ACCB- Adaptive Congestion Control with backoff Algorithm for CoAP

Sneha Deshmukh[1†] and Vijay T. Raisinghani[2††],

Mukesh Patel School of Technology Management and Engineering, NMIMS Deemed-to-be University, Mumbai, India

Abstract
Constrained Application Protocol (CoAP) is a standardized protocol by the Internet Engineering Task Force (IETF) for the Internet of things (IoT). IoT devices have limited computation power, memory, and connectivity capabilities. One of the significant problems in IoT networks is congestion control. The CoAP standard has an exponential backoff congestion control mechanism, which may not be adequate for all IoT applications. Each IoT application would have different characteristics, requiring a novel algorithm to handle congestion in the IoT network. Unnecessary retransmissions, and packet collisions, caused due to lossy links and higher packet error rates, lead to congestion in the IoT network. This paper presents an adaptive congestion control protocol for CoAP, Adaptive Congestion Control with a Backoff algorithm (ACCB). AACB is an extension to our earlier protocol AdCoCoA. The proposed algorithm estimates RTT, RTTVAR, and RTO using dynamic factors instead of fixed values. Also, the backoff mechanism has dynamic factors to estimate the RTO value on retransmissions. This dynamic adaptation helps to improve CoAP performance and reduce retransmissions. The results show ACCB has significantly higher goodput (49.5%, 436.5%, 312.7%), packet delivery ratio (10.1%, 56%, 23.3%), and transmission rate (37.7%, 265%, 175.3%); compare to CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. The results show ACCB has significantly higher goodput (60.5%, 482%,202.1%), packet delivery ratio (7.6%, 60.6%, 26%), and transmission rate (40.9%, 284%, 146.45%); compare to CoAP, CoCoA+ and AdCoCoA respectively in random walk scenario. ACCB has similar retransmission index compare to CoAp, CoCoA+ and AdCoCoA respectively in both the scenarios.

Keywords:
CoAP, congestion control, CoCoA+, AdCoCoA, ACCB.

1. Introduction

The Internet of things (IoT) has a significant role in today's applications with the fifth generation of cellular technology. IoT networks are a network of things capable of sending, receiving, or exchanging information. The IoT devices limited battery, memory, and processing capabilities. They can communicate over links that may have higher packet error rates and low throughput. These characteristics may lead to more retransmissions, which eventually may cause congestion in the network. Either loss of packet or acknowledgment (ACK) is commonly considered an indication of congestion. In IoT networks, packet or ACK loss can happen due to (i) lossy links, link errors, or weak signal strength and (ii) delay in receiving the ACK, which leads to spurious retransmissions. Hence, an efficient congestion control mechanism is required to minimize unnecessary retransmissions and accurately estimate the retransmission time out (RTO) value. The IETF has defined a protocol stack for these IoT devices. In the IoT protocol stack, Constrained Application Protocol (CoAP) operates over UDP as the transport protocol – supports lightweight internet applications.

CoAP uses a request/response interactive model between the application endpoints. A message identifier field help to detect duplicate messages. The Confirmable (CON) message requires an acknowledgment (ACK), whereas a Non-confirmable (NON) message doesn't require an ACK. A Reset (RST) message indicates to the sender that the received message is unable to process. CoAP provides optional reliability using the RTO mechanism and an exponential backoff policy. It chooses the initial RTO value randomly, and later it uses a binary exponential backoff (BEB) mechanism for computing the RTO value. The random initial value could lead to unnecessary retransmissions and, thus, network congestion [12]. Also, the CoAP backoff mechanism often fails to utilize the network dynamics to the best of its traffic conditions. As Currently, the CoAP Simple Congestion Control/Advanced (CoCoA) [11] is the standard protocol for CoAP by IETF. CoCoA considers Round Trip Time (RTT) for estimating Retransmission Timeout (RTO) along with a Variable Backoff Factor (VBF) and aging mechanisms to provide flexibility and monitored adaptation suited to the dynamic nature of IoT networks. Each IoT application has different characteristics and behavior, as a single standard algorithm may be adequate to handle the congestion in the network.

Recent studies [1], [2] show that CoCoA+ can perform better than basic CoAP in many scenarios but performs significantly poorly in bursty traffic environments in a large-scale network. In [3], [14], the authors have highlighted the drawbacks of CoCoA+, based simulation evaluation. Many researchers have proposed different mechanisms to overcome the weaknesses of CoCoA+. Few researchers aim to improve the performance of CoAP in a bursty traffic environment by measuring the RTT precisely and monitoring network conditions to minimize spurious retransmission. Few researchers use a rate-based mechanism [4], which utilizes the maximum bandwidth of the bottleneck link and ensures fairness. Existing protocols use fixed scaling factors for computations of the RTO, which might not be suitable for a wide range of IoT scenarios, and

Manuscript received October 5, 2022
Manuscript revised October 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.10.25

thus lead to higher retransmissions. Dynamic scaling factors could estimate RTO to be large enough to minimize retransmissions.

We propose a dynamic scaling factor for estimating RTT, RTTVAR, and RTO to overcome the fixed scaling values. The significant contribution of this work is an adaptive congestion control with a backoff algorithm named ACCB. ACCB estimates RTO using dynamic scaling factors for smoothed round trip time (RTT), RTT variation (RTTVAR), and RTO instead of fixed values. Also, ACCB has a dynamic backoff mechanism for retransmission. The proposed model is evaluated against CoAP, CoCoA+, and AdCoCoA.

This paper is structured as follows: Section 2 summarizes the existing congestion control mechanisms for CoAP. Section 3 describes the design and working of the proposed protocol. Section 4 presents the evaluation scenarios and analyzes the Contiki OS Cooja [20] simulation results. The conclusion in Section 5.

2. Literature survey

In [1] and [2], the authors have evaluated the performance of CoAP and CoCoA in environments varying from emulated Zigbee networks to large-scale IoT networks. In [5], CoCoA is considered in large-scale IoT networks in which GPRS is employed to connect IoT nodes. A. Betzler et al. compared CoCoA with congestion control protocols designed for TCP applications [2]. The authors reported that CoCoA performs similarly or better than TCP-based protocols. Similar interpretations are documented in [2]; the authors evaluated CoCoA with different traffic patterns. In [6], the authors proposed modified CoCoA with 4-State-Strong, which can distinguish loss due to lossy links and congestion. The reported results show better performance than CoCoA, but the loss rate is high. In [7], the authors proposed an enhanced version of CoCoA, named CoCoA-E. It uses the Eifel retransmission timer [15] for estimating RTO. The results show that CoCoA-E performs significantly better than CoCoA in the presence of lesser traffic fluctuations. In [5], the authors evaluate CoCoA-S (a variant of CoCoA), which considers only *strong*-RTO estimator, basic CoAP, CoCoA, and TCP-based protocols like Linux RTO [16], *peaker-hopper* TCP RTO [17]. The results show that CoCoA performance is better than the other evaluated protocols with lower traffic rates.

