EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Avoiding Congestion for Coap Burst Traffic

Thi Thuy Duong Le[1], Dang Hai Hoang[2,*], Thieu Nga Pham[1]

[1]University of Civil Engineering, Hanoi, Vietnam
[2]Posts and Telecommunications Institute of Technology, Hanoi, Vietnam

Abstract

Congestion is an important issue in Internet of Things (IoT) networks with constrained devices and a growing number of applications. This paper investigated the problem of congestion control for burst traffic in such networks. We highlight the shortcomings of the current constrained application protocol (CoAP) in its inability to support burst traffic and rate control. Subsequently, we propose an analytical model for CoAP burst traffic and a new rate-control algorithm for CoAP to avoid congestion. A CoAP sender increases or decreases the transmission rate depending on the congestion detection. Using simulations, we compared the performance of the proposed algorithm with the current CoAP in various traffic scenarios. Experimental results show that the proposed algorithm is efficient for burst traffic and provides better performance in terms of delay, throughput, retransmission, packet duplication, and packet loss compared to CoAP.

Keywords: Congestion control, Internet of Things, Rate control.

Received on 31 August 2022, accepted on 22 December 2022, published on 29 March 2023

doi: 10.4108/eetiot.v9i1.2655

*Corresponding author. Email: haihd@ptit.edu.vn

1. Introduction

Internet of Things (IoT) networks are widely applied in many fields such as industry, agriculture, healthcare, transportation, environment, and smart cities. Typically, IoT networks consist of three main components: (1) a subnet of various IoT devices to collect data from the environment, (2) a subnet of gateways for relaying data, and (3) multiple servers on the Internet to process collected data and provide services. Typical applications often require sending a large amount of collected data to a remote server on the Internet. This transmission can cause congestion, which leads to unexpected performance degradation. The problem of congestion control has been extensively studied in traditional computer networks but still is a challenge in IoT networks.

In contrast to traditional networks, IoT networks have different characteristics and dynamic links, with a high bit error rate. IoT devices typically have limited resource and processing capability. Therefore, the transmission control protocol (TCP) has been neglected in IoT networks [1]. Because of the limited resources and constraints, the development of lightweight protocols is encouraged. Several lightweight transport protocols have been developed for IoT networks, such as message queue telemetry transport (MQTT), advanced message queuing protocol (AMQP), and constrained application protocol (CoAP) [2]. MQTT and AMQP rely on TCP to transport data messages. In contrast, CoAP operates on top of a user datagram protocol (UDP), but it provides reliable connection-oriented data transport similar to TCP.

CoAP has been standardized by the Internet Engineering Task Force (IETF) with RFC 7252 [3]. Similar to TCP, CoAP provides a congestion control mechanism. However, the design of CoAP reduces some control facilities compared with TCP to keep the protocol lightweight. The congestion control of CoAP simply relies on timeout to retransmit the lost packet. Because of its shortcomings, many studies have proposed an enhancement for the CoAP. However, the remaining issues and limitations have been outlined in recent studies [2] [4] [5].

In this article, we investigate two remaining issues of the CoAP: 1) Lack of support for burst data transfer and 2) Lack of transmission rate control to avoid congestion. We propose an analytical model for the CoAP burst traffic. Based on this

model, we developed a control algorithm for CoAP to control the transmission rate for burst data transfer and alleviate network congestion. The remainder of this paper is organized as follows. Section 2 presents the background and related works. In Section 3, we present the proposed model and control algorithm. Finally, Section 4 presents the simulation results. Section 5 concludes the article.

2. Background and related work

This study focuses on the reliable mode of CoAP for burst traffic. According to [3][5], the reliable mode of CoAP is similar to that of TCP. CoAP uses acknowledgment (ACK) packets to confirm the transmission of confirmable (CON) packets.

Figure 1 shows the data exchange between a CoAP sender (IoT client) and CoAP receiver (server) in reliable transport mode. Let T_s denotes the sending time of a CON packet and Ta be the receiving time of an ACK packet. The time difference T_a-T_s represents a round-trip time (RTT). RTT is the time interval between a transmitted CON packet and the received ACK for the corresponding packet sent from the receiver. The retransmission timeout (RTO) is a time variable used to check the ACK. The initial RTO for each packet is predefined between 2 s and 3 s [3].

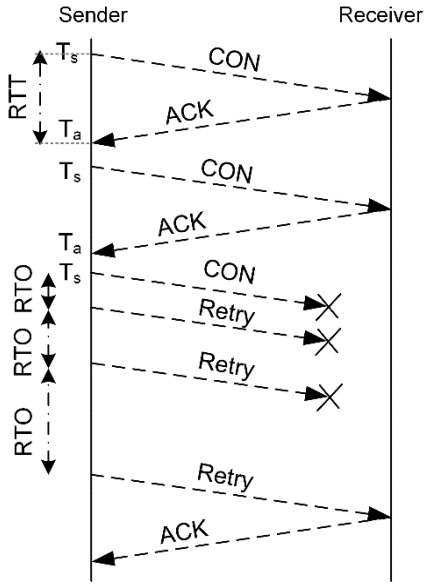

Figure 1. Data exchange between sender and receiver

The sender sends only the next CON packet after receiving the ACK from the server. If the sender does not receive an ACK for a CON packet during RTO, the CON packet is lost. The CoAP sender must initiate retransmission. Four retransmissions are allowed for each retransmitted packet. The RTO variable is doubled for each retransmission attempt, which is called the binary exponential backoff (BEB) policy. After four unsuccessful retransmission attempts, the lost packets are not retransmitted. The connection can be considered to have failed, or the sender continues to send subsequent packets. The higher layer on the server may require the sender to resend the block of lost packets. The retransmitted packets can be duplicated or disordered on the server. The higher layer discards duplicate packets and rearranges the order of the received packets. However, these issues are beyond the scope of this article.

