RESEARCH ARTICLE OPEN ACCESS

An Adaptive Congestion Control Algorithm with CoAP for the Internet of Thing

He Meng[1], Huang HongBing[2], Liu WeiPing[2]
[1](Department of Electronic Engineering, Jinan University, Huangpu Road west, No.601, Guangzhou, China)
[2](Department of Electronic Engineering, Jinan University, Huangpu Road west, No.601, Guangzhou, China)
* (Corresponding author: He Meng)

----------**********----------

Abstract:

CoAP is a dominant application protocol designed for the constrained devices in IoT (the Internet of Things) which is going to interconnect every physical object in our daily life together to make an intelligent world. To keep the communication between devices in IoT from congestion, CoAP originally provides a simple congestion control mechanism. Since this mechanism can hardly meet the requirements of many IoT applications, an extension called CoCoA is being standardized by IETF(the Internet Engineering Task Force) to optimize CoAP. However, the thresholds underpinned the VBF(variable backoff factor) and RTO(retransmission timeout) aging mechanism in traditional CoCoA are constants and cannot be changed under most conditions which limit the performance of CoCoA; and further the existed estimators of CoCoA ambiguity have the potential risk to cause some unexpected or unpredictable problems. Hence, in this paper, we propose a novel threshold adjustable scheme for the VBF and RTO aging mechanism and a new encapsulation method for message retransmission to solve the above problems in the traditional CoCoA. Our evaluation results show that our modified version CoCoA greatly helps to solve the problem of congestion in IoT which outperforms over the existed CoCoA in terms of throughput and network adaptability.

Keywords **—CoAP, congestion control, CoCoA, IoT.**

----------**********----------

I. INTRODUCTION

IoT, an extension of the current Internet, not only interconnects traditional devices, PC for example, but also daily objects such as light, air conditioning, and so on.[1] It's going to greatly improve the quality of our life. IoT provides equipment manufacturers, Internet service providers and application developers with a great market. Thereinto the two dominant are health care and manufacturing that takes up of 41% and 33% respectively[2].

To make the vision of IoT come true, a lot of relative technologies should be developed, including identification, sensing, communication, computation, services and semantics[2]. Communication is an essential element of IoT which makes communication protocols the key enabler of IoT. Devices from different equipment manufacturers must be possible to communicate with each other if they work under the same protocols. In other words, protocols will play the key role in eliminating the gap between devices. Also, they serve standard interfaces to programmers and services providers to develop various applications. Devices in IoT are so diverse in their individual memory, computation capability and bandwidth that it needs a specially designed protocol to coordinate with each other.

Among these protocols, a protocol suite developed from a standardized protocol stack and made up of the IEEE 802.15.4-2006 PHY layer, the IEEE 802.15.4e MAC layer, the IETF 6LoWPAN for adaption layer, the IETF RPL for routing protocol, UDP for transport layer and the IETF CoAP provid-

ing REST services[3], becomes widely accepted due to its power efficiency, low cost in both memory and computation.

CoAP[4] is specially designed for constrained devices in IoT, which is special at least in two aspects: one is that the devices in IoT are only equipped with limited memory and computation resources, working in networks characteristic of low bandwidth and relatively high bit error rate[5]; the other is that the transport layer protocol of CoAP, namely UDP, is unreliable to make the CoAP risk congestion, thus it needs an additional congestion control to prevent it. Although a simple congestion control mechanism has been established for the CoAP, it can hardly meet the requirement of many IoT application. Thus, a new mechanism called CoCoA[6], which is a standardizing IETF draft, continues to be developed to provide a powerful congestion control scheme for CoAP with minimal extra resources.

This paper aims to proposes a modified version of CoCoA to improve the performance of CoAP in IoT. We organize it as follows: first summarizing related work in section Ⅱ; then introducing our modified version of CoCoA in section Ⅲ; and then after in section Ⅳ, describing our network setup and discussing the evaluation results; finally, in section Ⅴ making a conclusion about this paper.

