

Received November 23, 2021, accepted December 9, 2021, date of publication December 13, 2021, date of current version December 31, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3135435

RCOAP: A Rate Control Scheme for Reliable Bursty Data Transfer in IoT Networks

DANG HAI HOANG¹ AND THI THUY DUONG LE²

¹Posts and Telecommunications Institute of Technology, Hanoi, Vietnam

²University of Civil Engineering, Hanoi, Vietnam

Corresponding author: Dang Hai Hoang (haihd@ptit.edu.vn)

ABSTRACT The Internet Engineering Task Force (IETF) standardized the Constrained Application Protocol (CoAP) for Internet of Things (IoT) devices to meet the demands of IoT applications. Due to the constrained IoT environment, CoAP was designed based on UDP as a lightweight protocol with simple congestion control, which leverages the basic binary exponential backoff. However, the basic congestion control of CoAP is unable to effectively perform reliable bursty data transfer in IoT networks. Recent studies have indicated that CoAP and its modifications still suffer from critical performance problems regarding congestion control, throughput, and delay. The current congestion control of the CoAP does not support bursty data transfer. In contrast to the current schemes that focus on a loss-based mechanism and a retransmission time-out (RTO) calculation, we propose a new rate control scheme, RCOAP, for reliable bursty data transfer in IoT networks. RCOAP uses the concept of regulating the transmission rate of CoAP sources. The key features of RCOAP are 1) estimating the initial sending rate by probing the bottleneck bandwidth, 2) adjusting the sending rate according to the dynamic network condition, and 3) distinguishing between losses due to congestion and losses due to wireless errors for the purpose of maintaining high throughput. Simulation results indicate that RCOAP is suitable for bursty data transfer. RCOAP shows a throughput increase of approximately 135% compared to the basic CoAP, CoCoA, and CoCoA+ under the same conditions while maintaining a low delay, loss rate, and a low number of retransmission attempts.

INDEX TERMS Constrained application protocol, congestion control, Internet of Things, rate control.


I. INTRODUCTION

Internet of Things (IoT) networks are becoming increasingly significant in different types of applications, such as healthcare, agriculture, environment monitoring, and automation. A typical IoT network consists of several resource-constrained IoT devices connected to a remote server. The primary task of IoT devices is to collect data from the physical environment and send the requested data to the server. Today, many IoT applications rely not only on the occasional transmission of small payloads from IoT devices but also must transfer large blocks of collected data to the server, resulting in an enormous amount of traffic in the network [1], [2]. Transmission control of such traffic is inevitable for alleviating the network congestion. Popular applications of IoT networks often require the transfer of a large amount of collected data to the center for further processing. Due to such bursty

traffic, the network can become overloaded, and packets can be dropped due to network congestion. The retransmission of lost packets can introduce further congestion. Without proper control, the network performance can become worse and undesirably degraded.

IoT networks are characterized by constrained and error-prone environments. Typical IoT devices have limited resources and processing capabilities. Furthermore, these devices are typically employed in high bit error rate environments with lossy communication links. Traditional reliable data transfer protocols such as TCP cannot be directly applied to IoT networks because of the different characteristics of such networks, which means that we must develop new data transfer protocols.

To fulfill the requirements of constrained devices and IoT applications, the Internet Engineering Task Force (IETF) has standardized the Constrained Application Protocol (CoAP) [3] for constrained IoT devices operating in lossy environments. CoAP works on top of the unreliable User

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Maaz Rehan .

Datagram Protocol (UDP) and is considered to be a lightweight version of HTTP for use in IoT devices. Similar to the Transmission Control Protocol (TCP), CoAP supports reliable connection-oriented data transfer using acknowledgment (ACK) messages. However, CoAP implements a lightweight reliability mechanism, without attempting to re-create the full feature set of a transport such as TCP [3]. The simple congestion control of CoAP is based on a retransmission timeout (RTO) with a binary exponential backoff (BEB). The basic design of the RTO in CoAP does not consider dynamic round-trip times (RTT). CoAP defines a fixed RTO value of 2 - 3 s. Thus, the basic CoAP does not reflect the dynamic network conditions and does not consider possible congestion threats. The RTT can change rapidly depending on the congestion in the network. Furthermore, a fixed RTO can lead to long idle delays that cause inefficiency and undesirable performance degradation.

To overcome the problems in the basic CoAP, another congestion control mechanism called CoAP Congestion Control/Advance (CoCoA) is under standardization [4], [5]. CoCoA attempts to remedy the basic congestion control of CoAP by exploiting continuous measurements of the dynamic round-trip times (RTT) to limit the frequency of retransmissions. The RTO values were adjusted based on the RTT measurements. Furthermore, CoCoA uses a variable backoff factor (VBF) and an RTO aging mechanism instead of a BEB for the RTO calculation. CoCoA has a weakness in the RTO estimator, which results in spurious retransmissions and additional network congestion. An enhancement of CoCoA, namely CoCoA+, was proposed in [6] to overcome the shortcomings of the basic CoAP and CoCoA in adjusting the RTO values. In fact, CoCoA+ is similar to CoCoA except for two key differences: 1) a modification of the RTO calculation using a smaller factor that helps to reduce the impact of high fluctuations of RTT on the RTO estimators, and 2) a new aging policy for the RTO values. Although CoCoA+ resolves the problem of weak RTO estimation in CoCoA and can provide better performance than the basic CoAP and CoCoA, recent studies such as [7]–[10] highlighted the drawbacks of CoCoA+, including constant RTO settings and poor performance in the case of small RTTs.

Due to the shortcomings of basic CoAP and its modifications, various enhancements have been proposed in recent years. Most of the proposals focused on the problem of choosing a correct RTO for retransmissions. The authors in [9] introduced several modifications to the RTO estimation. The maximum mean deviation of the RTO was calculated to avoid the impact of the RTT variations. A delay gradient-based congestion control was proposed in [6]. The proposed method predicts congestion using per-packet RTT measurements and the RTT gradient over time. This method introduces a probabilistic backoff factor (PBF) for adjusting the overall RTO. To overcome the fixed scaling values of the RTO, the authors in [7] proposed a dynamic scaling factor for estimating the RTO. The dynamic factors were calculated based on the difference between the measured RTT and the current RTT.

The RTO is estimated to be sufficiently large to minimize retransmissions [7]. The authors in [8] proposed the use of a fuzzy logic system for adjusting the RTO using a flexible backoff mechanism.

Until now, only a few authors have investigated the performance of CoAP in bursty data transfers. The authors in [11] introduced the retransmission counter as an option field in the message to estimate the RTT for every packet of burst traffic. In [12], the authors suggested a congestion control algorithm for CoAP based on the TCP BBR (bottleneck bandwidth round-trip propagation time) protocol. The BDP-CoAP mechanism [12] estimates the bottleneck bandwidth and round-trip propagation time to cope with lossy links and short-term unfairness. The paper in [13] proposed a rate-based approach for regulating the sending rate of CoAP sources. In this method, a data rate is allocated to each CoAP client based on a throughput estimation of the upward route. The authors in [14] evaluated the performance of a basic CoAP for video streaming applications. The experiments conducted by these authors indicated that the default RTO values have a significant impact on the video streaming throughput. The performance can be improved by tuning the RTO values according to the dynamic network conditions. Congestion control for reliable bursty data transfer remains a hot research topic.

With regard to another aspect, the stop-and-wait mechanism of CoAP, CoCoA, CoCoA+, and their modifications is not suitable for bursty data transfers. The most common design fixes the maximum number of outstanding CoAP transactions via a value, called $NSTART = 1$. This mechanism limits the concurrent number of packets that can be sent by a CoAP sender without receiving an acknowledgment, i.e., the number of *in-flight* packets. An extension of the current CoAP for block-wise transfers was addressed in [1], which provides an option to transfer large payloads in a block-wise fashion. Another extension of CoAP blockwise transfers is under standardization [15]. This specification is similar to the options defined in [1], but it is only used for unreliable data transfer.

In contrast to most current schemes, which have focused on loss-based approaches by adjusting RTO values for controlling the retransmission, this study considers an alternative approach using a rate-based mechanism. Specifically, we propose a new rate control scheme, called RCOAP, for reliable bursty data transfer. The key idea of RCOAP is to enable an amount of *in-flight* packets according to the available bandwidth of the bottleneck link. At the beginning of each new connection, an RCOAP sender sends several discovery packets to probe the bottleneck bandwidth. By discovering the link bandwidth, a max-min fair allocated rate can be determined. The initial sending rate is estimated based on the number of acknowledgments and variable round-trip delays. During the connection, the RCOAP sender regulates the transmission rate according to the dynamic network conditions. If a packet loss is detected, the sender probes the wireless channel to distinguish between losses due to wireless errors and congestion

losses. If the network is congested, RCOAP decreases the transmission by halving its rate. In the case of temporal wireless link errors, the rate does not have to be decreased and can be restored when the link has been recovered. RCOAP attempts to increase the transmission rate until a permitted maximum rate as long as no network congestion is detected. Our simulations demonstrated that RCOAP is effective for bursty data transfer in comparison with other schemes.

The remainder of this paper is organized as follows. Section II describes the background and related work. Section III presents the details of the proposed scheme. The evaluation results are presented in Section IV. Finally, Section V concludes the paper.

II. BACKGROUND AND RELATED WORK

For the sake of clarity, this section presents the key features of the basic CoAP and its modifications for congestion control.

The basic congestion control of the CoAP is illustrated in Fig. 1 and 2 for confirmable messages. CoAP uses a simple stop-and-wait mechanism for reliable transmission. After transmitting a message, the CoAP sender waits for the corresponding ACK message. RTT is the round-trip time, i.e., the time interval between a transmitted CON message and a received ACK from the receiver. The sender sends only the next CON message after receiving an ACK. The initial RTO is a randomly selected fixed value between 2 and 3 s. If the sender does not receive an ACK within the RTO (Fig. 2), the retransmission is triggered. For each retransmission, this value is doubled using a binary exponential backoff (BEB) to avoid congestion. Four retransmissions are allowed. After that, the exchange is considered to have failed.

Let the inter-packet interval (*IPI*) denote the time interval between two successive CON messages. In principle, the *IPI* can be smaller or larger than the RTT. If the *IPI* is smaller than the RTT, i.e., the next CON message is sent before the sender receives an ACK, then an outstanding transaction occurs. Note that the CoAP message and CoAP packet are interchangeable notations in this paper. A CoAP packet (or CoAP message) consists of a header of four bytes plus option fields and a payload, as explained in [3].

According to [3], the simple congestion control of a CoAP restricts the number of concurrent messages that can be sent without receiving an ACK. The maximum number of outstanding interactions is limited by a fixed value named NSTART (the default value is one). As recommended in RFC 7252 [3], the problem of outstanding transactions is left open for further development. Because the number of outstanding packets (i.e., packets *in flight*) is limited, the stop-and-wait mechanism of CoAP is inefficient for bursty data transfer.

The simple congestion control of CoAP ignores the possible changes in round-trip times that reflect the dynamic network conditions and possible congestion threats. To overcome the problem of fixed RTO values, various studies have focused on proposing a modification for adaptive RTO values. The CoCoA algorithm [4], [5] was proposed to remedy the basic congestion control of CoAP. CoCoA adopts

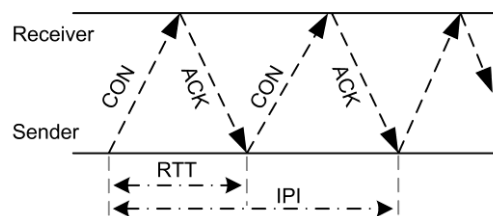


FIGURE 1. Exchange of CON and ACK messages in the basic CoAP.