Figure 1 shows *three* unsuccessful retransmissions of a CON packet and the last successful retransmission of a CON packet. Similar to TCP, packet loss indicates the occurrence of network congestion. In contrast to TCP, CoAP did not support burst traffic. This means that CoAP does not allow for *inflight* packets, that is, packets sent but not yet acknowledged. The CoAP sender can send only a new packet when it receives an ACK for the previous packet. As indicated in [3][5], CoAP restricts the number of concurrent packets that can be sent without receiving an ACK. Therefore, the CoAP does not support burst data transfer. Because of this limitation, CoAP exhibits poor performance if ACK packets are delayed. In this case, the sender remains in a long idle period, waiting for acknowledgment from the server. In the case of temporal packet loss, CoAP shows inefficient data transfer. The link bandwidth is wasted at long idle intervals.

The second deficiency of CoAP is the lack of rate control to avoid congestion. The simple control mechanism of CoAP is activated only when congestion occurs. The CoAP sender only adjusts the retransmission speed by halving the retransmission timeout based on the BEB. This implies that the RTO is doubled for each retransmission attempt. Thus, the RTO value plays an important role in CoAP. A large RTO value can lead to long idle delays, which causes inefficiency and poor performance. If the RTO value is small in comparison to the propagation delay, the sender can trigger the early retransmission, resulting in spurious retransmissions and an additional load for the network. Fixed RTO values do not reflect the dynamic nature of the networks, because the propagation delay can fluctuate according to the load and congestion situation in the network. The current CoAP uses fixed RTOs and ignores the changes in the round-trip time. The dynamic network conditions were not considered.

Because of these shortcomings, various studies have proposed modifying CoAP. The proposed variants of CoAP can be classified into *three* groups: 1) RTO modification [6]–[11], 2) enhancement of burst transfer [5] [12][13][14], and 3) enhancement of rate control [16]–[19].

Most studies have focused on RTO modifications for CoAP [6]–[11]. This is because a fixed RTO value of the CoAP is not suitable for dynamic network conditions. The authors in [6] proposed a dynamic update of RTO to restrict the frequency of retransmissions. Because of the variation in RTT, the authors in [7] proposed using two estimators to update RTOs. A variable backoff factor (VBF) for RTOs was used instead of the BEB policy in the CoAP. In [8], the authors proposed a small RTT multiplicative factor for computing dynamic RTOs. A probabilistic backoff factor (PBF) was proposed. A dynamic scaling factor was proposed

in [9] to estimate RTOs. The authors in [10] proposed a fuzzy logic system to compute RTOs using smooth RTT estimation and a flexible backoff mechanism. In [11], the maximum mean deviation of the RTOs was computed to avoid the impact of RTT variations and limit the overall RTO value. As presented, RTO modifications do not consider the burst transfer and rate control problems. The RTO adjustment affects only the retransmission rate not the transmission rate.

Several studies addressed the problem of burst traffic, such as [5] [12] [13] [14]. Because the basic CoAP [3] does not support burst traffic, new modifications were proposed in [5] [12]. Burst transfer has been proposed using block-wise transfers. The authors in [5] proposed an option for CoAP headers to transfer large payloads in a block-wise manner. A similar mechanism has been proposed in [12]. However, these mechanisms are only used either to separate large datagrams into blocks [5] or for unreliable data block transfer [12]. In addition, these mechanisms are used for flow control rather than for congestion control. In [13], the authors showed the problem of RTO computation for packets in a burst. A retransmission counter was proposed as an option field in the packet header to estimate the RTT for every packet of burst traffic. The burst transfer of streaming data was investigated in [14][15]. The impact of RTOs on video streaming applications was investigated.

Few studies have addressed transmission rate control for CoAP. The authors in [16] proposed a control mechanism for CoAP based on the TCP BBR (bottleneck bandwidth round-trip propagation time) protocol. This mechanism estimates the bottleneck bandwidth and round-trip propagation time to determine the new RTO and adjust the transmission rate. However, this mechanism cannot be used for burst traffic. In [17], the authors proposed a rate-based mechanism for regulating the sending rate of the CoAP sources. This mechanism is not feasible because it requires knowledge of bandwidth information along the connection path. A rate-based scheme was proposed in [18] using probe packets to estimate bottleneck bandwidth. However, the authors indicated the difficulty in estimating the bottleneck bandwidth.

3. Control algorithm for CoAP

As presented, modifications and enhancements of CoAP did not satisfy the requirements of burst data transfer and control of the transmission rate. In this section, we present the proposed control algorithm for CoAP to solve the mentioned problem. First, we present an analytical model for the CoAP burst traffic. Subsequently, we presented a control algorithm based on the proposed model.

3.1. Analytical model for CoAP burst traffic

Referring to Figure 1, the sequences of the CON and ACK packets can be described using a discrete-time model. This model was typically used in computer networks [19] [20]. In [19], Kleinrock analyzed the congestion control using queueing systems for TCP. In [20], Keshav used a discrete time model to illustrate TCP conversation over a series of network nodes in an end-to-end path. In this study, we used a discrete-time model for CoAP transactions. However, this model differs from TCP [19] [20] in various aspects. First, the TCP model describes throughput and delay as functions of the congestion window. The control decision increases or decreases window size. By contrast, the CoAP model uses inflight packets and adjusts the sending rate. The sending rate and delay are functions of the inflight packets. Second, the TCP model uses triple ACKs as indicators of packet loss. In contrast, CoAP considers ACK loss as a packet loss according to RFC 7252 [3]. Third, the control objective of the TCP model was the window size, whereas it was the sending rate in the proposed CoAP model.

