Journal of Ambient Intelligence and Humanized Computing (2023) 14:2951–2966
https://doi.org/10.1007/s12652-023-04534-8

ORIGINAL RESEARCH

iCoCoA: intelligent congestion control algorithm for CoAP using deep reinforcement learning

Praveen Kumar Donta[1,2] · Satish Narayana Srirama[2,3] · Tarachand Amgoth[1] · Chandra Sekhara Rao Annavarapu[1]

Received: 15 December 2020 / Accepted: 10 January 2023 / Published online: 26 January 2023

Abstract
The constrained application protocol (CoAP) is an application layer protocol in IoT, with underlying support for congestion control mechanism. It minimizes the frequent retransmissions, but does not optimize the throughput or adapt dynamic conditions. However, designing an efficient congestion control mechanism over the IoT poses new challenges because of its resource constraint nature. In this context, this article presents a new Intelligent congestion control algorithm (iCoCoA) for constraint devices, motivated by the success of the deep reinforcement learning in various applications. The iCoCoA learns from the various network features to decide the best Retransmission Timeout to mitigate the congestion in the dynamic environments. It also optimizes the throughput, energy, and unnecessary frequent retransmissions compared with the existing models. iCoCoA is developed and tested on the Cooja simulator and compared it with the standard protocols such as CoAP, CoCoA, and CoCoA+ in continuous and burst traffic conditions. The proposed iCoCoA mitigates congestion, outperforms 4–15% in throughput, 3–10% better packet delivery ratio, and 7–16% energy-efficiency with reduced number of retransmissions.

Keywords Constrained application protocol · Congestion control · Deep reinforcement learning · Internet of Things · Retransmission timeout

1 Introduction

Internet of things (IoT) is the most promising technology because of its appealing features such as scalability, cost-effective, low-complexity, self-organize, ease of use and deployment. With the proliferation of IoT applications, many devices are connected over the Internet every day (Jamshed et al. 2022). These devices are battery-powered, with limited buffer, communication bandwidth and processing capabilities. The primary goal of these devices is to exchange data among them or the cloud by interacting with the environment. Furthermore, the cloud extracts the knowledge from this data and communicates with the user through devices (Kaur and Sood 2017). For reliable data transmissions, the application layer of IoT is composed with a variety of protocols such as HTTP, CoAP, XMPP, MQTT, AMQP, DDS, WebSocket, etc (Donta et al. 2022; Sun and Ansari 2018). These protocols use either transmission control protocol (TCP) or user datagram protocol (UDP) transport control to fill this gap like the Internet (Sandell and Raza 2019; Mahajan et al. 2022). In this context, the standard protocols which are used in the regular Internet are not preferable for IoT because of its constraints in nature.

In IoT, as there are large number of devices with continuous monitoring, occasionally the traffic of the network exceeds the available capacities of channel contention or the buffer. It is usually uncontrollable and creates the congestion in IoTs (Donta et al. 2020; Sangaiah et al. 2020).

✉ Satish Narayana Srirama
satish.srirama@uohyd.ac.in

Praveen Kumar Donta
praveeniitism@gmail.com

Tarachand Amgoth
tarachand@iitism.ac.in

Chandra Sekhara Rao Annavarapu
acsrao@iitism.ac.in

1 Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, Jharkhand, India

2 Mobile & Cloud Lab, Institute of Computer Science, University of Tartu, 50090 Tartu, Estonia

3 School of Computer and Information Sciences, University of Hyderabad, Hyderabad 500 046, India

Congestion is an increasingly significant challenging issue in IoTs because it has more impact on various QoS parameters. Mainly, it degrades the throughput, packet delivery rate (PDR), and increases the packet retransmissions and losses, energy wastage, and end-to-end delay (Salkuti 2018). There are several congestion control techniques in various IoT protocols (from different layers), and this article focuses on the congestion control mechanism in the Constrained Application Protocol (CoAP) (Bormann et al. 2012).

The Constrained RESTful Environments (CoRE) group under the Internet Engineering Task Force (IETF) standardized CoAP (RFC 7252). It is a low-powered, low-bandwidth and light-weight constrained protocol for IoT and is inspired by the Hyper-text transfer protocol (HTTP) over the UDP. CoAP supports the conformable (CON) or NON-message transmissions. CON messages receive an acknowledgement (ACK) for successful message delivery, and there is no ACK for the NON messages (Bormann et al. 2012). The basic congestion control mechanism in CoAP, primarily considers the packet loss (within the specified time) for congestion detection. Thus it supports only the CON messages, and the CoAP uses Binary Exponential Backoff (BEB) function to compute the Retransmission timeout (RTO) for unsuccessful message delivery (Mišić et al. 2018). The initial RTO selects randomly between the interval [2s, 3s], and BEB doubles (up to 60 s) it for each retransmission (i.e. $RTO_{new} = RTO_{old} << 1$). For example, the four RTOs when the initial RTO is 2 are 4, 8, 16, and 32. The major limitations of CoAP are, it does not avoid the congestion; moreover, it increases the delay and also degrades the buffer utilization (Kim et al. 2019; Betzler et al. 2016a, 2016b).

