Congestion Control in CoAP Protocol: A Case study of Observing Resource with Block-Wise Transfer

Thavrak Chan[1], Chanwit Suwannapong[2] and Chatchai Khunboa[1,*]

[1,*] Department of Computer Engineering, Faculty of Engineering, Khon Kaen University, Mittraphap Road, Muang, Khon Kaen 40002, Thailand.
[2] Division of Computer Engineering, Faculty of Engineering, Nakhon Phanom University, Chayangkol Road, Tambol Khamtao, A.Muang, Nakhon Phanom 48000, Thailand

thavrak.c@kkumail.com, schanwit@npu.ac.th and chatchai@kku.ac.th*

Abstract. *The Constraint Application Protocol (CoAP) is a restricted protocol used for communication in the Internet of Things (IoT). It allows limited resource devices to connect to the Internet, exchange request/response messages, and block-wise transfer for large data transfers. CoAP also includes an observed mode, which allows a client to monitor resources on servers and receive notification messages via unicast when the state of the resource is changed. However, the default congestion control algorithm used by CoAP, called Binary Exponential Back-Off (BEB), is insufficient for group communication resource observation with block-wise transfer and can lead to significant congestion, resulting in a buffer overflow, data loss, and connection drop. To address this problem, we conducted a study to evaluate the effectiveness of two alternative algorithms for congestion control in CoAP: Fibonacci Pre-Increase Back-off (FPB) and Half Binary Exponential Backoff (HBEB) algorithms. We tested these algorithms using a Cooja simulation and compared their performance to the default BEB algorithm. Our results showed that HBEB outperformed both BEB and FPB in terms of throughput and packet loss ratio (PLR), where BEB provides the best in term of end-to-end delay.*

Received by 20 January 2023
Revised by 26 April 2023
Accepted by 28 April 2023

Keywords:

block-wise transfer, CoAP, congestion control, group communication, IoT, observe resource

1. Introduction

The Internet of Things (IoT) refers to the connection of various networking systems and everyday devices to the Internet. The Constrained Application Protocol (CoAP), developed by the Internet Engineering Task Force (IETF), is a specific web transmission protocol designed for using in devices with limited resources in IoT. CoAP uses a User Datagram Protocol (UDP) with IEEE 802.15.4, rather than the resource-intensive Transmission Control Protocol (TCP) to make operation easier and reduce system resource needs, as shown in Fig. 1. Moreover, CoAP is also used in machine-to-machine (M2M) communications [1], [2].

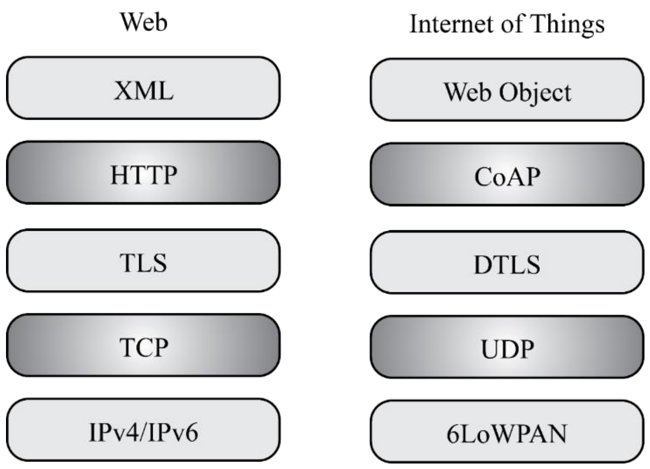


Fig. 1 HTTP and CoAP protocol stack

The design of CoAP is similar to HTTP in a request/response model which allows clients to send requests to servers and receive responses. Moreover, CoAP also includes additional features called Observing Resources (OBS), which allows for group communication and notification of resource changes. This provides benefits for smart cities and healthcare applications to monitor data from sensors in real-time [3]–[7].

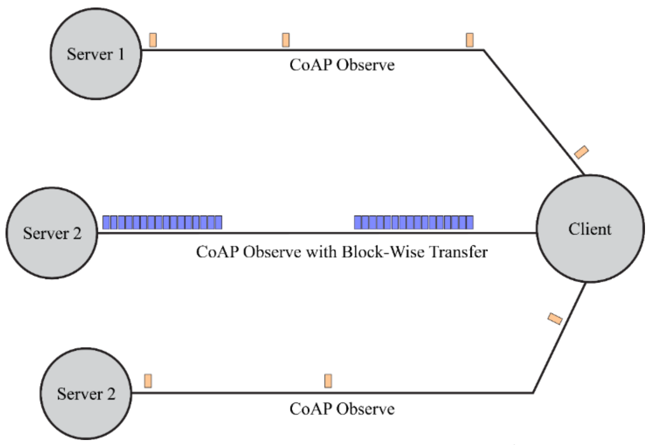


Fig. 2 CoAP Observe with Block-Wise Transfer

Furthermore, CoAP provides the block-wise transfer feature, which enables efficient data transfer in resource-limited environments. This feature allows large amounts of data to be divided into smaller blocks and transferred individually within a single operation, as illustrated in Fig. 2.