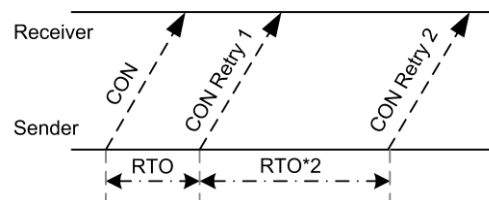


FIGURE 2. Retransmissions in the basic CoAP.

continuous measurements of the round-trip times (RTT) to limit the frequency of retransmissions. Namely, two RTT estimators are used by CoCoA, a “strong RTO estimator” and a “weak RTO estimator” [4], [5]. The first estimator is for transmissions without retransmission, and the second estimator is for transmissions that require retransmissions. The RTO values were adjusted based on the RTT measurements. The overall RTOs are calculated using an exponential weighted moving average of the “strong” and the “weak” RTO estimators. Furthermore, CoCoA uses a variable backoff factor (VBF) instead of a BEB for RTO calculation. Three values of VBF are applied depending on the values of RTO, as follows: 1) VBF = 3 for RTO less than 1 second, 2) VBF = 2 for RTO between 1 and 3 seconds, and 3) VBF = 1.5 for RTO greater than 3 seconds. Using this RTO aging mechanism, CoCoA outperformed basic CoAP. However, there is ambiguity in the calculation of the weak RTO estimator. The overall RTO values can be overestimated, resulting in unnecessarily long idle delays. If the initial RTO values are low or exceed the default RTO of 2 s, all retransmissions can be initiated within a short time, resulting in additional congestion.

The authors in [6] showed several shortcomings of CoCoA and proposed an enhancement called CoCoA+. Using a smaller RTT variance multiplication factor, CoCoA+ reduces the impact of the weak RTT estimator in the calculation of the overall RTOs. However, this calculation is limited to only the first transmission and first retransmission. CoCoA+ is unable to select the correct RTO value for bursty traffic [10]. Inaccurate measurements of RTTs can lead to spurious retransmissions. The authors in [9]–[12] indicated that CoCoA+ performed significantly worse than the basic CoAP in various network conditions, especially for bursty traffic.

Choosing a right RTO value is a problem in dynamic network scenarios. Most recent studies have focused on dynamic RTO estimation rather than on fixed RTO values. The authors in [6] proposed a delay gradient approach using a gradient of RTT over time and a probabilistic backoff factor (PBF) for adjusting the overall RTO. The remaining issue of this

mechanism is that the retransmission can occur quickly in many network scenarios, especially in the case of small RTTs and bursty traffic. A dynamic scaling factor was proposed in [7] for estimating the RTO. The proposed method attempts to minimize retransmissions based on a large RTO estimation. Because the estimation of the RTO values depends on the calculation of the dynamic scaling factors, this scheme could have a problem with quick changes in the RTT values. This problem can lead to large overall RTOs and long idle delays. If the RTO value is large, the sender cannot receive further ACKs. This circumstance in turn affects the estimation of the correct RTT values. The authors in [8] proposed a fuzzy logic system for adjusting the RTO using a smooth RTT estimation and a flexible backoff mechanism. The relative strength was defined as the ratio of the *strong* and the *weak* RTO estimators. The RTT was determined using the number of ACKs. The number change is used as the threshold for detecting congestion. The RTO value is estimated with trend adjustment, which is based on the prediction of the difference between the RTTs in consecutive intervals. The major problem is that RTT values fluctuate. It is difficult to keep track of quick RTT changes. Although the RTO can be flexibly adjusted, this mechanism is unable to predict the temporal loss conditions. A failed estimation of RTO could lead to spurious retransmission, which has a large impact on the performance in the case of bursty traffic. Another method for precise RTO estimation was proposed in [9]. This method is based on the maximum mean deviation of the RTO to avoid fluctuations in the RTT and to limit the overall RTO value.

A comprehensive survey of CoAP congestion control mechanisms is presented in [10]. The survey highlighted the shortcomings of basic CoAP and its enhancements. Existing challenges and issues were also discussed. One of the most important challenges is to provide congestion control in IoT networks. Performing complex calculations for RTTs as well as RTOs is inefficient in a constrained network environment. The problem of bursty data transfer has not been adequately investigated.

Only a few studies have addressed the problem of bursty traffic. As presented in [11], existing schemes still have limitations in calculating RTOs when bursty traffic exists. The problem is that the RTT is not correctly identified. The scheme proposed in [11] utilizes the retransmission counter as an option field in the message to estimate the RTT for every packet of burst traffic. This method can only improve the accuracy of RTO estimations. A rate-based control method was proposed in [12], which leverages the mechanism for estimating the bottleneck bandwidth and round-trip propagation time. The proposed BDP-CoAP [12] uses an RTO estimator for the delivery rate measurements for bottleneck bandwidth estimation. The estimator is adjusted either more or less aggressively to vary the rate of transmission accordingly. To the best of our knowledge, this study is the first work that has addressed the *in-flight* packets with regard to CoAP. However, the problem of BDP-CoAP is the overestimation of the available bandwidth, which results in inefficient

performance, especially under a high traffic load. Furthermore, this method regulates the rate based on only a bottleneck bandwidth estimator and an RTO estimator. Another rate-based scheme was proposed in [13], which uses throughput estimation to regulate the sending rate of CoAP sources. The authors investigated the performance in the case of light and bursty traffic conditions and the problem of unfair bandwidth allocation in different traffic scenarios. However, it is difficult to estimate the link capacity under dynamic network conditions. Wrong allocation can lead to inaccurate allocation of the transmission rate. The problem of high RTO values for video streaming applications was investigated in [14]. The authors indicated that high RTO values in the current CoAP variants have a considerable impact on streaming throughput. For bursty traffic such as streaming applications, a greater NSTART value for CoAP should be considered.

Based on the stop-and-wait mechanism, current CoAP schemes are not suitable for bursty traffic. Several extensions to the current CoAP specification have been proposed, such as those in [1], [15]. The scheme in [1] provides an option for transferring large payloads in a block-wise manner. Another study on CoAP block-wise transfers is under standardization [15]. However, these schemes are either only for unreliable data transfer [15] or are based on separating large datagrams into blocks [1]. A streaming control scheme was proposed in [16], which focused on error handling and flow control. The scheme proposed a new CoAP option field in the CoAP header to store the message sequence number. This number is used to detect losses and generate the ACKs. The number of ACKs depends on the sending buffer of the sender. In this way, the scheme can reduce the number of ACK messages. Thus, it can improve the performance gain. According to our analysis, congestion control for reliable bursty data transfer has not been adequately addressed.

The issue of congestion control (CC) has been long studied. According to surveys in [17]–[19], existing CC schemes can be classified into two types: 1) end-to-end CC and 2) network-assisted CC. End-to-end approaches only rely on communication between senders and receivers, and they do not require any information from the network nodes. Examples are several TCP variants [20] and the CoAP [3]. Network-assisted approaches require the support of network nodes in a hop-by-hop manner, or the support of lower layers such as the medium access control (MAC) layer. Examples are schemes using explicit congestion notification (ECN) [21]–[23], or mechanisms at the MAC layer [13], [24]–[26].

Loss detection is a critical issue for any CC scheme. At the MAC layer, various mechanisms can be employed to handle the frame losses such as frame acknowledgment, retransmission, retransmission timeout, and backoff [24], [27]–[29]. Using the end-to-end approach, we avoid some issues of interaction with the MAC layer as indicated in [4], [6], [19] to keep CoAP lightweight. The loss detection in end-to-end approaches is mainly based on the RTTs, the transmission delay, and the ACKs from the end receiver. The issues of the MAC layer and its interactions are out of scope of this work.

For end-to-end approaches, the problem of distinguishing wireless errors from congestion losses has been early investigated [30], [31]. Different methods have been proposed to solve this issue [30]–[37]. According to [32], there are five approaches for the differentiation of packet loss reasons: 1) using RTT statistics and throughput [30]–[32], 2) using the RTT variation and the measured delay of ACKs [33], 3) using the inter-arrival time of successive packets to predict the expected arrivals at the receiver [34], 4) using the difference between the sender rate and the rate of the corresponding received ACKs [35], and 5) using relative one-way trip time (ROTT) [36].

In this paper, we focus on the end-to-end approach. The key idea of this paper is to enable in-flight packets for reliable bursty data transfer by adding some functionalities to the basic CoAP on top of UDP.

III. RCOAP: A RATE-BASED CONTROL SCHEME

As explained in Section II, the basic CoAP and its modifications are mainly based on the stop-and-wait mechanism. This mechanism defines basic window-based congestion control, where a fixed window size, called NSTART, is defined by a default value of *one* [3]. On the other hand, the sender rate plays a key role in congestion control. If congestion is detected, the sender rate should be decreased to avoid further congestion. CoAP and its variants use a loss-based mechanism to detect congestion, i.e., the loss of ACK messages, and trigger retransmission based on the RTO adjustment and a backoff algorithm. Current schemes focus only on the retransmission rate (i.e., on the RTO calculation).

The amount of data *in flight* (i.e., the packets in transit in the network not yet acknowledged) is still insufficiently considered. This amount allows a higher transmission for bursty traffic and should be bounded to the bandwidth-delay product. Typically, many applications in IoT networks generate data by monitoring events and sending data in bulk to the center. Depending on the dynamic network condition, this data stream can lead to bursty data traffic. For these reasons, rate-based control is desirable to adapt the sending rate to the dynamic condition of the networks [13]. On the other hand, appropriate congestion control must distinguish between wireless losses (i.e., losses due to wireless link errors) and congestion losses. The reduction of the sending rate in the case of temporal wireless losses can lead to undesirable performance degradation [16]. These problems have not been fully investigated in previous studies.

As we discussed in the previous sections, the basic CoAP and its variants are not suitable for bursty traffic due to the fixed window size with $NSTART = 1$. Suppose that the sender has collected 100 packets to send. Using $NSTART = 1$ and the default leisure time of 5 s [3], the sender would require around 500 s to complete the bursty data transfer of 100 packets using the stop-and-go mechanism. If the window is larger than one packet, e.g. using $NSTART = 2$, the transfer time can be reduced to 250 s in the case of a fixed sending rate. If the sender rate could be higher adjusted as in RCOAP,

the completion time will be much shorter. Thus, bursty data transfer using RCOAP might benefit from dynamic rate control for in-flight packets.

In this section, we present a new rate-based control scheme, called RCOAP, for reliable bursty data transfer in IoT networks. The RCOAP is an extension of the CoAP over the UDP with transmission rate control. Our scheme has the following key features:

- 1) At the beginning of the connection, an RCOAP sender probes the available bandwidth by sending discovery packets to estimate the initial transmission rate.
- 2) During the connection, the RCOAP sender adjusts the transmission rate based on the current round-trip delay and packet losses. The number of *in-flight* packets is regulated according to the bandwidth-delay product, i.e., the capacity of the bottleneck link in the network connection and the variable round-trip delay.
- 3) If packet loss is detected, the RCOAP sender probes the transmission link to distinguish between losses due to wireless errors and congestion losses.
- 4) The transmission rate is increased if the network is not congested, is halved in the case of congestion, and is recovered in the case of temporal wireless errors.

The architecture of RCOAP is shown in Fig. 3. RCOAP is mainly implemented at the sender, but some additional functionalities are required at the receiver to support the acknowledgment and the detection of losses. All key features are implemented at the CoAP layer on top of UDP.