Some of the RTO computations over CoAP use Round Trip Time (RTT) to estimate or control the congestion (Rathod et al. 2019; Suwannapong and Khunboa 2019; Akpakwu et al. 2020). Most of these techniques are using the TCP congestion control mechanism based on the previous RTT. These techniques are not works dynamically according to the change of network properties. It also take more resources and produce static RTO values. The non-continuous conditions of the IoT environments, with highly variable multiple complex network features such as RTTs, buffer sizes, bandwidths, flow sizes, and burst traffic conditions between the devices or devices and server (Uroz and Rodríguez 2022). These variable factors create dynamic problems and require dynamic decisions to control the congestion in the CoAP. Hence, there is a need for efficient and dynamic RTO computation techniques over CoAP for efficient congestion control. So, there is a need of Intelligent protocol, which works dynamically according to the changes in the network features. We strongly believe that the deep reinforcement learning (DRL) algorithm is the best solution to address the congestion problem of the CoAP in the above mentioned conditions.

The DRL is a machine learning (ML) approach, which learns with experiences by interacting with the environment. DRL is being used in various application such as gaming, robotic, computer vision, Internet congestion control, etc., (Praveen Kumar et al. 2019; Xiao et al. 2019; Nie et al. 2019). Success of these applications have motivated to choose DRL for addressing the congestion issue in the CoAP. The major benefits identified from the DRL are (1) the DRL does not require any predetermined data sets to train the system unlike other ML approaches such as supervised or unsupervised, (2) It provides the best decisions based on the trial and error methods by considering the exploitation or exploration algorithms with the previous optimal decisions (Xiao et al. 2019), (3) Unlike RL, DRL does not require additional space to maintain a Q-table and thus it is also not required to compute all the Q-values associated with each state. The major contributions of this article are as follows:

- The proposed Intelligent Congestion Control algorithm (iCoCoA) uses DRL algorithm to predict and mitigate congestion by computing dynamic RTOs.
- The iCoCoA considers various network features such as number of retransmissions, RTTVAR, RTT and previous RTO to estimate the efficient RTO, whereas the existing CoAP, CoCoA, CoCoA+, pCoCoA, and CoCoA++ algorithms estimate RTO by considering *only RTT* value and sometimes these RTTs are noisy.
- The proposed iCoCoA efficiently manages the limited buffer and minimizes unnecessary computations of the agent during training and running process.
- iCoCoA is implemented on the Contiki v3.0 Cooja simulator and its efficiency is compared against standard (by IETF) algorithms CoAP, CoCoA, and CoCoA+ algorithms.

The remaining sections of this article is arranged as follows. In Sect. 2, we review congestion control approaches for CoAP. In Sect. 3, we formulate the problem. In Sect. 4, we describe the proposed iCoCoA in detail. In Sect. 5, we compare the simulation results of the existing approaches with the proposed iCoCoA, with various parameters. The paper is concluded in Sect. 6.

2 Related work

In the recent years, several congestion control mechanisms have been introduced across the different layers of IoT, but this paper focuses only on congestion control in CoAP. The extensive literature on other aspect of CoAP is available in Donta et al. (2022). In this section, we review various existing but related congestion control approaches used in CoAP.

In Betzler et al. (2013), an end-to-end congestion control mechanism Congestion Control/Advanced (CoCoA) has been developed for CoAP. It uses the TCP's retransmission timer computing strategy (RFC-6298) to calculate the overall RTO (Sargent et al. 2011). The CoCoA enhances the CoAP with two RTO estimators called strong RTO and weak RTO depending on the previous RTTs. The strong estimator uses the RTT of successful transmissions in the first attempt and whereas weak estimator considers the RTT of at least one retransmission. The overall RTO of CoCoA is computed based on the previous overall RTO and the weighted average of either weak or strong estimator. The major limitations identified in CoCoA algorithm are producing the overall RTO value with a very successive time and it also calculates two estimators to decide the overall RTO. Later CoCoA+ was introduced by enhancing the CoCoA from the authors of CoCoA in Betzler et al. (2015). In this, Variable Backoff Factor (VBF) was introduced in place of the BEB. Additionally, the computational strategy of the weak estimator was also upgraded. The BEB doubles the previous RTO, whereas VBF uses different variable backoff values for high or low initial RTO as shown in Eq. (1) to avoid the frequent retransmissions.

$$\mathrm{VBF} = \begin{cases} 3 & \text{initial RTO} < 1\mathrm{s} \\ 2 & \text{initial RTO for the interval } [1\mathrm{s}, 3\mathrm{s}] \\ 1.3 & \text{Otherwise} \end{cases} \tag{1}$$

Still CoCoA+ depends on the weak and strong estimators at both the endpoints to determine the overall RTO. The priority of the weak or strong estimator is less compared with the previous overall RTO. So, the computed overall RTOs are very close to the RTTs. Besides, the per-packet estimation of RTT not always the proper measure of the congestion in both CoCoA and CoCoA+, because sometimes the RTTs are noisy (Rathod et al. 2019).