The use of CoAP OBS allows multiple servers to transfer data simultaneously, which can result in congestion at a client. However, the default CoAP congestion control algorithm called Binary Exponential Back-Off (BEB) has been found to be ineffective in reducing OBS congestion [8]. In this study, we examine the use of two alternative algorithms called FPB and HBEB to improve throughput and reduce packet loss in the context of block-wise transfer with CoAP OBS. Our aim is to identify effective congestion control strategies for this common scenario in IoT.

The remainder of the paper is organized as follows. In Section 2, we summarize CoAP functions and its congestion control algorithm. Section 3, we study the effect of FPB and HBEB on the congestion control mechanism for CoAP Observe with block-wise transfer and present the simulation setup and communication protocol stack settings that we determine the performance of our proposed mechanism and compare it with the standard CoAP mechanism. The results of these evaluations are presented in Section 4. In the final section, we provide the conclusions of this paper.

2. Literature Review

CoAP is a lightweight protocol for resource-constrained devices that enables efficient communication between devices over the Internet. It is commonly utilized in IoT applications where device processing, memory, and network bandwidth are constrained. CoAP is built on the Representational State Transfer (REST) architectural style, which uses HTTP-like protocols to access and modify resources on a server (GET, POST, PUT and DELETE). CoAP messages can be transmitted via multicast or unicast and are commonly sent over UDP. CoAP uses four types of messages to request and manipulate resources on a server:

• Confirmable (CON): A confirmable message requires a response from the server and will be retransmitted if a response is not received within a specific time period. Then, Confirmable messages are used for reliable communications such as POST and PUT requests.

• Non-Confirmable (NON): A non-confirmable message does not require a response from the server and will not be retransmitted if a response is not received. Non-confirmable messages are used for unreliable communication such as GET requests.

• Acknowledgment (ACK): An acknowledgment message is a response to a confirmable message and indicates that the request has been received and processed by the server.

• Reset (RST): A reset message is used to cancel a confirmable message that is in progress or to indicate that a non-confirmable message has been received and processed by the server.

CoAP also includes congestion control mechanisms designed to help reduce network congestion by spacing out retransmissions of messages when a response has not been received within a specific time period. The default BEB mechanism in CoAP provides an interval between retransmissions and allows the interval to increase gradually over time to ensure that retransmissions are spaced out and do not flood the network.

CoAP offers Observe Resource (OBS) [9] in addition to the conventional request/response mechanisms known as CoAP-based transmission, as shown in Fig. 3 (a). In the CoAP OBS, a client performs as an observer registers resources to interested servers. Then, each server will notify the client whenever a registered resource changes, as shown in Fig. 3 (b). The notification transmitted by the default server uses the NON message. In this study, we concentrated on the CoAP OBS and applied the CON message to provide the congestion control mechanisms.