The most important problems for any congestion control (CC) scheme are: 1) how to control the congestion according to the bandwidth bottleneck in the network? and 2) how to deal with the packet losses? Two CC approaches have been studied in the community: 1) end-to-end approaches without any support of network nodes, and 2) network-assisted approaches using the support of other nodes or lower layers in a hop-by-hop fashion [17]–[19]. Examples of end-to-end CC are the basic TCP and its variants [20], and the basic CoAP [3]. Examples of network-assisted CC approaches are active queue management, explicit congestion notification (ECN), and medium access control (MAC) mechanisms [21]–[26]. Various MAC layer functionalities are used to support the hop-by-hop transmission, such as wireless error handling, ACK frames, retransmission, transmission timeout, and backoff timer [24], [27]–[29]. Hop-by-hop MAC mechanisms can handle frame losses and retransmit lost frames using explicit ACKs and retransmission timeouts. However, since we want to keep RCOAP lightweight similar to the basic CoAP on top of UDP, we neglect the interaction with the MAC layer in this work and only focus on the CoAP layer.

At the transport layer, there are different approaches to estimate the bottleneck bandwidth and differentiate the losses [30]–[37]. The authors in [32] reviewed various end-to-end solutions for detection and differentiation of losses. In [30], the authors proposed several loss predictors based on

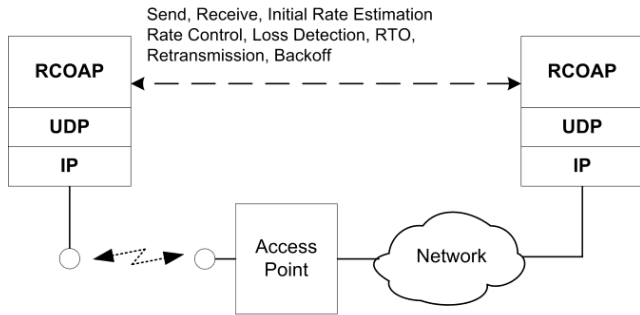


FIGURE 3. The overall architecture of RCOAP.

RTTs and/or throughput. The paper [31] indicated that RTT and bottleneck bandwidth are the most important parameters, regardless of how many links a connection traverses. The authors in [33] indicated the possibility to infer losses due to wireless errors only based on RTOs and RTT measurements. The paper [34] suggested using the receive packet time interval (RPTI) to identify packet loss types. The authors in [35] presented various end-to-end loss discriminators and proposed a mechanism for marking *in* and *out* packets. The authors in [36] proposed using spikes in relative one-way trip time (ROTT) to distinguish between wireless losses and congestion losses. As indicated in [10], [37], existing end-to-end CC schemes mainly focus on the measurement of RTTs and RTOs according to the network conditions.

RCOAP is developed based on end-to-end approaches. In our scheme, we use some discovery packets to probe the bottleneck bandwidth and distinguish congestion losses from wireless losses. Discovery packets are only used once during the first estimated RTT of the connection for estimating the initial rate. The detection of losses is based on the measured RTTs and the inter-arrival of ACK packets at the sender.

The RCOAP consists of four states: 1) initial, 2) normal, 3) loss-detection, and 4) backoff, as shown in Fig. 4.

A. INITIAL STATE

The RCOAP starts the initial state at the beginning of a new connection. The purpose of this state is to determine the initial transmission rate for the RCOAP transaction. There are various techniques for measuring the available link bandwidth, e.g., the link capacity estimation or the max-min fair allocation rate [12], [13].

Without loss of generality, we define the sender rate R_t at time t as

$$R_t = \frac{N_t}{T} \tag{1}$$

where T is the time interval ($T > 0$), and N_t is the number of packets transmitted during this interval. For reliable data transfer, a packet is successfully transmitted when the sender receives an ACK. In Fig. 5, we denote S_i as the number of CoAP packets sent with a start transmission rate R_{start} during time interval T_1 . If the sender receives an ACK at time t_1 , the round-trip time (RTT) for the first successful

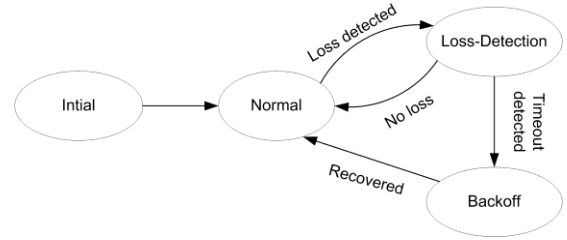


FIGURE 4. Four states of RCOAP.

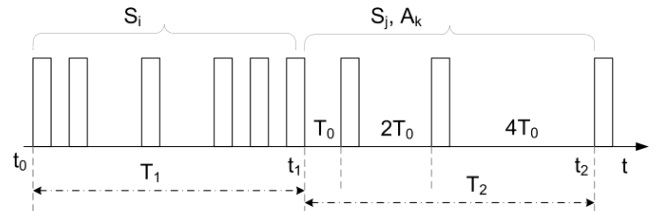


FIGURE 5. Time interval for estimating the initial transmission rate.

packet transmission is estimated as follows:

$$RTT = T_1 = t_1 - t_0 \tag{2}$$

During T_2 , the sender continues to send the next S_j packets with R_{start} . If the network is not congested, the sender can receive A_k acknowledgments (ACKs) for the transmitted packets. The number of ACKs depends on the round-trip delay and the available bandwidth of the bottleneck link. If the network is congested or in the case of temporal wireless errors, some packets can be lost, and only a few ACKs can arrive at the sender. The value of A_k could be *zero* in the case of heavy congestion.

We assume that a packet loss is detected at the time $t_1 + T_0$, as shown in Fig. 5. The lost packet is retransmitted, whereas the sender continues to transmit the other $j-1$ packets *in flight*. The time-out T_0 is doubled for every retransmission. The maximum number of retransmissions in CoAP is *four*. If the last retransmission is successful, the sender is assumed to receive an ACK at the end of T_2 . Therefore, we chose two round-trip times ($2RTT$) as the time interval for estimating the transmission rate of RCOAP, i.e., $T_1 + T_2 = 2 * RTT$. The total number of received ACKs during T_1 and T_2 can be estimated as follows:

$$N_ACK = \max(1, A_k) \tag{3}$$

The choice of the start transmission rate R_{start} depends on the bottleneck bandwidth of the network. In principle, R_{start} can be derived from a max-min fair allocation algorithm similar to [13]. Let us assume that there are n nodes in a tree-topology IoT network. Let r_i be the maximum allocated rate at node i , and c_i be the maximum capacity of the link between node i and its parent node. B_i is the set of nodes that are the children of node i . If node i is a leaf node in the tree, the start transmission rate R_{start} for a transaction at node i can be

determined by the following equation:

$$R_{start} = \min \left(c_i, r_i + \sum_{j \in B_j} r_j \right) \quad (4)$$

In other words, the transmission rate is either the available capacity c_i of node i towards the parent node or the sum of the maximum link rate r_i of node i and its parent route (including the bottleneck link), whichever is smaller. If R_{start} is chosen to be too large (i.e., $R_{start} \gg c_i$), then the new connection will cause network congestion very quickly. Otherwise, the link bandwidth is not efficiently utilized, and the performance will remain low for a time interval that is proportional to the bandwidth-delay product of the bottleneck link.

Another concern is the choice of discovery packets during the initial state. There are two options: 1) discovery packets are empty, and 2) discovery packets are actual packets of the transaction. In the first option, it is not necessary to resend if discovery packets are lost. In the second option, we must retransmit the packets that are lost during the initial interval. A simple method is to use the first option.

In this scheme, we use a simple method for estimating the initial transmission rate, as follows. Within two round-trip times ($2RTT$), the sender transmits a number of *in-flight* discovery packets using a start transmission rate (R_{start}) with an inter-packet interval (IPI_{start}). The receiver sends an ACK to each received message. The number of ACKs is proportional to the bandwidth-delay product and depends on the bottleneck bandwidth of the network. The sender counts the number of received ACKs for these probing packets. At the end of the initial state, the RCOAP sender sets the transmission rate R as in (1) by using the number of ACKs and the initial time interval.

At the beginning of a new connection, the sender sets the start transmission rate R_{start} as follows:

$$R_{start} = \min(c_i, r_i + \text{sum}(r_j)) \quad (5)$$

The inter-packet interval is initialized as

$$IPI_{start} = \frac{1}{R_{start}} \quad (6)$$

The duration of the initial time interval is approximately two round-trip times, as explained above. At the end of the initial state, the sender determines the transmission rate R for bursty data transfer as follows:

$$R = \min \left(R_{start}, \frac{\max(1, N - ACK)}{2 \times RTT} \right) \quad (7)$$

where N_ACK is the number of ACKs received during the initial state. Consequently, the inter-packet interval is calculated as follows:

$$IPI_{init} = \frac{1}{R} \quad (8)$$

Then, the sender leaves the initial state and changes to the normal state. The algorithm for the initial state is shown in Fig. 6 and is explained as follows.

Algorithm 1: Initial State

```

1: FUNCTION InitialState
2:   State = Init
3:    $R_{start} \leftarrow \min(c_i, r_i + \text{sum}(r_j))$ 
4:    $IPI_{start} \leftarrow 1 / R_{start}$ 
5:    $T \leftarrow t_0 + IPI_{start}$ 
6:    $T_1 \leftarrow RTT_m$ 
7:   packet  $\leftarrow$  sendNextPacket()
8:   N_ACK  $\leftarrow$  0
9:   WHILE (ACK == 0)
10:    IF ( $t \geq T$ )
11:      packet  $\leftarrow$  sendNextPacket()
12:       $T \leftarrow t + IPI_{start}$ 
13:    IF ( $t > T_1$ )
14:      TransactionFailed()
15:    return
16:    $RTT \leftarrow$  CalculateRTT()
17:   N_ACK  $\leftarrow$  N_ACK + 1
18:   WHILE ( $t \leq 2 * RTT$ )
19:    IF ( $t \geq T$ )
20:      packet  $\leftarrow$  sendNextPacket()
21:       $T \leftarrow t + IPI_{start}$ 
22:    IF (ACK != 0)
23:      N_ACK  $\leftarrow$  N_ACK + 1
24:    $R \leftarrow \min(R_{start}, \max(1, N\_ACK) / 2 * RTT)$ 
25:   State = Normal
26: END FUNCTION

```

FIGURE 6. Initial state algorithm for RCOAP.

The initial state starts at line 2. R_{start} is estimated based on c_i , r_i , and r_j (line 3). T_1 is necessary to check the ability to receive the first acknowledgment (line 6). RTT_m is defined by a value of four RTO_{init} , i.e., 8 s ($RTO_{init} = 2$ s). RCOAP sends the first packet (line 7) and waits for the first acknowledgment (lines 9 – 15). During time interval T_1 , RCOAP continuously sends discovery packets (line 11) using the inter-packet interval IPI_{start} (lines 10 – 12). If no ACK is returned during T_1 (line 13), the transaction fails (line 14). If the first ACK arrives, the round-trip time (RTT) is updated (line 16), and the number of ACKs is increased by *one* (line 17). During the time interval $2 * RTT$ (line 18), RCOAP continuously sends discovery packets (line 20) using the inter-packet interval IPI_{start} (lines 19 – 21). For every ACK received, the number of ACKs is updated accordingly (lines 22 and 23). At the end of the initial state, the transmission rate is determined, and RCOAP goes to the normal state (lines 24 and 25). In the worst case, if the retransmission occurs immediately during T_1 , the RTT could be large. It is reasonable to adjust RTT_m to limit the number of *in-flight* packets.

At the beginning of a connection, we assume that the RCOAP sender has the maximum link rates r_j of network nodes in order to calculate R_{start} according to (5). However, R_{start} is only valid during the first RTT. Upon receiving the first ACK for discovery packets, the R_{start} value

is immediately replaced by the estimated rate R using (7). In fact, the number of ACKs for discovery packets represents the current bottleneck bandwidth. Thus, the assumption is not necessary for RCOAP. Without such information, the sender can estimate R_{start} by using the method in [31]. Otherwise, one additional RTT can be used at the connection beginning for sending successive discovery packets to estimate R_{start} using the inter-arrival time of ACK packets.