Further extension of CoCoA+ with optimized RTO estimator is done in precise CoCoA (pCoCoA) (Bolettieri et al. 2018) and CoCoA++ (Rathod et al. 2019). The pCoCoA uses only one RTO (smooth RTO) rather than maintaining two RTO estimators. Additionally, pCoCoA uses the retransmission count at each ACK during the CON message. Because of this feature, it avoids the duplicate retransmissions of a packet. Its computational complexity of RTO calculation is minimum when compared with CoCoA+ or CoCoA. The CoCoA++ also maintains a single RTO estimator, and it computes an overall RTO by integrating with the CAIA Delay-Gradient (CDG) and Probabilistic Backoff Function (PBF). CDG gets the congestion information from the TCP's congestion window (queue) and packet loss. CoCoA++ replaces the VBF with PBF during the RTO computation, and it does not consider per-packet RTT like others discussed above. In CoCoA++, there is an ambiguity that either minimum or maximum delay-gradient results in the best overall RTO. Genetic CoCoA++ has been introduced in Yadav et al. (2020) for CoAP by extending the CoCoA++ protocol.

The congestion control Random Early Detection (CoCo-RED) has been developed in Suwannapong and Khunboa (2019) using revised random early detection (RevRED) and a Fibonacci Pre-Increment Backoff (FPB) function to compute the RTO estimator. For each retransmission in CoCo-RED, the overall RTO value is determined by multiplying the *i th* Fibonacci value with the initial RTO value. The CoCo-RED is enhanced recently by Suwannapong and Khunboa (2021) to manage the buffer and traffics. In these two approaches, the RTO computation strategy is straightforward and has low computational overhead. But, increasing the number of retransmissions also increases the RTO value exponentially. Overall, most of the advancements done in the CoAP are concerning avoiding congestion by computing an optimal overall RTO estimator in a static environment. Figure 1 shows the four continuous RTO values with the initial RTO of 2 of CoCoA, CoCoA+, pCoCoA, and CoCo-RED. Demir and Abut (2020) use machine learning-based CoAP to address the congestion. They use support vector machine to estimate the congestion level in the network. Zhang et al. (2022) proposed an upper confidence bound strategy to make the CoAP dynamic. A fuzzy logic based adaptive CoAP is introduced by Aimtongkham et al. (2021) to determine the adaptive RTO. The CoAP message format is analyzed in both TCP and UDP formats by Agyemang et al. (2022).

Xiao et al. (2019), Nie et al. (2019) have used DRL to address the congestion control for TCP protocol. In Xiao et al. (2019), a DRL-based smart congestion control

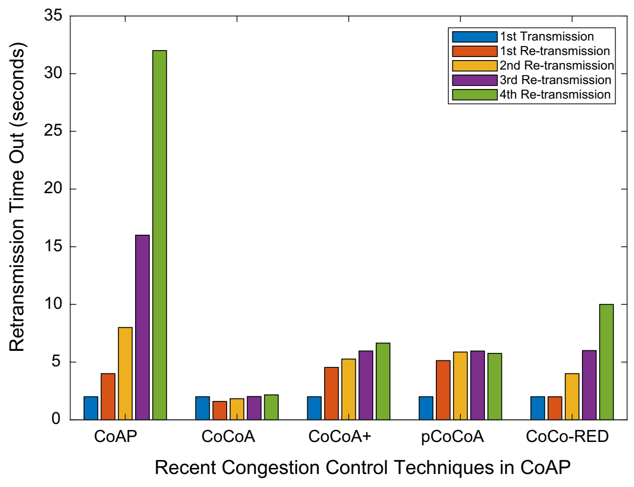

Fig. 1 Comparison of RTOs in four transmissions to the recent Congestion Control Techniques for CoAP

protocol has been developed. It controls the congestion based on past experiences. In Nie et al. (2019), the authors used the Asynchronous Actor-Critic Agents (A3C) approach, a DRL method, to address the congestion issue over the TCP and also manages the TCP initial window size.

The existing methods which are discussed in this section are similar to Internet congestion control methods. These are also consider the static environment and previous RTTs to decide the best RTO for further transmissions. So, dynamic, efficient and intelligent protocols require to mitigate the challenges for IoT including the congestion issues. In this context, proposed iCoCoA method uses UDP transport and applies DRL for correctly predicting the RTO to minimize unnecessary retransmission for congestion mitigation.

3 Problem formulation

In this section, we present the problem formulation of the proposed iCoCoA. The energy consumption (EC) of an IoT device mainly considers the energy drain for data acquisition by the sensor embedded in it, processing and the transmissions (Martinez et al. 2015). Based on these assumptions, we compute the EC of a device i using Eq. (2):

$$E_i = E_p(i) + E_d(i) + E_{tx}(i) + \varepsilon \tag{2}$$

where $E_p(i)$ is the EC for processing the data and depends on the $E_d(i)$ in terms of data type (arithmetic or non-arithmetic), and selected hardware architecture, clock cycles used, etc. $E_d(i)$ denotes the energy drain during the data acquisition of node i and it is computed as follows:

$$E_d(i) = \begin{cases} P(i) \times E_s & \text{For Continuous} \\ P(i) \times E_s \times \mathbb{P}_t(i) & \text{For Event driven} \end{cases} \tag{3}$$

where E_s indicates the energy needed for a sample of sensed data or payload, and $\mathbb{P}_t$ indicate the probability of the occurrence of the event during a unit time interval t. The $P(i)$ is the total number of packets collected by a node i.