II. RELATED WORK

A. CoAP

CoAP is a REST-ful application protocol designed for constrained devices. It implements a subset of HTTP, which excludes out all the features that are unnecessary for IoT, instead introduces mechanisms suitable for IoT applications, such as built-in resource discovery, multicast support, asynchronous message exchange and so on[7]. There are four kinds of messages in CoAP, namely CON, NON, ACK and RST. Within the duration of RTO, the four kinds messages send and receive according to congestion control mechanism defined in CoAP. For example after a sender transfers a CON message to a receiver, the sender waits for an ACK message from the receiver. If the sender receives the expected ACK message, the CON message will be supposed to be transferred successfully, otherwise unsuccessfully and in this case the sender will retransmit the CON message for MAX_RETRANSMIT(4 by default) times at most. Usually, the RTO is initialized randomly to be the values between 2s and 3s, and will be changed in every retransmission process in the BEB(binary exponential backoff) manner.

The RTO is the key parameter of congestion control, but it doesn't take RTT into consideration which may cause catastrophic result [8]. If it is too small (compared with RTT), plenty of unnecessary retransmissions may appear, while if it is too large, valuable bandwidth will waste.

B. CoCoA

In order to solve the problems of congestion control mechanism in CoAP, CoCoA introduces TCP's RTO estimation algorithm(RFC 6298)[9]. CoCoA defines VBF and RTO aging mechanism in consideration of dynamic network condition of IoT applications, and establishes a congestion control mechanism for NON message that based on CON message's congestion control mechanism in naked CoAP.

Instead of only one RTO estimator as in RFC 6298, two estimators, namely strong estimator and weak estimator, are created and run by a single sender concurrently towards the same receiver to increase the probability of obtaining a valid RTO. The strong estimator works in initial transmission while the weak estimator works in the first or the second retransmission. To avoid the ambiguity whether an ACK massage is triggered by the transmission or the retransmissions, CoCoA simply assumes that the ACK message is based on the initial transmission. When a valid RTT is measured, the following equations will be performed:

$$RTTVAR_X = (1-\beta)\times RTTVAR_X + \beta\times(RTT_X - RTT_{X\_new}) \quad (1)$$

$$RTT_X = (1-\alpha)\times RTT_X + \alpha\times RTT_{X\_new} \quad (2)$$

where X characters "strong" or "weak" that stands for the respective strong or weak estimator, while α=1/8 and β=1/4. And then, update the the RTO_X:

$$RTO_X = RTT_X + \max(G, K_X \times RTTVAR_X) \quad (3)$$

where G is a constant value representing a clock granularity, while K_{strong}=4 and K_{weak}=1. And the overall RTO is:

$$RTO_{overall} = \gamma_X \times RTO_X + (1-\gamma_X) \times RTO_{overall} \quad (4)$$

where γ_{strong}=0.5 and γ_{weak}=0.25.

From Eq(1-4) we can see that weak estimator predicts a large possibility of geting RTO, but an over estimation for RTO value at the same time. However, if adjusting the weak estimator's parameters, for example, K_{weak} =1 instead of 4 and γ_{weak} =0.25 instead of 0.5, the over estimated RTO value will be resuded, though with the problem of the ambiguity still being actually unsolved.

The CoAP base specification applies BEB to RTO updating when the second or further retransmission occurs. However, initial RTO may not be too small or too large when compared to current network condition due to the dynamic feature of IoT. Noting that though most of packet losses in IoT tend to be the result of high bit error rate of the lossy link rather than real congestion problem, CoAP's congestion control mechanism treats it as a congestion problem and increases the RTO estimation. To deal with this problem, CoCoA defines VBF and RTO aging mechanism. When the initial RTO is smaller than 1s, the back off factor will be set to 3 to avoid using up all retransmissions in a short period. However, when the initial RTO is larger than 3s, the back off factor will be adjusted to 1.5 to improve the efficiency of bandwidth usage. Otherwise, the initial RTO is applied with BEB. When the initial RTO value has not been updated for a long time, it's probable that the initial RTO is aging and needs replacing[6]. Then, CoCoA will double the initial RTO if it has been smaller than 1s and has not been updated more than 16 times; and if it has not been updated more than 4 times and larger than 3s, it will be forced to update as follows:

$$RTO_{initial} = 1s + 0.5 \times RTO_{initial} \quad (5)$$

As having taken dynamic nature of IoT network into account, VBF and RTO aging mechanism is an appropriate technology to cope with the problems in most IoT applications. Nevertheless, the performance of VBF and RTO mechanism is dependent on predetermined parameters that cannot be adjusted in the process of messages sending and receiving. The unadjustablity of these parameters may lead to unexpected results in some cases, for example, the fixed thresholds may be not reasonable becase the RTO of a transaction within a local area network may be obviously smaller than that between a local device and a remote device.

