Sensors and Materials, Vol. 29, No.7 (2017) 855–858
MYU Tokyo

S & M 1374

Letter

Stepwise Adjustment of Constrained Application Protocol Observing Period for Internet of Things Applications Using Wireless Sensor Networks

Jung-Hyok Kwon, Sungmo Kim, and Eui-Jik Kim*

Department of Convergence Software, Hallym University,
1 Hallymdaehak-gil, Chuncheon-si, Gangwon-do 24252, South Korea

(Received March 1, 2016; accepted February 27, 2017)

Keywords: buffer overflow, constrained application protocol, Internet of Things, observing period, wireless sensor networks

In this paper, we present a stepwise adjustment of the constrained application protocol observing period (SACOP) for Internet of Things (IoT) applications using wireless sensor networks (WSNs), which dynamically adjust observing periods depending on the Rx queue status of the constrained application protocol (CoAP) client. The operation of SACOP consists of the following two consecutive phases: *overflow alert* and *observing period adjustment.* In the former phase, a client sends a buffer overflow alert when the queue of the client reaches the predefined queue threshold. In the latter phase, the servers change their own observing period level depending on whether or not they receive a buffer overflow alert message. Therefore, SACOP can significantly reduce the number of dropped messages caused by buffer overflow. A simulation showed that SACOP achieved a higher network performance than the legacy approach with regard to the number of dropped messages.

1. Introduction

Recently, the Internet of Things (IoT) has received considerable attention from both academia and industry. In the IoT world, remote monitoring and control using wireless sensor networks (WSNs) is a representative application, and resource-constrained devices are generally used due to concerns about cost.(1) To support web service for constrained devices, the Internet Engineering Task Force (IETF) Constrained Restful Environments (CoRE) working group has developed the constrained application protocol (CoAP). CoAP is a specialized web transfer protocol designed to meet the requirements of simplicity and low overhead in resource-constrained environments.(2)

In general, a WSN is composed of a massive number of sensor devices, communications among which are likely to cause network congestion. CoAP supports a basic form of congestion control through the exponential backoff mechanism, and CoAP Simple Congestion Control/Advanced (Cocoa), which is the advanced version of CoAP, improves the congestion control functionality of CoAP by using real-time retransmission timeout (RTO) measurements.(3)

CoAP and Cocoa commonly try to mitigate congestion by reducing the number of retransmissions of servers (i.e., sensor devices). However, most IoT applications necessarily use

*Corresponding author: e-mail: ejkim32@hallym.ac.kr

http://dx.doi.org/10.18494/SAM.2017.1562

ISSN 0914-4935 © MYU K.K.

the CoAP observe option, by which numerous sensor devices (i.e., servers) periodically send their observing messages to the gateway (i.e., client). This observe option does not require any acknowledgement or retransmission. Therefore, such an approach cannot be a suitable solution for the congestion problem in IoT applications.

In this paper, we propose a stepwise adjustment of the constrained application protocol observing period (SACOP) for remote monitoring and control applications using WSNs. SACOP adjusts the observing period of CoAP servers in a step-by-step manner according to the buffer status of the client.(4) The operation of SACOP consists of the following two consecutive phases: *overflow alert* and *observing period adjustment*. In the former phase, the client alerts the servers that its buffer overflow is imminent by broadcasting a buffer overflow alert (BoA) message, when its queue status reaches a predefined threshold. In the latter phase, servers that have received a BoA message adjust their observing period level, $op(i)$ upward, or, as we shall see, the servers can adjust $op(i)$ downward in this phase by inference in response to a dearth of BoA messages from the client. Therefore, SACOP can significantly reduce dropped messages caused by buffer overflow at the client side. A simulation showed that SACOP yields better performance than the legacy CoAP with regard to the number of dropped messages.

The rest of this paper is organized as follows. Section 2 presents the design of SACOP. Results of simulations and conclusions are given in Sects. 3 and 4, respectively.