B. NORMAL STATE

The normal state is considered to be a non-congestion state. The sender periodically transmits the packets in a step-like manner. For each ACK received, the actual RTT is updated, and the transmission rate is adjusted accordingly. In fact, the transmission rate is increased by one packet per actual round-trip time if no packet loss is detected. In this way, RCOAP acts similarly to TCP when the network is not congested. Fairness can be obtained for TCP flows that often appear in the networks. A variable called $wdsn$ can be used to limit the maximum number of *in-flight* packets in correspondence with the TCP behavior. To make RCOAP not aggressive, the maximum transmission rate is instead upper bounded by the value of R_{max} , i.e., the maximum available bandwidth of the bottleneck link.

RCOAP determines the actual transmission rate based on RTT as follows.

If no packet loss is detected:

$$RTT_{new} = (1 - \alpha) \times RTT_{old} + \alpha \times RTT_{cur} \quad (9)$$

$$R_{new} = \min(R_{cur} + 1/RTT_{new}, R_{max}) \quad (10)$$

If no ACK is received within an initialized time-out, the transmitted packet with the associated message ID is considered to be lost. Then, RCOAP stops the data transfer and goes to the loss-detection state. The algorithm for the normal state is shown in Fig. 7.

The normal state begins at t_0 with the current transmission rate that is represented by the inter-packet interval $IPI_{current}$. T is the time required for the next transmission (line 3). T_n denotes the time instant at which the transmission rate can be increased. If no loss is detected (lines 5 – 14), the transmission rate increases after every T_n interval (line 11). Packets are periodically sent after every time interval T (lines 6 and 7). For every ACK received (line 8), the actual round-trip time RTT_{new} is updated (line 9). After each T_n , the transmission rate is increased by one packet per round-trip delay (line 11). The new inter-packet interval IPI_{new} is adjusted accordingly (line 12). The actual values for T and T_n are updated (lines 13 and 14). If a packet loss is detected, the RCOAP changes to the loss-detection state (line 15).

C. LOSS-DETECTION STATE

RCOAP enters the loss-detection state when a packet loss is detected. RCOAP must distinguish two possibilities: loss due to temporal wireless link errors (i.e., wireless errors) or loss due to congestion (e.g., buffer overflow at the receiver or in

Algorithm 2: Normal State

```

1:  FUNCTION NormalState
2:  State = Normal
3:  T ← t0 + IPIcurrent
4:  Tn ← t0 + RTTcurrent
5:  WHILE (No_Loss)
6:    IF (t ≥ T)
7:      packet ← sendNextPacket()
8:    IF (ACK)
9:      RTTnew ← (1 - α) × RTTold + α × RTTcurrent
10:   IF (t > Tn)
11:     Rnew ← min(Rcurrent + 1 / RTTnew, Rmax)
12:     IPInew ← 1 / Rnew
13:     T ← t + IPInew
14:     Tn ← t + RTTnew
15:   State = Loss_Detection
16: END FUNCTION

```

FIGURE 7. Normal state algorithm for RCOAP.

the intermediate network nodes). If the loss is due to temporal wireless errors, it is unnecessary to decrease the rate because the link will shortly be recovered (typical fading would last for approximately tens to a few hundreds of milliseconds). Therefore, RCOAP could maintain high performance. In the case of congestion, RCOAP should decrease the transmission rate similar to the behavior of TCP flows, to avoid further congestion. The algorithm for the loss-detection state is shown in Fig. 8.

At the beginning of the loss-detection state, the reason for the packet loss is still unknown. The current transmission rate is stored (line 3) at the sender to recover when the temporal wireless losses are recovered. RCOAP tries to maintain fair congestion control with the TCP flows by halving its transmission rate (lines 4 – 6). The duration for checking the link condition is one estimated RTT (line 7).

Then, the RCOAP sender retransmits the lost packet (line 9) and updates the RTO (line 10) based on the measurements of new RTTs (the same as in [6]), as follows:

$$RTT_{new} = (1 - \alpha) \times RTT_{old} + \alpha \times RTT_{cur} \quad (11)$$

$$RTTVAR_{new} = (1 - \beta) \times RTTVAR_{old} + \beta \times |RTT_{old} - RTT_{cur}| \quad (12)$$

$$RTO_{new} = 0.5 \times (RTT_{new} + 4 \times RTTVAR_{new}) + 0.5 \times RTT_{old} \quad (13)$$

During the time interval of one estimated RTT, RCOAP checks for the reason for packet loss (line 11-17). If an ACK is received for the retransmitted packet (line 12), RCOAP assumes that the packet loss is due to wireless link error, i.e., the network is not congested. The transmission rate of the RCOAP is recovered to the previous rate R_{old} (line 13). On the other hand, RCOAP sends further packets periodically to probe the network conditions (lines 15, 16, and 17). The loss-detection state lasts for one estimated round-trip time (line 11). At the end of this time interval, the RCOAP returns

Algorithm 3: Loss-Detection State

```

1:  FUNCTION LossDetectionState
2:  State = Loss-Detection
3:   $R_{old} \leftarrow R_{current}$ 
4:   $R_{new} \leftarrow R_{current} / 2$ 
5:   $IPI_{new} \leftarrow 1 / R_{new}$ 
6:   $T \leftarrow t_0 + IPI_{new}$ 
7:   $T_n \leftarrow t_0 + RTT_{current}$ 
8:  ACK  $\leftarrow$  False
9:  packet  $\leftarrow$  Retransmission()
10:  $RTO_{new} \leftarrow$  UpdateRTO()
11: WHILE ( $t \leq T_n$ )
12:   IF (ACK)
13:      $R_{new} \leftarrow R_{old}$ 
14:     ACK  $\leftarrow$  True
15:   IF ( $t \geq T$ )
16:     packet  $\leftarrow$  sendNextPacket()
17:      $T \leftarrow t + 1 / R_{new}$ 
18:   IF (ACK == True)
19:     State = Normal
20:     return
21:   ELSE
22:     State = Backoff
23: END FUNCTION

```

FIGURE 8. Loss-detection state algorithm for RCOAP.

to the normal state if the sender receives an ACK from the receiver (lines 18 and 19). The acknowledgment indicates that the network is not congested, and the packets can be acknowledged. The old transmission rate can be recovered after one RTT. Thus, RCOAP can maintain a high throughput for bursty data transfer.

If the network is congested, no ACK is received until the end of the loss-detection state. In this case, RCOAP goes to the backoff state to handle the congestion problem (line 22).

D. BACKOFF STATE

In this state, RCOAP first maintains the decreased transmission rate at the end of the loss-detection state, i.e., a halved rate similar to the TCP behavior. During this state, the sender attempts to transmit discovery packets to check the congestion state. If no ACK is received, the network is congested. The sender rate is further decreased multiplicatively. The RCOAP sender triggers a backoff algorithm for retransmission. The retransmission of packets fails when the maximum number of four retransmissions is exceeded.

If any ACK is received, the sender assumes that the congestion has been resolved. Then, the RCOAP returns to the normal state. Note that in the case of multiple losses, the

sender rate can be decreased several times. The number of ACKs for discovery packets can be used to indicate the congestion level. The RCOAP sender remains in the backoff state until it receives an ACK from the receiver.

The algorithm for the backoff state is shown in Fig. 9. At the beginning of the backoff state (at time t_0), the transmission rate of RCOAP is the halved rate (line 3), as explained in the previous state. T_n is the time interval for sending discovery packets to check the congestion situation (line 6). This time interval is approximately one estimated RTT. The sender periodically checks if any ACK is received (line 8). In addition, the sender tries to send some discovery packets within one estimated RTT to check the network condition (lines 9, 10, and 11). The number of ACKs for these discovery packets indicates the congestion level. If the network is heavily congested, no ACK or only a few ACKs can be received for the transmitted packets. The sender can use this number to set the transmission rate after the backoff period using (7). When the network is still congested, several retransmissions are triggered (line 13), if the maximum number of retransmissions is not exceeded. The new RTO value is updated (line 14) by using a variable backoff mechanism [6] for retransmission, as follows:

$$RTT_{new} = (1 - \alpha) \times RTT_{old} + \alpha \times RTT_{cur} \quad (14)$$

$$RTTVAR_{new} = (1 - \beta) \times RTTVAR_{old} + \beta \times |RTT_{old} - RTT_{cur}| \quad (15)$$

$$RTO_{cur} = 0.5 \times (RTT_{new} + 4 \times RTTVAR_{new}) + 0.5 \times RTT_{old} \quad (16)$$

$$RTO_{new} = RTO_{cur} \times VBF \quad (17)$$

where VBF is the variable backoff factor. The VBF depends on the initial RTO of the RCOAP, as follows:

$$VBF = \begin{cases} 3 & \text{if } RTO_{init} < 1s \\ 2 & \text{if } 1 \leq RTO_{init} \leq 3s \\ 1.3 & \text{if } RTO_{init} > 3s \end{cases} \quad (18)$$

VBF presents the degradation steps for the backoff strategy in the case of heavy network congestion.

If the retransmission count reaches a total of four retransmissions, the retransmission of the lost packet is considered to have failed (line 16). This lost packet can be stored in the sender buffer for a later retransmission, when the network congestion has been resolved. The received packets can be disordered at the receiver. Therefore, the application must rearrange the sequences of the received packets.

Once the RCOAP sender receives an ACK packet (line 17), it is assumed that the congestion is resolved. Then, the sender returns to the normal state (line 24) with the current rate. In the case of heavy network congestion, the RCOAP sender will further reduce its rate using multiplicative decreases (lines 20 – 23). If the sender does not receive any ACK for a certain timeout period, the transaction is considered to have failed. The exchange can be suspended, or the sender moves back to the initial state.

Algorithm 4: Backoff State

```

1:  FUNCTION BackoffState
2:  State = Backoff
3:   $R_{new} \leftarrow R_{current} / 2$ 
4:   $IPI_{new} \leftarrow 1 / R_{new}$ 
5:   $T \leftarrow t_0 + IPI_{current}$ 
6:   $T_n \leftarrow t_0 + RTT_{current}$ 
7:  ACK  $\leftarrow$  False
8:  WHILE (ACK == False)
9:      IF ( $t \geq T$ )
10:         packet  $\leftarrow$  sendNextPacket()
11:          $T \leftarrow t + 1 / R_{new}$ 
12:      IF (NotMaxRetransmission)
13:         packet  $\leftarrow$  Retransmission()
14:          $RTO_{new} \leftarrow$  UpdateRTO()
15:      ELSE
16:         PacketTransmissionFailed()
17:      IF (ACK)
18:         ACK  $\leftarrow$  True
19:      IF ( $t \geq T_n$ )
20:          $R_{new} \leftarrow R_{current} / 2$ 
21:          $IPI_{new} \leftarrow 1 / R_{new}$ 
22:          $T \leftarrow t + 1 / R_{new}$ 
23:          $T_n \leftarrow t + RTT_{new}$ 
24:  State = Normal
25:  END FUNCTION

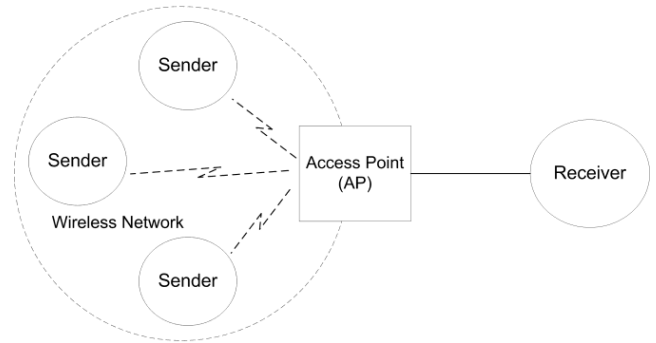
```

FIGURE 9. Backoff state algorithm for RCOAP.**IV. PERFORMANCE EVALUATION**

In this section, we present the performance evaluation of our RCOAP scheme and compare its performance with that of three other standard CoAP schemes, including the basic CoAP [3], CoCoA [5], and CoCoA+ [6]. We use the network simulator NS-3 [38] for the simulation environment. NS-3 does not support CoAP. We found a partial CoAP implementation developed by Maesoser for NS-3 on Github [39]. However, this partial CoAP implementation is for mDNS, multicast, and single hop. This design differs from our concept, which uses dynamic rate control and multi-hop networks to convey bursty data from senders to a remote center. We developed and implemented our RCOAP and other CoAP schemes in the network simulator NS-3. All of these modules have been implemented over UDP in the NS-3 protocol stack using the specifications provided in [3]–[6].

