International Journal of Engineering & Technology, 7 (4.38) (2018) 885-887

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Adaptive Resource Observation for Congestion Alleviation using Constrained Application Protocol

Jung-Hyok Kwon1, Sol-Bee Lee1, Jaehoon Park2, Kyu-Sung Hwang3, Yongseok Lim4, and Eui-Jik Kim1,∗

1Department of Convergence Software, Hallym University, 1 Hallymdaehak-gil, Chuncheon-si, Gangwon-do 24252, South Korea
2Department of Electronic Engineering, Hallym University, 1 Hallymdaehak-gil, Chuncheon-si, Gangwon-do 24252, South Korea
3School of Electronic Engineering, Kyungil University, 50 Gamasil-gil, Hayang-eup, Gyeongbuk 38428, South Korea
4Network Convergence Research Center, Korea Electronics Technology Institute, 11 World Cup buk-ro 54-gil, Mapo-gu, Seoul 03924, South Korea

Abstract

This paper presents an adaptive resource observation (ARO) for congestion alleviation using constrained application protocol (CoAP), which prevents buffer overflow of the client by adjusting observing period of the associated servers. The operation of ARO consists of two main phases; 1) buffer overflow estimation, 2) observing period adaptation. In the former, the client estimates whether buffer overflow will occur by comparing its service rate with packet arrival rate, then it determines the new observing period that can prevent buffer overflow of the client. The latter is used to adjust the observing period of servers considering the predefined the minimum and maximum queue threshold. ARO can significantly reduce the number of dropped packets caused by buffer overflow. The simulation results show that ARO achieves a higher network performance than legacy CoAP.

Keywords: *Adaptive Resource Observation, Buffer Overflow Estimation, Congestion Alleviation, Constrained, Application Protocol, Observing Period Adaptation.*

1. Introduction

Recently, the demand for Internet of things (IoT) monitoring services such as health monitoring, environmental monitoring, and vehicle tracking applications has been sharply increased.1, 2 In general, a number of resource-constrained devices and low power lossy networks (LLNs) are used in IoT monitoring services.3 Therefore, the use of lightweight web-based transfer protocol is strongly recommended for reliable IoT monitoring services. For this, the constrained application protocol (CoAP) is standardized by the Constrained Restful Environments (CoRE) working group in Internet Engineering Task Force (IETF).4 The CoAP is a specialized web transfer protocol designed based on the representational state transfer (REST) architecture. It uses 4-byte size of fixed header over the user datagram protocol (UDP) and easily interfaces with hypertext transfer protocol (HTTP). Thus, the CoAP has been investigated by many research institutes to provide reliable communication in the constrained environments.
As we mentioned above, a number of resource constrained

devices are usually employed in IoT monitoring services. Therefore, the network congestion may occur frequently, which may result in a large number of dropped packets. To address this problem, CoAP provides the basic congestion control mechanism
that can alleviate the network congestion by reducing the number of retransmissions. However, since it uses the fixed length of timeout and backoff factor regardless of the network condition, it cannot entirely solve the existing congestion problem. To improve the basic congestion control mechanism, CoAP simple congestion control/advanced (CoCoA) is proposed.5 The CoCoA uses the round trip time (RTT) estimation based on the retransmission timeout (RTO) and the variable backoff factor (VBF), thus it considerably reduces the unnecessary retransmissions compared to the CoAP. However, considering that most IoT monitoring services use the CoAP observation mechanism, where the sensor device (i.e., server) periodically transmits the packets including change of surrounding environment, the existing congestion control mechanisms are not suitable solution for congestion problem in IoT monitoring services.6 This is because the CoAP observation mechanism does not use any acknowledgement and retransmission. Therefore, to support reliable IoT monitoring services, a new approach is needed to solve the congestion problem in environment using CoAP observation mechanism.
In this paper, an adaptive resource observation (ARO) for congestion alleviation using CoAP. The ARO aims to prevent buffer overflow of the client by adjusting observing period of the associated servers. The operation of ARO consists of the buffer overflow estimation and the observing period adaptation phases. In the buffer overflow estimation, the client estimates whether buffer overflow will occur by comparing its service rate with packet arrival rate, then it determines the new observing period that can prevent buffer overflow of the client. The observing period adaptation is used to adjust the observing period of servers considering the predefined the minimum and maximum queue threshold. The ARO can significantly reduce the number of dropped packets caused by buffer overflow. To evaluate the performance of the ARO, we conduct experimental simulation under various scenarios. The results show that ARO achieves a higher network performance than legacy CoAP.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The rest of this paper organized as follows. In Section 2, the system model is described. In Section 3, the design of ARO is presented in detail. The performance evaluation is described in Section 4. Finally, we conclude this paper in Section 5.