$$P(i) = \begin{cases} \sum_{t=0}^{T} P_t(i) & \text{For Continuous} \\ \sum_{t=0}^{T} \left(P_t(i) \times \mathbb{P}_t(i)\right) & \text{For Event driven} \end{cases} \tag{4}$$

where $P_t(i)$ means the number of samples acquired during a unit time interval t at mote i, and T means the total simulation time. The $E_{tx}(i,j)$ denotes the energy dissipated during the data transmission from device i to j, is computed as shown in Eq. (5) Donta et al. (2020).

$$E_{tx}(i,j) = \left(\alpha_{tx} + \alpha_{fs} \times \Delta_{ij}^2\right) \times \Gamma_i \tag{5}$$

where α_{tx} is the EC for processing the data by circuits, α_{fs} is the energy dissipation for amplification, Δ_{ij} is the distance between the devices i to j, and the Γ_i is the number of data transmissions by mote i, and it is computed as follows:

$$\Gamma_i = \sum_{k=0}^{P(i)} (\eta(k) + 1) \tag{6}$$

where $\eta(k)$ denotes the number of retransmissions required to the sensed data packet k. The ε indicates the additional EC to handle the resource and managing the tasks. Additionally, the average EC of the network is computed as follows:

$$E = \frac{1}{n} \sum_{i=1}^{n} E_i \tag{7}$$

where n indicates the number of clients/devices in the network. The packet delivery ratio (PDR (Ψ)) is the ratio of the total number of packets received by the server by excluding ACKs ($\mathcal{R}$) and the total number of packets transmitted by other nodes to the server by excluding ACKs ($\mathcal{T}$) during the time T. PDR is computed as follows:

$$\Psi = \frac{\mathcal{R}}{\mathcal{T}} \tag{8}$$

where $\mathcal{R} \leq \mathcal{T}$, and $\mathcal{T}$ is calculated using the Eq. (9)

$$\mathcal{T} \cong \sum_{i=1}^{n} P(i) \tag{9}$$

from Eqs. (8) and (9), the number of packets lost (Φ) during T can be estimated as $\Phi = \mathcal{T} - \mathcal{R}$ or the percentage of packet lost is $\Phi_a = (1 - \Psi) \times 100$. The throughout ($\sigma$) of the network is determined based on the total amount of packets received by the server during the time T as shown in Eq. (10).

$$\sigma = \frac{\mathcal{R}}{T} \tag{10}$$

The end-to-end delay/latency (d) of a packet is computed as the total time taken by a packet to travel from the source to destination. The d includes the queuing delay (d_q), radio propagation delay (d_r), signal processing delay (d_s), and transmission delay (d_t). From these, $d_r(k) \approx d_s(k) \leq 1$, so we neglect $d_r(k)$ and $d_s(k)$ because of no effect on outcome. The latency of the packet k is calculated as shown in Eq. (11)

$$d(k) = \begin{cases} d_q(k) + d_t(k) & \text{For Successful} \\ \left(d_q(k) + d_t(k)\right) \times \eta(k) & \text{For retransmitted} \end{cases} \tag{11}$$

The average d of the network is computed as shown in Eq. (12)

$$d = \sum_{i=1}^{n}\left(\sum_{k=1}^{P(i)}(d(k))\right) \times \frac{1}{\mathcal{R}} \tag{12}$$

The RTT is the time delay for a packet to send from source and receive an ACK from the server, it may be asymmetric, and not always equal. Simply, the sum of $d(k)$ and the time taken to receive an ACK ($\mathcal{A}$) i.e. shown in Eq. (13)

$$\lambda(k) = d(k) + \mathcal{A}(k) \tag{13}$$

the average RTT time is computed using Eq. (14)

$$\lambda = \sum_{i=1}^{n}\left(\sum_{j=1}^{P(i)}\left(\sum_{k=0}^{\eta(j)}\lambda(k) \times \frac{1}{\eta(j)}\right) \times \frac{1}{P(i)}\right) \times \frac{1}{T \times n} \tag{14}$$

The maximum σ always minimizes the E. The E also can be minimized by minimizing the η, where η indicates the average number of retransmissions computed as shown below:

$$\eta = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{P(i)}\sum_{j=1}^{P(i)}\eta(j)\right) \tag{15}$$

The minimization of η value maximizes the σ, and it also minimizes the d and λ. The σ is also maximized when Φ value is minimized. It will be minimized when Ψ is maximized. Finally, we achieve the Eq. (16) through η with optimal RTOs.

With the observations from Fig. 2, The primary goal of the congestion control is for handling trade-off between maximizing σ, minimizing λ and other parameters (Jay et al. 2019). To trade-off the design goal of low λ and high σ, we adopt a utility function (Xiao et al. 2019) as shown in Eq. (16)

$$U(\lambda, \sigma) = \big((1-\varphi) \times U_\vartheta(\sigma)\big) - \big(\varphi \times U_\vartheta(\lambda)\big) \tag{16}$$

where $\varphi \in [0, 1]$ is the relative importance of the σ and λ and the $U_\vartheta(x)$ is computed as follows

$$U_\vartheta(x) = \begin{cases} \log(x) & \text{if } \vartheta = 1 \\ \frac{x^{1-\vartheta}}{1-\vartheta} & \text{Otherwise} \end{cases} \tag{17}$$

where ϑ is the fairness value ranging (0,∞), and x is either σ or λ (Xiao et al. 2019). The goal of the proposed algorithm is to optimize the Eq. (16).