A. EXPERIMENT SETTINGS

The network topology of the simulation is shown in Fig. 10. CoAP senders are implemented at wireless nodes in a Wi-Fi network. We assume that each sender has collected data (e.g., sensing data) and they build flows of bursty data to send to the receiver. The receiver is located at the center and is connected to the senders through a Wi-Fi access point (AP node). At the

**FIGURE 10.** Simulation model.

physical and MAC layers, the nodes in the Wi-Fi network implement the standard IEEE 802.11b [38]. The mobility model is the *ConstantPositionMobilityModel* [38] with a distance of 10 m. The link bandwidth between the AP and the receiver was set to 1 Mbps with a link delay of 500 ms for the rate estimation experiments. The link delay was 64 ms for other performance simulations.

B. RATE ESTIMATION AND LOSS DETECTION

Fig. 11 presents the rate estimation for a RCOAP sender. As explained in Section 3, the initial state takes two round-trip times (approximately 1 s in our simulation). The start transmission rate is set to 180 Kbps. This value is determined based on the max-min fair share of the bottleneck link bandwidth for all RCOAP senders, as explained in Section II.

After the initial interval (approximately 1 s), the transmission rate is initialized to 160 Kbps, which is the maximum allowed rate for each flow. Then, the sender periodically transmits the RCOAP packets according to this maximum rate.

We consider the behavior of RCOAP in the case of a packet loss detection during the time interval from 4s to 6s. Once a packet loss is detected, the sender decreases its transmission rate to a halved rate. The loss-detection state takes a time from 4.5 s to 6.5 s. During this time, the sender periodically sends discovery packets to check the network conditions. Because the loss is resolved at time 6.5 s, the sender increases its transmission rate stepwise until it reaches the maximum allowed rate (160 Kbps in this simulation). There are some peak discovery rates without a rate decrease at 12 s, and then at 17 s due to the short fading of the wireless channel errors.

If the start sending rate is set to a value lower than the available fair share of the bottleneck bandwidth, the initial rate is determined to be a lower value, as shown in Fig. 12. The sender needs some time to increase its transmission rate stepwise until it obtains the maximum allowed rate at time 6 s. Again, the initial state lasts for 1 s, in which the sender checks the available link bandwidth to estimate the transmission rate. Fig. 12 also shows two short wireless link errors at 12 s and then at 17 s. The RCOAP sender checks for the reason for packet losses and determines that there is no need to decrease the transmission rate in this case.

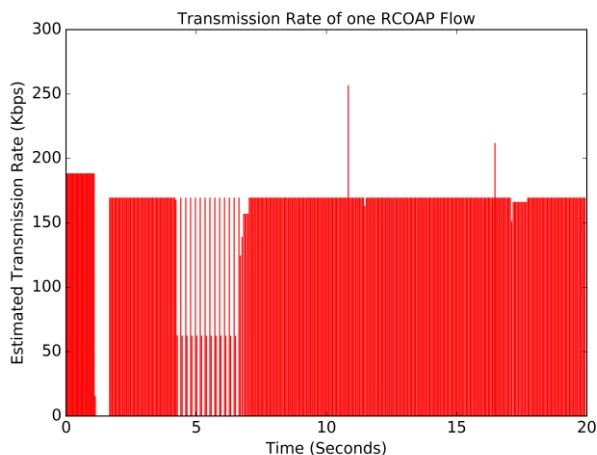


FIGURE 11. Rate estimation using a high start rate.

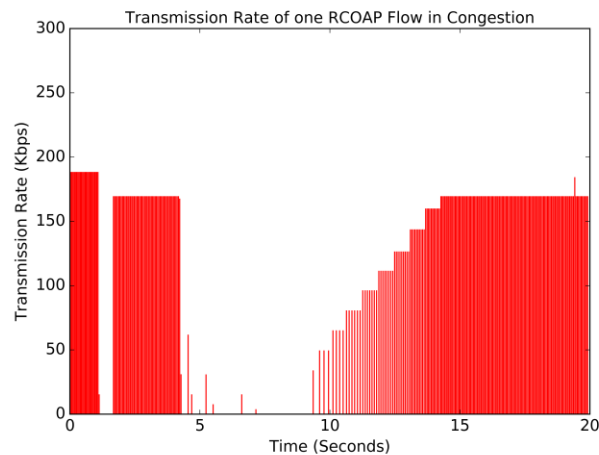


FIGURE 13. Congestion and backoff behavior.

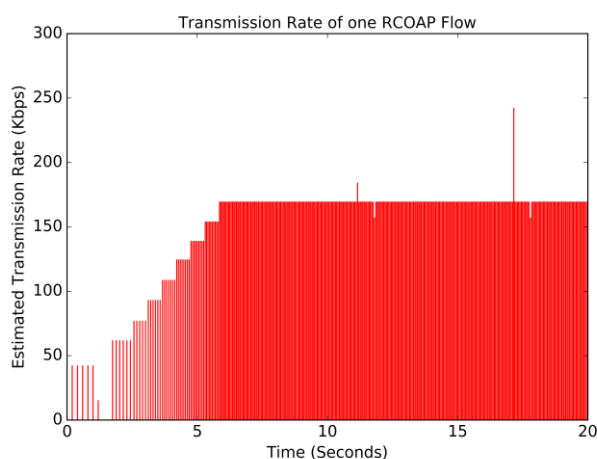


FIGURE 12. Rate estimation using a low start rate.

In Fig. 13, we show the behavior of RCOAP in the case of congestion and backoff. The initial state ended at time 1 s. At time 4.5 s, the sender detects a packet loss due to network congestion. RCOAP halves its transmission rate, similar to the behavior of TCP, and triggers the retransmission of lost packets. During the loss-detection state, the RCOAP sender checks for the network condition by continuously sending the discovery packets. The rate is further multiplicatively decreased, because no ACK is received until 9 s. Because the sender does not receive any ACK after the time-out (RTO), it goes to the backoff state and continues to retransmit the lost packet until it exceeds the maximum retransmission number. In our simulation, the congestion was resolved at time 9 s. Thus, the sender increases its transmission rate stepwise until it reaches the maximum allowed rate of 160 Kbps.

As we explained in Section III, various methods can be used to identify and differentiate packet losses [33]–[37]. Wireless losses are random losses and vary in nature. Congestion losses are usually correlated; i.e., if a packet loss is detected, further consecutive packets are considered to be lost. This behavior has been observed from the phenomenon of buffer overflow in routers. According to the methods

discussed in [32]–[34], RCOAP uses RTT, the inter-arrival of ACKs, and the mismatch between the message IDs of the transmitted packet and the ACK packet to detect the packet loss types. At the beginning of the loss-detection state, RCOAP tries to retransmit the lost packet. Within one estimated RTT, the RCOAP sender checks for the ACK. If the ACK is received for the retransmitted packet (i.e., with the same message ID), RCOAP assumes that the single packet loss is due to wireless transmission error. Otherwise, RCOAP identifies a network congestion loss.

We used two experiments to show how RCOAP is able to detect packet loss types. To this aim, we modified our simulator to mimic the packet loss types and measure the transmission rate (in packets/s) to indicate the behavior of RCOAP in both cases. The simulation was conducted using $RTT = 500$ ms. The start inter-packet interval (*IPI*) was 800 ms. The purpose of these experiments is to show the ability of RCOAP to distinguish wireless loss from congestion loss.

In the first experiment, we mimicked the wireless loss by dropping a single packet. Fig. 14 depicts the RCOAP transmission rate when a packet loss was due to wireless errors. At time 5.5 s, the RCOAP sender detected a single packet loss. The sender entered the loss-detection state, and halved its transmission rate. After one estimated RTT, the RCOAP sender identified that the single packet loss was due to wireless errors. Then, the previous transmission rate was recovered at time 7 s. These results give evidence for the behavior of RCOAP in the case of a packet loss due to wireless errors. Other experiments using different wireless error rate models are out of the scope of this paper. This issue could be a topic for further research.

In the second experiment, we mimicked the congestion loss by dropping some consecutive packets (at least two) as shown in Fig. 15. When a packet loss was detected, the sender entered the loss-detection state and halved its rate at time 6 s. After one estimated RTT, the sender detected consecutive packet losses due to network congestion. Thus,

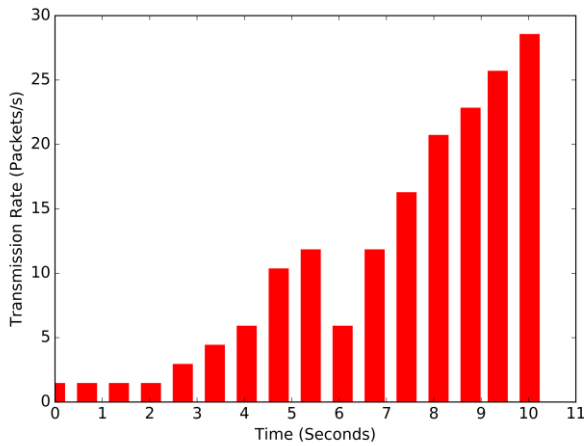


FIGURE 14. Transmission rate in the case of a packet loss due to wireless errors.

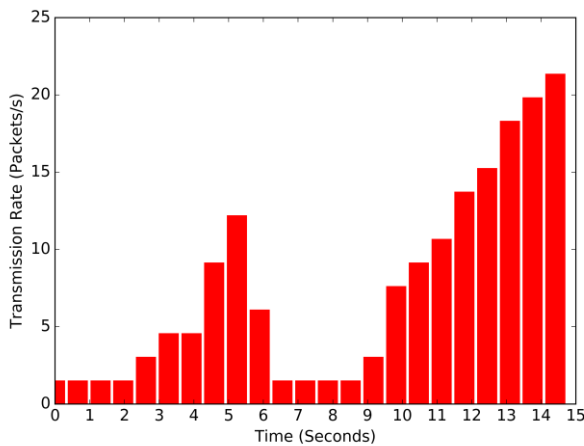


FIGURE 15. Transmission rate in the case of a packet loss due to network congestion.

it entered the backoff-state. During this state, some discovery packets were sent to check the network condition. When the congestion had been resolved at the time of approximately 9.5 s, the sender returned to the normal state and increased the transmission rate stepwise for each RTT according to algorithm 2 in Section III. These results give evidence for the behavior of RCOAP in the case of a packet loss due to network congestion.

C. THROUGHPUT PERFORMANCE

The model in Fig. 10 is used for the performance evaluation with three competing flows for each scheme alternately. The link delay and link bandwidth between the AP and receiver were set to 64 ms and 1 Mbps, respectively.