In [8], the authors proposed an Enhanced version of CoCoA, which measures RTT accurately using a *retransmission count field*. The authors reported that the proposed protocol performs better than CoAP, CoCoA, and CoCoA-E. In [3], [14], the authors evaluated CoCoA+ with bursty traffic and reported its shortcomings of CoCoA+. CoCoA+ performs worst with bursty traffic; RTO and RTT are too close, which may result in spurious retransmissions. A novel approach called pCoCoA is designed to overcome the limitations of CoCoA+. pCoCoA measures RTT precisely using a timestamp and aims to reduce spurious retransmissions. The results show that pCoCoA minimizes unnecessary retransmissions without affecting the throughput and delay. However, some of the fixed values introduced in pCoCoA might not be suitable for many IoT scenarios. In [4], the authors propose a new rate-based congestion control mechanism for CoAP, named CoAP-R. pCoCoA and CoAP-R aim to improve the performance of CoAP in a bursty traffic environment. CoAP-R is designed to achieve maximum bandwidth from the bottleneck link capacity and allocate the network resources based on the max-min fairness. The results show that CoAP-R distributes the network resources equally amongst the senders and reduces the delay compared to CoAP and CoCoA.

In [9], the author proposes CoCoA++, a delay gradient-based mechanism with a probability backoff factor. CoCoA++ is compared with CoCoA+ using a simulator and real testbed. The results show that CoCoA++ has low RTO values, reduces transmission delay, and increases transmission rate compared to CoCoA+ in different IoT scenarios. In [10], the authors propose a rate-based congestion control protocol for CoAP derived from TCP *Bottleneck Bandwidth and Round-trip propagation time* (BBR) [18], named BDP-CoAP. BDP-CoAP design copes with lossy links and the short-term unfairness of channel access in IoT networks. The results show that BPD-CoAP significantly improves the fairness of data connections and reduces the number of retransmissions while achieving throughput comparable to CoAP and CoCoA+.

In [22], the author has used learning automata to design a congestion control protocol for CoAP. A group of tunable parameters is used to control congestion in the network and enhance the network performance. CCCLA performs better than CoAP CC, CoCoA, and TCP-Siam. In [23], the authors enhance CoCoA using a machine-learning mechanism to tune the RTO parameter values using node count, packet size, and PDR. mlCoCoA achieves higher throughput than CoAP CC and CoCoA. In [24], the author has designed three RTO estimators which help to determine the exact network status. CACC performs better than CoAP and CoCoA in static scenarios.

An improved adaptive CoAP [25] determines the RTO value using packet loss ratio and RTT. RTT-CoAP [26] has a novel approach to detecting congestion in the network using the growth of RTT variance coupled with thresholds on CoAP message losses. CoAP Eifel [27] is a modified version of CoAP which uses only strong RTT to estimate the RTO value. AdCoCoA [28] considers link quality, link delay, and RTT deviation to calculate the RTO value. CACC [29] considers strong, weak, and failed RTT to identify the exact network status and provide an adaptive congestion control mechanism. FASOR [30] determines whether packet loss is due to the wireless link environment or congestion by considering three unique features – self-adaptive retransmission timer backoff, slow RTO computation, and fast RTO computation mechanism. DCC-CoAP [31] has an

efficient way of predicting network congestion; it uses a combination of distance between nodes and round RTT measurements, which limits the losses of CoAP messages.

RCoAP [32] is a rate control-based scheme for CoAP designed for reliable bursty data transfer. It consists of four states – *initial, normal, loss detection,* and *backoff.* psoCoCoA [33] is a variation of CoCoA, which applies random and optimal parameter-driven simulation to optimize default CoAP parameters and update the fitness and velocity positions to adapt to the traffic conditions.

In this section, we summarized the existing protocols designed to improve the performance of CoAP. The following section discusses our adaptive congestion control's design and working mechanism with the backoff algorithm (ACCB) for CoAP.

3. Adaptive Congestion Control with Backoff algorithm (ACCB)

As discussed in section II, most existing congestion control protocols for CoAP use fixed scaling factors for the RTO estimator. These fixed factors might not be suitable in many IoT scenarios, for example, connected cars or self-driven cars. The algorithm considers the varying density (low to high) and mobility speed of vehicles (slow to fast) and adapts as per the characteristic of the application. Hence, dynamic scaling factors may help to improve the performance of CoAP for such applications. ACCB is an enhancement of AdCoCoA, which (i) estimates dynamic scaling factors using multiple network parameters instead of a single parameter and (ii) uses the Adaptive backoff (ABF) mechanism instead of the variable backoff (VBF) mechanism. AdCoCoA determines dynamic scaling factor using only RTT variance, whereas ACCB determines using RTT variance, retransmission ratio, and packet loss ratio.

Figure 1 shows the overview of the ACCB mechanism for CoAP. ACCB measures RTT using a timestamp; hence only strong RTT contributes to RTO estimation. On sending the CON message, if the receiver is unknown, then the default CoAP mechanism is used for the first time. On estimating the first measured RTT, ACCB calculates Smoothed RTT, RTTVAR, and RTO, the same as AdCoCoA. ACCB uses dynamic scaling and smoothing factors for subsequent measured RTTs to calculate RTT, RTTVAR, and RTO values. ACCB calculates dynamic scaling and smoothing characteristics for estimating RTT, RTTVAR, and RTO.