C. Current State

Since its first being proposed by C.Bormann in 2012, CoCoA has attracted more and more attention from related community. A.Betzler puts forward improvement to enhance the base CoCoA[5] , and extends its application from reliable CoAP to unreliable CoAP through thorough evaluation and comparison[8,10,11]. The latest version of CoCoA draft has been released in March this year, which is the seventh IETF draft of CoCoA. Though a lot of improvements has been made in it, to make CoCoA a more suitable congestion control mechanism for IoT applications, however there are still some work to be done in the following aspects: 1) the ambiguity whether an ACK massage is based on the transmission or the retransmissions in weak estimator; 2) the appropriate constants for VBF mechanism and RTO aging mechanism to decide whether and how to take actions.

III. IMPROVEMENT FOR COCOA

In this section, we propose our solutions to overcome the shortcomings mentioned in section II.

A. *Solution for ambiguity in weak estimator*

To cope with the ambiguity whether an ACK massage is based on the transmission or the retransmissions in weak estimator, CoCoA simply assume that the received ACK message is always triggered by the initial transmission which may lead to over estimation of RTO value[12]. Problem may be even worse when retransmission is caused by bit error under lossy links. When it happens, a wiser decision is to retransmit frequently to improve throughput. However, in this case the large initial RTO will mislead the sender not to retransmit. Thus, assuming the received ACK message is triggered by

the latest retransmission seems to be more reasonable[12], though it still fails to effectively eliminate the ambiguity. Another solution is to add an option to the CoAP message header to carry the information of retransmission count(Fig.1).

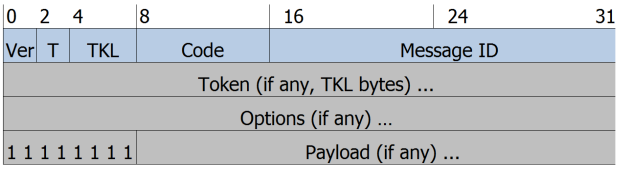

Fig. 1 CoAP message format

This solution is at the cost of valuable bandwidth, and high energy consumption especial for the case of wireless communication which will usually cost about more than 10 times energy compared with that in-node computations[13]. Obviously, it's a good idea to offload cost from communication to computation. To this end, we create a unique Message ID for each message in our work, including initial transmission and retransmission messages. When a CON message is going to be retransmitted, instead of retrieving the previous CON message from buffer, a new copy of it with the same payload but different Message ID will be encapsulated. Therefore, the energy consumption transfers from the message transmission to the generation of new messages packages in the memory of devices.

In addition, this solution is able to strengthen the congestion control mechanism of CoAP despite of some limits. When a CON message transports an idempotent request (GET, PUT and DELETE), the unique message ID is OK for the receiver to process any request more than once. And for non-idempotent requests (POST) which depends on application semantics, if the trade-off is favorable, the unique message ID technology can further reduce the cost of keeping track of retransmission states.

B. *Improvement for VBF*

VBF is a skill to adjust the backoff factor according to the initial RTO, in order to reduce the negative effect resulting from a too large (more than upper threshold) or too small (less than lower threshold) value of initial RTO. Specifically, the backoff factor is adjusted to its upper limit if the initial RTO is less than the lower threshold, and to the lower limit if it is larger than the upper threshold. However, the right thresholds are related to the real network environments and dynamic in nature, thus should not be fixed to be constant.

To make the thresholds adaptive to actual network environment, we take RTO_{strong} as a reference. The lower threshold is replaced by 1/3 * RTO_{strong} which keeps RTO value not be larger than RTO_{strong} until the second retransmission, while the upper threshold is modified as 5/3 * RTO_{strong} to guarantee that RTO value increases in a relative fast speed. When RTO_{strong} is set to the default value, that is RTO_{strong}=2s, the lower and upper threshold will almost get back to their respective default value 0.7s and 3.3s.

C. *Improvement for aging time*

As discussed in subsection B, the RTO value in real network environment is dynamic. It will be adjusted in a forcing manner if it is out of a specific range and not being updated for a relatively long period. However, if within this specific range, it has not an expire time and will not be adjusted at all. In fact, the special range corresponds to the interval between the lower threshold and the upper threshold representing a relatively reasonable RTO estimation for the current network. Combining the discussion in subsection B, the range can be specified as (1/3 * RTO_{strong}, 5/3 * RTO_{strong}).