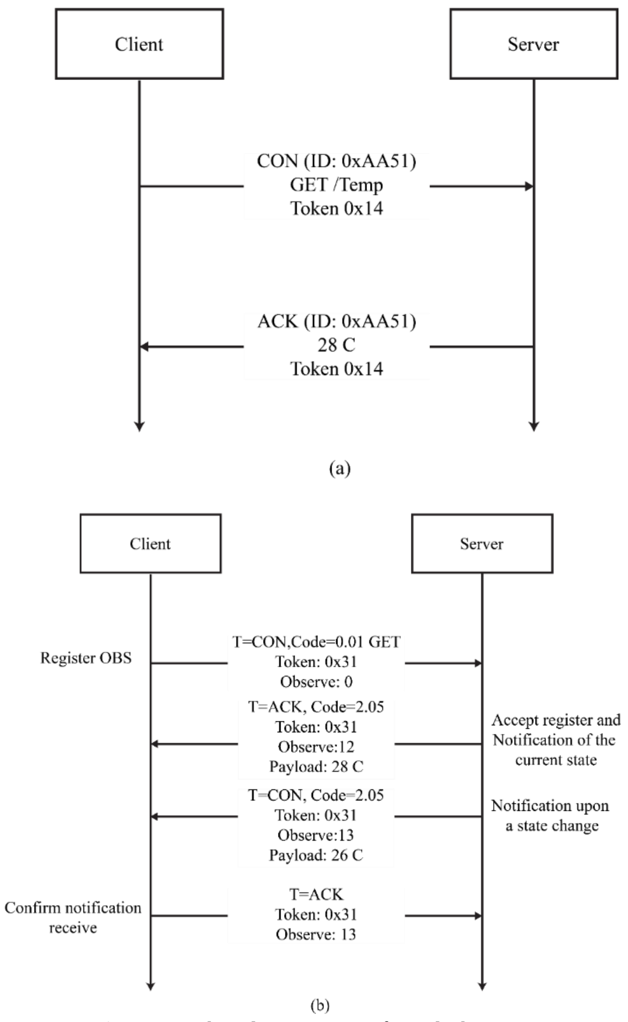
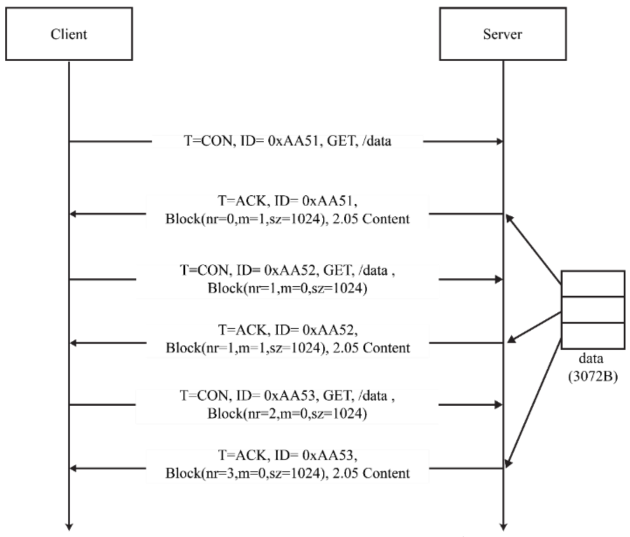


Fig. 3 CoAP-based message transfer and Observing resource message transfer

Furthermore, the client can register for a large data of resources in the observe mode, and those cannot be sent in a single CoAP packet. As a result, the server must partition into numerous data blocks, known as block-wise transfer. As shown in Fig. 4, each block contains the number of the most recently transmitted block (#Block), the block size (Size), and the number of subsequent blocks (M). Also, data retransmission uses a timeout.

Since CoAP works on top of UDP data transfer protocol, it is necessary to use extra congestion control. Default CoAP uses a straightforward stop-and-wait congestion control strategy to manage network congestion automatically by using Retransmission TimeOut (RTO) value to retransmit lost packets at exponential intervals until they obtain an ACK. However, doing so will increase network traffic congestion.



Client
Server
T=CON, ID= 0xAA51, GET, /data
T=ACK, ID= 0xAA51, Block(nr=0,m=1,sz=1024), 2.05 Content
T=CON, ID= 0xAA52, GET, /data , Block(nr=1,m=0,sz=1024)
T=ACK, ID= 0xAA52, Block(nr=1,m=1,sz=1024), 2.05 Content
T=CON, ID= 0xAA53, GET, /data , Block(nr=2,m=0,sz=1024)
T=ACK, ID= 0xAA53, Block(nr=3,m=0,sz=1024), 2.05 Content
data (3072B)

Fig. 4 CoAP Block-Wise Transfer

Several studies have been conducted to examine CoAP congestion control mechanisms in order to achieve optimal performance in various scenarios. For example, the approaches in [10]–[17] allow for dynamically determining RTO value based on round-trip time (RTT) samples and considering multiple outstanding interactions. However, the practical implementation has limitations in terms of constraints on RTT observation time and the possibility of ignoring burst traffic [18]. In addition, they are complex to implement and may not be backward compatible. Also, using a high RTO value in CoAP, especially in networks with high bit error rates such as CoAP OBS may result in long idle times, which can be a problem for resource-constrained devices like sensors and motes that rely on battery power. In order to address this issue, it is crucial to consider the limited battery power of IoT devices and to design protocols that are both efficient and easy to implement in terms of energy consumption [19]. The default CoAP algorithm minimizes complexity by not requiring the calculation and management of end-to-end connection information.

In general, the default CoAP behavior is entirely unaware of the state of the network. Numerous studies have shown that using a basic CoAP congestion control strategy might lead to substandard network performance [20]. Congestion control algorithms play an essential part in a wide range of IoT applications. As a result, numerous efforts have been made to identify and reduce congestion in the IoT. Two back-off mechanisms have been implemented in CoAP observe group communication algorithms introduced in [21], namely FPB and HBEB. The two proposed back-off mechanisms are to improve the packet drop ratio and throughput by changing the RTO value in CoAP observe mode without using RTT measurements. Hence, the working of the original CoAP algorithm remains largely unaffected.