2. Design of SACOP

SACOP prevents buffer overflow at the CoAP client side by adjusting the observing period of the CoAP servers. SACOP includes two operational phases: *overflow alert* and *observing period adjustment.*

In the first phase, the client configures the threshold value of its own buffer queue, which is a reference value for determining whether a buffer overflow event is imminent. Then, the client waits for messages transmitted from the associated servers. When the queue status of the client reaches the predefined queue threshold, it broadcasts a BoA message to alert for the upcoming buffer overflow event.

In the observing period adjustment phase, SACOP uses two types of adjustment procedures according to the entity that triggers the adjustment procedure: client-initiated adjustment and server-initiated adjustment. The former is used for reducing the observing period of servers, while the latter extends it. The observing period is adjusted in a step-by-step manner. For this, SACOP uses "observing period level", $op(i)$, $i \in [0, m]$, where m is a preconfigured number of observing periods. If the $op(i)$ changes, the observing period is calculated afresh as follows.

$$T_{observ}(op(i)) = T_{observ,init} \times op(i), \tag{1}$$

where $T_{observ,init}$ is the initial observing period when $i = 0$ and is preconfigured by the system. Figure 1 shows the overall operation of the observing period adjustment phase, for which the servers use two variables: *Send_msg_cnt* and *Max_msg_cnt*, which respectively indicate the number of transmitted messages and the maximum number that *Send_msg_cnt* can attain. The server increases its *Send_msg_cnt* by 1, whenever transmitting the observing message to the client. If *Send_msg_cnt* reaches *Max_msg_cnt*, it is initialized to 0.

When the server receives the BoA message from the client, the client-initiated adjustment

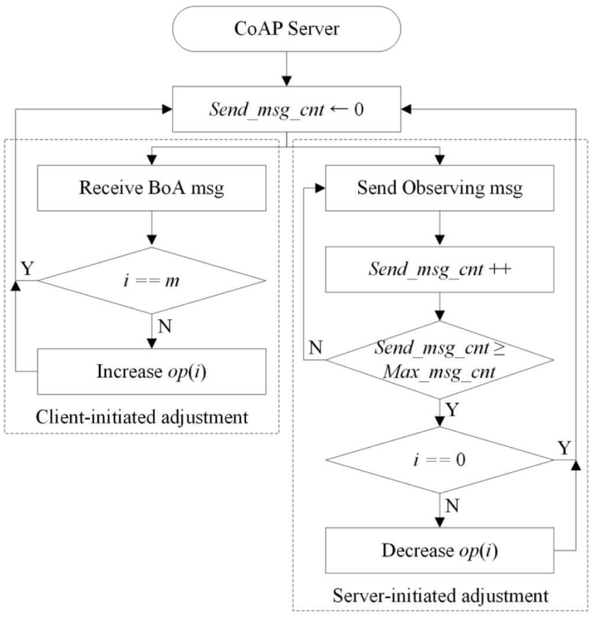

Fig. 1. Observing period adjustment.

procedure is triggered and it checks *i*. The server increases *op*(*i*) if *i* is not *m*. Otherwise it maintains the current observing period. Then, the server initializes *Send_msg_cnt* to 0.

The server-initiated adjustment uses the concept of "aging" with which the CoAP server reduces the observing period by itself. As long as the server does not receive a BoA message, it repeatedly transmits messages with the same observing period and increases its *Send_msg_cnt*. If *Send_msg_cnt* reaches the *Max_msg_cnt*, the server checks *i*. If *i* is not 0, it decreases its *op*(*i*) by one level.

3. Performance Evaluation

In this section, we describe the evaluation of the performance of SACOP using MATLAB. To verify the superiority of SACOP, we compare SACOP with the legacy CoAP. In this simulation, we assume that one client and multiple observing servers communicate with each other using CoAP within a 10×10 m^2 square room. We specify the queue size to 1 MB and the clock rate to 1 MHz. The threshold value of the buffer queue is specified to 0.9 MB, which is 90% of the queue size, and *Max_msg_cnt* is set to 5. The detailed parameters are listed in Table 1.