Figure 2 presents the periods of sending and receiving packets for the CoAP burst traffic. Let k denote an RTT period and $T(k)$ be the time duration of this period. A CoAP sender can send several inflight packets during each period k. The sending rate was adjusted in a discrete time manner. That is, the decision on rate control can be made at the time of packet sending.

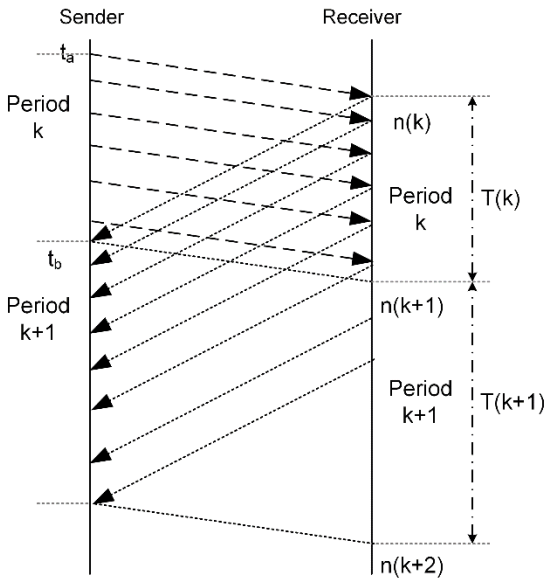

Figure 2. Periods of burst traffic

Let $\lambda(k)$ denote the sending rate during period k, $\mu(k)$ the delivery rate computed at the receiver in period k, and $T(k)$ the time interval of period k. The amount of data packets (inflight packets) transmitted in period k can be computed as follows:

$$L(k) = \lambda(k) \times T(k) \quad (1)$$

Among transmitted packets $L(k)$, there are $\mu(k) \times T(k)$ packets that have been processed by the receiver (i.e., the received packets and ACKs). Let $n(k)$ denote the instantaneous number of packets that arrive at the destination

waiting for processing. The cumulative number of packets in the next period *(k+1)* is denoted as *n(k+1)*. We have:

$$n(k+1) = n(k) + \lambda(k) \times T(k) - \mu(k) \times T(k) \quad (2)$$

From (1) and (2), we have:

$$n(k+1) = n(k) + L(k) - \mu(k) \times T(k) \quad (3)$$

From (3), we have:

$$\mu(k) = \frac{1}{T(k)}(L(k) + n(k) - n(k+1)) \quad (4)$$

Using (1) and (4), we can have:

$$\lambda(k) = \mu(k) + \frac{\Delta n}{T(k)} \quad (5)$$

where $\Delta n = n(k+1) - n(k)$ (6)

Δn represents the number of increased or decreased packets between the periods. This amount depends on the sending rate of the sender and processing capability of the receiver. According to [20], we can define a utilization factor ρ as follows:

$$\rho = \frac{\lambda}{\mu} \quad (7)$$

The system is stable if $\rho \leq 1$, which means that $\mu \geq \lambda$. That is, the sending rate must be less than or equal to the delivery rate under stable conditions. In other words, the delivery rate must be greater than or equal to the sending rate to avoid congestion. Without loss of generality, we can assume that the minimal delivery rate $\mu(k)$ at step k is equal to the sending rate $\lambda(k-1)$ at step $k-1$ because of a small time interval between $k-1$ and k. We can rewrite (5) as follows:

$$\lambda(k) = \lambda(k-1) + \frac{\Delta n}{T(k)} \quad (8)$$

The quotient $\frac{\Delta n}{T(k)}$ in (8) represents the amount of increased or decreased packets in each period k. The smallest increase is *one* if Δn is equal to *one*. If we do not want to make the control more aggressive, we can choose the value $\Delta n = 1$ for the increase of the sending rate in case of no congestion. Thus, we can rewrite (8) as follows:

$$\lambda(k) = \lambda(k-1) + \frac{1}{T(k)} \quad (9)$$

The increase of *one* packet per $T(k)$ is reasonable owing to a possible large number of inflight packets at this moment. As indicated in [21], senders can treat the network as a black box and interact with the receiver using only requests and responses. In [20], Kleinrock showed that it is possible to model an end-to-end connection in the form of a physical pipe. The diameter of the pipe describes the maximum bottleneck bandwidth for all flows. The pipe length describes the propagation delay. Intuitively, the delivery rate must be less than or equal to the maximum bottleneck bandwidth to avoid congestion.

Assume that the pipe can be described in a Cartesian coordinate system, where the x-axis represents the propagation delay, and the y-axis represents the diameter of the pipe. Let Y denote the portion of the diameter used by a CoAP flow and X be the propagation delay of the flow. The product of X and Y represents the number of inflight packets of such a flow. Thus, we can define a function that represents the number of inflight packets for each flow. Owing to the non-linear characteristics of the parameters, we must use an exponential function. Using an exponential function is the best way to model a nonlinear variable [20]. We define a utility function $U(L)$ for inflight packets as follows:

$$U(L) = \mu(L)^{-\alpha} T(L) \quad (10)$$

where $\mu(L)$ is a function of L representing the delivery rate at the receiver, L is the number of inflight packets, $T(L)$ is the delay function of L, α is a control factor, and $\alpha > 0$.