3. Research Methodology

To alleviate congestion, CoAP calculates RTO with BEB to generate a random interval between 2–4 seconds. If a retransmission occurs, RTO is double leading to a long idle time to send another packet in case that congestion has stayed for a long period. In this study, we apply the FPB and HBEB to lower the RTO. Beginning with the RTO value of 2 seconds, overall time before the timeout was about 32 seconds for the BEB, 12 seconds for the FPB, and 8 seconds for the HBEB, as shown in Fig. 5. Similar to the default CoAP, our study permits retransmissions up to four times before classifying the notice as an error or failure.

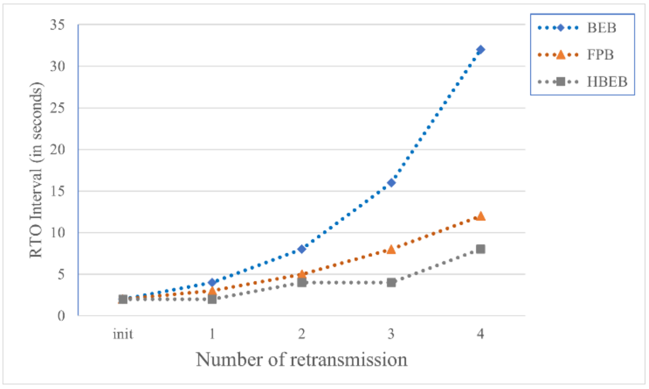


Fig. 5 The comparison of back-off method BEB, FPB, and HBEB

3.1 Fibonacci Pre-Increase Back-Off (FPB)

Fibonacci Pre-Increase Back-Off (FPB) [21] is a congestion control algorithm that uses a sequence of numbers based on the Fibonacci sequence to determine the amount of time to wait before retransmitting a message that the receiver has not acknowledged. In the FPB algorithm, the sender starts by waiting a certain number of milliseconds before retrying and then increases the wait time for each subsequent retry based on the Fibonacci sequence. FPB is defined as equation (1) presented in Algorithm 1:

$$RTO_{current} = RTO_{previous} \times Fibonacci[i] \quad (1)$$

Where $\mathrm{RTO}_{\mathrm{current}}$ is the amount of time to wait on the next retransmission, $\mathrm{RTO}_{\mathrm{previous}}$ is the amount of time waiting in the current retransmission, and Fibonacci[i] is the calculation of the Fibonacci number of the *i* retransmission.

```
1.  define ini_rto [2,3]
2.  int RTO = ini_rto
3.  int retransmissions = 0
4.  void send_request()
5.  if (!received_response)
6.  retransmissions++
7.   $RTO_{current} = RTO_{previous}$ * fib(retransmissions);
8.  wait($RTO_{current}$)
9.  send_request()
10. int fib(int n)
11. if (n <= 1) return n
12.  return fib(n-1) + fib(n-2)
```

Algorithm 1 FPB algorithm

3.2 Half Binary Exponential Back-Off (HBEB)

Half Binary Exponential Back-Off (HBEB) [21] is a congestion control algorithm that uses a binary exponential back-off approach to determine the RTO before retransmitting a message that the receiver has not acknowledged. In the HBEB algorithm, the sender starts by waiting a certain number of seconds before retrying and then increases the RTO for each subsequent retry by doubling the previous RTO. However, unlike BEB, the HBEB algorithm only increases the RTO for every other retry rather than for every retry. HBEB is defined as equation (2) and presents in Algorithm 2 :

$$RTO_{current} = RTO_{previous} \times 2^{k} \quad (2)$$

Where $RTO_{current}$ is the amount of time to wait on the next retransmission, $RTO_{previous}$ is the amount of time waiting in the current retransmission, and k is equal to 1 for odd retransmission and equal to 0 for even retransmission. For example, if the initial wait time is 2 seconds and the message is not acknowledged on the first attempt, the RTO will not be increased on the second attempt. It will only be increased on the third attempt, and so on. This means that the RTO will be increased less frequently in the HBEB algorithm than in the BEB algorithm.

```
1.  define ini_rto [2,3]
2.  int RTO = ini_rto
3.  int retransmissions = 0
4.  void send_request()
5.  if (!received_response)
6.  retransmissions++
7.   if (retransmissions % 2 == 1)
8.   $RTO_{current} = RTO_{previous}$
9.   else
10.  $RTO_{current} = RTO_{previous}$ * 2
11. wait($RTO_{current}$)
12. send_request()
```

Algorithm 2 HBEB algorithm

3.3 Evaluation Setup

This section presents a setup for performance evaluations of the three congestion control algorithm: the default CoAP BEB, FPB, and HBEB with simulation setup, traffic scenarios, network topologies and performance metrics.