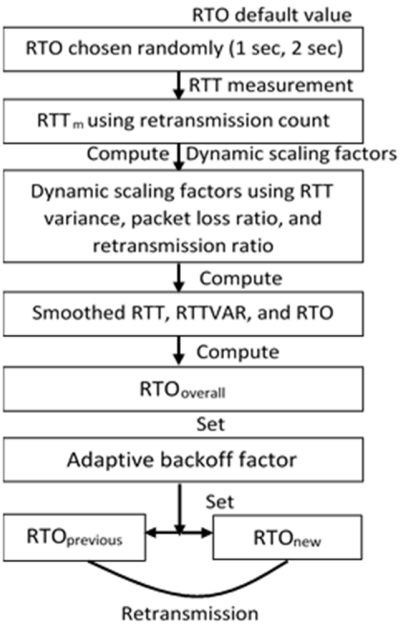


Figure 1: An overview of the ACCB mechanism

To calculate *RTT deviation, retransmission ratio*, and p*acket loss ratio*.

RTT deviation = $((RTT_m - RTT_{current})/ RTT_{current})$ (eq 1)

If the *RTT deviation* exceeds one, the *RTT deviation* value equals 0.9.

$$retransmission\ ratio = \frac{\text{number of packets retransmitted}}{number\ of\ packets\ transmitted} \quad \text{(eq 2)}$$

If the *retransmission ratio* is zero, the *retransmission ratio* value equals 0.1.

$$packet\ loss\ ratio = \frac{\text{number of packets loss}}{number\ of\ packets\ transmitted} \quad \text{(eq 3)}$$

If the *packet loss ratio* is zero, then the *packet loss ratio* value equals 0.1.

To calculate dynamic smoothing factors *alpha* and *beta.*

alpha = ((*RTT deviation* * 0.5) + (*retransmission ratio* * 0.25) + (*packet loss ratio* * 0.25)) (eq 4)

beta = alpha * 0.5 ; (eq 5)

To calculate smoothed RTT, RTTVAR, and RTO values.

$RTT_{current} = (1 - alpha) * RTT_m + alpha * RTT_{current}$ (eq 6)

$RTTVAR_{current} = (1 - beta) * RTTVAR_{previous} + beta * |RTT_m - RTT_{current}|$ (eq 7)

$RTO_{current} = RTT_{current} + K * RTTVAR_{current}$ (eq 8)

Where K is the scaling factor. If the *RTT deviation* value exceeds one, K =6; else, K=4.

$RTO_{overall} = 0.5*RTO_{previous} + 0.5 * RTO_{current}$ (eq 9)

The Adaptive backoff factor (ABF) mechanism calculates the RTO value on retransmissions. If $RTO_{overall}$ less than one second, $RTO_{new} = RTO_{previous} + (1 + RTT\ deviation)$*; else,* $RTO_{new} = RTO_{previous} + RTT\ deviation$.

In this section, we described the design and working of the ACCB algorithm. The following section discusses the

simulations used to validate ACCB against CoAP, CoCoA+, and AdCoCoA.

4. Performance evaluation

In this section, we evaluate the performance of ACCB with CoAP, CoCoA+, and AdCoCoA through simulations. This section includes the simulator's configuration, the traffic scenario, the network topologies, and the performance metrics used to carry out the performance evaluations.

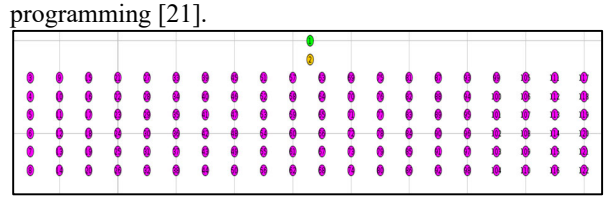
4.1 Simulation setup

We use the Cooja simulator [20], a simulation platform included in Contiki 3.0 toolset [20], that allows emulating off-the-shelf wireless sensor node hardware. At the physical (PHY) and MAC layers, the nodes implement IEEE 802.15.4, using a transmission rate of 250 kbps in the 2.4 GHz radio band. The Cooja simulator and Zolerita Z1 motes are used for clients and servers. Z1 motes have 92KB of ROM, enabling us to code applications and implement congestion control mechanisms. T-mote Sky motes are used to configure the border router because we use IPv6 Routing Protocol for Low-Power and Lossy Networks [19] in storing mode, which requires a larger RAM capacity provided by Sky motes. The relevant simulation parameters are summarized in Table 2.

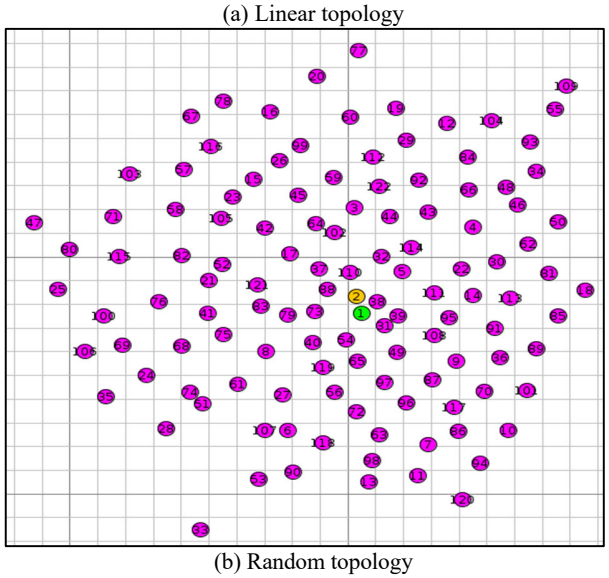
Table 1. Simulation parameters

Operating system	Contiki 3.0
Simulator	Cooja
Area	Linear – varies from 20 m * 10 m to 500 m * 20 m, Random - ranges from 50m * 50 m to 400 m * 400 m
Radio Medium	Unit Disk Graph Medium (UDGM) - Distance loss
Radio duty cycling	NullRDC
Mote type	T-mote sky and Zolerita Z1
Number of nodes	40, 80, 120, and 160
Topologies	Linear (chain) and Random
Node transmission range	10 m
Node interference range	20 m
Interference level	10%, 25%, 35%, 50%
Traffic type	Constant bit rate (periodic)
Speed of nodes	Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr
Packet size	100 bytes
Retransmission limit	4
Simulation time	300 seconds

Figure 2 shows the topologies used, nodeID 1 represents the RPL border router, nodeID 2 represents the CoAP server, and the rest are CoAP clients. After the RPL setup, the clients periodically send messages targeted at the servers. Every scenario is run 15 times with a different random seed with a simulation time of 300 seconds. The simulation results are plotted against the number of nodes, interference level, and speed of nodes using a box-and-whisker plot diagram. To determine the difference among the protocols, we used Tukey's Honestly Significance test with a confidence level of 90%, evaluated using R programming [21].