The utility function $U(L)$ represents the relationship between delivery rate and packet delay with variable L (inflight packets). Delivery rate is defined as the ratio of the number of received packet at the destination and the time unit. This ratio corresponds to the receiving flow rate. From (10), we have

$$\log(U(L)) = \log(T(L)) - \alpha \log(\mu(L)) \quad (11)$$

By taking the differential for both sides, we can have:

$$\frac{dU(L)}{U(L)} = \frac{dT(L)}{T(L)} - \alpha \frac{d\mu(L)}{\mu(L)} \quad (12)$$

The utility function $U(L)$ is maximum if its derivative is equal to *zero*. That is,

$$\frac{dU(L)}{U(L)} = \frac{dT(L)}{T(L)} - \alpha \frac{d\mu(L)}{\mu(L)} = 0 \quad (13)$$

Therefore, we have:

$$\alpha \frac{d\mu(L)}{\mu(L)} = \frac{dT(L)}{T(L)} \quad (14)$$

The quotient $\frac{d\mu(L)}{\mu(L)}$ represents the relative variation in the delivery rate, whereas the quotient $\frac{dT(L)}{T(L)}$ represents the relative variation of the packet delay with the number of inflight packets L. The value α represents a relative variation ratio of both presented quantities.

The utility function increases with the delivery rate and packet delay. This function reaches its maximum at point, as described by (14). Subsequently, the function decreases. This is the case of congestion when the number of inflight packets becomes too large. The goal of control is to limit the number of inflight packets before the maximum point of the utility function. The meaning of the control factor α is as follows:

- If $\alpha < 1$, the increase speed of the delay variation is faster than that of the delivery rate variation. The objective of the control will be in the direction of a lower delay.

- If $\alpha > 1$, the increase speed of the delay variation is slower than that of the delivery rate variation. The objective of this control will be in the direction of higher delivery rate.

- If $\alpha = 1$, the packet delay increases according to the delivery rate. The objective of this control is to maintain a balance between the delivery rate and packet delay.

Let $B(L)$ denote the number of inflight packets at the end of period k. We consider two cases: 1) without packet loss and 2) with packet loss.

In the case without packet loss, all transmitted packets *L* arrive at the destination in period *k*. The number of received packets is denoted by *B(L)*. At the maximum point of *U(L)*, we can determine the delivery rate μ(*L*) as follows:

$$\mu(L) = \frac{L}{T(L)} \quad (15)$$

Thus, from (14), we can have:

$$\alpha \frac{d\mu(L)}{dL} \frac{dL}{\mu(L)} = \frac{dT(L)}{T(L)} \quad (16)$$

$$L = \alpha T(L) \frac{dL}{dT(L)} \quad (17)$$

Because of the assumption of no packet loss, from Equation (1) we can deduce that:

$$\frac{dL}{dT(L)} = \lambda(L) \quad (18)$$

Thus, from (17) and (18), we can obtain:

$$B(L) = L = \alpha \times T(L) \times \lambda(L) \quad (19)$$

Equation (19) indicates the amount of inflight packets *B(L)* at the maximum point of the utility function in the case of no packet loss.

We now consider the case of packet loss. Suppose that packet loss occurs owing to congestion during period *k*. Let *B(L)* denote the number of inflight packets at the maximum of utility function *U(L)* in case of packet loss. The delivery rate μ(*L*) at packet loss time is determined as follows:

$$\mu(L) = \frac{B(L)}{T(L)} \quad (20)$$

By substituting μ(*L*) into (11), we have:

$$log(U(L)) = (1+\alpha)log(T(L)) - \alpha log(B(L)) \quad (21)$$

The utility function *U(L)* is maximum if its derivative is equal to *zero*. That is,

$$\frac{dU(L)}{U(L)} = (1+\alpha)\frac{dT(L)}{T(L)} - \alpha\frac{dB(L)}{B(L)} = 0 \quad (22)$$

Thus, we can compute *B(L)* as follows:

$$B(L) = \frac{\alpha}{(1+\alpha)} T(L) \frac{dB(L)}{dT(L)} \quad (23)$$

As explained above, the sending rate must be less than or equal to the delivery rate to avoid congestion. The maximum sending rate just before packet loss occurs, is determined as follows:

$$\lambda(L) = \frac{dB(L)}{dT(L)} \quad (24)$$

By substituting (24) into (23), we have:

$$B(L) = \frac{\alpha}{(1+\alpha)} T(L) \lambda(L) \quad (25)$$

By comparing (25) and (19), we can conclude that *B(L)* in case of packet loss must be less than *B(L)* in case without packet loss by a factor of $\frac{\alpha}{(1+\alpha)}$. If we choose α = 1, we have

$$\frac{\alpha}{(1+\alpha)} = 0.5 \quad (26)$$

This means that *B(L)* in case of packet loss is half of *B(L)* in case without packet loss. That is, the sending rate must be adjusted to maintain half of inflight packets to obtain a maximum of the utility function in case of packet loss. Because the same number of inflight packets occurs before and after packet loss, the sending rate must be reduced to a half in case of packet loss.

Therefore, we obtain the following control mechanism.

- Without packet loss, the CoAP sender can increase the sending rate by *one* as follows:

$$\lambda(k) = \lambda(k-1) + \frac{1}{T(k)} \quad (27)$$

- In the case of packet loss, (i.e., when congestion occurs), the CoAP sender must decrease the sending rate by half.

$$\lambda(k) = 0.5 \times \lambda(k-1) \quad (28)$$

where $\lambda(k)$ is the sending rate at step *k*, $\lambda(k-1)$ is the sending rate at the previous step *k-1*, *T(k)* is the round-trip time measured at step *k*, and *k* is the time when the sender receives an ACK. The equations (27) and (28) represent the proposed rate control mechanism for CoAP in this paper.