3.3.1 Simulation Setup

The experiment runs on Cooja Simulation in Contiki OS 3.0, a well-known IoT operating system that supports the 6LoWPAN communications over IEEE 802.15.4. Moreover, it can upload the hex file created by the Cooja Simulation into the actual WSN mote for a real environment evaluation [22]. The simulation parameters are displayed in Table 1.

Setting	Value
Mote	Zolertia (Z1)
Routing protocol	Routing protocol for lower-power and lossy networks (RPL)
Wireless channel model	UDGM, transmission range = 15m, interference rage = 30m
Node distant	10 m
Transport and Network	UDP + uIPv6 + 6LoWPAN
Radio duty cycling (RDC)	Null-RDC, null-MAC
Radio band	2.4 GHz
Radio channel	26
Chanel check rate	128 Hz
Physical PHY	IEEE.802.15.4 PHY
Congestion mechanism	Default CoAP BEB, FPB and HBEB
Maximum retransmissions	4
Simulation time out	300 s
Maximum open transaction	11

Table 1 Cooja simulation parameter setup

In this simulation, Zolertia Z1 mote is selected, which is a low-power wireless sensor network (WSN) node supporting 6LoWPAN. The hardware specifications of the Z1 mote are displayed in Table 2.

Setting	Value
Microcontroller	MSP430F267 low power
RAM	8KB
ROM	92KB
Data rate	250Kbps
Clock Speed	16MHz

Table 2 mote hardware specification

3.3.2 Network Topology

The research in this article examines the potential effects of various congestion control strategies for RTO value on the overall network performance. To compare the performance of the default CoAP BEB, FPB, and HBEB, we use grid traffic scenarios that have been used in a variant of research on CoAP congestion control [8], [10], [15], [16], [21], [23]–[25]. The arrangement of the nodes model in 3x3 and 3x4 is shown in Fig. 6. Each node is separated by 10 meters, the transmission range is 15 meters, and the interference range is 30 meters. The communication in these scenarios uses OBS.

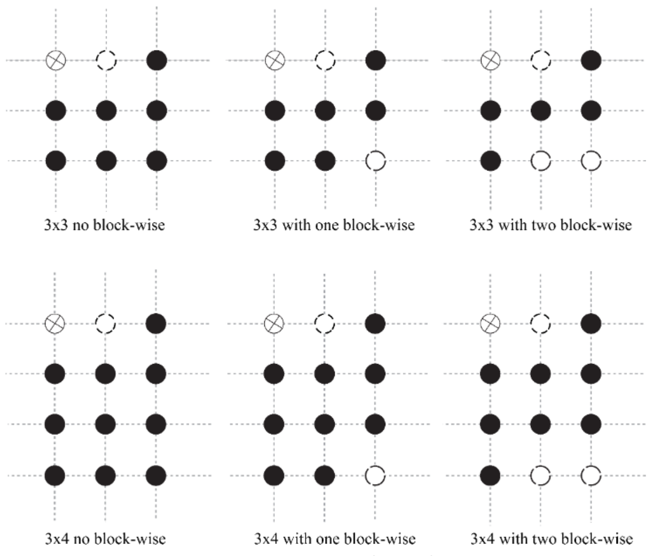


Fig. 6 Simulation node setup used for performance analysis

3.3.3 *Traffic Scenarios*

In OBS, one CoAP client is registered one by one to all CoAP observe servers. As soon as the OBS server completely registers with the CoAP client, it sends the periodic resource continuously to the client every second if the data of the resource changes.

We model our experiment with one border router (circle with a cross) node and one CoAP client (circle with six dashes) node. There are up to two CoAP Observe nodes (circle with three dashes) with the block-wise transfer. The rest of the nodes (black circle) are generally observed nodes, as illustrated in Fig. 6. Each node performs the following functions:

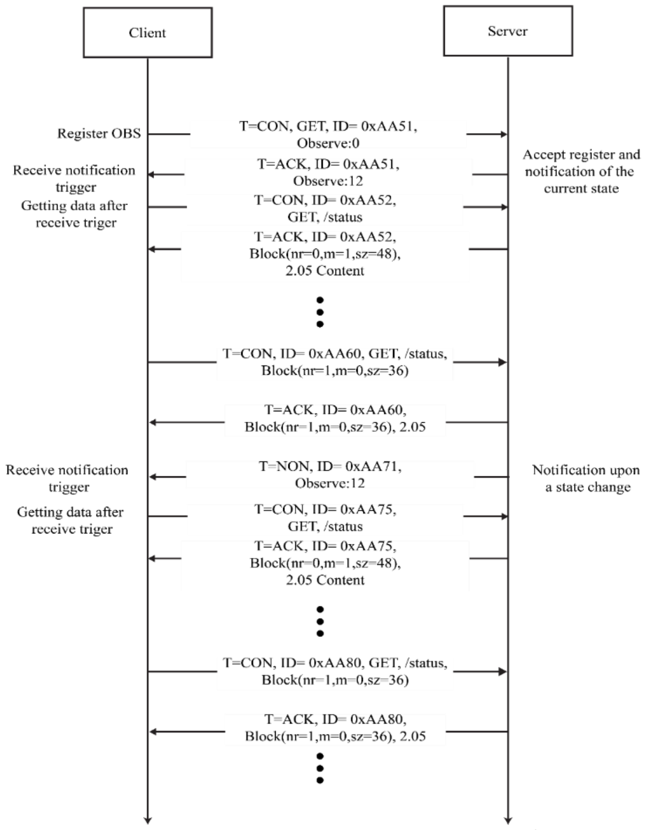
• The border router routes data between an internal network and the Internet.

• The CoAP client is used for getting data from all CoAP servers.

• The regular CoAP OBS is the server with periodic sending data (resource) with a size of 8 bytes and will notify the client every second if the resource has changed its value.

• CoAP OBS with the block-wise transfer is the server with a more significant resource payload of 1024 bytes. Therefore, we use the observer feature of CoAP as a trigger for sending block-wise data. The block size can be varied between 16 bytes and 1024 bytes per block. Due to memory constraints in the Z1 mote, we use a maximum 48 bytes block size of CoAP. Fig. 7 shows the structure of a CoAP OBS with block-wise transfer.

Six scenarios are tested for the effects of the RTO mechanisms, as shown in Fig. 6. These scenarios compare FPB and HBEB with default BEB on regular OBS and OBS with block-wise transfer performance in terms of packet loss ratio (PLR), end-to-end delay, and throughput. In every scenario, we test for 300 seconds after the client has registered all servers. Each scenario was repeated 50 times with different random seed numbers in the Cooja simulation.


Client
Server
Register OBS
T=CON, GET, ID= 0xAA51, Observe:0
Accept register and notification of the current state
Receive notification trigger
T=ACK, ID= 0xAA51, Observe:12
Getting data after receive triger
T=CON, ID= 0xAA52, GET, /status
T=ACK, ID= 0xAA52, Block(nr=0,m=1,sz=48), 2.05 Content
T=CON, ID= 0xAA60, GET, /status, Block(nr=1,m=0,sz=36)
T=ACK, ID= 0xAA60, Block(nr=1,m=0,sz=36), 2.05
Receive notification trigger
T=NON, ID= 0xAA71, Observe:12
Notification upon a state change
Getting data after receive triger
T=CON, ID= 0xAA75, GET, /status
T=ACK, ID= 0xAA75, Block(nr=0,m=1,sz=48), 2.05 Content
T=CON, ID= 0xAA80, GET, /status, Block(nr=1,m=0,sz=36)
T=ACK, ID= 0xAA80, Block(nr=1,m=0,sz=36), 2.05
Fig. 7 CoAP OBS with the block-wise transfer

4. Result and Discussion

CoAP OBS behavior allows a client to request notifications from multiple servers about the changes of particular resources. This allows the client to receive updates from the server in real-time without continuously polling the updates. However, all servers constantly send data to the client whenever their resource changes. As shown in Table 3, the PLR is over 50%. At both grid scenarios, 3x3 and 3x4 without the block-wise transfer, the use of HBEB performs the best in terms of throughput and packet loss. However, BEB performs the best in terms of end-to-end delay.

With one and two block-wise transfers, congestion is higher because the block-wise node sends a large amount of data into the network. This causes delays in transferring resources, reduces network performance, or fails in communication. However, once the block-wise transfer is finished, congestion is quickly reduced. BEB has a negative impact on high congestion networks because it doubles the RTO value with each retransmission causing the sender to wait longer before retransmitting a packet. In highly congested networks, many packets are waiting in the queue which leads to packet drops and retransmissions. Then, the throughput of the network is lower.

In contrast, compared to BEB, the gradual increase in RTO of HBEB values in a highly congested network provides the benefit of reducing congestion and increasing the chances of successful delivery. As shown in **Fig. 5**, the lower increasing period of HBEB provides enough time to reduce congestion. Then, packets are retransmitted quickly. This improves the overall network throughput. Therefore, the results show that HBEB outperforms both BEB and FPB in terms of throughput and packet loss as shown in **Table 3**.