In the simulation, all three data flows use the same system configurations as follows. The senders were implemented at wireless nodes using IEEE 802.11b and the protocol stack of NS-3 [38]. The distance to the AP node is 10 m. The mobility model is the ConstantPositionMobilityModel [38]. The data packet and the discovery packet (including IP/UDP/CoAP headers) are 106 bytes and 41 bytes, respectively. The ACK

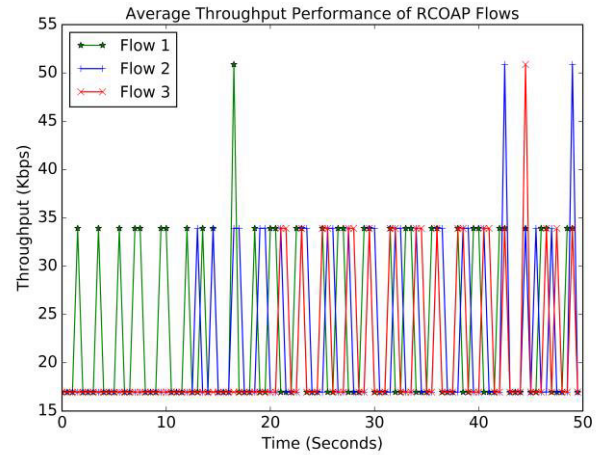


FIGURE 16. Average throughput performance of RCOAP flows.

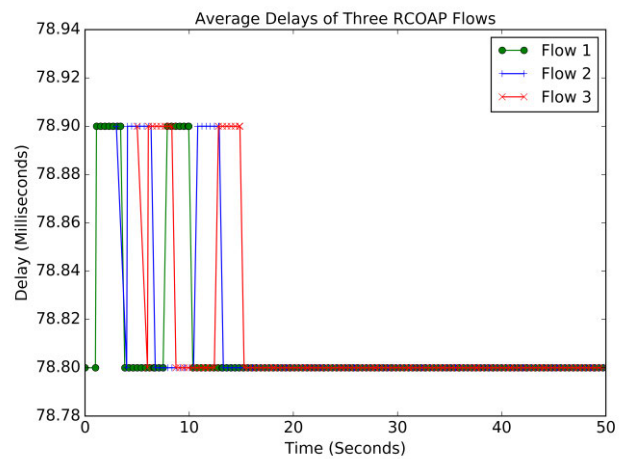


FIGURE 17. Average transmission delays of RCOAP flows.

timeout is 2 s. The maximum number of retransmissions is 4. The start rate is 16 Kbps for all three flows. The simulation time is 50 s for all flows. The only difference is the start time. Flow 1 starts at 0 s, flow 2 at 3 s, and flow 3 at 5 s. The RCOAP receiver is implemented at an NS-3 node connected to the AP node using an Ethernet link.

Fig. 16 depicts the throughput performance for three competing RCOAP flows in the case of non-congestion for 50 s. Three flows start their transmission at 0, 3, and 5 s, respectively. The packet size consists of 106 bytes, including the CoAP header.

As shown in Fig. 16, the initial interval of each sender differs depending on the actual round-trip time. We calculated the average throughput of each flow using a confidence interval of 500 ms. It can be seen that the RCOAP is fair for competing flows. The fairness can be computed using the Jain’s well-known fairness index. There are some peaks due to the higher probing rates for checking the temporal wireless errors during the loss-detection state of the flows.

Fig. 17 shows the average transmission delays of packets that belong to three competing RCOAP flows at different

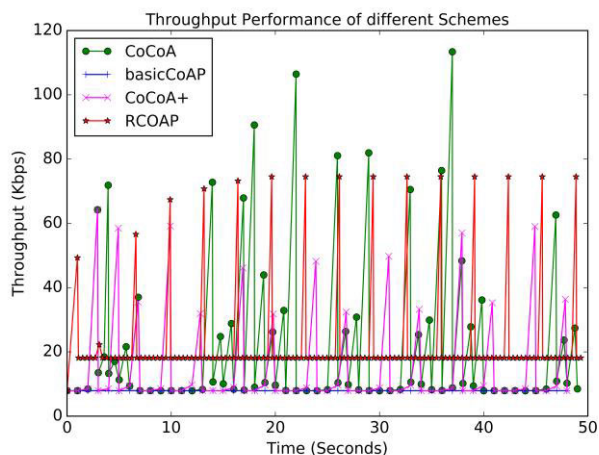


FIGURE 18. Throughput performance evaluation.

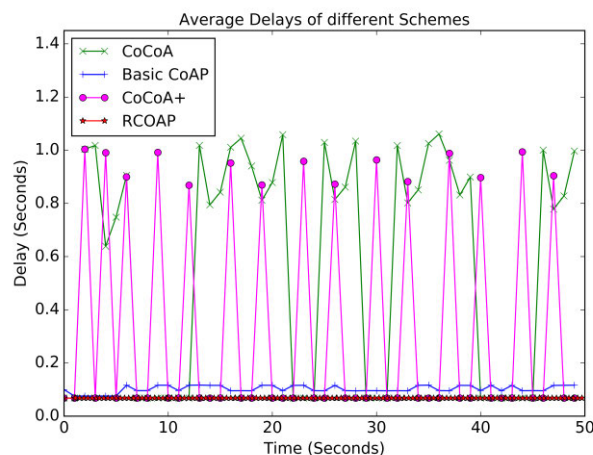


FIGURE 19. Transmission delays of different schemes.

start times. Due to the high probing rate at the initial phase, the packet delays vary from 77.87 ms to 78.90 ms for all of the flows. When the senders were stable in the normal state, the average delays of all flows fluctuate around 78.50 ms over the entire simulation time of 50 s.

In Fig. 18, we compare the throughput of various schemes, including basic CoAP, CoCoA, CoCoA+, and RCOAP. We ran simulations in 50 s for each scheme alternately under the same conditions using three competing flows. As shown in the figure, the throughput values vary with some peaks due to the changes in the RTOs and retransmissions. The statistical results on these data can be seen in detail in Tables 2, 3, and 4. It can be seen that the average throughput values of CoCoA, basic CoAP, and CoCoA+ are almost 16 Kbps, whereas the throughput of RCOAP is approximately 20 Kbps.

Fig. 19 shows the average transmission delays for the different schemes. In this simulation, three flows of the basic CoAP, CoCoA, and CoCoA+ had a fixed transmission rate using the IPI of 500 ms. RCOAP flows had an initial IPI of 500 ms. Then, RCOAP flows dynamically changed the transmission rate up to the maximum allowed rate according to a minimum inter-packet interval IPI_{min} of 400 ms. The link bandwidth was 1 Mbps with a link delay of 64 ms. This experiment aims to show the outperformance of RCOAP under the same network condition for all schemes. Under this simulation condition, the delay values of CoCoA and CoCoA+ were high, and they fluctuated. Their average delays are between 400 ms and 700 ms. This fluctuation occurs because these schemes have many packets to retransmit (see also Tables 2, 3, and 4). The RTO values are frequently updated for retransmissions. These changes in turn lead to further fluctuations in the delay. In contrast, basic CoAP and RCOAP provide better delay performance. The average delay values were approximately 100 ms and 77 ms in the basic CoAP and RCOAP, respectively. This finding occurs because basic CoAP and RCOAP provide fast retransmission and reduce the retransmission attempts by detecting ACK losses.

The delay due to retransmissions can be spotted. The reason is as follows. In the case of packet losses, the number of retransmissions can reach the maximum value of *four* in all schemes. However, CoCoA and CoCoA+ frequently adjust the RTO value. Thus, retransmissions can be quickly triggered, which can lead to further congestion. Each retransmission will cause an additional delay in the packet. That is why there are some spikes in the delay in the case of CoCoA and CoCoA+ schemes.

The basic CoAP uses the fixed RTO value. For each retransmission, this value is doubled using a binary exponential backoff (BEB). Although the basic CoAP needed to retransmit many packets (see Tables 2, 3, and 4 for the number of retransmissions by the basic CoAP), all packets were successfully retransmitted within the next round trip time due to the fixed RTO value. That is why the average delays were smaller compared to CoCoA and CoCoA+.

For RCOAP, no packet loss was observed in this case. Although the sender rate dynamically increased, it was limited by the maximum allowed rate using $IPI_{min} = 400$ ms. The number of retransmissions was *zero* (see Tables 2, 3, and 4 for the number of retransmissions by RCOAP). That is why the delay values for RCOAP were almost constant in Fig. 19. These results indicate that RCOAP significantly outperforms other schemes under the same conditions.

The previous experiment also demonstrates the reliability of the schemes. The packet loss ratio was *zero* in all schemes, although several packets were required to be retransmitted. A packet will be lost if its retransmission is not successful after *four* retries. The results indicate that all retransmissions were successfully completed. All schemes were reliable.

Another set of experiments was conducted to show the behavior of the schemes in a more dynamic high-load network environment using TCP background traffic. The basic CoAP, CoCoA, and CoCoA+ nodes send confirmable packets to the sink with a fixed rate by $IPI = 500$ ms. The initial rate of the RCOAP sender is set by $IPI = 500$ ms. The

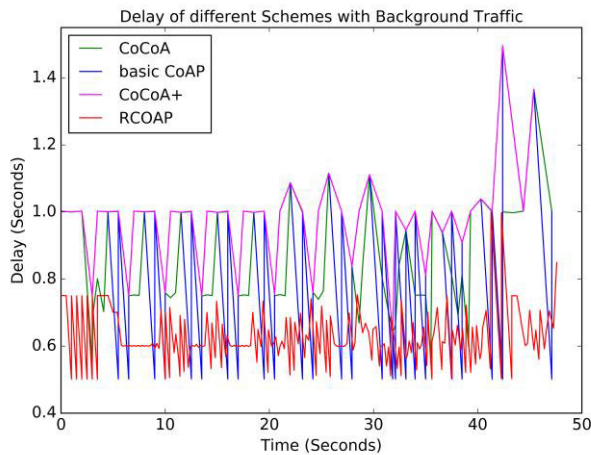


FIGURE 20. Delays of schemes using background traffic when the minimum IPI of RCOAP is 400 ms.

maximum allowed rate of the RCOAP sender is limited by $IPI_{min} = 400$ ms. A source node generates background traffic using NS-3 TCP socket for each experiment. The shared link bandwidth is 1 Mbps with an estimated round trip time of 500 ms. The purpose of these experiments is to compare different schemes under the high-load condition.

Fig. 20 indicates the delays for different schemes in the case of TCP background traffic. The average delays were 800.81 ms, 804.57 ms, 941.24 ms, and 613.25 ms for CoCoA, basic CoAP, CoCoA+, and RCOAP, respectively. The packet delays varied according the offered load of the TCP flow. When the offered load was higher, packet delays for CoCoA and CoCoA+ quickly increased and highly fluctuated. On the contrary the delay for RCOAP fluctuated, but it was lower than 750 ms. The results show that RCOAP provides better delay performance than the other schemes under the same dynamic high-load conditions.

Fig. 21 shows the throughput comparison of the schemes. The average throughput values were sampled every 500 ms. The measured values were approximately 5.6 Kbps, 5.8 Kbps, 4.6 Kbps, and 7.0 Kbps for CoCoA, basic CoAP, CoCoA+, and RCOAP, respectively. The number of successful transmitted packets of RCOAP was 216. On the contrary, these values were 81, 80, and 54 for CoCoA, basic CoAP, and CoCoA+, respectively. As shown in Fig. 21, the flows of CoCoA and CoCoA+ were starved at time 42 s when the TCP background traffic quickly increased. These results demonstrate that RCOAP behaves more efficiently than the other schemes under the same high-load network conditions when network congestion increases.