4 Proposed iCoCoA protocol

This section provides the detailed discussion on the proposed iCoCoA. Initially, we provide the discussion on the DRL and its elements. Subsequently, we discuss about Experience Replay Buffer (ERB) followed by the design of the agent for the proposed model. Furthermore, we

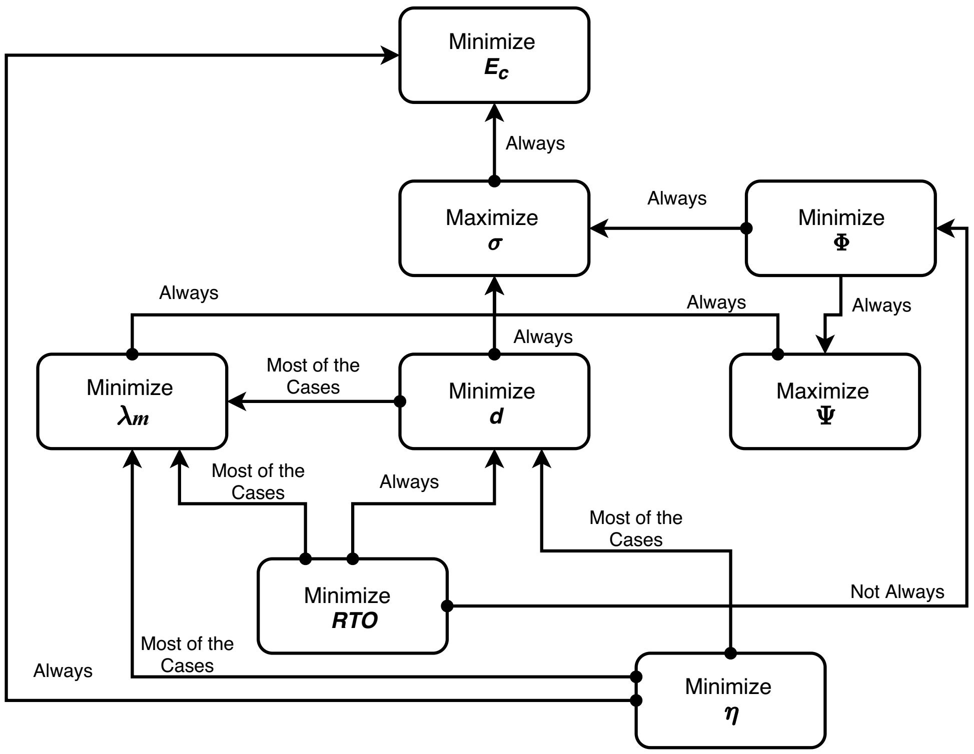

Fig. 2 Dependency of various network features

present the training and running process of the iCoCoA. The working model of the proposed iCoCoA is summarized using Fig. 3. The client and server are the main components of this model to transmit their data packets and the control signal. The agent works in server to determine the best and most dynamic RTO based on previous RTO and other features, which further help to mitigate the congested situations in the network.

4.1 Deep reinforcement learning

The proposed method uses a deep-Q-network (DQN), it is a category of the DRL approach. In general, the agent and environment are the two basic elements of the DQN (Sutton and Barto 2018). The agent trains by interacting with the environments in a fixed time slot t and it operates from the CoAP server. During each t, the agent receives μ inputs as state s_t to take an action a_t depending on a policy $\pi_\theta(s_t, a_t)$ and receives a reward $r(s_t, a_t)$. The agent updates the s_t values in each iteration at the predetermined t. The agent considers various network features such as minimum RTT (λ_m), RTTVAR (δ), initial RTO (τ) and the number of retransmission (η) as a state information. The λ_m value is computed using Eq. (18) which is similar to TCPs computation used in Sargent et al. (2011):

