International Journal of Communication Networks and Information Security
ISSN: 2073-607X, 2076-0930
Volume ***14*** *Issue* ***02 Year 2022*** *Page* ***111:123***

A Survey on Congestion Control Protocols for CoAP

[1] Sneha R Deshmukh, [2]Vijay T Raisinghani
Department of Information Technology, Mukesh Patel School of Technology Management and Engineering,
NMIMS Deemed-to-be University, Mumbai, India
[1] sneha.deshmukh@nmims.edu, [2] vijay.raisinghani@nmims.edu

Article History

Received: 13 Feb 2022
Revised: 26 May 2022
Accepted: 12 August 2022

Abstract

The Internet of things (IoT) comprises things interconnected through the internet with unique identities. Congestion management is one of the most challenging tasks in networks. The Constrained Application Protocol (CoAP) is a low-footprint protocol designed for IoT networks and has been defined by IETF. In IoT networks, CoAP nodes have limited network and battery resources. The CoAP standard has an exponential backoff congestion control mechanism. This backoff mechanism may not be adequate for all IoT applications. The characteristics of each IoT application would be different. Further, the events such as unnecessary retransmissions and packet collision caused due to links with high losses and packet transmission errors may lead to network congestion. Various congestion handling algorithms for CoAP have been defined to enrich the performance of IoT applications. Our paper presents a comprehensive survey on the evolution of the congestion control mechanism used in IoT networks. We have classified the protocols into RTO-based, queue-monitoring, and rate-based. We review congestion avoidance protocols for CoAP networks and discuss directions for future work.

Keywords: CoCoA+, pCoCoA, AdCoCoA, CoAP-Eifel

1. Introduction

The internet of things (IoT) is a network of devices (physical and virtual) that can auto-configure. The protocols used for this network are standard and enable interoperability [33]. Further, the network is dynamic. These IoT devices exchange data over a network, which may have lossy links, packet transmission errors, and poor throughput. These conditions could lead to packet losses and unnecessary packet retransmissions, which could cause congestion in the network. Hence, a proficient congestion handling technique is required to reduce uncalled-for resending and determine the correct retransmission timeout (RTO) value.

In the IoT protocol stack, Constrained Application Protocol (CoAP) [1] has reliable transmission as an option utilizing RTO and aggressive backoff. UDP is the transport CoAP protocol for CoAP. CoAP is designed for light applications which require congestion handling. CoAP has a request and response model for end-to-end operation. The RTO is initially set at random. After that, binary exponential backoff (BEB) is used for calculating the RTO. However, this random value at the start can cause retransmissions which could cause avoidable congestion. We broadly classify the congestion control mechanisms for CoAP networks into three categories – RTO-based, queue-monitoring, and rate-based.

RTO-based mechanisms: CoCoA [2] calculates RTO by using the value of round-trip time

(RTT). CoCoA+ [3] has step-function techniques with scaling factors to calculate the RTO for retransmission. CoCoA-E [4] is a modified version of CoCoA+, which uses a new scaling factor based on the Eifel retransmission timer. Bhalerao et al. [5] have further improved the RTO estimation mechanism of CoCoA+ using an adaptive technique based on retransmission attempts. Lee et al. [6] have defined a method for congestion control that uses RTT. This mechanism considers retransmission count for accurately measuring the RTT value and estimating the RTO value. pCoCoA [7] estimates the RTT precisely using timestamps and a modified RTO estimation mechanism. Hence, it helps minimize unnecessary retransmissions.

An improved adaptive CoAP [8] determines the RTO value using packet loss ratio and RTT. RTT-CoAP [9] has a novel approach to detecting congestion in the network. It uses the checks variance of RTT against predefined thresholds for loss of COAP messages. CoAP Eifel [10] is a modified version of CoAP which uses only strong RTT to estimate the RTO value. In AdCoCoA [11], link quality, delay, and RTT deviation are considered to calculate the RTO value. CACC [12] categorizes RTT into three types, as per the receipt of an ACK – without retransmission (*strong*) or with retransmission (*weak*; one or more retransmissions) or no ACK at all (*failed*), to determine the condition of the network, and adapt the congestion control mechanism. FASOR [13] determines whether the loss of packets is because of poor links or congestion in the network. FASOR has three distinctive features – an adaptation of the RTO backoff and RTO computation which can be fast or slow. DCC-CoAP [14] efficiently predicts network congestion; it combines the separation of nodes and multiple RTT measurements, reducing CoAP message loss. CoCoA++ [15] uses the delay gradients method to measure network congestion and introduces a new backoff mechanism called probabilistic backoff to deal with congestion.

Queue monitoring mechanisms: In a Bird flock model [16], a node monitors its neighbor node's buffer at each hop to avoid congestion. The backpressure congestion-control mechanism [17] uses the MAC and network layer's information to prevent congestion at each hop. Rate-based mechanism: CoAP-R [18] is designed to utilize the maximum available bandwidth and uses max-min fairness to allocate bandwidth. BDP-CoAP [19] design copes with lossy links and problems of unfair access over the *short-term*, which typically occurs in IoT networks.

The paper contents are as follows – Section II provides an outline of the CoAP protocol. Review of various CoAP congestion handling techniques is in section III. Section IV presents further research directions and the conclusion.