Fig. 22 shows the delays for different schemes when the maximum allowed rate of the RCOAP sender was limited by $IPI_{min} = 200$ ms. The number of successful transmitted packets of RCOAP was 270 in this experiment. As shown in Fig. 22, the delay fluctuated, but it remained lower than 750 ms. The RCOAP flow using $IPI_{min} = 200$ ms was more aggressive due to the higher allowed rate. In response to network congestion, the RCOAP sender quickly decreased

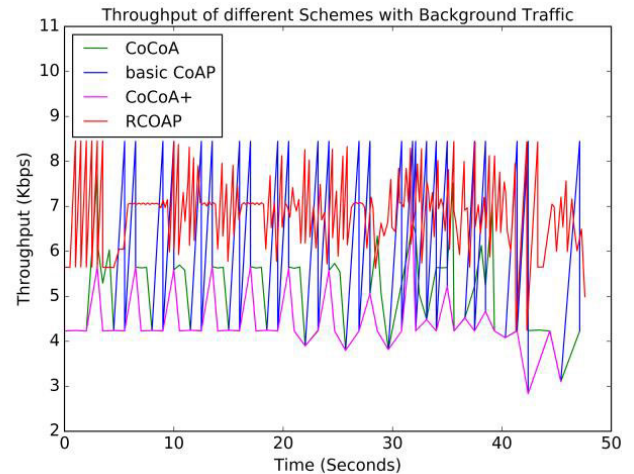


FIGURE 21. Throughput of schemes using background traffic.

its rate, resulting in scattered intervals of delay variation. These results show that the RCOAP scheme provides stable performance even under high-load conditions.

The ability of RCOAP to differentiate wireless losses from congestion losses has been indicated using the results in Fig. 14 and Fig. 15. The RCOAP sender detected wireless errors at 5.5 s as shown in Fig. 14. Thus, the sending rate did not decrease. In Fig. 15, the RCOAP sender detected a packet loss due to network congestion at 6 s. The sender decreased its sending rate by halving its rate.

The basic CoAP, CoCoA, and CoCoA+ cannot distinguish the reason for packet losses. An experiment was conducted to explore the packet loss ratio of these schemes compared to RCOAP. In this experiment, the inter-packet interval (IPI) was changed from 10 ms to 5 s using a step of around 10 ms. As indicated in [3], the default leisure time of CoAP is 5 s. For each IPI , a simulation was run for each scheme. We measured the number of delivered packets N_s and received packets N_r to calculate the average loss packet ratio R_L as $R_L = 100\% * (N_s - N_r)/N_s$. To compare the schemes, we convert the range of the IPI values into the offered loads. The highest load (80%) is corresponding to the IPI value of 10 ms, and the lowest load (10%) is for the IPI value of 5 s. Using the experiment, we found that the packet loss occurred at the load of 25% ($IPI = 700$ ms) in the basic CoAP, CoCoA, and CoCoA+. In RCOAP, a small loss rate was detected at the load of 50%. The loss rate increased quickly as the load was higher than 65% ($IPI \leq 50$ ms). These results indicate that RCOAP can detect and retransmit lost packets due to wireless errors and congestion to maintain a lower packet loss ratio. This result is shown in Fig. 23.

Fig. 23 depicts the packet loss ratios of the different schemes according to the increasing offered load. As shown in the figure, the packet loss ratio of CoCoA and CoCoA+ increases when the load is 45%. The average loss rate of CoCoA was higher than that of CoCoA+.

The basic CoAP exhibits the best performance in terms of packet loss. This finding shows the advantage of using

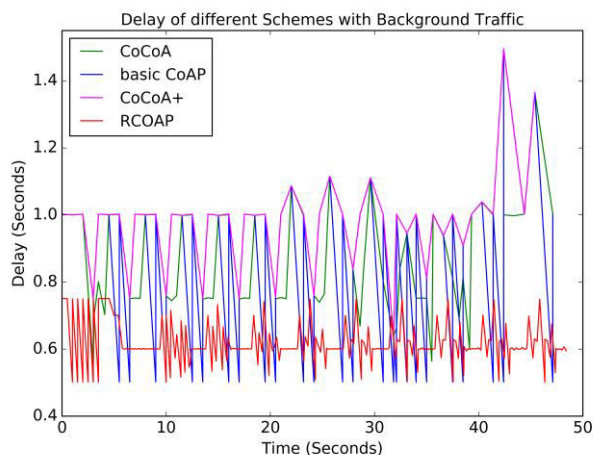


FIGURE 22. Delays of schemes using background traffic when the minimum IPI of RCOAP is 200 ms.

fixed RTO values. RCOAP maintains a low loss rate, when the offered load is below 65% and increases the loss rate by a higher load.

Table 1 shows the detailed performance evaluation of the three competing RCOAP flows. Although the flows start at different times, they have almost the same average delay of approximately 78 ms. The three flows have a comparable average throughput according to the number of delivered packets. The flow that starts early can send more packets. The loss ratio and retransmission number were zero for all flows. In other words, the flows are fair to each other by sharing a common link bandwidth. Tables 2, 3, and 4 present a performance comparison of RCOAP with three other schemes, each using three competing flows. RCOAP has low average delays, similar to the basic CoAP, while the delays in CoCoA and CoCoA+ are very large because of the large number of retransmissions that cause frequent RTO updating periods. In addition, the number of retransmissions for flows 1 and 2 was higher than that of flow 3 in basic CoAP, CoCoA, and CoCoA+. There is an unfairness between competing flows in these schemes.

In contrast, RCOAP provides a higher average throughput of approximately 135% compared to the other schemes. Moreover, competing RCOAP flows have a lower packet loss ratio, lower retransmission rate, and higher delivery rate compared to the other schemes. With these performance characteristics, RCOAP is more efficient and suitable for bursty data transfer in IoT networks.

The simulation was run 50 times for each scheme. We measure the amount of received packets during each 500 ms. The throughput is then calculated as the number of received packets divided by 500 ms. Since the simulation time is 50 s, we have 100 intervals. Thus, the average throughput for each flow is the sum of all throughput values divided by 100. The average measured throughput is about 16.96 Kbps for each flow in the basic CoAP, CoCoA, and CoCoA+. The mean throughput of RCOAP flows is around 22.9 Kbps. Thus, the average throughput of RCOAP is approximately

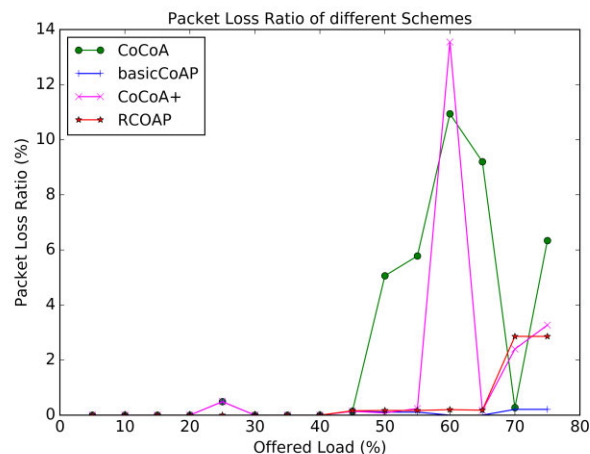


FIGURE 23. Packet loss ratio of different schemes.

22.9 Kbps/16.96 Kbps = 135% compared to other schemes. That is, RCOAP has a throughput improvement of about 35%.

We performed the simulation 50 times for each experiment and each scheme to measure the average throughput. In addition, we also tested the schemes using several long sessions of 100 s and with different inter-packet intervals to measure the packet delay, throughput, the number of delivered packets, the number of lost packets, and the number of retransmissions.

An experiment was conducted to perform statistical testing for the schemes. In this experiment, we used different inter-packet intervals (IPI) for representing the start rate. Let T_M denote the average throughput of RCOAP, T_B is the average throughput of the basic CoAP, T_A is the average throughput of CoCoA, and T_P is the average throughput of CoCoA+. We denote by L_M the loss rate of RCOAP, by L_B the loss rate of the basic CoAP, by L_A the loss rate of CoCoA, and by L_P the loss rate of CoCoA+. Table 5 shows the statistical results for throughput ratios of RCOAP compared to each other scheme, and the loss ratio of each scheme according to various IPI values. As shown in Table 5, the loss rate in all schemes is increased quickly for IPI values smaller than 200 ms. We found that, RCOAP works better for IPIs between 200 ms and 250 ms in this case. Smaller IPIs can cause a higher loss probability. Despite the higher throughput compared to other schemes, higher IPIs are not recommended for bursty data transfer due to long session completion time.

Table 6 shows the percentage of improvement of the RCOAP in comparison to the basic CoAP, CoCoA, and CoCoA+.

D. DISCUSSION

RCOAP uses some small empty discovery packets for estimating the initial sending rate and checking the reason for packet losses. These control packets can introduce some overhead, but they can help to increase performance. An experiment was conducted to show this result. We used the same simulation model as shown in Fig. 10, and set the bottleneck bandwidth at 25 Kbps. We used 41 bytes for a discovery

TABLE 1. Performance evaluation for three RCOAP flows.

Flows	Average Delay (ms)	Average Throughput (Kbps)	Loss Ratio (%)	Number of delivered Packets in 50s	Number of Retransmissions
RCOAP Flow 1	78.6	23.24	0	137	0
RCOAP Flow 2	78.5	21.88	0	129	0
RCOAP Flow 3	78.4	20.86	0	123	0

TABLE 2. Performance evaluation for flow 1.

	Average Delay (ms)	Average Throughput (Kbps)	Loss Ratio (%)	Number of delivered Packets	Number of Retransmissions
Basic CoAP	102.4	16.96	0	50	43
CoCoA	585.7	16.96	0	50	43
CoCoA+	316.7	16.96	0	50	43
RCOAP	78.8	23.24	0	137	0

TABLE 3. Performance evaluation for flow 2.

	Average Delay (ms)	Average Throughput (Kbps)	Loss Ratio (%)	Number of delivered Packets	Number of Retransmissions
Basic CoAP	106.2	16.96	0	47	45
CoCoA	582.3	16.96	0	47	44
CoCoA+	371.8	16.96	0	47	45
RCOAP	78.8	21.88	0	129	0

TABLE 4. Performance evaluation for flow 3.

	Average Delay (ms)	Average Throughput (Kbps)	Loss Ratio (%)	Number of delivered Packets	Number of Retransmissions
Basic CoAP	87.8	16.96	0	45	5
CoCoA	601.7	16.96	0	45	6
CoCoA+	368.6	16.96	0	45	5
RCOAP	78.8	20.86	0	123	0

TABLE 5. Comparison of throughput and loss rate.

Inter-packet Interval (<i>IPI</i>) (ms)	T_M/T_B	T_M/T_A	T_M/T_P	L_M	L_B	L_A	L_P
5000	4.54	4.54	4.54	0.00	0.00	0.00	0.00
3000	2.57	2.57	2.57	0.00	0.00	0.00	0.00
1000	1.56	1.56	1.56	0.00	0.49	0.49	0.49
500	1.14	1.14	1.14	0.00	0.00	0.00	0.00
300	1.45	1.45	1.45	0.00	0.00	0.00	0.00
200	1.74	1.74	1.74	0.16	0.14	0.14	0.14
100	2.16	0.94	0.95	0.20	0.11	10.94	13.55

packet, and 106 bytes for a data packet (including IP, UDP, and CoAP headers). The simulation time was 100 s for a burst transfer session. We ran the simulation 50 times to measure the average throughput.

In the condition without losses, the sender only sent some discovery packets once during the first RTT. The estimated

RTT was around 64 ms. Thus, the time used for the discovery was 64 ms/100 s = 0.064% of the total session time. Without rate control, the sender would send using the fixed rate according to the default leisure time of 5 s [3]. The average measured throughput was 21.2 Kbps for one flow. Using rate control, the average throughput was 23.4 Kbps,

TABLE 6. Features of the schemes.

Protocol	Backoff Algorithm	RTT Estimation	RTO Aging	Bottleneck Bandwidth Estimation	Differentiation of Losses	Rate Control	Bursty Traffic Support	Average Percentage of Improvement compared to CoAP
Basic CoAP [3]	BEB	None	No	No	No	No	No	-
CoCoA [5]	BEB	Strong and Weak	Yes	No	No	No	No	19% [5]
CoCoA+ [6]	VBF	Strong and Weak	Yes	No	No	No	No	19% [6]
RCOAP	VBF	Strong and Weak	Yes	Yes	Yes	Yes	Yes	35%

i.e., around 10% higher. This was because using discovery packets, the sender was able to adjust the sending rate close to the estimated bottleneck bandwidth. After the simulation time of 50 s, the number of received data packets was around 138. Two discovery packets were used during the first RTT. Thus, the overhead cost for discovery packets was $2 * 41 \text{ bytes} / (138 * 106 \text{ bytes}) = 0.56\%$.