3.2. A rate control algorithm for CoAP

Based on the developed model, we propose a rate control algorithm for CoAP burst traffic, as follows:

Start-up phase

- The CoAP sender starts with an initialized transmission rate. This rate is unimportant, because it is replaced at the end of the start-up phase.
- During *two* estimated RTTs, the sender transmits packets and counts the number of received ACKs (*nACK*). The lost packets are not retransmitted.
- The sender updates the RTT estimation for each received ACK.
- The start-up phase is completed after *two* estimated RTTs. The transmission rate is computed as the ratio of *nACK* and *2 x RTT*. If *nACK* is equal to *zero*, the sender assumes that all transmitted packets have been lost. In this case, the sender must restart the connection.
- Subsequently, the sender enters the steady phase.

Steady phase

- The CoAP sender sends packets continuously using the computed transmission rate at the end of start-up. The sender sets an RTO for each transmitted packet.
- If RTO expires and no ACK is received for the transmitted packet, the timeout function retransmit the lost packet. *Four* retransmissions are allowed for each packet. The RTO is updated for each retransmission attempt using BEB. After *four* unsuccessful packet retransmissions, the packet is considered lost. Subsequently, the timeout function marks packet loss for loss detection.

- If packet loss is detected, the sender enters the backoff phase. In case without packet loss, the sender increases the transmission rate using Eq. (27) after each RTT.
- The steady phase repeats the subsequent loop.

Backoff phase

- At the beginning of this phase, the sender immediately reduces the transmission rate by half using Eq. (28) to avoid congestion.
- Subsequently, the sender performs a backoff loop to check ACK.
- If an ACK is received, the sender assumes that congestion has been resolved. Subsequently, the sender sends a new packet and returns to the steady phase.
- If no ACK is received, the sender assumes that the congestion remains. Accordingly, the sender must reduce the current transmission rate by half after each estimated RTT, to avoid further congestion.
- The backoff loop is repeated if the sender does not receive an ACK during the maximum transaction time, as defined in [3]. If no ACK is received when this maximum time is reached, the transaction is considered to have failed. Subsequently, the sender must restart.

4. Simulation experiments

Because of its lightweight design, the proposed control algorithm can be easily implemented in the protocol stack of an IoT device and a remote server, as shown in Figure 3. CoAP+ is the modified version of CoAP that uses the proposed control algorithm.

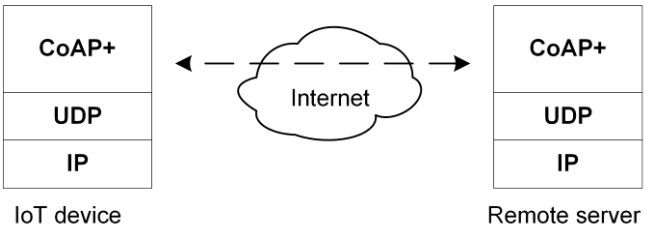

Figure 3. Implementation of the proposed algorithm

We used the Network Simulator NS-3.36 [22] for the simulation evaluation of the proposed CoAP+ and basic CoAP. All the simulation scenarios used a star network topology, as shown in Figure 4.

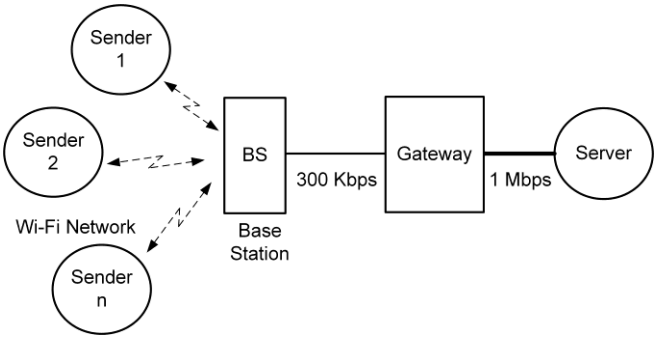

Figure 4. Simulation model

All senders, including *ten* basic CoAP and *ten* CoAP+ flows, were implemented at the wireless nodes of a Wi-Fi network. This Wi-Fi network uses a base station (BS) and is connected to the Internet through a gateway. The senders transmit the collected packets to a central server. We assume that all senders had sufficient collected data to simulate burst traffic. The Wi-Fi network was established using the standard parameters of IEEE 802.11 in NS-3 [22]. We evaluated CoAP+ and CoAP using two simulation scenarios: occasional congestion and heavy congestion. Although various link bandwidths and delays can be selected for the simulation experiments, the aim of these experiments was to compare the schemes under the same network conditions. All measured values were computed using average values for all *ten* flows. The simulation time was 1000 s. We conducted each experiment *ten* times to compute confident measurements.

Occasional congestion scenario

In these scenarios, the link bandwidth between the base station (BS) and gateway was 300 Kbps, with a delay of 70 ms. The link bandwidth between the gateway and server was 1 Mbps with a delay of 50 ms. These parameters are used to create occasional congestion conditions.

Figure 5 shows comparison of the average delays for CoAP+ and CoAP flows. As indicated, the delay in CoAP was larger than that in CoAP+ because of the congestion. Large delays were observed for CoAP during 20 and 420 s because of retransmissions. For CoAP, many packets did not arrive at the server because of timeout. CoAP retransmitted these packets using a doubled RTO at each retransmission attempt. In contrast, CoAP+ did not require retransmission owing to its rate control. The average delay was 844.76 ms with confidence intervals of (826.19, 863.33) in CoAP+ and was 18702.23 ms with confidence intervals of (17320.67, 20083.79) in CoAP, respectively. Confidence intervals were computed at a confidence level of 99%.