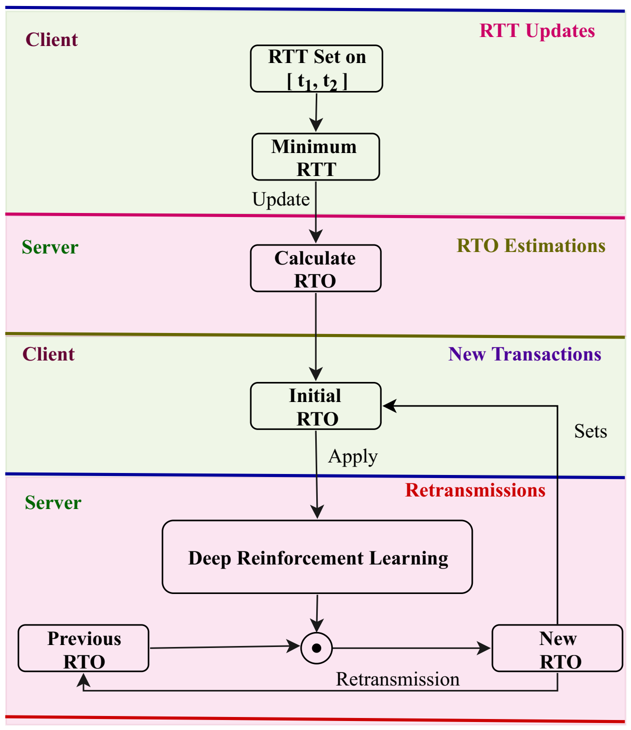

Fig. 3 The working model of the Proposed iCoCoA

$$\lambda_m = \begin{cases} \min_x \lambda_m(x) \text{ on } x \in [t_1, t_2] & \eta = 0 \\ (1-\alpha) \times \lambda_m + \alpha \times \lambda'_m & \text{Otherwise} \end{cases} \tag{18}$$

where λ'_m is the new λ_m, $\alpha = 0.125$, t_1 and t_2 are the two consecutive time slots. The δ can be estimated using Eq. (18).

$$\delta = \begin{cases} \frac{\lambda_m}{2} & \eta = 0 \\ (1-\beta) \times \delta + \beta \times \left(\lambda_m - \lambda'_m\right) & \text{Otherwise} \end{cases} \tag{19}$$

where $\beta = 0.25$, and the τ is computed using Eq. (20):

$$\tau = \lambda_m + max(G, k \times \delta) \tag{20}$$

where G is the granularity time (1 ms), and the value of $k = 4$. From these, we form a state set to compute the new RTO. The combination of the $\mu = 4$ states at t are denoted in Eq. (21).

$$s_t = \left\{\tau_t, \lambda_{mt}, \delta_t, \eta_t\right\} \tag{21}$$

The action space A considers four possible actions to control the congestion with the new RTO (τ'') computations. The possible action sequences are to update the previous RTO by increasing or decreasing it, the previous RTO τ', consider the initial RTO (τ_t), or drop the packet (no further transmission). The action space at t is defined as shown in Eq. (22).

$$A = \left\{\tau''_t = Eq.\,(20), \tau''_t = \tau'_t, \tau''_t = \tau_t, \tau''_t = 0\right\} \tag{22}$$

The selection of any possible action a_t is decided by the agent after it receives the state s_t information using a *policy* (π_θ). Note that the DQN used in iCoCoA is a model-free and off-policy approach (Krizhevsky et al. 2017), which trains the agent over various adjustable parameters (θ) to maintain the $\pi_\theta(s_t, a_t)$ to determine the best possible action a_t to the current state s_t depends on the Eq. (23).

$$\pi_\theta\left(s_t, a_t\right) = arg \max_a Q(s, a; \omega) \tag{23}$$

where ω denotes the weights of the DQN. The distribution approach which followed by Eq. (23) ensures apposite exploration of the states.

Another important considerable parameter in the agent is the reward function (R_t). Designing an accurate R_t is a challenging issue for DQN to control the congestion over CoAP. The agent receives a scalar value as a reward R_t from each desirable action a_{t+1} for a state s_{t+1}. The primary goal of the agent is to maximize the expected cumulative reward that it receives from the reward function, which aims to improve the throughput by controlling the congestion. The reward function R_t we consider in the proposed work is shown in Eq. (24).

$$R_t = r_t + \gamma^1 R_{t+1} + \gamma^2 R_{t+2} + \cdots = \sum_{i=t}^{\infty} \gamma^{i-t} r_i(s_i, a_i) \tag{24}$$

where $\gamma \in [0, 1]$, and $r_i(s_i, a_i)$ is defined as shown in Eq. (25). The aim of R_t is to keep the network channel busy, but not overflow. The agent consider the immediate reward if the γ value is closer zero. If the γ is closer to one, the future reward with highest weight is considered by the agent.

$$r_i(s_i, a_i) = \sigma_i \times \log\left(\frac{\lambda_i}{\sigma_i}\right) \tag{25}$$

The Q-value function $Q^{\pi}(s_t, a_t)$ in this article basically uses a given input state s_t to determine optimal action a_t for a given policy π_θ is determined based on the Eq. (26) Mnih et al. (2015).

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_s\left[R_t | s_t, a_t, \pi\right] \tag{26}$$

where the expanded Q-value function is shown in Eq. (27).

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s'}\left[Y_t^{DQN} | s_t, a_t, ;\pi\right] \tag{27}$$

where Y_t^{DQN} is represented as shown in Eq. (28).

$$Y_t^{DQN} = r_t + \gamma \max_{a'} Q^*(s_t', a_t'; \omega') \tag{28}$$

where $Q(s_t, a_t, \omega) \approx Q^{\pi}(s_t, a_t)$, and the optimized *loss function* at *i th* iteration for DQN is computed as shown below:

$$L_i(\omega_i) = \mathbb{E}_{s,a,r,s'}\left[\left(Y_i^{DQN} - Q(s, a; \omega_i)\right)^2\right] \tag{29}$$

At each iteration, the previous value of ω_i' $(= \omega_{i-1})$ holds when fixing Eq. (29). But in the final iteration ω_i' will be ignored because, this stage of optimization uses the variance of the targets.