2. Constrained Application Protocol (CoAP)

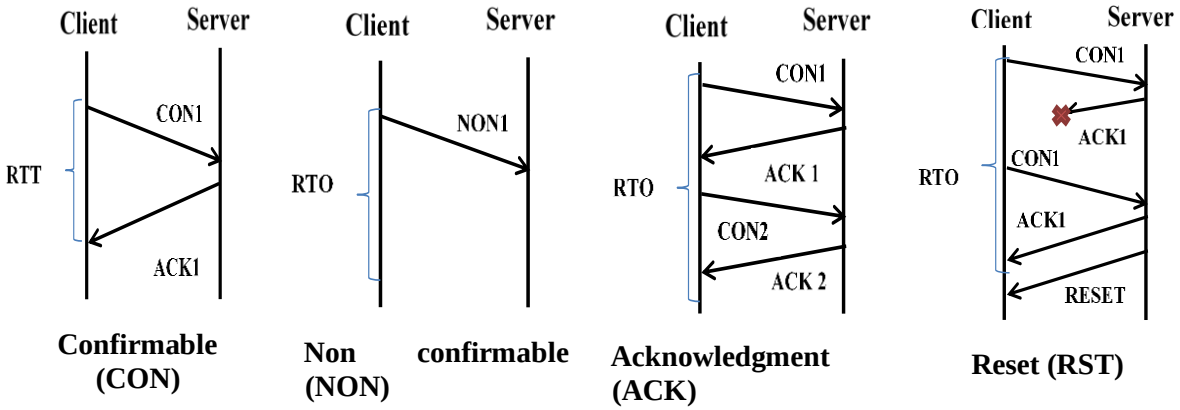

Figure 1: CoAP message types

CoAP (RFC 7252) [1] has a client-server type of model with requests and responses. CoAP uses a stop-and-wait approach, retransmissions, and exponential backoff for confirmable (CON) messages. Figure 1(sourced from reference [1]) shows four types of CoAP messages: Reset (RST), Acknowledgement (ACK), Non-confirmable (NON), and Confirmable (CON). A message identifier helps in duplicate identification. An ACK is returned for a CON message. ACK is not required for NON. RST from the receiver to the sender indicates that the receiver could not process the message.

CoAP implements an optional reliability mechanism. When no ACK not received for a transmitted CON, the sender retransmits the message.

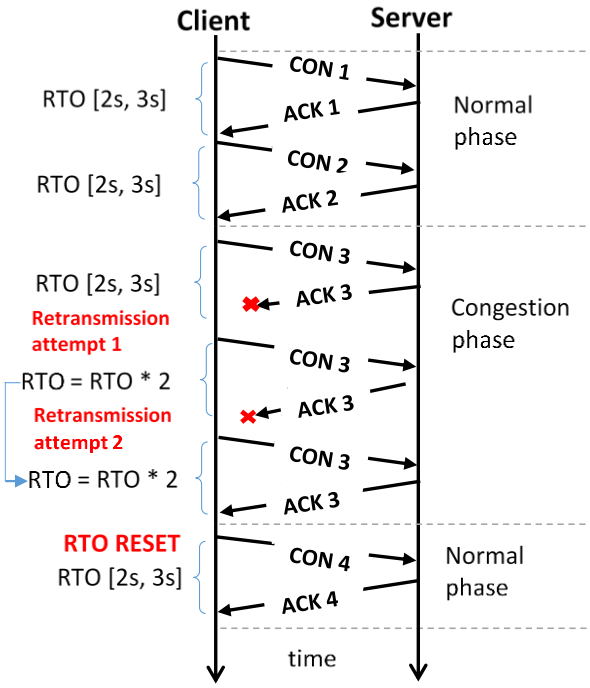

Figure 2: Working Binary exponential backoff (BEB) in CoAP

On sending a CON message, the sender chooses an initial value from 2s to 3s for the RTO. The sender assumes the CON message is lost. Suppose an ACK is not received from the destination before the expiry of the RTO timer. The sender does retransmission and doubles the current RTO value to reduce needless retransmissions to avoid congestion, as shown in figure 2 (sourced from reference [1]). The sender does not consider RTT to compute the RTO. As a result, the RTO value would tend to be inaccurate and could lead to unnecessary retransmissions. The RTO value will grow exponentially if the retransmissions increase, leading to an increase in the retransmission delay.

The following section discusses the evolution of the various techniques for congestion control for IoT (CoAP) networks.

3. Congestion control mechanisms for CoAP

We have classified the mechanisms based on the approach used to control network congestion – RTO, queue-monitoring, and rate-based mechanisms. The protocols CoCoA [2], CoCoA+ [3], CoCoA-E [4], 4-State Strong CoCoA [5], Enhanced CoCoA [6], pCoCoA [7], RTT-CoAP [9], improved adaptive CoAP [8], CACC [12], FASOR [13], CoCoA++ [15], CoAP Eifel [10], AdCoCoA [11], DCC-CoAP are RTO based congestion control mechanisms. Back pressure [17] and bird flock [16] are queue-monitoring-based congestion control mechanisms. The protocols CoAP-R [18] and BDP-CoAP [19] are rate-based congestion control mechanisms.

3.1 CoCoA

In contrast to CoAP [1], the CoCoA [2] [26] sender measures RTT values to determine RTO before transmitting the first CON message. RTT is the time from sending a CON message to receipt of an ACK for this message. CoCoA computes two RTO for every receiver. *Weak* and *strong* RTO values calculations use a weighted average of the weak and strong RTTs. A node maintains separate *strong* and *weak* values for $RTT_{current}$, $RTTVAR_{current}$ (RTT variance) and $RTO_{current}$ for each destination, where $RTT_{current} = (1-\alpha) * RTT_{previous} + \alpha * RTT_{current}$; $RTTVAR_{current} = (1-\beta) * RTTVAR_{current} + \beta * |RTT_{previous} - RTT_{current}|$; and $RTO_{current} = RTT_{current} + K * RTTVAR_{current}$. The smoothing factors α, β, and K values

are 0.25, 0.125, and 4, respectively. The last updated RTO is considered $RTO_{current}$, and a weighted average of RTOprevious and RTOcurrent is used to calculate the $RTO_{overall}$, where $RTO_{overall\_current}$= 0.5* $RTO_{current}$ + 0.5 * $RTO_{overall\_previuos}$. RTO_{init} is a random value between 1 to 1.5 times $RTO_{overall}$, for the first transmission. The RTO value is doubled for every retransmission.