Grid size	Back-off algorithm	PLR (%)	End-to-End delay (ms)	Throughput (Kbps)
3x3-0	BEB	53.59	**5.61**	8.30
	FPB	52.57	6.28	10.06
	HBEB	**52.72**	6.24	**15.18**
3x4-0	BEB	54.41	**3.89**	11.18
	FPB	54.05	5.02	11.67
	HBEB	**53.82**	4.24	**12.34**
3x3-1	BEB	54.15	5.59	10.03
	FPB	54.02	**5.39**	11.76
	HBEB	**53.96**	5.43	**12.44**
3x4-1	BEB	**53.97**	**3.99**	14.04
	FPB	54.59	4.46	15.74
	HBEB	55.31	4.02	**16.44**
3x3-2	BEB	55.05	4.29	10.30
	FPB	54.42	**4.27**	11.12
	HBEB	**54.36**	4.53	**11.42**
3x4-2	BEB	55.25	**3.17**	15.88
	FPB	55.20	3.88	16.14
	HBEB	**54.98**	3.61	**17.18**

Table 3 Overall performance metric value comparison (number in bold is the better performance)

5. Conclusion

The goal of this research was to examine the potential effects of different congestion control strategies on the RTO value in CoAP observe group communication with the block-wise transfer. We compared three strategies: default CoAP BEB, FPB, and HBEB in terms of their impacts on packet loss ratio, end-to-end delay, and throughput. In our simulation, we implemented CoAP in the Cooja simulation of Contiki OS 3.0 with multiple CoAP servers sending back-to-back traffic to an OBS client in scenarios with no block-wise transfers, one block-wise transfer and two block-wise transfers. The results showed that network traffic is often highly congested in CoAP OBS with the block-wise transfer. As a result, using the HBEB resulted in the best throughput and packet loss ratio. These findings suggest that HBEB is an effective option for congestion control in CoAP, especially in scenarios involving group communication resource observation with the block-wise transfer.

References

[1] P. Sharma, I. Pandey, and P. M. Pradhan, "Hardware Implementation and Comparison of IoT Data Protocol for Home Automation Application," in *2022 IEEE Delhi Section Conference (DELCON)*, pp.1–6, 2022 doi: 10.1109/DELCON54057.2022.9752957.

[2] D. Ray, P. Bhale, S. Biswas, S. Nandi, and P. Mitra, "DAISS: Design of an Attacker Identification Scheme in CoAP Request/Response Spoofing," in *TENCON 2021-2021 IEEE Region 10 Conference (TENCON)*, pp. 941–946, 2021, doi: 10.1109/TENCON54134.2021.9707405.

[3] D. Ugrenovic and G. Gardasevic, "CoAP protocol for Web-based monitoring in IoT healthcare applications," Nov. 2015. doi: 10.1109/TELFOR.2015.7377418.

[4] M. D. Babakerkhell and N. Pandey, "Analysis of different IOT based healthcare monitoring systems," *Int J Innov Technol Explor Eng (IJITEE)*, 2019.

[5] P. K. Rath, N. Mahapatro, S. Sahoo, and S. Chinara, "IoT Based Health Monitoring System for Hospital Management," 2021.

[6] A. Aldribi and A. Singh, "Blockchain Empowered Smart Home: A Scalable Architecture for Sustainable Smart Cities," *Mathematics*, vol. 10, no. 14, p. 2378, 2022.

[7] S. Beloualid, S. El Aidi, A. El Allali, A. Bajit, and A. Tamtaoui, "Applying Advanced IoT Network Topologies to Enhance Intelligent City Transportation Cost Based on a Constrained and Secured Applicative IoT CoAP Protocol," *Advances in Information, Communication and Cybersecurity: Proceedings of ICI2C'21*, vol. 357, p. 195, 2022.

[8] E. Ancillotti and R. Bruno, "Comparison of CoAP and CoCoA+ congestion control mechanisms for different IoT application scenarios," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1186–1192, 2017, doi: 10.1109/ISCC.2017.8024686.

[9] J. Youn and H. Choi, "CoAP-based Reliable Message Transmission Scheme in IoT Environments," *Journal of the Korea Society of Computer and Information*, vol. 21, pp. 79–84, 2016, doi: 10.9708/jksci.2016.21.1.079.

[10] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoCoA+: An advanced congestion control mechanism for CoAP," *Ad Hoc Networks*, pp. 126–139, 2016, doi: 10.1016/j.adhoc.2015.04.007.

[11] R. Bhalerao, S. S. Subramanian, and J. Pasquale, "An analysis and improvement of congestion control in the CoAP Internet-of-Things protocol," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 889–894, 2016. doi: 10.1109/CCNC.2016.7444906.