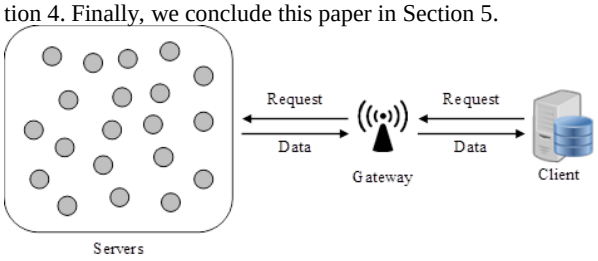


Fig. 1: System architecture.

2. System Architecture

Figure 1 shows the system architecture for IoT monitoring service that consists of a client, a gateway and the multiple servers. The client is the user device (i.e., laptop, user mobile, tablet, etc.) that manages requests to the servers to obtain the sensing data through gateway. The servers are the sensor devices that detects the changes their surrounding environment, and transmit the sensing data according to the client's request via gateway. The gateway is responsible for forwarding messages between the servers and the client. All types of devices communicate each other using the CoAP via wireless links. In the architecture, we assume that the client and servers have the limited physical resources such as the low-power, the limited computational capability, and the limited storage.

To receive the sensing data from the servers periodically, the CoAP observation technique is used by the client and servers. In order to establish the observation relationship between them, the client transmits the observation request message to the targeted server for registration. When receiving the message, the server periodically transmits the sensing data to the client. We assume that the transmission interval (i.e., observing period) of the servers is predefined by the system depending on the application, and we further assume that all servers use the same observing period. The legacy CoAP does not take into account the queue status of the client. Therefore, when a large number of data is transmitted to the client simultaneously, the buffer overflow can the client due to lack of queue. The buffer overflow increases the number of dropped packets. To address this problem, we propose the ARO that prevents buffer overflow of the client by adjusting observing period of the associated servers. Since the ARO adaptively adjust the observing period of server considering the queue status of the client, it significantly reduces the number of dropped packets.

3. Design of Aro

The ARO aims to prevent buffer overflow of the client by adjusting observing period of the associated servers. In ARO, we assume that the client periodically checks the number of associated servers and the observing period of servers. We further assume that all servers use the same observing period. The ARO is conducted by the client, and the operation of ARO consists of two phases: 1) buffer overflow estimation and 2) observing period adaptation.

In the former, the client compares the service rate with the packet arrival rate for buffer overflow estimation. The service rate means the processing capability of the client per second. The value of service rate can be determined according to the hardware such as the central processing unit (CPU). The packet arrival rate means the amount of packets received by the client per second. The buffer overflow may occur when the service rate is lower than the packet arrival rate. Therefore, the client periodically compares the service rate with the packet arrival rate to estimate the buffer overflow. For this, the client firstly calculates the packet arrival rate that can be given as equation (1).

(1)

where is the number of associated servers, is the expected number of packets received by the client per second, and is the packet size. In here, the is easily calculated by the client since the server sends the packet to the client using the fixed transmission interval called observing period. If the packet arrival rate is higher than service rate, the client calculates new observing period ( ) as given in equation (2).

$$NewOP = OldOP \times \frac{\text{Packet arrival rate}}{\text{Service rate}} \tag{2}$$

where is the old observing period that is currently used by the server.