3.2 CoCoA+

CoCoA+ [3], an improved version of CoCoA, consists of an RTO estimator, a variable backoff factor (VBF) policy for retransmission attempts, and an aging mechanism for updating an obsolete RTO value. The RTT, RTTVAR, and RTO computations of CoCoA+ are the same as CoCoA, with a few modifications added, as shown in figure 3[3]. Following are the improvements introduced in CoCoA+ [3] (sourced from reference [3]). (1) RTT change: For weak RTT computation, the value of smoothing factor (K) changed from 4 to 1. (2) RTO change: for calculating $RTO_{overall}$, *weak* RTO weight changed from 0.5 to 0.25. (3) Backoff policy change: For the retransmission, the old RTO value is multiplied by a value ranging from 1 to 3 to determine RTO_{new}. If the RTOprevious is less than 1s, between 1-3 s, or greater than 3s, then it is multiplied by 3, 2, or 1.3, respectively. (4) RTO aging: To update an obsolete RTO value, aging that updates the RTO at regular intervals. If RTOcurrent > 2s has not changed for the past 30s, then it is changed to (2 +RTOcurrent)/ 2.

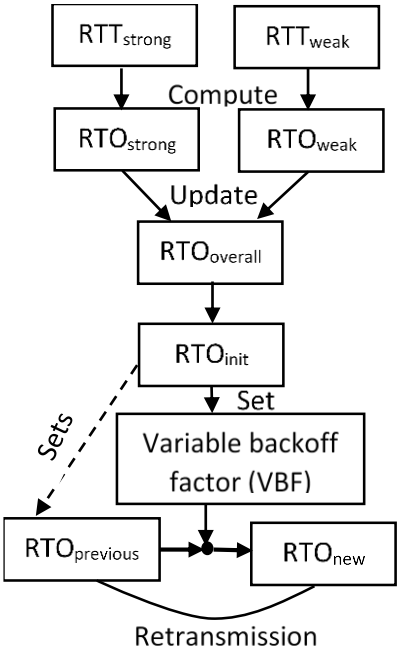

Figure 3: CoCoA+ mechanism

3.3 CoCoA-E

CoCoA-E [4] is an improvement on CoCoA+, which estimates RTO using the retransmission timer of Eifel [21][22], which had initially been proposed for estimating TCP timeouts. The authors, while analysing the Eifel retransmission timer for a large sending rate, have claimed that the standard values of α (1/4), β (1/8), and K (4) (RFC 6298 [23]) would not be perfect. Hence, they suggest a different technique for determining RTO using a new factor γ rather than α and β. γ = RTT/RTO. If γ is less than 0.5, then γ is the smoothing factor; else, 1- γ is used. The RTO value adjustment is as per the value of γ. The smoothed RTT and RTT variance calculations are the same as CoCoA[2].

3.4 CoCoA -4 states

Bhalerao et al. [5] have improved CoCoA+ [3] using a state estimator with an adaptive variable backoff factor policy for retransmissions. The protocol maintains the state of retransmissions for each source-destination pair. A transaction starts with state one. Each time a CoAP packet is acknowledged before the RTO timer expires, the transaction's condition is unchanged. On each retransmission, the state value is increased by 1 to a maximum of 4.

Variable backoff factors and weight control the amount of time a transaction must wait before retransmitting a packet after a timeout. Each time a retransmitted packet is acknowledged before the RTO timer expires; the transaction's state is lowered. bf1, bf2, bf3, and bf4, are the backoff factors used for each transaction, with the value increasing from bf1 to bf4. Based on the RTO estimate and the current state of the transaction, an appropriate variable backoff factor is used to prevent excessive backoff. In addition, a weight w is used, which increases according to the transaction state; for computing the $RTO_{current}$ value, $RTO_{current} = w * RTO_{current} + (1-w) * RTO_{previous}$.

3.5 Enhanced CoCoA

The formula to compute RTT and RTTVAR is the same as CoCoA, but this mechanism does not maintain strong and weak RTTs. Lee et al. [6] have proposed a scheme that improves CoCoA+ by introducing a packet's *retransmission count* (RC) to estimate every packet's RTT correctly. The RC field in the CoAP packet helps to measure the RTT accurately. Hence, it improves the accuracy of RTO estimation and thus minimizes unnecessary retransmissions. In addition, a fixed lower bound for RTTVAR is used while computing the RTO to ensure that the RTT and RTO values are not close. The mechanism for aging the RTO and backoff using varied factors is the same as CoCoA+ [3].

3.6 pCoCoA

In contrast to CoCoA+ [3], pCoCoA [7] eliminated the use of *weak* RTO estimation, modified RTTVAR and smoothed RTT (*SRTT*) formulas, and introduced a dynamic smoothing factor for computing an accurate RTO value to reduce spurious retransmissions. A *transmission count* (TC) parameter is included in the CoAP message header for weak RTO elimination, which maps a specific ACK to its CON message. TC=1, when a sender has sent a CON message. TC increments by one on every retransmission until the TC equals the maximum retransmission count (value=4). The receiver copies this TC value in the corresponding ACK. CON and its TC-mapped ACK are used to compute the RTT precisely. If the sender sends the packet for the first time, pCoCoA uses the default CoAP RTO mechanism. Subsequently, it uses the VBF backoff policy for retransmissions and the RTO aging mechanism borrowed from CoCoA+. If there is a difference between the TC value of a CON and the ACK, it implies that spurious retransmission occurred; the RTTVAR is multiplied by scaling factor K=6, else K=4.