(a) Linear topology



(b) Random topology

Figure 2: Topologies (a) Linear (b) Random

The metrics used in the performance evaluation are i) the *goodput*, measured as the total amount of data successfully received per unit of time; ii) the *packet delivery ratio*, computed as the ratio of the total number of received CoAP messages over the number of sent CoAP messages in a given time interval; iii) the *retransmission rate*, evaluated as the percentage of retransmitted packets over the total number of packets sent by the CoAP clients; and iv) the *packet sending rate* at the application level, the total number of packets sent by the CoAP clients in a given time interval.

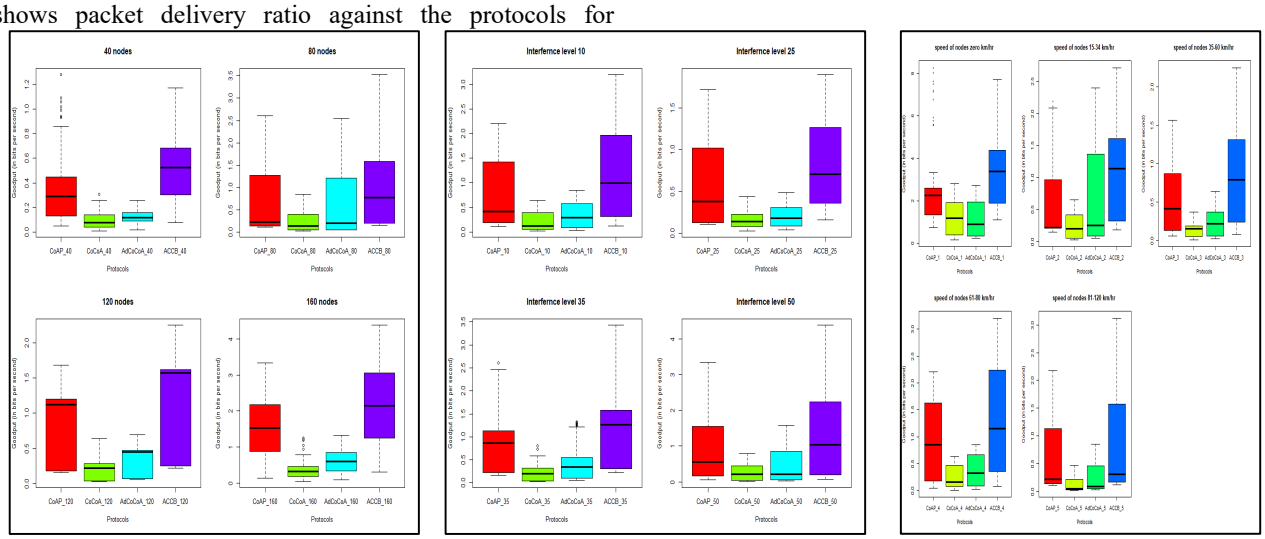
Linear scenario:

Figure 3 shows the comparative goodput of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The goodput achieved in the linear scenario with ACCB is statistically higher by approx. 49.5%, 436.5%, 312.7% than CoAP, CoCoA+, and AdCoCoA as p-value is ($0 < 0.10$) respectively. Figure 3 (a) shows goodput against the protocols for the number of nodes 40, 80, 120, and 160. The goodput of ACCB is significantly higher by approx. 41%, 417.1%, 221.1% than CoAP, CoCoA+, and AdCoCoA respectively. As the number of nodes increases, the goodput also increases. Figure 3 (b) shows goodput against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is significantly higher by approx. 45.6%, 441.4%, 245.4% than CoAP, CoCoA+, and AdCoCoA respectively. As the interference level increases,

the goodput of decreases. Figure 3 (c) shows goodput against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The goodput of ACCB is significantly higher by approx. 45.6%, 410.9%, 205.6% than CoAP, CoCoA+, and AdCoCoA respectively. As the speed of nodes increases, the goodput decreases.

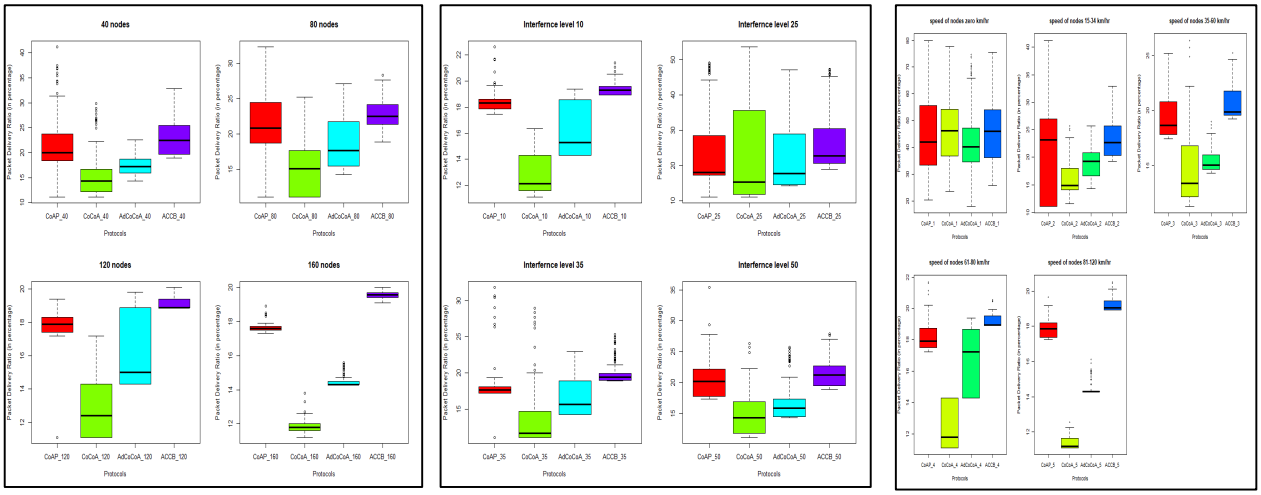
Figure 4 shows the comparative packet delivery ratio of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The packet delivery ratio achieved in the linear scenario with ACCB is statistically higher by approx. 10.12%, 56.05%, 28.3% than CoAP, CoCoA+, and AdCoCoA as p-value is 1.7E-11 < 0.10) respectively. Figure 4 (a) shows the packet delivery ratio against the protocols for the number of nodes 40, 80, 120, and 160. For 40 nodes, the goodput of ACCB is significantly higher by approx. 9.2%, 53.3%, 26.9% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 4 (b) shows packet delivery ratio against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is statistically higher by approx. 10.2%, 39.5%, 20.4% than CoAP, CoCoA+, and AdCoCoA as respectively. Figure 4 (c) shows packet delivery ratio against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The packet delivery ratio of ACCB is statistically higher by approx. 6.4%, 41.3%, 23.1% than CoAP, CoCoA+, and AdCoCoA as respectively.