[12] C. Bormann, A. Betzler, C. Gomez, and I. Demirkol, "CoAP Simple Congestion Control/Advanced," Internet Engineering Task Force, Internet Draft draft-ietf-core-cocoa-00, 2017. [Accessed: Jan. 28, 2022].

[13] H. Meng, H. HongBing, and L. WeiPing, "An Adaptive Congestion Control Algorithm with CoAP for the Internet of Thing," *Int. J. Comput. Tech*, vol. 4, pp. 46–51, 2017.

[14] I. Jarvinen, I. Raitahila, Z. Cao, and M. Kojo, "FASOR Retransmission Timeout and Congestion Control Mechanism for CoAP," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2018, doi: 10.1109/GLOCOM.2018.8647909.

[15] S. Bolettieri, G. Tanganelli, C. Vallati, and E. Mingozzi, "pCoCoA: A precise congestion control algorithm for CoAP," *Ad Hoc Networks*, vol. 80, pp. 116–129, 2018, doi: 10.1016/j.adhoc.2018.06.015.

[16] V. Rathod, N. Jeppu, S. Sastry, S. Singala, and M. P. Tahiliani, "CoCoA++: Delay gradient based congestion control for Internet of Things," *Future Generation Computer Systems*, vol. 100, pp. 1053–1072, 2019, doi: 10.1016/j.future.2019.04.054.

[17] G. A. Akpakwu, G. P. Hancke, and A. M. Abu-Mahfouz, "CACC: Context-aware congestion control approach for lightweight CoAP/UDP-based Internet of Things traffic," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, p. e3822, 2020.

[18] R. K. Yadav, N. Singh, and P. Piyush, "Genetic CoCoA++: genetic algorithm based congestion control in CoAP," in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp.808–813, 2020. doi: 10.1109/ICICCS48265.2020.9121093.

[19] M. A. Tariq, M. Khan, M. T. Raza Khan, and D. Kim, "Enhancements and Challenges in CoAP—A Survey," *Sensors*, vol. 20, no. 21, 2020, doi: 10.3390/s20216391.

[20] I. Järvinen, L. Daniel, and M. Kojo, "Experimental evaluation of alternative congestion control algorithms for Constrained Application Protocol (CoAP)," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 453–458, 2015, doi: 10.1109/WF-IoT.2015.7389097.

[21] C. Suwannapong and C. Khunboa, "EnCoCo-RED: Enhanced congestion control mechanism for CoAP observe group communication," *Ad Hoc Networks*, vol. 112, p. 102377, 2021, doi: 10.1016/j.adhoc.2020.102377.

[22] I. Essop, J. C. Ribeiro, M. Papaioannou, G. Zachos, G. Mantas, and J. Rodriguez, "Generating Datasets for Anomaly-Based Intrusion Detection Systems in IoT and Industrial IoT Networks," *Sensors*, vol. 21, no. 4, 2021, doi: 10.3390/s21041528.

[23] V. J. Rathod and M. P. Tahiliani, "Geometric Sequence Technique for Effective RTO Estimation in CoAP," in *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–6, 2020. doi: 10.1109/ANTS50601.2020.9342748.

[24] N. Makarem, W. Bou Diab, I. Mougharbel, and N. Malouch, "On the design of efficient congestion control for the Constrained Application Protocol in IoT," *Computer Networks*, vol. 207, p. 108824, 2022, doi: 10.1016/j.comnet.2022.108824.

[25] S. Deshmukh and V. T. Raisinghani, "AdCoCoA- Adaptive Congestion Control Algorithm for CoAP," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2020, doi: 10.1109/ICCCNT49239.2020.9225315.

Biographies

Thavrak Chan received a B.Sc. degree in information technology and engineering from the Royal University of Phnom Penh, Cambodia, in 2016. Currently, he is continuing his master’s degree at the Department of Computer Engineering at Khon Kaen University.

Chanwit Suwannapong received a B.Eng., M.Eng, and a Ph.D. degree in computer engineering from Khon Kaen University, Thailand, in 2011, 2013, and 2020 respectively. He has published many publications in the area of Wireless Sensor Networks, Ad Hoc Networks, and Smart Agriculture.

Chatchai Khunboa is an associate professor in the Department of Computer Engineering at Khon Kaen University, Thailand. Dr. Khunboa received a B.Eng. from Khon Kaen University in 1992, his MSc degree in Telecommunications from the University of Pittsburgh in 2000, and his Ph.D. degree in Information Technology from George Mason University in 2005. His research interests include Wireless Sensor Networks, the Internet of Things, and Software-defined Networks (SDN).