For each destination, a node maintains $SRTT_{current}$, $RTTVAR_{current}$, $SRTO_{current}$, and $RTO_{current}$. The values of smoothing factors (SF) $\alpha=1/8$, $\beta=1/4$, $\Upsilon=1/32$, and $\delta=1/2$. pCoCoA selects one of the smoothing factors (α, β, and Υ) for estimating RTTVAR based on the RTT variance in the network. The SRTT, RTTVAR and SRTO are estimated as follows: $SRTT_{current} = (1- \alpha)\ SRTT_{previous} + \alpha * RTT_{current}$; $RTTVAR_{current} = (1- SF)\ RTTVAR_{previous} + SF * |SRTT_{current} - RTT_{current}|$; $SRTO_{current} = SRTO_{previous} + \max(K*RTTVAR_{current}, mdev_{max})$; $RTOcurrent = (1-\delta) * SRTO_{current} + \delta * RTO_{previous}$. The $mdev_{max}$ is a lower bound to limit the influence of fluctuation in RTTVAR on the RTO computation. Suppose $RTTVAR_{previous}$ is less than the gap between SRTT and RTT. In that case, the lower weight (β) applies to $RTTVAR_{previous}$ or higher weight (α), which helps suppress the effect of a sudden change in RTT variance in the network. These weights will help to estimate a large enough RTO value to minimize spurious retransmissions. The $mdev_{max}$ increases gradually by a small amount and decreases cautiously based on the RTTVAR to ensure the RTO value is large enough to prevent spurious retransmissions.

3.7 RTT CoAP

RTT-CoAP [9] is an enhancement of CoAP, which uses a rate-based algorithm. RTT-CoAP aims to reduce traffic burstiness and packet loss by controlling the interarrival time of packets transmitted. This protocol considers only RTT measurements (r) without retransmission and uses timestamps to measure RTT precisely. A CoAP sender maintains *long-term* RTT ($SRTT_L$), *short-term* RTT ($SRTT_S$), and RTT variance in this algorithm. The variables $SRTT_L$, $SRTT_S$ and $RTTVAR_L$ are computed as follows: $SRTT_S=\alpha_S*r+ (1-\alpha_S) * SRTT_S$; $SRTT_L=\alpha_L*r + (1-\alpha_L) * SRTTS$; and $RTTVARL = \beta_L * |SRTT_L-r| + (1-\beta_L) * RTTVAR_L$. A larger smoothing factor (α_S) is used to smooth out the short-term RTT variations due to spikes in the network. A smaller smoothing factor (α_L) is used to reduce the impact of noise on RTT measurements while estimating long-term RTT. The RTT variance determines the state of the

network state and estimates the rate for sending. RTT-CoAP has defined four network states corresponding to varying levels of congestion from low to high.

In a low congestion state, the sending rate increases aggressively with no change observed in RTT variance. In a normal state, RTT variations and packet loss ratio are determined to decide the sending rate. In a medium variability state, if short-term RTT variance grows, the network might get congested, and sending rate is decreased. In a high variability state, the network is congested if the value of short-term RTT is greater than that of the long-term RTT value. Hence, the sending rate decreases aggressively. RTT-CoAP increases or decreases the sending rate by a fixed amount to adjust the packet transmission speed.

3.8 Improved Adaptive CoAP

Ouakasse and Rakrak's protocol [8] uses the fraction of packets lost (PLR) to adjust the value of RTO for every retransmission. Initially, the RTO value is set to a random value ranging from 2 to 3 seconds. The CoAP sender sends a CON message; on receiving the ACK, it estimates PLR. If the PLR is less than 50% then RTO_{recent} = RTT * PLR + (1- PLR) * $RTO_{previous}$. If the more than 50% of the packets are lost, then RTO_{recent} = $RTO_{previous}$ * PLR+ (1- PLR) * RTT. The PLR is used to determine the condition of the network. Using PLR helps estimate an RTO that is large enough to avoid excessive waiting time (if RTO is larger than RTT) and unnecessary retransmissions (if RTO is close to RTT).

3.9 CoCoA++

In contrast to CoCoA+ [3], CoCoA++ [15] monitors network conditions for fixed interval and uses a delay gradient mechanism [29] to estimate RTO. The RTO estimation mechanism is borrowed from CoCoA+. Two gradients help to monitor RTT variations. CoCoA++ updates RTO when periodically the delay gradients are estimated. CoCoA++ has introduced a new probabilistic backoff factor (PBF) mechanism to estimate the backoff probability.

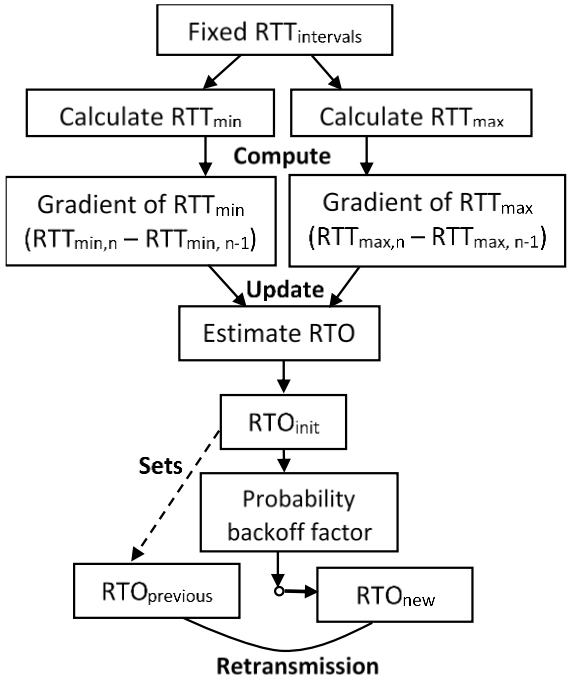

Figure 4: CoCoA++ mechanism (source: reference [15])

As shown in figure 4 [15] (figure derived from [15]), CoCoA++ stores the RTT samples and computes RTTmin and RTTmax within an interval of time n. The minimum and maximum values obtained from RTT samples are termed RTT_{min} and RTT_{max}. The gradients g_{min} and g_{max} help track the RTT variance change for RTT_{min} and RTT_{max}, respectively, over n and n-1. $\bar{g}_{min}$ and $\bar{g}_{max}$ denote average gradients calculated over multiple time intervals. The protocol uses the RTT variance difference to compute the backoff probability. The probability increases as change the RTT variance increases.