Figure 5 shows the comparative retransmission rate of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The retransmission rate achieved in the linear scenario with ACCB is the same as CoAP, CoCoA+, and AdCoCoA, respectively.



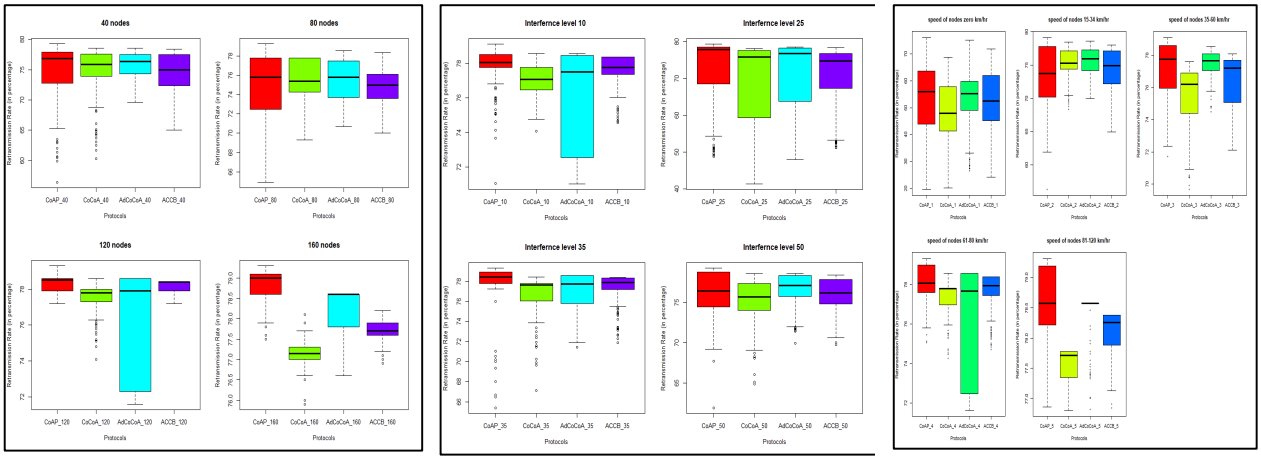
(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 3: Goodput v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



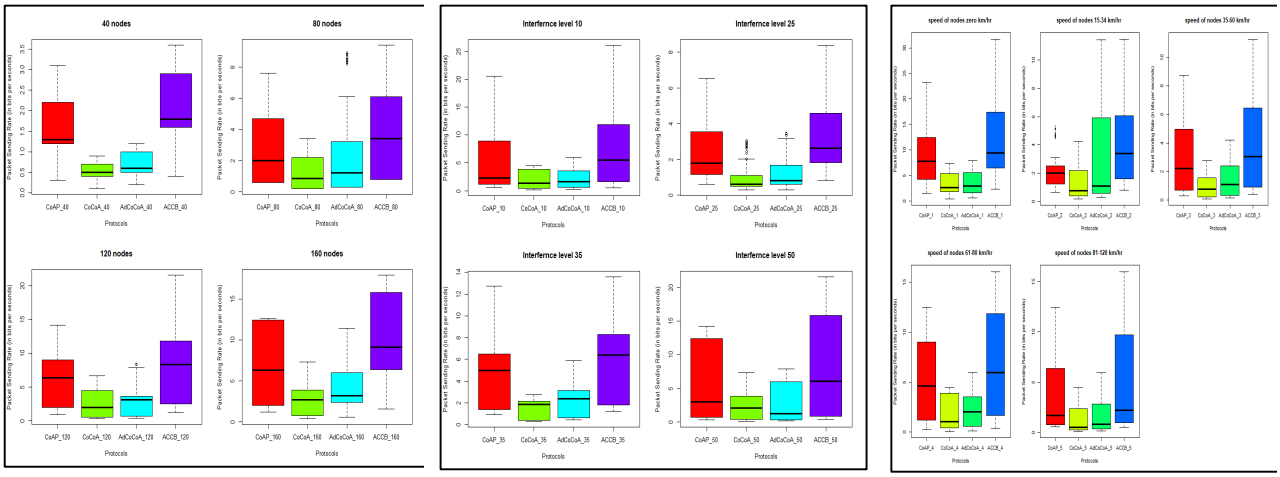
(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 4: Packet delivery ratio v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 5: Retransmission rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



(a)Varying number of nod *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 6: Packet sending rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying the speed of nodes

Figure 5 (a), (b), and (c) show the retransmission rate against protocols by varying the number of nodes, interference level, and speed of nodes. The retransmission rate is directly proportional to the number of nodes. As the interference level increases, the retransmission rate also increases. As the speed of nodes increases, the retransmission rate also increases.

Figure 6 shows the comparative packet sending rate of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The packet sending rate achieved in the linear scenario with ACCB is statistically higher by approx. 37.7%, 265%, 175.3% than CoAP, CoCoA+, and AdCoCoA as p-value is ($0 < 0.10$) respectively. Figure 6 (a) shows the packet sending rate against the protocols for the number of nodes 40, 80, 120, and 160. The packet-sending rate of ACCB is significantly higher by approx. 39.9%, 255%, 136% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 6 (b) shows packet sending rate against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The packet-sending rate of ACCB is significantly higher by approx. 37.5%, 272.3%, 180.7% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 6 (c) shows packet sending rate against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The packet-sending rate of ACCB is significantly higher by approx. 41.7%, 260.4%, 165.9% than CoAP, CoCoA+, and AdCoCoA respectively.