Figure 2 shows the variation of the observing period. In the simulation, we set the number of devices to 15 and the simulation time to 100 s. The legacy CoAP maintains the same observing period throughout the entire simulation time. However, with SACOP, the observing period begins to change at 15 s because SACOP dynamically adjusts the observing period via the *op*(*i*). Specifically, the average variation time of the observing period for the SACOP test is 5.2 ms.

Figure 3 shows the number of dropped messages for varying numbers of CoAP servers. When the number of devices is less than 10, no messages are dropped with either CoAP or SACOP. However, when there are more than 12 CoAP servers, the legacy CoAP begins to drop messages. Meanwhile, SACOP does not drop any messages throughout the entire simulation time. The average number of dropped messages of the legacy CoAP is 1587.1 while for the SACOP it is 0.

Table 1
Simulation parameter.

Parameter	Value	Parameter	Value
Payload	125 B	*op*(0)	1.0
Queue size	1 MB	*op*(1)	1.2
Clock rate	1 MHz	*op*(2)	1.4
Simulation time	100 s	*op*(3)	1.6
Threshold value	0.9 MB	*op*(4)	1.8
$T_{observ,init}$	10 ms	*op*(5)	2.0
Max_msg_cnt	5	Number of servers	0–20

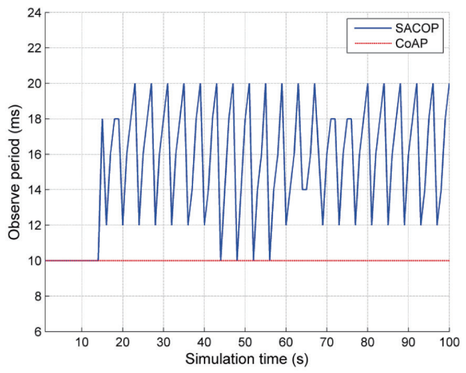

Fig. 2. (Color online) Variation of the observing period.

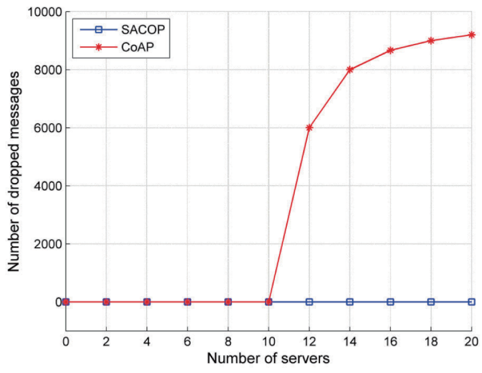

Fig. 3. (Color online) Variation of the number of dropped messages.

4. Conclusions

In this paper, we propose a SACOP that prevents buffer overflow at the CoAP client side by adjusting the observing periods of CoAP servers in a stepwise manner. SACOP consists of *overflow alert* and *observing period adjustment* phases, in which a client alerts for buffer overflow by broadcasting a BoA message and the servers adjust their observing period by increasing or decreasing the *op*(*i*). The simulation showed that SACOP achieves better performance in terms of the number of dropped messages than legacy CoAP. Therefore, SACOP is expected to be an efficient solution for the congestion problem of IoT applications using WSNs.

Acknowledgments

This research was supported by Hallym University Research Fund, 2017 (H20170051).

References

1 A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash: IEEE Commun. Surveys Tuts. **17** (2015) 2347.
2 Z. Shelby, K. Hartke, and C. Bormann: The Constrained Application Protocol (CoAP) RFC 7252 (2014).
3 C. Bormann, A. Betzler, C. Gomez, and I. Demirkol: CoAP Simple Congestion Control/Advanced draft-bormann-core-cocoa-02 (2014).
4 K. Hartke: Observing Resources in the Constrained Application Protocol (CoAP) RFC 7641 (2015).