If RTT_{max} equals the maximum possible value, and RTT_{min} growth has not stopped ($g_{min}>0$), it indicates congestion. In this situation, the congestion window will be decreased by 30%, increasing by 1. Further, it will reduce the load in the network and minimize unnecessary retransmission.

3.10 FASOR

An RTO-based congestion control mechanism named FASOR [13] is capable of working in *buffer bloat* [30] conditions and copes with high link error rate scenarios. FASOR [27] consists of three components: Fast and Slow RTO estimation and a novel retransmission timer backoff mechanism. If the RTT samples are definite, then a fast RTO calculation is used. If the RTT samples are unclear, then a slow RTO computation is done to prevent excessive packet buffering and extreme congestion. This approach controls the flow, reduces delays, and handles cases with link errors.

The RTT variation initialization formula is modified RTTVAR = RTT measurement / 2K, where K is a scaling factor (value =4) for achieving fast convergence with minimum exchanges. The Fast RTO calculation is derived from TCP RTO calculation [21], but there is no lowest value fixed for the RTO. The RTO value is initially kept at 2 seconds and considers actual RTT samples. Slow RTO considers ambiguous RTT samples. For every retransmission, Slow RTO increases the RTO by 1.5 times.

Karn's algorithm and Slow RTO work similarly. The FASOR backoff mechanism has three states of backoff. If the sender starts with the least conservative state (FAST), it sends the packet and waits for ACK. On successful ACK, the sender will remain in the least conservative state. If the sender requires retransmissions, it uses a backoff factor and retransmits the packet. If more than one retransmission occurs, the sender goes to the intermediate conservative state. In this state, the backoff timer is large enough to minimize the retransmissions.

Further, if more retransmissions occur, the sender goes to the most conservative states. In this state, the sender uses a larger backoff factor to minimize the retransmissions. As the number of retransmissions reduces, the sender gradually returns to the least conservative from the most conservative state, which minimizes unnecessary retransmissions.

3.11 Context-Aware Congestion Control

Context-aware congestion control (CACC) [12] has three RTO estimators to determine whether the loss of packets loss caused by collision or congestion. The RTO calculations are based on whether the packet transmission is successful, delayed, or failed. Correspondingly, *strong*, *weak*, or *failed* RTT is determined. If there are many *strong* or *failed*, RTTs collision is assumed. If there are many *weak* and *failed* RTTs, then congestion is assumed.

CACC measures RTT accurately using a retransmission counter. This mechanism runs three RTO estimators – *strong* RTT, *weak* RTT, and *failed* RTT. A *strong* RTT means on receipt of an ACK for a CON message without any retransmission. A *weak* RTT means receiving an ACK with packet retransmissions; *failed* RTT is obtained when ACK is not received even after maximum retries for packet transmission. The computation is the same as in CoCoA+ [3] for the first RTT measurement, smoothed RTT, RTTVAR, and RTO.

The terms *SR*, *WR*, and *FR*, represent the number of times *strong*, *weak*, and *failed* RTT has been determined over the last four transmissions. If *FR* is higher than *SR*, it reflects collision in the network. The weighted average of *failed* and *strong* RTT estimates smoothed RTT. If *WR* is higher than *FR*, there is congestion at the nodes. The weighted average of *failed* and *weak* RTT estimates smoothed RTT. If *SR* is higher than *WR* or *FR*, then the weighted average of *strong* and *weak* RTT estimates smoothed RTT. A weighted average of $RTTVAR_{previous}$ and the difference between RTT and SRTT is used to calculate RTTVAR. Scaling factors $\alpha=0.25$, $\beta=0.125$, and $K=4$ are used for calculating SRTT, RTTVAR, and RTO. A lower bound restricts the RTO value from becoming very low, which avoids spurious retransmissions of CON messages. The $RTO_{overall}$ and RTO aging mechanism is the same as that in CoCoA+.

3.12 CoAP Eifel

In contrast to CoAP[1], CoAP Eifel [10] sender considers RTT measurements obtained without any retransmission to compute RTO. The initial RTO value is *two* ticks*, where *ticks* are operating system-

dependent fractions of seconds. For RTO calculation, the node maintains smoothed RTT ($SRTT_{current}$), RTT variance ($RTTVAR_{current}$), and RTO value ($RTO_{current}$). A scaling factor GAIN = 1/3 estimates smoothed RTT and RTO. A scaling factor $\overline{GAIN}$ (either 1/3 or $(1/3)^2$) is used for estimating RTTVAR. If RTT variance is greater than zero, then a higher scaling factor (GAIN) is used; a lower scaling factor ($GAIN^2$) is used. The smoothed RTT and RTT variance are updated on every RTT measurement for the span of the connection. The sender assumes that the CON message is lost if an ACK is not received from the destination before the RTO timer expires. The sender initiates retransmission and uses binary exponential backoff to double the current RTO value.

3.13 AdCoCoA

AdCoCoA[11] improves CoCoA+ [3], and uses retransmission counter to calculate RTT. Figure 5 [11] shows the detailed working mechanism of the AdCoCoA protocol. AdCoCoA considers network conditions like link quality, link delay, and RTT deviations for estimating dynamic scaling factors. The dynamic scaling factors estimate smoothed RTT, RTT variance, and RTO values. AdCoCoA uses the standard CoAP RTO value for the first transaction. Subsequently, it uses the VBF backoff policy for retransmissions and the RTO aging mechanism borrowed from CoCoA+ [3]. A node maintains $RTT_{current}$, $RTTVAR_{current}$, and $RTO_{current}$ for each destination, as shown in figure 9. The change in RTT measured and recent RTT is used to determine δ (RTT change). Using δ, it estimates the scaling factors δα, δb, and δk, used to estimate $RTT_{current}$, $RTTVAR_{current}$, and $RTO_{current}$, respectively.