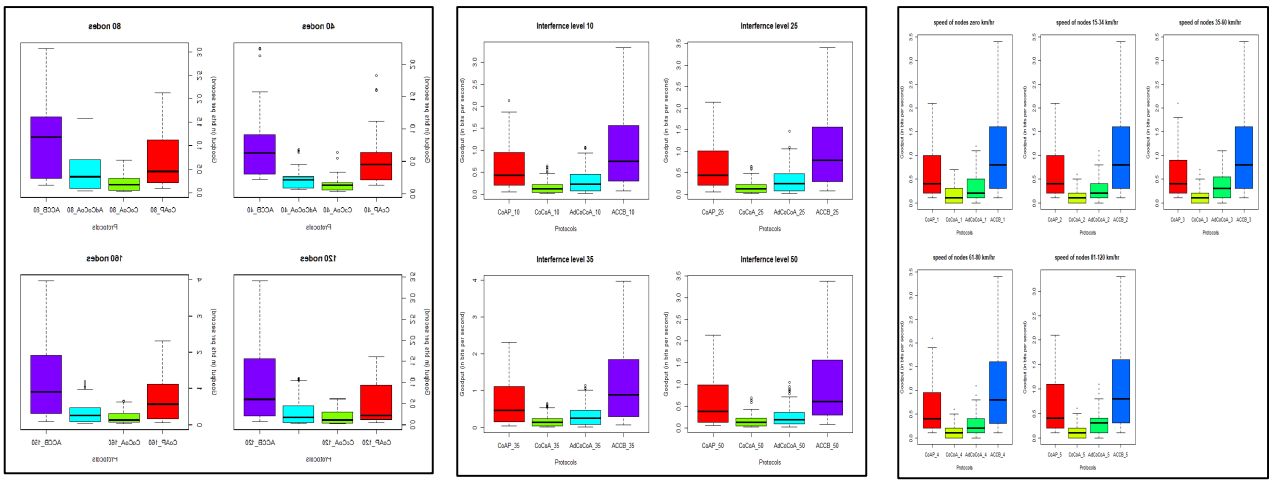
ACCB estimates RTO per RTT measured and considers the multiple network status parameters. The dynamic smoothing factor scales the RTO as per the change fraction compared to the current RTT, ensuring the maximum CoAP message is received. Whereas AdCoCoA considers single network status parameters to calculate dynamic scaling factors. CoCoA uses fixed scaling factors for calculating RTT, RTTVAR, and RTO. CoAP does not consider any network status parameter and doubles the RTO value on retransmission.

Random scenario:

Figure 7 shows the comparative goodput of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The goodput achieved in the random scenario with ACCB is statistically higher by approx. 60.5%, 482%, 202.1% than CoAP, CoCoA+, and AdCoCoA as p-value is (0 < 0.10) respectively. Figure 7 (a) shows goodput against the protocols for the number of nodes 40, 80, 120, and 160. The goodput of ACCB is significantly higher by approx. 63.8%, 482.4%, 211.7% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 7 (b) shows goodput against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is significantly higher by approx. 61.6%, 500.2%, 228% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 7 (c) shows goodput against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The goodput of ACCB is significantly higher by approx. 65.5%, 552.2%, 218% than CoAP, CoCoA+, and AdCoCoA respectively.

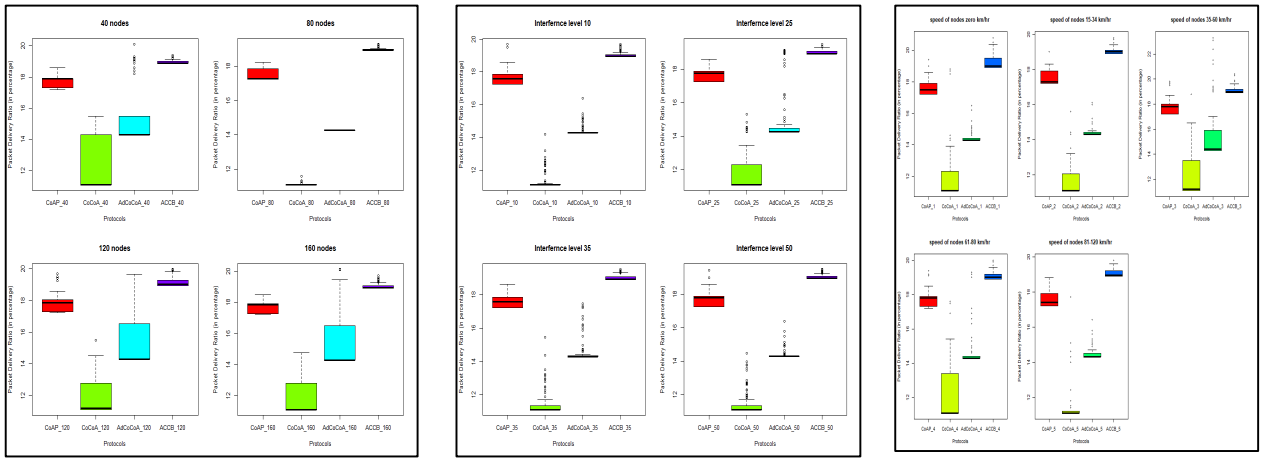
Figure 8 shows the comparative packet delivery ratio of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The packet delivery ratio achieved in the random scenario with ACCB is significantly higher by approx. 7.6%, 60.6%, 26% than CoAP, CoCoA+, and AdCoCoA p-value is (2.5E-11 < 0.10) respectively. Figure 8 (a) shows the packet delivery ratio against the protocols for the number of nodes 40, 80, 120, and 160. The goodput of ACCB is significantly higher by approx. 7.7%, 60.6%, 25.3% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 8 (b) shows packet delivery ratio against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is significantly higher by approx. 7.8%, 64.8%, 30% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 8 (c) shows packet delivery ratio against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The packet delivery ratio of ACCB is significantly higher by approx. 8.1%, 59%, 29.1% than CoAP, CoCoA+, and AdCoCoA respectively.

Figure 9 shows the comparative retransmission rate of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The retransmission rate achieved in the random scenario with ACCB is the same as CoAP, CoCoA+, and AdCoCoA, respectively. Figure 9 (a), (b), and (c) show the retransmission rate against protocols by varying the number of nodes, interference level, and speed of nodes. As the number of nodes increases, the retransmission rate also increases.