In the latter, the client notifies the to the servers when the buffer queue of client reaches the predefined maximum queue threshold. On the other hand, it notifies the to the servers, when the buffer queue of client reaches the predefined minimum queue threshold. Upon receiving the or , the servers adjust their observing period referring to the notification of the client.

4. Performance Evaluation

To evaluate the performance of ARO, we conduct the experimental simulation using MATLAB. The simulation results are compared to that of the legacy CoAP to verify the effectiveness of the ARO. In the simulation, we check the variation of the number of dropped packets and the queue status when the simulation time and the number of servers change. For this, we set the old observing period is 10 ms and service rate of client is set to 1 Mbps. In addition, the queue size of the client is set to 1 Mbytes, and the minimum and the maximum queue thresholds are set to 0.1 Mbytes and 0.9 Mbytes, respectively. The detailed simulation parameters are listed in table 1.

Figures 2 and 3 show the number of dropped packets when the simulation time and the number of servers change. We set the number of servers to 15 for figure 2, and the simulation time to 50 s for figure 3. In both figures, the ARO has none of the dropped packets for the variation of simulation time and the number of servers. On the other hand, in the case of legacy CoAP, the number of dropped packets increases as the simulation time and the number of servers increases. This is because the ARO prevents the buffer overflow of the client by adaptively adjusting CoAP observing period according to the result of buffer overflow estimation.

Figures 2 and 3 show the number of dropped packets when the simulation time and the number of servers change. We set the number of servers to 15 for figure 2, and the simulation time to 50 s for figure 3. In both figures, the ARO has none of the dropped packets for the variation of simulation time and the number of servers. On the other hand, in the case of legacy CoAP, the number of dropped packets increases as the simulation time and the number of servers increases. This is because the ARO prevents the buffer overflow of the client by adaptively adjusting CoAP observing period according to the result of buffer overflow estimation.

Table I. Simulation parameters.

Parameter	Value	Parameter	Value
Old observing period	10 ms	Queue size	1 Mbytes
Number of servers	0–20	Service rate of client	1 Mbps
Simulation time	0–100 s	Minimum queue threshold	0.1 Mbytes
Packet size	125 Bytes	Maximum	0.9 Mbytes

		queue threshold	

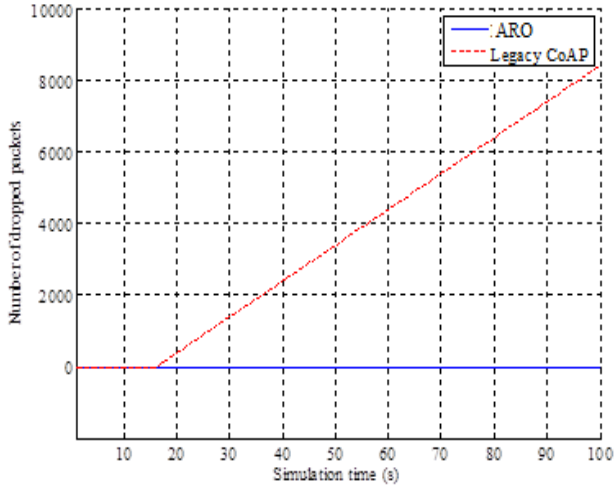


Fig. 2. Number of dropped packets for simulation time change.

Figures 4 and 5 show the variation of the queue status of the client. For this, we set the simulation configurations same as the number of dropped message case (i.e., figures 2 and 3), and increases until the simulation time reaches 15 s. This is because packet arrival rate is higher than service rate of the client. However, the maximum queue status of ARO is lower than the legacy CoAP. The reason is that the client transmits the

to the servers to increase the observing period of them when the queue status reaches the maximum queue threshold. Similarly, in figure 5, the queue status of both cases increases when the number of servers is more than 10 servers are deployed. However, the maximum queue status of ARO is lower than the legacy CoAP for the same reason as figure 4. In detail, the average queue status of the legacy CoAP is 0.5 Mbytes and ARO is 0.45 Mbytes.