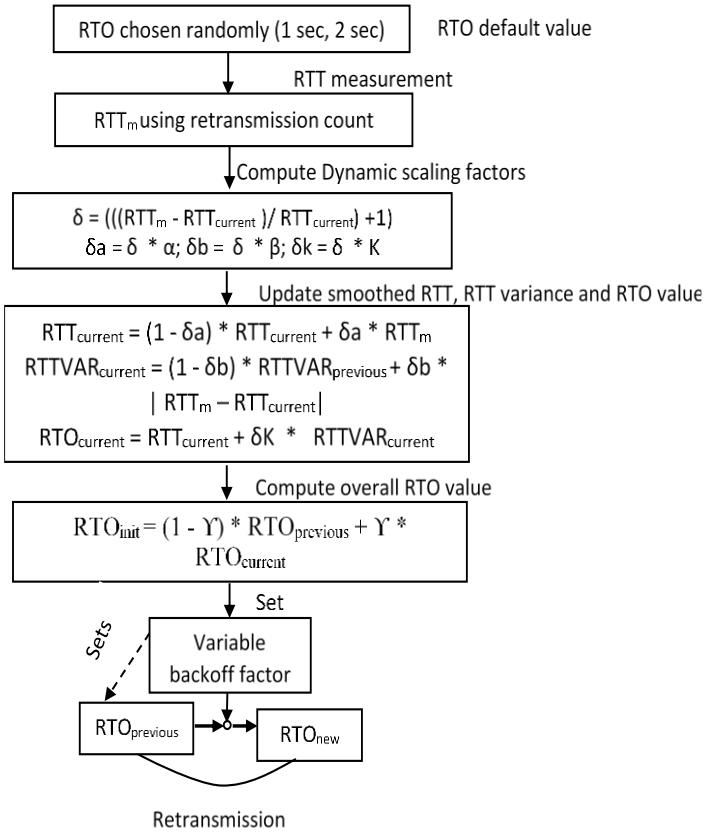

Figure 5: AdCoCoA working mechanism (source: reference [11])

3.14 Distance-based congestion control CoAP (DCC-CoAP)

DCC-CoAP [14] measures RTT using a timestamp and determines the distance between CoAP, the requestor, and the responder using the *Euclidean distance* formula [31]. The distance between two nodes is determined using the node's coordinates (x, y) (either requestor or responder). The scaling factor is determined based on this distance to estimate the weighted RTO value. If the node is far from the responder, then a large scaling factor is used; the small scaling factor is used to estimate weighted RTO. The final $RTO_{def} = RTT_{strong}$ + WRTO. DCC-CoAP uses a variable backoff factor mechanism for estimating RTO value on retransmissions, the same as CoCoA+ [3].

3.15 Genetic CoCoA++

Genetic CoCoA++ [28] is an improvised version of CoCoA++ [15], which uses genetic algorithm along with CAIA Delay-Gradient (CDG) [29] to estimate RTO. While CoCoA++ uses RTT_{min} and RTT_{max}, this protocol uses RTT_{min} only for computation which helps to achieve better results. RTT_{min} is the lowest value selected from the values measured over a fixed interval of 5 seconds. The difference between $RTT_{min\_current}$ and $RTT_{min\_previous}$ helps estimate the network congestion. The delay gradient and probabilistic backoff factor computation are the same as CoCoA++ [15].

Above, we described the RTO-based mechanisms. Next, we describe queue-monitoring mechanisms.

3.16 Bird Flocking Congestion Control Mechanism

Bird flock [16] is used to avoid congestion in CoAP/RPL [32] /6LoWPAN [20] networks, which operate at each hop. The Routing protocol for low-power and lossy networks (RPL) [32] creates an acyclic graph rooted at the destination node. Each node has a spherical model with two layers – zone of repulsion (ZoR) and zone of attraction (ZoA). The ZoR of a node are the immediate successors (one hop away), and the ZoA of a node contains two hop away nodes, the successor's successor.

Each node has a buffer for received packets. The current buffer status of nodes in ZoR and ZoA is communicated to the current node, which will help it select the next hop node for forwarding a packet. Each node maintains two parameters in its routing table for each successor node, the available buffer of the immediate successor (QsZoR – one-hop information) and the available buffer of the successor's successor (QsZoA – two-hop information). The criteria to select a successor for forwarding a packet is that the successor and the successor's successor should have the lowest congestion, as compared to others, both in ZoR and ZoA. If the successor has a higher packet transmission rate, then the congestion in ZoA is lower. The reciprocal of this success rate indicates the level of congestion (QsZoA). The successor with lesser QsZoA is selected.

For estimating QsZoR, the current node eavesdrops on the successors' activities. The node increases the QsZoR for a successor when its successor receives a packet. When the packet is sent ahead, the QsZoR is decreased. The nodes track the packets and ACKs using messages and identify packets retransmitted if the RTO expires before receipt of an ACKtted. On every retransmission, the RTO is recomputed as α * scaling factor. If the QsZoR is higher, then α will be higher, delaying the retransmission, thus decongesting the successor path.

3.17 Back Pressure Congestion Control Mechanism

Castellani et al.[17] implemented congestion control at the network layer using algorithms based on backpressure routing. This mechanism is designed for both CON and NON-messages. The backpressure techniques are implemented in the network layer, which the authors call a layer three device (L3D). The following are the types of L3Ds –*IdealBP*, *griping*, *deaf*, and *fuse*. All the nodes in a network will use the same L3D. Below we explain the mechanism of these L3Ds.

The *griping* device sends an explicit message to the sender to notify congestion if its queue length exceeds a predefined threshold on receiving a datagram. The *IdealBP* sender selects the next hop node for forwarding a datagram only if the next hop's queue length is lesser than that of the sender and has a predetermined threshold. The sender uses the Additive Increase Multiplicative Decrease (AIMD) policy transmission rate. If a deaf receiver's queue length exceeds a predefined threshold, it stops sending layer two acknowledgments.