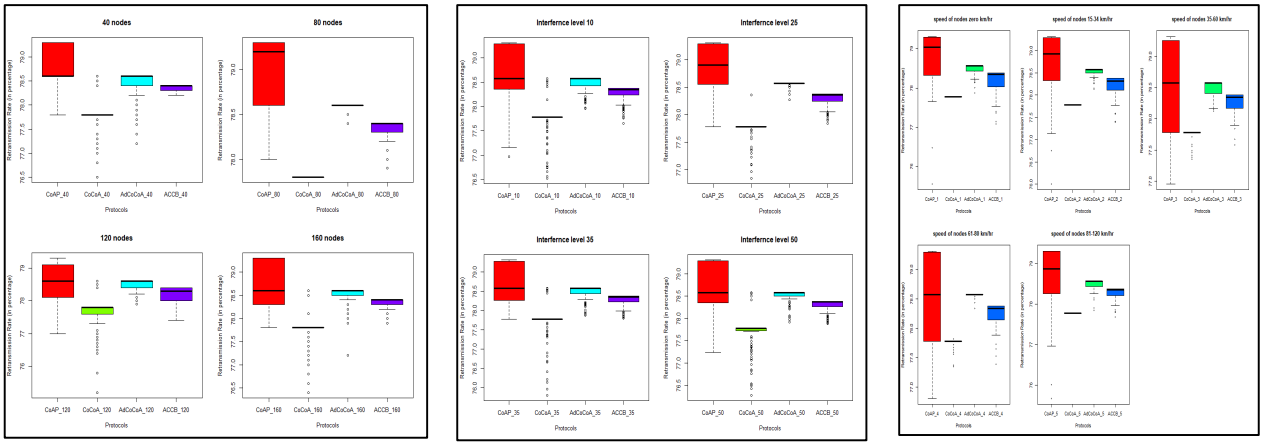
(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 7: Goodput v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



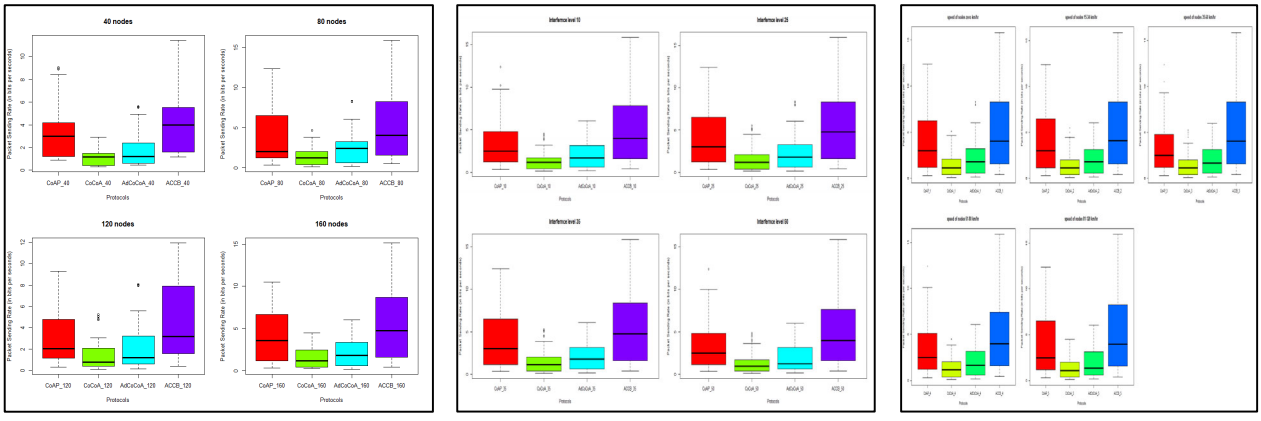
(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 8: Packet delivery ratio v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 9: Retransmission rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 10: Packet sending rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying the speed of nodes

As the interference level increases, the retransmission rate also increases. As the speed of nodes increases, the retransmission rate also increases.

Figure 10 shows the comparative packet sending rate of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The packet sending rate achieved in the random scenario with ACCB is statistically higher by approx. 40.9%, 284%, 146.4% than CoAP, CoCoA+, and AdCoCoA as p-value is ($0 < 0.10$) respectively. Figure 10(a) shows goodput against the protocols for the number of nodes 40, 80, 120, and 160. The goodput of ACCB is significantly higher by approx. 42.1%, 277.4%, 126.9% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 10(b) shows goodput against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is statistically higher by approx. 41.7%, 286.1%, 150.1% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 10(c) shows goodput against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The goodput of ACCB is statistically higher by approx. 42.2%, 285.4%, 148.8% than CoAP, CoCoA+, and AdCoCoA respectively.

ACCB estimates dynamic smoothing factors per the change determined by RTT deviation, retransmission ratio, and packet loss ratio. These dynamic scaling factors help estimate RTO large enough, ensuring maximum CoAP message to be received. CoCoA uses fixed scaling factors for calculating RTT, RTTVAR, and RTO. Whereas AdCoCoA considers a single network status parameter to calculate dynamic scaling factors. CoAP does not consider any network status parameter and doubles the RTO value on retransmission.

In this section, we describe the simulation setup, evaluation parameters, and simulation results.

5. Conclusion

In recent studies, most existing protocols use fixed scaling factors for estimating RTO. This paper proposes an Adaptive congestion control with a backoff algorithm (ACCB) for CoAP in IoT networks. ACCB has a dynamic-scaling-factor-based mechanism for RTT, RTTVAR, and RTO estimation. ACCB adaptively computes the RTO depending on the changes in the network status. ACCB considers RTT deviation, retransmission ratio, and packet loss ratio to calculate dynamic scaling and smoothing factors. Adaptive factors and ABF mechanisms would help improve performance and minimize retransmissions. Our simulation results show that ACCB performs better than CoAP, CoCoA+, and AdCoCoa in linear and random scenarios. ACCB has significantly higher goodput 49.5%, 436.5%, and 312.7% compare to CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. ACCB has significantly higher packet delivery ratio 10.1%, 56%, 23.3% compare to CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. ACCB has significantly higher packet sending rate 37.7%, 265%, and 175.3% compare to CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. The results show ACCB has significantly higher goodput (60.5%, 482%,202.1%), packet delivery ratio (7.6%, 60.6%, 26%), and packet sending rate (40.9%, 284%, 146.45%), compared to CoAP, CoCoA+ and AdCoCoA respectively in random walk scenario. ACCB has similar retransmission rate compared to CoAP, CoCoA+ and AdCoCoA in both the scenarios.