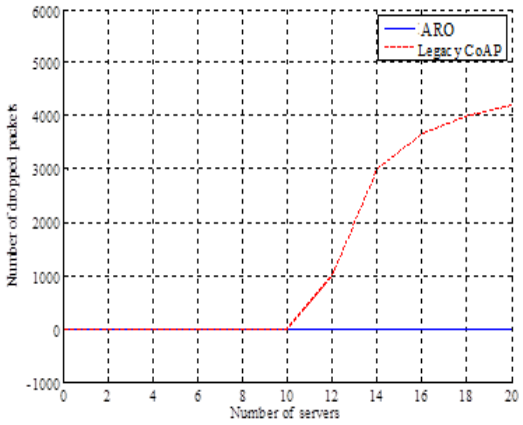


Fig. 3. Number of dropped packets for various number of servers.

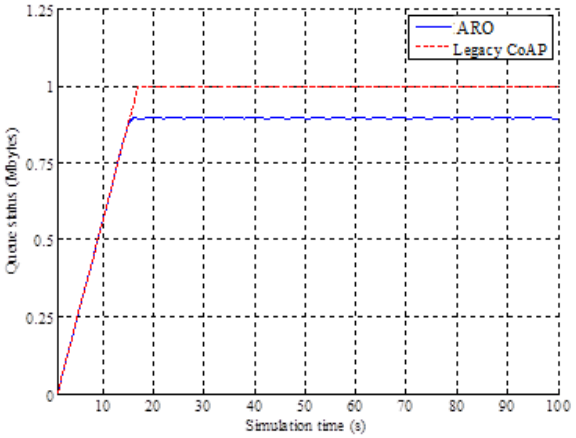


Fig. 4. Number of dropped packets for simulation time change

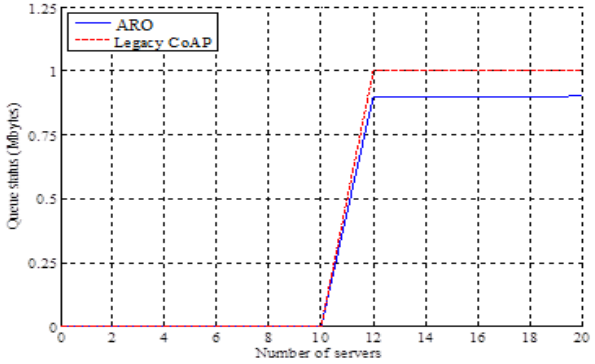


Fig. 5. Queue status for various number of servers

5. Conclusion

This paper presents the ARO designed to prevent buffer overflow of the client by adjusting observing period of the associated servers. For this, the operation of ARO consists of buffer overflow estimation and observing period adaptation phases. The buffer overflow estimation is used to estimate the occurrence of the buffer overflow and determine the new observing period. The observing period adaptation is used to adjust their observing period. To verify the effectiveness of the ARO, we conduct experimental simulation using MATLAB. The results show that ARO significantly reduces the number of dropped packets compared to the legacy CoAP. In addition, ARO can maximize utilization of the queue of the client while preventing a buffer overflow.

Acknowledgments: This work was supported by the Hallym Leading Research Group Support Program of 2017 (HRF-LGR-2017-0003).

References

1. S. Balampanis, S. Sotiriadis, and E. G. Petrakis, IEEE Cloud Comput. 3 6 (2016).
2. N. C. Luong, D. T. Hoang, P. Wang, D. Niyato, D. I. Kim, and Z. Han, IEEE Commun. Surv. Tut. 18 4 (2016).
3. R. V. chander, S. Mukherjee, and S. Elias, Comput. Electr. Eng. (2017), in press.
4. Z. Shelby, K. Hartke, and C. Bormann, The constrained application protocol (CoAP), RFC 7252 (2014).
5. C. Bormann, A. Betzler, C. Gomez, and I. Demirkol, CoAP simple congestion control/advanced draft-bormann-core-cocoa-04 (2017).
6. K. Hartke Observing resources in the constrained application protocol, RFC 7641 (2015).