4.2 Experience replay buffer (ERB)

However, the IoT environment performs the dynamic changes in the network features, sometimes the set of features to cause of congestion are repeated. This kind of repeated situations does not require new solutions or further learning process. Unlike recent RL approaches, the DRL takes this advantage by using ERB. The ERB maintains a set of past experiences of the agent, and allow the agent to stabilize training and break undesirable temporal correlations to minimize the computational time. In which, each time-stamp t, the set of values $\Gamma_t = (s_t, a_t, R_{t+1}, s_{t+1})$ updates the ERB in each iteration and the dataset becomes $\mathcal{D}_t = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{|\mathcal{D}|}\}$. Generally, the size of the ERB $|\mathcal{D}|$ in DRL is set to be multiples of 10K (Xiao et al. 2019). Due to the memory constraints in IoT, we set the $|\mathcal{D}| = \lceil n \times log_e(n) \rceil$.

The agent considers the set of input state values from ERB in the iCoCoA. The data stored in ERB decides either it requires further training process or not. Initially, the ERB is empty, and it fills during the training and running process. If the set of input features are available in the ERB, it provides the stored reward and action without additional computations. If the input network features are not available in the ERB, it moves to further training process and the outcome of the training results are stored into the ERB. The ERB updates the buffer according to the first-in-first-out (FIFO) when it is full. So, it maintains only the most recent data because of the limited available memory.

4.3 Design of agent

The agent periodically checks the network to detect the changes in the network features, to improve the learning process by adopting the changing conditions. The iCoCoA learns from these experiences to determine the new RTO, to control the unnecessary retransmission over the network. The agent uses the Deep Convolutional Neural Network (DCNN) to produce the actions of the system. The DCNN is processing the data in a sequence of layers, and each layer performs a differentiable function to transform the input from one format to another format to produce the desired output. From Fig. 4, the three main layers between input and fully-connected (FC) layers of the DCNN architecture are one or more convolutional (CONV), Non-linearity, and pooling layers, respectively.

The DCNN chooses M consecutive state features from the ERB periodically and convert them into μ (value is equal to the number of states) frames of size $\lfloor\sqrt{M}\rfloor \times \lfloor\sqrt{M}\rfloor$. These frames are taken as input by the CONV layer. The CONV layer is the core building block of a DCNN, that does most of the computations with a set of learnable parameters. In this work, we consider two CONV layers, and each associated with a non-linearity layer. In general, a Non-linearity layer uses a rectified linear unit (ReLU) activation function, whereas the iCoCoA adopts an experimental linear unit (ELU).

$$ELU(x) = \begin{cases} \mathcal{S}_x \times (e^x - 1) & x \le 0 \\ x & x > 0 \end{cases} \tag{30}$$

where $\mathcal{S}_x$ indicate the state value belongs to x. The primary goal of the ELU is to suppress the negative values.

The first CONV layer uses the input volume size $\sqrt{M} \times \sqrt{M} \times \mu$ by considering the receptive field of 3×3 neurons, with a stride of one, and the zero-padding value of one. Each receptive field extracts a feature at every part of the input frame using the Eq. (31):

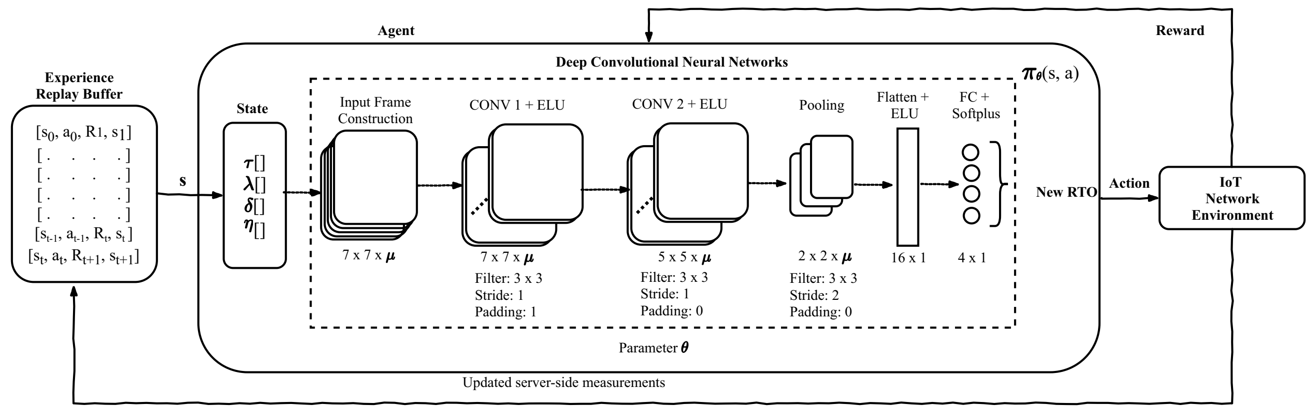

Fig. 4 Working model of DQN agent for iCoCoA

$$f(x) = \sum_{i=1} w_i x_i + b \tag{31}$$

where w_i is the random weight of each neuron example w={1, 0, -1}, and the b is the bias. The expected output volume size of the first CONV layer is $\lceil\sqrt{M}\rceil \times \lceil\sqrt{M}\rceil \times \mu$, and it will be the input to the second CONV layer. The second CONV layer constructed with the 3×3 filter with the stride of one and no zero-padding hyperparameters. This layer uses the Eqs. (30) and (31) internally, and produce the output volume for the input to the pooling layer.