The *deaf* sender handles layer three retransmissions using a backoff timer mechanism. A *fuse* device follows either agent *deaf* mechanism based on its queue size and predefined thresholds. If the queue length is less threshold, then the *fuse* receiver adopts *griping* behavior. Further, it adopts *deaf* device behavior when the network layer is full, does not send MAC layer ACKs, and sends an explicit notification to the sender as a *griping* device.

Above, we described the queue-monitoring mechanisms. Next, we describe the techniques which use sending rate for congestion control.

3.18 CoAP-R

COAP-R [18] is an enhancement of CoAP[1], which considers the transmission rate for controlling the sending rate of CoAP senders. A COAP-R generates a tree topology with the receiver node considered the root. Each CoAP node in the tree receives packets to send to the receiver. Each CoAP sender estimates the link's capacity to its ancestor node by measuring the time taken for the packet to leave the source and reach the receiver at the next hop. COAP-R runs two parallel link capacity estimators, one for a *long-term* link and another for a *short-term* one. The receiver node gathers the updates of least the bandwidth subtrees with active nodes. The receiver node uses a progressive filling algorithm [24] to fairly calculate the transmission rate for all senders in the tree and share transmission rate details with all the tree nodes.

3.19 BDP-CoAP

Bandwidth-Delay Product BDP-CoAP [19] considers transmission rate. It is an enhancement that considers bandwidth and RTT for estimating transmission rates. BDP-CoAP uses the highest and lowest transmission rate measurements to estimate bottleneck bandwidth. BDP-CoAP algorithm has three core functions (i) to transmit packets, (ii) to compute pacing gain (iii) to receive an acknowledgment. The algorithm considers the bandwidth measurement obtained on receiving an ACK without retransmission. The bandwidth-delay product is computed as throughput times RTT. If the packets are to be transmitted >= BDP, the sender waits for an ACK or RTO expiry; if RTO expires, it retransmits the packet. On receiving the ACK, the node updates bottleneck bandwidth, RTT, and a scaling factor pacing gain. Based on the retransmission count, the scaling factor value is determined to estimate the packet transmission rate of the node.

Table 1 shows a comprehensive assessment of the protocols discussed in this paper.

Table 1. Consolidated summary of CoAP protocols for handling congestion

Protocol	**Backoff mechanism**	**RTT estimators**	**RTO aging**	**Derived from**	**Scaling factors (S, D, F)***
CoAP	BEB	None	None	None	N
CoCoA	BEB	Strong and weak	Yes	LinuxRTO	F
CoCoA+	VBF	Strong and weak	Yes	CoCoA	F
CoCoA-S	VBF	Strong and weak	Yes	CoCoA	F
CoCoA-E	VBF	Strong and weak	Yes	CoCoA and Eifel	F
4-State-Strong	4-state VBF	Four estimators	Yes	CoCoA	F
Enhanced CoCoA	VBF	Strong	Yes	CoCoA	F
pCoCoA	VBF	Strong	Yes	CoCoA	F
RTT CoAP	Increase/decreases as per estimated δ	Strong	Yes	CoAP, and CoCoA	F
Improved Adaptive CoAP	VBF	Strong	Yes	CoCoA	D
CoCoA++	PBF	CAIA Delay-Gradient	No	CoCoA	F
FASOR	-	Strong	Yes	CoCoA	S and D
CACC	Dynamic RTO	Strong, weak, and failed	No	CoCoA	Sand D
CoAPefiel	-	Strong	Yes	CoAP	F

AdCoCoA	VBF	Strong	Yes	CoCoA	Sand D
DCC-CoAP		strong	yes	CoAP and CoCoA	Sand D
Genetic CoCoA++	PBF	CAIA Delay-Gradient	No	CoCoA	F
CoAP-R	Rate control	Monitoring RTT variations	No	CoAP and RTT monitoring	F
BDP-CoAP	Rate control	Strong	Yes	TCP BBR	F
***(N=None, S=Static, D=Dynamic, F=Fixed).**					

4. Conclusion

This paper comprehensively reviewed the progression of CoAP congestion control protocols. To understand the trends, we categorized the protocols into three categories based on the mechanism – RTO, queue-monitoring, and rate-based mechanism. All the RTO-based protocols aim to minimize the number of retransmissions attempts by estimating the RTO value appropriately. Whereas queue monitoring-based protocols aim to prevent congestion by controlling the packet-sending rate. The rate-based protocols consider the bandwidth available to control the senders' transmission rate. Based on the review of the existing protocols, we have identified some future research directions. While evaluating the current protocols, the authors had not considered the effects of wireless transmission errors and challenges in real networks. The existing protocols need further evaluation in the presence of realistic wireless transmission errors. As per the results presented in [10], a single protocol is inadequate for all scenarios. Hence, a hybrid protocol is needed to control congestion in various scenarios. For example, Hybrid TCP [33] has two modes – RTT monitoring is used to activate a specific method. RTO calculation based on the type of IoT application should be explored further. RTO could be estimated periodically by monitoring RTT changes for static applications like intelligent meter readings. For dynamic applications like connected cars, RTO could be evaluated after each RTT. Estimating the RTO value more precisely based on a single parameter might be insufficient. Considering other parameters (packet loss, delay, link loss, etc.) would help to determine RTO precisely and would also help to regulate the packet sending rate. Our analysis shows a strong need for a hybrid or flexible mechanism to handle congestion in CoAP networks.

References

[1] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP),"RFC 7252, p. 112, 2014.

[2] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "Congestion Control in Reliable CoAP Communication,"Proc. 16th ACM Int. Conf. Model. Anal. Simul. Wirel. Mob. Syst., pp. 365–372, 2013.

[3] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoCoA+: An advanced congestion control mechanism for CoAP,"Ad Hoc Networks, vol. 33, pp. 126–139, 2015.

[4] E. Balandina, Y. Koucheryavy, and A. Gurtov, "Computing the Retransmission Timeout in CoAP,"Internet Things, Smart Spaces, Next Gener. Netw., pp. 352–362, 2013.

