on the californium(cf) CoAP framework. The network size is maintained at seven nodes, and the behavior that nodes join in or exist the group may be active or passive. The experiments simulated third scene mentioned above, that is part of the nodes are in the same sleep feature, and make a comparison with the unicast control mechanism.

The experiments use seven computers to simulate the seven nodes in IoT, one computer serves as client, one as GC proxy, and the other five computers serve as server nodes. Why using only seven nodes is that the number of nodes in a smart house is limited and small. We used Wireshark to capture the communicate packets to compute the time from the request is sent to the final response is received. We conducted five experiments for every situation, and then compute the average of the five results, the shorter the time, the more efficient. In the experiments, nodes' sleep feature is set sleep duration 3s and work duration 5s, and not all the nodes are the same sleep feature.

The experiment data as follows: (unit is s)

TABLE I. EXPERIMENTAL RESULTS

	1	2	3	4	5	aver age
Combination of synchronous and asynchronous mechnism	0.01 075	0.01 283	1.01 246	0.00 973	2.10 912	0.63 233
Unicast-based control mechanism	0.02 324	0.01 854	1.23 783	0.01 370	2.20 667	0.69 996

The average of combination of synchronous and asynchronous control mechanism is 0.63233s. The average of unicast-based control mechanism is 0.69996s. And the number of packages is another indictor to compare the two mechanisms. In the experiment, the number of packages in combination of synchronous and asynchronous mechanism is 8 in average, and the number of packages in unicast-based control mechanism is 12 in average. The data shows the effective of combination of synchronous and asynchronous is better than the unicast-based control mechanism in average. We can see the 3rd and 5th data is bigger than the other data in table, which because of sleepy node is in sleep state when GC proxy forwards the request. What's more, for the number of message, combination of synchronous and asynchronous is less than unicast-based control mechanism.

Obviously, the result of combination of synchronous and asynchronous control mechanism is optimal than the unicast-based control mechanism.

V. CONCLUSION

This paper proposes control mechanism of sleep node aiming at solving the unreliable CoAP group communication. The mechanism improves effective of group communication in certain degree, and alleviates network congestion caused by the large amount of messages. The simulation experiment shows: the algorithms proposed by this paper can improve the group communication effective, save network energy consumption and reduce the amount of data transferred.

REFERENCES

[1] L. Tan, N. Wang. Future Internet: The Internet of Things[C]. 2013 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), 2015, 5:pp.376-380.

[2] Z. Shelby, K. Hartke, C. Bormann. The Constrained Application Protocol (CoAP)[S]. IETF: RFC 7252, June 2014, pp.1-92.

[3] A. Rahman, E. Dijk. Group Communication for the Constrained Application Protocol[S]. IETF: RFC 7390, 2014, pp.1-37.

[4] "Constrained RESTful environments (core)."[online], Available: http://datatracker.ietf.org/wg/core/[Accessed:28-Dec-2012].

[5] I. Ishaq, J. Hoebeke, F. Van den Abeele, et al. Group communication in constrained environments using CoAP-based entities[C]. 2013 IEEE International Conference on Distributed Computing in Sensor Systems. IEEE, 2013, pp.345-350.

[6] WEI geng-yu, GUO yu-meng. Design and implementation of reliable group communication base on CoAP[J]. Internet of things technologies, 2014, 4(12): pp.24-27.

[7] Z. Shelby. Constrained RESTful environments(CORE) Link Format[S]. IETF: RFC 6690, 2012, pp.3-19.

[8] J. Hui, R. Kelsey. Multicat protocol for low-power and lossy networks(MPL)[S]. IETF: RFC 7731, 2016, pp.3-25.

[9] A. Rhaman,JC. Zuniga. Sleeping and Multicast Consideration for CoAP draft-rahman-core-sleeping-00.txt[S]. IETF, 2010.

[10] M. Nottingham. Web Linking[S]. IETF: RFC 5988, 2010, pp.3-9.

[11] M. Nottingham, E. Hammer-Lahav. Defining Well-Known Uniform Resource Identifiers(URIs)[S]. IETF: RFC 5785, 2010, pp.1-6.

[12] T. Zotti, E. Dijk. Sleepy CoAP Nodes draft-zotti-core-sleepy-nodes-04[S]. IETF, 2015, pp.2-23.

[13] Chulho Park. IP multicasting for mobile hosts in wireless networks[C]. 2013 International Conference on Information Science and Applications(ICISA), 2013, pp.1-2.

[14] N. Kaur, SK. Sood. An energy-efficient architecture for the Internet of Things(IoT)[J]. IEEE Systems Journal, October 2015, pp.1-10.

[15] CHENG ming-yue, MA ya-jie. A node sleeping scheduling algorithm for WSNs based on spatial correlation[J]. Transducer and Microsystem Technologies, 2015, 34(11), pp.143-146.

[16] C. Lerche, N. Laum, F. Golaowski. Connecting the web with the web of things: lessons learned from implementing a CoAP-HTTP proxy[C]. 2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems(MASS 2012), pp.1-8.

[17] HE zhao-sun, SHI gao-tao, LIAO ming-hong. Research and implementation of random-sleep node dissemination algorithm for sensor network[J]. Computer engineering, 2007, 33(8): pp.115-117.