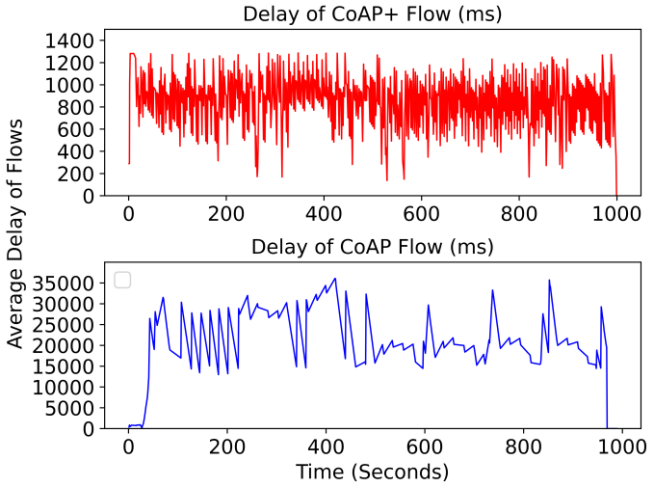

Figure 5. Average delay in occasional congestion

Figure 6 shows a comparison of the average throughputs of CoAP+ and CoAP. The throughput fluctuated owing to the occasional congestion. CoAP+ attempted to leverage bandwidth to improve its performance when congestion was resolved, whereas CoAP did not. Therefore, CoAP+ can achieve better throughput in the case of congestion. The average throughput was 0.669 Kbps with confidence intervals of (0.657, 0.680) for CoAP+, and was 0.664 Kbps with confidence intervals of (0.591, 0.737) for CoAP, respectively. The confidence intervals were computed using a confidence level of 99%.

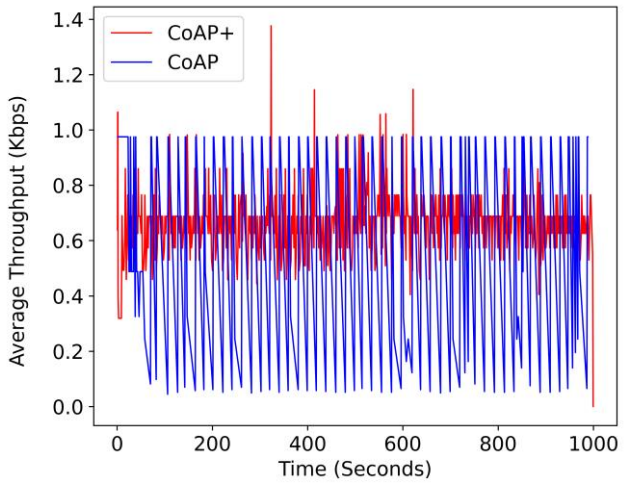

Figure 6. Throughput in occasional congestion

The results show that CoAP+ shows better performance in terms of delay, throughput, retransmission, packet duplication, and loss rate than CoAP under the same network conditions with occasional congestion. Table 1 shows a performance comparison between CoAP+ and CoAP. The average values were computed for *ten* flows for each scheme.

Table 1. Performance evaluation

Average	CoAP+	CoAP
Packets sent	703	201
Packets acknowledged	703	199
Retransmitted packets	2 (0.03%)	180 (90.81%)
Duplicated packets	0 (0.0%)	174 (87.44%)
Successful received packets	703 (100%)	195 (97.89%)
Lost packets	0 (0.0%)	2 (0.10%)
Average delay	844.76 ms	18702.33 ms
Average throughput	844.76 Kbps	0.664 Kbps

During the simulation, we classified the collected packets in the tracing data according to their types: 1) packets sent, 2) packets acknowledged (including acked and retries), 3) retransmitted packets, 4) duplicated packets, 5) successful received packets (only successful acked), and 6) lost packets. The values in Table 1 presents the average number of packets of each type for *one* flow (i.e., total packets of each type divided by 10 flows) using rounded integers. Let i denote the packet type (i=*1*: sent packets sent, i=*2*: acknowledged packets, i=*3*: retransmitted packets, i=*4*: duplicated packets, i=*5*: successful received packets, i=*6*: lost packets). Let M_i denote the total number of packets of type i from *ten* flows. All values M_i were measured during the simulation time for each scheme.

Let N_i be the average number of packets of type i of each flow. We compute the average values for each of *ten* flows as follows:

$$N_i = M_i / 10 \quad (29)$$

Let X_i denote the percentage of measured packets of type i for each of *ten* flows, we have:

$$X_i = \frac{N_i}{N_2} \times 100\% \quad \text{with } i = 3,4,5,6 \quad (30)$$

where N_2 is the average number of acknowledged packets for *one* of *ten* flows.

In this experiment, the number of packets measured for CoAP+ was as follows: M_i = {*7036*, *7036*, *2*, *0*, *7036*, *0*} with i = *1,2,…,6*. Therefore, we have N_i = {*703.6*, *703.6*, *0.2*, *0*, *703.6*, *0*} with i = *1,…,6*. Accordingly, we have X_i = {*0.03%*, *0.0%*, *100%*, *0.0%*} with i = *3,4,5,6* for the CoAP+ scheme. In the CoAP, the number of packets was measured as follows: M_i = {*2012*, *1991*, *1808*, *1741*, *1949*, *20*} with i = *1,…,6*. Therefore, we have N_i = {*201.2*, *199.1*, *180.8*, *174.1*, *194.9*, *2*} with i = *1,2,…,6*. Accordingly, we have X_i = {*90.81%*, *87.44%*, *97.89%*, *0.10%*} with i = *3,4,5,6*. Note that Table 1 shows the rounded integers of the average values for a flow.