[5] R. Bhalerao, S. Srinivasa Subramanian, and J. Pasquale, "An Analysis and Improvement of Congestion Control in the CoAP Internet-of-Things Protocol,"IEEE Annu. Consum. Commun. Netw. Conf., 2016.

[6] 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. Sens. Networks, vol. 12, no. 11, 2016.

[7] S. Bolettieri, G. Tanganelli, C. Vallati, and E. Mingozzi, "pCoCoA: A precise congestion control algorithm for CoAP,"Ad Hoc Networks, vol. 80, pp. 116–129, 2018.

[8] F. Ouakasse and S. Rakrak, "An improved adaptive CoAP congestion control algorithm,"Int. J. Online Eng., vol. 15, no. 3, pp. 96–109, 2019.

[9] E. Ancillotti, S. Bolettieri, and R. Bruno, "RTT-based congestion control for the internet of things," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2018, vol. 10866 LNCS, pp. 3–15.

[10] V. J. Rathod, S. Krishnam, A. Kumar, G. Baraskar, and M. P. Tahiliani, "Effective RTO estimation using Eifel Retransmission Timer in CoAP,"Proc. CONECCT 2020 - 6th IEEE Int. Conf. Electron. Comput. Commun. Technol., no. ii, 2020.

[11] S. Deshmukh and V. T. Raisinghani, "AdCoCoA- Adaptive Congestion Control Algorithm for CoAP,"2020 11th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2020, 2020.

[12] G. A. Akpakwu, G. P. Hancke, and A. M. Abu-Mahfouz, "CACC: Context-aware congestion control approach for lightweight CoAP/UDP-based Internet of Things traffic,"Trans. Emerg. Telecommun. Technol., vol. 31, no. 2, pp. 1–19, 2020.

[13] I. Jarvinen, I. Raitahila, Z. Cao, and M. Kojo, "FASOR Retransmission Timeout and Congestion Control Mechanism for CoAP," in 2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings, 2019.

[14] S. Bansal and D. Kumar, "Distance-based congestion control mechanism for CoAP in IoT,"IET Commun., vol. 14, no. 19, pp. 3512–3520, 2020.

[15] M. P. T. Vishal Rathod, Natasha Jeppu, Samanvita Sastry, Shruti Singala, "CoCoA++: Delay gradient based congestion control for Internet of Things,"Futur. Gener. Comput. Syst., vol. Volume 100, p. Pages 1053-1072, May 2019.

[16] H. Hellaoui and M. Koudil, "Bird Flocking Congestion Control for CoAP/RPL/6LoWPAN Networks,"IoT-Sys '15 Proc. 2015 Work. IoT challenges Mob. Ind. Syst., pp. 25–30, 2015.

[17] A. P. Castellani, M. Rossi, and M. Zorzi, "Back Pressure Congestion Control for CoAP/6LoWPAN Networks,"Ad Hoc Networks, vol. 18, pp. 71–84, 2013.

[18] 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 19th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2018, 2018.

[19] E. Ancillotti and R. Bruno, "BDP-CoAP: Leveraging Bandwidth-Delay Product for Congestion Control in CoAP," in 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 656–66.

[20] Shelby, Zach, and Bormann, Carsten, "6LoWPAN: The Wireless Embedded Internet," in Wiley Publishing, 2010.

[21] R. Ludwig and K. Sklower, “The Eifel retransmission timer,” ACM SIGCOMM Comput. Commun. Rev., vol. 30, no. 3, pp. 17–27, 2000.

[22] R. Ludwig and R. H. Katz, "The Eifel algorithm: Making TCP robust against spurious retransmissions,"Comput. Commun. Rev., vol. 30, no. 1, pp. 30–36, 2000.

[23] V. Paxson and M. Allman, “Computing TCP’s retransmission timer (RFC6298),” Ietf, pp. 1–11, 2011.

[24] D. Bertsekas and R. Gallager, Data Networks. Prentice-Hall, 1992.

[25] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," ACM Queue, vol. 14, no. 5, pp. 50:20–50:53, October 2016.

[26] Kranenburg, R.V., 2008. The Internet of Things: A Critique of Ambient Technology and the All-Seeing Network of RFID, Institute of Network Cultures.

[27] I. J¨arvinen, M. Kojo, I. Raitahila, and Z. Cao, "Fast-Slow Retransmission and Congestion Control Algorithm for CoAP," Internet Draft, Jun. 2018, Work in progress.

[28] R. K. Yadav, N. Singh, and P. Piyush, "Genetic CoCoA++: Genetic Algorithm based Congestion Control in CoAP," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 808-813, DOI: 10.1109/ICICCS48265.2020.9121093.

[29] K.K. Jonassen, Implementing CAIA delay-gradient in Linux (Master's thesis), Department of Informatics, University of Oslo, Norway, 2015.

[30] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," ACM Queue, vol. 9, no. 11, Nov. 2011.

[31] Shahi, G. S., Batth, R. S., & Egerton, S. (2020). MRGM: An adaptive mechanism for congestion control in smart vehicular network. International Journal of Communication Networks and Information Security, 12(2), 273-280.

[32] Angurala, M., Bala, M., & Khullar, V. (2022). A survey on various congestion control techniques in wireless sensor networks. International Journal on Recent and Innovation Trends in Computing and Communication, 10(8), 47-54. doi:10.17762/IJRITCC.V10I8.5667

[33] Liberti, Leo & Lavor, Carlile & Maculan, Nelson & Mucherino, Antonio. (2012). Euclidean Distance Geometry and Applications. SIAM Review. 56. 10.1137/120875909.

[34] J. Tripathi, J. de Oliveira, and J. P. Vasseur, "A performance evaluation study of RPL: Routing protocol for low power and lossy networks," in CISS'10, March 2010, pp. 1–6.

[35] J.Katto, K.Ogura, Y.Akae, T.Fujikawa, K.Kaneko, and S.Zhou: "Simple Model Analysis and Performance Tuning of Hybrid TCP Congestion Control," IEEE Globecom 2008, Dec 2008.