IV. EVALUATION

In this section we describe our experimental setup and the evaluation results.

A. *Experimental Setup*

To create a CoAP network, a Raspberry Pi (model B v3) is served as a CoAP server center with each of its threads corresponding to a CoAP server. And a desktop (CPU 3.1GHz, RAM 3.8GB and Core i5) operating with Ubuntu Linux acts as a CoAP client center with each of its threads representing a CoAP client. The desktop and the Raspberry Pi are interconnected via a single switch to communicate with each other.

The Java Californium (Cf) CoAP[1] , which is an open source CoAP implementation written in Java, is run on both the CoAP clients and the servers. Ne-

[1] http://www.eclipse.org/californium/

tEM (Network Emulation)[2], a tool providing functionality for testing protocols by emulating the properties of wide area networks, is run on the desktop to simulate the real environment of IoT network and helps to control the BER and latency of the network. There is a "Hello" resource in every CoAP server for clients to GET. During the experiment, every CoAP client individually and randomly visits CoAP servers by sending CON requests.

In the first scenario, we place 5 servers in the Raspberry Pi with the number of CoAP clients in the desktop varying from 20 to 180. The packet loss is set to 20%, and the network delay is 200 ±and the network latency 200ms with a fluctuation within 20ms. The clients send CON requests to servers back-to-back.

In the second scenario, 60 clients and 5 servers are deployed in the desktop and Raspberry Pi respectively. The network delay is set to 80 ± 10msthe network lantency is set to 80ms with a fluctuation within 10ms. The clients send CON requests to servers back-to-back . Meanwhile, NetEM is used to adjust the bit error rate of lossy link, from 4% to 27%.

In the third scenario, 60 clients and 5 servers are placed. The clients also send CON requests to servers back-to-back. The packet loss is set to 20%. Instead of varying the bit error rate of lossy link, the network latency is altered from 20 to 110ms with a fluctuation within 10ms to measure the throughput.

B. ***Evaluation Results***

For the sake of convenience , the improved version of CoCoA is namea as CoCoAI.

It can be seen in Fig.2 that in general the throughput evaluated in all cases will increase with the increase of the number of clients as expected. However, it can also be seen that both the CoCoAI-based and CoCoA-based throughput are obviously larger than CoAP-based throughput when the client number is less than 150. However, if the number of the clients is more than 150, both CoCoAI and CoCoA are no longer effective due to the jamming of network caused by more clients occupying the same number of servers. It is also worth to note that CoCoAI will outperform over CoCoA especially when the network is going to be jammed.

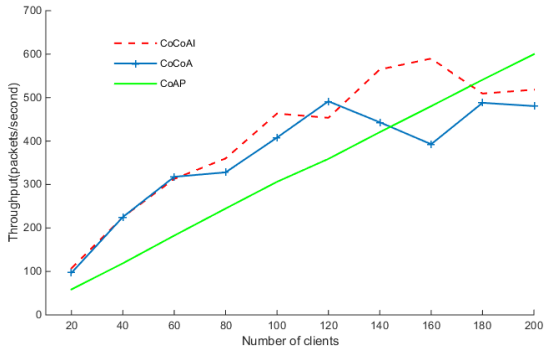

Fig. 2 Throughput against number of clients

In terms of packet loss, as can be seen from Fig.3, CoCoA and CoCoAI behave better than the naked CoAP congestion control algorithm in the same way. As the increase of bit error rate, the probability for a sender to get the ACK message becomes smaller. Then, CoCoA and CoCoAI propel the sender to send message more frequently that effectively increase the probability to receive an ACK message, while CoAP does nothing which makes it harder to get an ACK message.

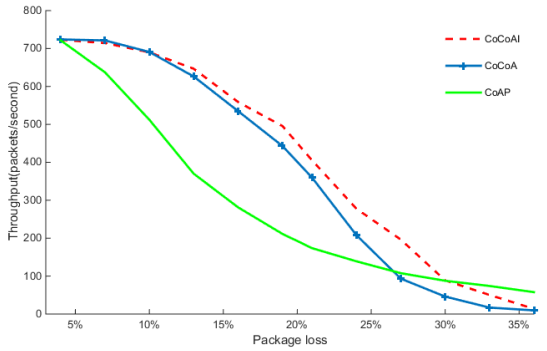

Fig. 3 Throughputs against packet loss

Also, as is shown from Fig.4, the CoCoA and CoCoAI which are the congestion control algorithms with the aging mechanism effectively outperform over CoAP which is without in terms of throughput.