Heavy congestion scenarios

In these scenarios, the link bandwidth between the base station (BS) and gateway was 300 Kbps, with a delay of 70 ms. The link bandwidth between the gateway and server was 1 Mbps with a delay of 140 ms. This link delay was doubled

to cause a large RTT resulting in a high likelihood of heavy congestion.

Figure 7 shows a delay comparison between CoAP+ and CoAP. As indicated, the CoAP+ and CoAP flows became congested quickly after the start-up. The CoAP had more delayed packets than CoAP+ because more packets were required to be retransmitted in CoAP. In contrast, CoAP+ was quickly recovered after congestion. Therefore, CoAP+ had fewer delayed packets.

Large delays and retransmissions were observed for CoAP during 0 and 270 s because of the high number of packet losses. In contrast, CoAP+ had fewer retransmissions owing to its rate control. The average delay was 3651.53 ms with confidence intervals of (2606.36, 4696.35) in CoAP+ and was 32207.25 ms with confidence intervals of (28878.25, 355536.24) in CoAP, respectively. All confidence intervals were computed with a level of 99%.

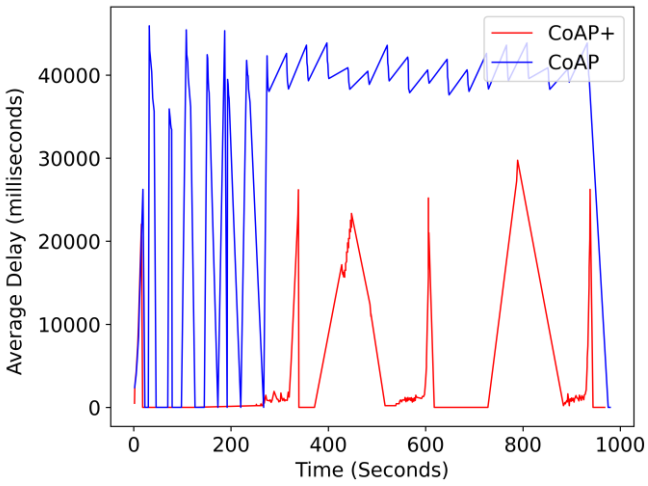

Figure 7. Average delay in heavy congestion

Figure 6 shows a throughput comparison of CoAP+ and CoAP under heavy-congestion conditions. CoAP+ flows attempted to leverage bandwidth to improve the performance when congestion was resolved. In contrast, the CoAP flows were not controlled to alleviate congestion. The average throughput was 0.668 Kbps with confidence intervals of (0.533, 0.802) for CoAP+ and 0.667 Kbps with confidence intervals of (0.592, 0.763) in CoAP, respectively. Confidence intervals were computed at a confidence level of 99%.

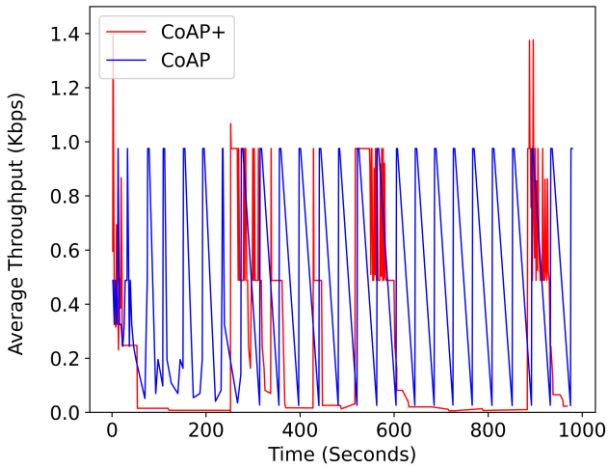

Figure 7. Throughput in heavy congestion

Note that, although the average throughput was the same for both CoAP+ and CoAP, CoAP+ sent more packets than CoAP. The number of packets sent was 227 for CoAP+, and 161 for CoAP. CoAP had more retransmitted and duplicated packets than those of CoAP+. CoAP+ successfully received 166 packets, whereas CoAP had only 124 successful received packets. CoAP+ had a higher number of lost packets than CoAP because CoAP+ sent more packets than CoAP under the same conditions. Although CoAP received a high number of packets, most of them were retransmitted (86.25%) and duplicated (86,25%). Table 2 shows a performance evaluation of CoAP+ and CoAP under heavy congestion conditions.

Table 2. Performance evaluation in heavy congestion

Average	**CoAP+**	**CoAP**
Packets sent	227	161
Packets acknowledged	226	160
Retransmitted packets	93 (41.15%)	138 (86.25%)
Duplicated packets	78 (34.51%)	138 (86.25%)
Successful received packets	166 (73.45%)	124 (77.50%)
Lost packets	57 (25.22%)	31 (19.38%)
Average delay	3651.33 ms	32207.25 ms
Average throughput	0.668 Kbps	0.667 Kbps

The values in Table 2 were calculated similar to Table 1 using equation (29) and (30). All packets were collected during the simulation time according to their types and divided by the number of flows (i.e., by *ten*) to calculate the average value for a flow. The values reflected the real number of packets that were traced during the simulation. In this experiment, the measured average values in CoAP+ were: N_i = {*227, 226, 93, 78, 166, 57*} with i = *1,2,...,6.* Accordingly, we have X_i = {*41.15%, 34.51%, 73.45%, 25.22%*} with i = *3,4,5,6* for the CoAP+ scheme. In the CoAP, the measured average values were: N_i = {*161, 160, 138, 138, 124, 31*} with i = *1,2,...,6.* Accordingly, we have X_i = {*86.25%, 86.25%, 77.50%, 19.38%*} with i = *3,4,5,6.*

All values were rounded integers. Note that, some packets might not reach the destination because of their long delays and they were dropped when the simulation ended.