The estimated RTO is large enough; packets are received in the first or minimum attempts. ACCB helps increase the packet delivery ratio and sending rate and achieve higher goodput. ACCB performs better because it determines dynamic scaling factors based on the RTT deviation, retransmission ratio, and packet loss ratio. Due to dynamic scaling factors, packet loss and retransmission attempts are reduced. For future research, we plan to extend our work to validate the proposed algorithm by varying packet size, the density of nodes, traffic type, and inter-arrival time.

References

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

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

[3] 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.

[4] E. Ancillotti, R. Bruno, C. Vallati and E. Mingozzi, "Design and Evaluation of a Rate-Based Congestion Control Mechanism in CoAP for IoT Applications," 2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Chania, 2018, pp. 14-15, DOI: 10.1109/WoWMoM.2018.8449736.

[5] A. Betzler, C. Gomez, I. Demirkol and J. Paradells, "CoAP congestion control for the internet of things," in IEEE Communications Magazine, vol. 54, no. 7, pp. 154-160, July 2016, DOI: 10.1109/MCOM.2016.7509394.

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

[7] Balandina E., Koucheryavy Y., Gurtov A. (2013) Computing the Retransmission Timeout in CoAP. In: Balandin S., Andreev S., Koucheryavy Y. (eds) Internet of Things, Smart Spaces, and Next Generation Networking. Lecture Notes in Computer Science, vol 8121. Spring- er, Berlin, Heidelberg.

[8] Lee, Jung June, Kyung Tae Kim, and Hee Yong Youn. "Enhancement of Congestion Control of Constrained Application Protocol/Congestion Control/Advanced for Internet of Things

Environment." International Journal of Distributed Sensor Networks, (November 2016).
[9] Vishal Rathod, Natasha Jeppu, Samanvita Sastry, Shruti Singala, Mohit P. Tahiliani, CoCoA++: Delay gradient based congestion control for Internet of Things, Future Generation Computer Systems, Volume 100, 2019, Pages 1053-1072.
[10] E. Ancillotti and R. Bruno, "BDP-CoAP: Leveraging Bandwidth-Delay Product for Congestion Control in CoAP," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 656-661, DOI: 10.1109/WF-IoT.2019.8767177.
[11] Bormann, C., Betzler, A., Gomez, C., & Demirkol, I. (2017). "CoAP simple congestion control/advanced." Working Draft, IETF Secretariat, Internet-Draft draft-bormann-core-cocoa-01.
[12] Shelby, Z., K. Hartke, and C. Bormann. "The Constrained Application Protocol (CoAP)." (2014).
[13] August Betzler, Carles Gomez, Ilker Demirkol, Josep Paradells, "CoCoA+: An advanced congestion control mechanism for CoAP," Ad Hoc Networks, Volume 33, 2015, Pages 126-139.
[14] Bolettieri S, Vallati C, Tanganelli G, Mingozzi E. Highlighting some shortcomings of the CoCoA+ congestion control algorithm. In: Ad-hoc, Mobile, and Wireless Networks: 16th International Conference onAdHocNetworks and Wireless, ADHOC-NOW2017, Messina, Italy, September 20-22, 2017, Proceedings. Cham, Switzerland: Springer International Publishing; 2017:213-220.
[15] R. Ludwig and K. Sklower "The Eifel Retransmission Timer," ACM SIGCOMM Computer Communication Review, Vol. 30, Issue 3, pp. 17-27, July 2000.
[16] Pasi Sarolahti and Alexey Kuznetsov. 2002. Congestion Control in Linux TCP. In Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference. USENIX Association, USA, 2002, pp. 49–62.
[17] H. Ekstrom and R. Ludwig, "The Peak-Hopper: A New End-to-End Retransmission Timer for Reliable Unicast Transport," Proc. IEEE INFOCOM 2004, vol. 4, Mar. 2004, pp. 2502–13.
[18] 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.
[19] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF RFC 6550, March 2012.
[20] The official web page of the Cooja simulator in Contiki OS, URI: http://www.contiki-os.org/. Retrieved on 08.04.2020.
[21] R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
[22] CCLCA Soulmaz Gheisari, Ehsan Tahavori, "CCCLA: A cognitive approach for congestion control in Internet of Things using a game of learning automata," Computer Communications, Volume 147, 2019, Pages 40-49, ISSN 0140-3664, doi.org/10.1016/j.comcom.2019.08.017.
[23] DEMIR, Alper Kamil, and Fatih ABUT. "mlCoCoA: a machine learning-based congestion control for CoAP," Turkish Journal of electrical engineering & computer sciences,2018. doi:10.3906/elk-
[24] Akpakwu GA, Hancke GP, Abu-Mahfouz AM. "CACC: Context-aware congestion control approach for lightweight CoAP/UDP-based Internet of Things traffic," Transaction Emerging Tel Tech. 2019; e3822. https://doi.org/10.1002/ett.3822.
[25] F. Ouakasse and S. Rakrak, "An improved adaptive CoAP congestion control algorithm," *Int. J. Online Eng.*, vol. 15, no. 3, pp. 96–109, 2019.
[26] 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.
[27] 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.
[28] S. Deshmukh and V. T. Raisinghani, "AdCoCoA- Adaptive Congestion Control Algorithm for CoAP," *2020 11th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2020*, 2020.
[29] 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.
[30] 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.
[31] S. Bansal and D. Kumar, "Distance-based congestion control mechanism for CoAP in IoT," *IET Commun.*, vol. 14, no. 19, pp. 3512–3520, 2020.
[32] D. H. Hoang and T. T. D. Le, "RCOAP: A Rate Control Scheme for Reliable Bursty Data Transfer in IoT Networks," in *IEEE Access*, vol. 9, pp. 169281-169298, 2021, DOI: 10.1109/ACCESS.2021.3135435.
[33] Godfrey A. Akpakwu & Gerhard P. Hancke & Adnan M. Abu-Mahfouz, 2022. "An optimization-based congestion control for constrained application protocol," International Journal of Network Management, John Wiley & Sons, vol. 32(1), January. DOI: 10.1002/nem.2178.

Sneha Deshmukh is a faculty member at the Dept of Information Technology, MPSTME, NMIMS Deemed-to-be University. Her research interests are wireless sensor networks and education technology.

Vijay Raisinghani (S. Member IEEE) is a faculty member at the Dept of Information Technology, MPSTME, NMIMS Deemed-to-be University. His research interests are computer networks and education technology.