In the case of losses, the sender would send a discovery packet together with one data packet within one RTT during the loss detection phase. Thus, the total number of discovery packets will be $n * \text{RTT}/2$, where n is the number of loss events during the session. Since the estimated RTT is much smaller than the session duration for a burst transfer, the overhead cost mainly depends on the number of loss events.

V. CONCLUSION

Congestion control is still a challenging problem for IoT networks because of the constrained IoT environment. Suitable control mechanisms should be sufficiently simple, but efficient. For this reason, CoAP was proposed and standardized. However, there are some performance problems with basic CoAP and its variants for bursty traffic. In the case of regular traffic, these schemes can provide good performance. Nevertheless, basic CoAP and its variants show poor performance in terms of the throughput, delay, and retransmission attempts for bursty traffic. The reason is that they mainly focus on RTT measurements and RTO adjustments for retransmissions.

In this paper, we propose a new rate control scheme called RCOAP for bursty data transfer in IoT networks. The key idea of RCOAP is to regulate the transmission rate according to the available bandwidth of the bottleneck link. RCOAP permits a number of *in-flight* packets with reference to the bandwidth-delay product. The RCOAP sender estimates the appropriate initial rate by sending discovery packets to probe the link bandwidth at the beginning of the connection. During the connection, the rate is either additively increased or multiplicatively decreased depending on the predicted congestion state. Furthermore, RCOAP can distinguish between congestion losses and temporal wireless losses by sending discovery packets to probe the availability of the wireless link.

We compared RCOAP against basic CoAP, CoCoA, and CoCoA+ in bursty traffic scenarios using competing flows. The results demonstrated that RCOAP is efficient for bursty data transfer in terms of the throughput, delay, packet loss

ratio, delivered packets, and retransmission attempts. In particular, RCOAP can provide a higher throughput of more than 135% compared to basic CoAP, CoCoA, and CoCoA+ while maintaining a lower delay, lower loss rate, and lower number of retransmission attempts. Moreover, RCOAP also provides stable performance even under high-load conditions. Further extensions of this work could study the performance of a large network, as well as scenarios with mixed TCP traffic.

ACKNOWLEDGMENT

The authors thank the Posts and Telecommunications Institute of Technology (PTIT) for the facilities. They thank the anonymous reviewers for their valuable comments.

REFERENCES

- [1] C. Bormann and Z. Shelby, *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*. Accessed: Jun. 24, 2021. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc7959>
- [2] C. Suwannapong and C. Khunboa, "Congestion control in CoAP observe group communication," *Sensors*, vol. 19, no. 15, p. 3433, Aug. 2019, doi: 10.3390/s19153433.
- [3] *The Constrained Application Protocol (CoAP)*, document RFC 7252, Accessed: Jun. 24, 2021. [Online]. Available: <https://tools.ietf.org/html/rfc7252>
- [4] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoAP congestion control for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 154–160, Jul. 2016.
- [5] C. Bormann, A. Betzler, C. Gomez, and I. Demirkol, (Feb. 2018). *CoAP Simple Congestion Control/Advanced, Internet-Draft*. Accessed: Jul. 24, 2021. [Online]. Available: <https://tools.ietf.org/id/draft-bormann-core-cocoa-03.txt>
- [6] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoCoA+: An advanced congestion control mechanism for CoAP," *Ad Hoc Netw.*, vol. 33, pp. 126–139, Oct. 2015.
- [7] S. Deshmukh and V. T. Raisinghani, "AdCoCoA-adaptive congestion control algorithm for CoAP," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Kharagpur, India, Jul. 2020, pp. 1–7.
- [8] P. Aimtongkham, P. Horkaew, and C. So-In, "An enhanced CoAP scheme using fuzzy logic with adaptive timeout for IoT congestion control," *IEEE Access*, vol. 9, pp. 58967–58981, 2021.
- [9] S. Boletieri, G. Tanganelli, C. Vallati, and E. Mingozzi, "pCoCoA: A precise congestion control algorithm for CoAP," *Ad Hoc Netw.*, vol. 80, pp. 116–139, Nov. 2018.
- [10] M. A. Tariq, M. Khan, M. T. R. Khan, and D. Kim, "Enhancements and challenges in CoAP—A survey," *Sensors*, vol. 20, pp. 1–29, Nov. 2020, doi: 10.3390/s20216391.
- [11] J. J. Lee, K. T. Kim, and H. Y. Youn, "Enhancement of congestion control of constrained application protocol/congestion control/advanced for Internet of Things environment," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 11, pp. 1–13, Nov. 2016.
- [12] E. Ancillotti and R. Bruno, "BDP-CoAP: Leveraging bandwidth-delay product for congestion control in CoAP," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, Limerick, Ireland, Apr. 2019, pp. 656–661.

- [13] E. Ancillotti, R. Bruno, C. Vallati, and E. Mingozzi, "Design and evaluation of a rate-based congestion control mechanism in CoAP for IoT applications," in *Proc. IEEE 19th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Chania, Greece, Jun. 2018, pp. 14–15.
- [14] W. U. Rahman, Y.-S. Choi, and K. Chung, "Performance evaluation of video streaming application over CoAP in IoT," *IEEE Access*, vol. 7, pp. 39852–39861, 2019.
- [15] M. Boucadair and J. Shallow. (May 2021). *Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission, Internet-Draft*. Accessed: Jun. 20, 2021. [Online]. Available: <https://tools.ietf.org/id/draft-ietf-core-new-block-14>
- [16] J. H. Jung, M. Gohar, and S. J. Koh, "CoAP-based streaming control for IoT applications," *Electronics*, vol. 9, no. 8, 2020, Art. no. 1320, doi: 10.3390/electronics9081320.
- [17] H. Jiang, Q. Li, Y. Jiang, G. Shen, R. Sinnott, C. Tian, and M. Xu, "When machine learning meets congestion control: A survey and comparison," *Comput. Netw.*, vol. 192, Jun. 2021, Art. no. 108033.
- [18] H. Haile, K.-J. Grinnemo, S. Ferlin, P. Hurtig, and A. Brunstrom, "End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks," *Comput. Netw.*, vol. 186, Feb. 2021, Art. no. 107692.
- [19] C. Gomez, A. Arcia-Moret, and J. Crowcroft, "TCP in the Internet of Things: From ostracism to prominence," *IEEE Internet Comput.*, vol. 22, no. 1, pp. 29–41, Jan./Feb. 2018.
- [20] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, *The NewReno Modification to TCP's Fast Recovery Algorithm*, document RFC 6582, Apr. 2012. [Online]. Available: <https://tools.ietf.org/html/rfc6582>
- [21] K. Ramakrishnan and S. Floyd, *A Proposal to Add Explicit Congestion Notification (ECN) to IP*, document RFC 2481, Jan. 1999. [Online]. Available: <https://tools.ietf.org/html/rfc2481>
- [22] G. Buchholz, T. Ziegler, and T. Van Do, "TCP-ELN: On the protocol aspects and performance of explicit loss notification for TCP over wireless networks," in *Proc. 1st Int. Conf. Wireless Internet (WICON)*, 2005, pp. 172–179.
- [23] T. Saedi and H. El-Ocla, "TCP CERL+: Revisiting TCP congestion control in wireless networks with random loss," *Wireless Netw.*, vol. 27, no. 1, pp. 423–440, Jan. 2021.
- [24] P. Thubert, *IPv6 Over Low-Power Wireless Personal Area Network (6LoWPAN) Selective Fragment Recovery*, document RFC 8931, Nov. 2020. [Online]. Available: <https://tools.ietf.org/html/rfc8931>
- [25] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann, *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, document RFC 6775, Nov. 2012. [Online]. Available: <https://tools.ietf.org/html/rfc-6775>
- [26] F. Filali, "Link-layer fragmentation and retransmission impact on TCP performance in 802.11-based networks," in *Proc. IFIP Mobile Wireless Commun. Netw. Conf. (MWCN)*, 2005, pp. 1–6.
- [27] X. Vilajosana, K. Pister, and T. Watteyne, *Minimal IPv6 Over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*, document RFC 8180, May 2017. [Online]. Available: <https://tools.ietf.org/html/rfc8180>
- [28] T. Szigeti, J. Henry, and F. Baker, *Mapping Diffserv to IEEE 802.11*, document RFC 8325, Feb. 2018. [Online]. Available: <https://tools.ietf.org/html/rfc8325>
- [29] N. Benamar, J. Harri, J. Lee, and T. Ernst, *Basic Support for IPv6 Networks Operating Outside the Context of a Basic Service Set Over IEEE Std 802.11*, document RFC 8691, Dec. 2019. [Online]. Available: <https://tools.ietf.org/html/rfc8691>
- [30] S. Biaz and N. H. Vaidya, "Distinguishing congestion losses from wireless transmission losses: A negative result," in *Proc. 7th Int. Conf. Comput. Commun. Netw.*, 1998, pp. 722–731.
- [31] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.
- [32] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 703–717, Oct. 2003.
- [33] I. Jarvinen, M. Kojo, I. Raitahila, and Z. Cao, *Fast-Slow Retransmission Timeout and Congestion Control Algorithm for CoAP*, document RFC. [Online]. Available: <https://tools.ietf.org/id/draft-jarvinen-core-fasor-02.html>
- [34] Y. Chen, L. Lu, X. Yu, and X. Li, "Adaptive method for packet loss types in IoT: An Naive Bayes distinguisher," *Electronics*, vol. 8, no. 2, p. 134, Jan. 2019.
- [35] S. Biaz and N. H. Vaidya, "'De-randomizing' congestion losses to improve TCP performance over wired-wireless networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 596–608, Jun. 2005.
- [36] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, "Achieving moderate fairness for UDP flows by path-status classification," in *Proc. 25th Annu. IEEE Conf. Local Comput. Netw. (LCN)*, Tampa, FL, USA, Nov. 2000, pp. 252–261.
- [37] C. Lim, "A survey on congestion control for CoAP over UDP," *Int. J. Internet, Broadcast. Commun.*, vol. 11, no. 1, pp. 17–26, 2019.
- [38] *NS-3 Network Simulator*. [Online]. Available: <https://www.nsnam.org/>
- [39] S. Maesoser. *A Partial CoAP Implementation With mDNS Support, Multicast*. [Online]. Available: <https://github.com/maesoser/ns3-coap/>



DANG HAI HOANG received the Diploma Ing. (M.Eng.) degree in technical cybernetics and automation and the Dr.-Ing. and Dr.-Ing.habil. degrees in telematics and communication systems from the Technical University of Ilmenau, Germany, in 1984, 1999, and 2002, respectively.

Since 2009, he has been an Associate Professor at the Posts and Telecommunications Institute of Technology, Hanoi, Vietnam. His current research interests include information security, communication protocols, communication systems, QoS mechanisms, and control systems.



THI THUY DUONG LE received the B.S. degree in telecommunications and electronic engineering and the M.S. degree from the Technical University of Hanoi, Vietnam, in 2002 and 2008, respectively. She is currently pursuing the Ph.D. degree with the Posts and Telecommunications Institute of Technology, Hanoi, Vietnam.

Since 2005, she has been a Lecturer with the Faculty of Information Technology, University of Civil Engineering, Hanoi. She is a Senior Lecturer. Her current research interests include computer and communication systems, wireless sensor networks, QoS mechanisms, and network performance.

•••