[2] https://wiki.linuxfoundation.org/networking/netem

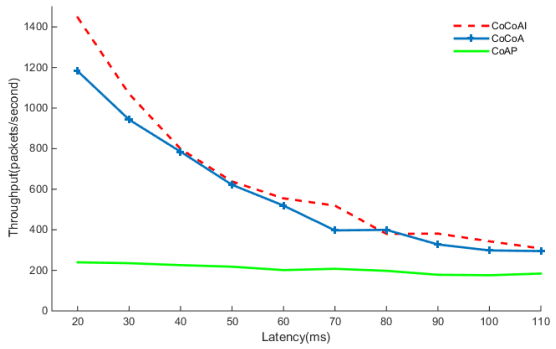

Fig. 4 Throughput against lantency

Furthermore, from Fig.2-4, it can be found CoCoAI is more efficient and powerful than other two congestion control algorithms to be used in a complex environment of network.

V. CONCLUSION

This paper proposes a series of effective improvements for CoCoA to overcome its shortcomings occurring once used in actual applications. The main improvements for CoCoA focus on two aspects: one is to eliminate the ambiguity existing in the process of its weak estimator by redistributing the same waiting-for-being-retransmitted message with a new Msssage ID; another is to dynamically adjust the thresholds (upper and lower) to settle down the low adaptability to network caused by the fixed thresholds setting in the VBF and aging mechanism of it. And the experiment results show that our improvements can help CoCoA outperform over the original CoAP and CoCoA in terms of throughput under the network conditions that are with the same clients number, package loss rate and lantency, indicating that our solution is better than the original CoAP and CoCoA for the control of the congestion of the dynamic IoT network.

ACKNOWLEDGMENT

This work was supported in part by the National High Tel R&D Program of China (No.2015AA015501).

REFERENCES

[1] Want R, Schilit B N, Jenson S. Enabling the internet of things[J]. Computer, 2015, 48(1): 28-35.

[2] Al-Fuqaha A, Guizani M, Mohammadi M, et al. Internet of things: A survey on enabling technologies, protocols, and applications[J]. IEEE Communications Surveys & Tutorials, 2015, 17(4): 2347-2376.

[3] Palattella M R, Accettura N, Vilajosana X, et al. Standardized protocol stack for the internet of (important) things[J]. IEEE communications surveys & tutorials, 2013, 15(3): 1389-1406.

[4] Shelby Z, Hartke K, Bormann C. The constrained application protocol (CoAP)[J]. 2014.

[5] Betzler A, Gomez C, Demirkol I, et al. CoCoA+: An advanced congestion control mechanism for CoAP[J]. Ad Hoc Networks, 2015, 33: 126-139.

[6] Bormann C, Betzler A, Gomez C, et al. CoAP Simple Congestion Control/Advanced (Work in Progress), July 2014[J]. URL http://tools.ietf.org/id/draft-bormann-core-cocoa-02.txt.

[7] Sheng Z, Yang S, Yu Y, et al. A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities[J]. IEEE Wireless Communications, 2013, 20(6): 91-98.

[8] Betzler A, Gomez C, Demirkol I, et al. CoAP congestion control for the Internet of Things[J]. IEEE Communications Magazine, 2016, 54(7): 154-160.

[9] Paxson V, Allman M, Chu J, et al. Computing TCP's retransmission timer[R]. 2011.

[10] Betzler A, Gomez C, Demirkol I, et al. Congestion control in reliable CoAP communication[C]//Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems. ACM, 2013: 365-372.

[11] Betzler A, Gomez C, Demirkol I. Evaluation of advanced congestion control mechanisms for unreliable CoAP communications[C]//Proceedings of the 12th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks. ACM, 2015: 63-70.

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

[13] Raza S, Shafagh H, Hewage K, et al. Lithe: Lightweight secure CoAP for the internet of things[J]. IEEE Sensors Journal, 2013, 13(10): 3711-3720.