The results indicate that CoAP+ exhibits better performance in terms of delay, throughput, retransmission, packet duplication, and received packets than CoAP under the same network conditions with heavy congestion conditions.

5. Conclusion

Congestion is an important issue in IoT networks with constrained devices and a growing number of applications. This paper investigates the problem of congestion control for CoAP burst traffic. Because the current CoAP uses a simple congestion algorithm, it does not support burst data transfer and lacks rate control to avoid congestion. By analyzing the shortcomings of CoAP, this paper proposes an analytical model for CoAP burst traffic and a rate control algorithm. The proposed CoAP+ algorithm provides a suitable solution by controlling the transmission rate to avoid congestion. The simulation results showed that CoAP+ provides better performance for burst traffic than the current CoAP under the same network conditions.

Future studies can investigate bottleneck bandwidths to detect congestion early for advanced congestion control.

References

[1] Gomez C., Archia-Moret A., Crowcroft J., et.al., TCP in the Internet of Things: from ostracism to prominence, *IEEE Internet Computing*, 2018, vol. 22, Issue 1, pp. 29-41.

[2] Tariq M.A., Khan M., Khan M.T.R., Kim D., Enhancements and Challenges in CoAP–A Survey, *Sensors*, 2020, DOI: 10.3390/s20216391, vol. 20, 2020 (6391), pp. 1-29.

[3] RFC 7252, The Constrained Application Protocol (CoAP), available: https://rfc-editor.org/info/rfc7252.

[4] Haile H., Grinnemo K., Ferlin S., et.al., End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks, *Computer Networks*, 2021, vol. 186.

[5] Bormann C., Shelby Z., Block–Wise Transfers in the Constrained Application Protocol (CoAP), [Online]. Available: https://rfc-editor.org/info/rfc7959.

[6] Betzler A., Gomez C., Demirkol I., Paradells J., CoAP congestion control for the internet of things, *IEEE Commun. Mag.*, 2016, vol. 54, no. 7, pp. 154–160.

[7] Bormann C., Betzler A., Gomez C., Demirkol I., CoAP Simple Congestion Control/Advanced, Internet-Draft, Feb. 2018. [Online]. Available: https://tools.ietf.org/id/draft-bormann-core-cocoa-03.txt.

[8] Betzler A., Gomez C., Demirkol I., Paradells J., CoCoA+: An advanced congestion control mechanism for CoAP, *Ad Hoc Netw.*, Oct 2015, vol. 33, pp. 126–139.

[9] Deshmukh S., Raisinghani V.T., AdCoCoA–Adaptive Congestion Control Algorithm for CoAP, in *Proc. of 11th IEEE Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, Jul. 2020, pp. 1-7.

[10] Aimtongkham P., Horkaew P., So-In C., An Enhanced CoAP Scheme Using Fuzzy Logic with Adaptive Timeout for IoT Congestion Control, *IEEE Access*, Apr. 2021, vol. 9, pp.58967-58981.

[11] Bolettieri S., Tanganelli G., Vallati C., Mingozzi E., pCoCoA: A precise congestion control algorithm for CoAP, *Ad hoc Network*, Nov. 2018, vol. 80, pp.116-139.

[12] Boucadair M., Shallow J., Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission, Internet-Draft, May 2021. [Online]. https://tools.ietf.org/id/draft-ietf-core-new-block-14.

[13] Lee J.J., Kim K.T., Youn H.Y., Enhancement of congestion control of Constrained Application Protocol/Congestion Control/Advanced for Internet of Things environment, *Int. J. of Distributed Sensor Networks*, Nov. 2016, vol. 12 (11), pp. 1-13

[14] Rahman W.U., Choi Y.S., Chung K., Performance Evaluation of Video Streaming Application Over CoAP in IoT, *IEEE Access*, Apr. 2019, vol. 9, pp.39852-39861.

[15] Jung J.H., Gohar M., Koh S.J., CoAP–Based Streaming Control for IoT Applications, *Electronics*, Aug. 2020, vol. 9 (8) 1320, DOI: 10.3390/electronics9081320, pp. 2-19.

[16] Ancillotti E., Bruno R., BDP–CoAP: Leveraging Bandwidth-Delay Product for Congestion Control in CoAP, *in Proc. of 5th IEEE World Forum on Internet of Things (WF-IoT)*, Ireland, Apr. 2019, pp. 656-661.

[17] Ancillotti E., Bruno R., Vallati C., Mingozzi E., Design and Evaluation of a Rate–Based Congestion Control Mechanism in CoAP for IoT Applications, *in Proc. 19th IEEE Int. Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Greece, Jun. 2018, pp. 14–15.

[18] Hoang D.H., Le T.T.D., RCOAP: A Rate Control Scheme for Reliable Bursty Data Transfer in IoT Networks, *IEEE Access*, 2021, vol. 9, doi: 10.1109/ ACCESS.2021. 3135435, pp. 169281-169298.

[19] Kleinrock L., Internet congestion control using the power metric: Keep the pipe justfull, but no fuller, *Ad Hoc Networks*, 2018, 05-015, pp.1-16.

[20] Keshav S., A Control-theoretic Approach to Flow Control, in *ACM SIGCOMM, Computer Communication Review*, Sept. 1991, Vol. 21, Issue 4, pp 3–15.

[21] Jain R., A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks, *CM SIGCOMM Computer Communication Review*, Oct. 1989, Volume 19, Issue 5, pp 56–71.

[22] NS-3 Network Simulator, version 3.36, available: https://www.nsnam.org