The primary purpose of the pooling layer is to reduce the spatial dimensions of the CONV layer output. In the agent of iCoCoA, we consider a single pooling layer of 3×3 neurons in the receptive field, with sliding of two and no zero-padding. The output volume of pooling layer is input to the Flatten layer and the Flatten layer process of the output of the pooling layer to convert it into to single dimension vector. During this conversion, Flatten uses Softplus or SmoothReLU as a Non-linearity layer as shown below:

$$f(x) = \log(1 + e^x) \tag{32}$$

Finally, the FC layer extracts the desired number of resultant feature values from the preceding layers. Further, the policy π_θ will choose the desired action based on the Q-values available in the FC layer.

4.4 Training and running

The DRL gains knowledge from the experiences, and it generates the right decisions by training with different network conditions and various features. Initially, the agent chooses an action randomly because of the dataset unavailability for the training process. After a few iterations, the agent decides and performs the actions based on π_θ using Eq. (23) with the output Q-values. The resultant action and reward are stored in the ERB, along with a set of network features. Further simulation runs over time will frequently change the network environment, and keep on varying the features. With these features, the proposed iCoCoA operates training by the agent using a DCNN approach, and it decides an appropriate action for a given set of states to determine RTO value.

After training, the state, reward, and action set will remain stored into the ERB, which are useful during the online running process. The ERB determines the changes in the input network features before the agent starts its training process. So, it reduces unnecessary computations over the duplicate features and also speedup the online running process with earlier action decisions. Note that it is necessary to use off-policy π_θ while retrieving the values form the ERB. Similar to Mnih et al. (2015), the training clips the loss function value while updating the Eq. (29) to [– 1, 1] and the values between the interval (– 1, 1) are clipped to absolute values. Thus the negative values are clipped to -1 and positive values to 1. Along with these, the rewards also clipped to 1 for all positive values, – 1 for all negative values and leaving 0 if no change. The stability of the proposed algorithm will improve with this form of clipping on the loss function and rewards.

5 Experimental results

We compare the existing but related standard algorithms such as CoAP, CoCoA and CoCoA+ with the proposed iCoCoA. The simulation setup of the network and the implementation of algorithms were tested in the Contiki v3.0 using Cooja simulator. We consider the Zolertia (Z1) mote with the specification of 8 KB RAM, 96 KB ROM, MSP430F2167 (v4.7.3) MCU model with CC2420 Radio for both CoAP server and client. These nodes are deployed randomly in a rectangular plane, and all the nodes are statically placed. The channel model used in the simulation is the Unit Disk Graph Medium with the T_x range of 10 m and an interface range of 25 m. Further parameters which we consider during the simulations are listed in Table 1. The comparison parameters consider an average number of retransmissions with variable initial RTO values, PDR, throughput, energy consumption, fairness index of the congestion and energy consumption. In this study, we tested both bursty and continuous network traffic scenarios. In the bursty traffic, the network traffic varies such as sudden peak or fall in various parts of the network. In continues scenario, the traffic is flow at a regular speed without any interruption. The simulation study perform multiple tests under various conditions with more number of iterations.

Table 1 Simulation parameters

Parameter	Value
1. Routing protocol	RPL
2. Max no. of retransmissions	Four
3. MAC level max retransmissions	Eight
4. Number of motes	100
5. Number of mote types	3 (RPL, Client, and CoAP Server)
6. CoAP ACK-Timeout	3 s
7. CoAP request buffer size	Four
8. Network Protocol	IPv6 + 6LoWPAN
9. Physical	IEEE 802.14.4, 250Kbps PHY
10. Maximum overall RTO	60 s
11. CONV1 Frame size (M)	49
12. Size of Payload	$0–2^8 - 1$

To avoid the replication of the result, we presented a few of the results and analysis here.

5.1 Average number of retransmissions

The number of packet retransmissions is directly proportional to the level of congestion. It means increasing the congestion affects packet loss or ACK delays, and it leads to increasing the number of retransmissions. It also affects the unwanted energy consumption of the nodes. The average number of retransmissions (η) in this article is computed using Eq. (15). Figure 5 shows the η during the simulation time between 0-300 seconds for both continuous and burst scenarios.

From Fig. 5a, we observe that no retransmissions are there until few iterations because of less traffic. After some time, the network traffic increases, gradually raising the retransmissions. The η of four algorithms are varied, and the proposed iCoCoA results in less or equal number of retransmissions in most of the cases. From Fig. 5b, the performance of the iCoCoA is better and performs less number of retransmissions and the η value rarely touches four. In a continuous network scenario, the iCoCoA achieves more than 18–29% less retransmission, whereas, in a burst scenario, it improved up to 13–27% compared with existing approaches. The cause of fewer retransmissions is because of the efficient RTO computation based the network traffic and the experience of the previous congestion cases.

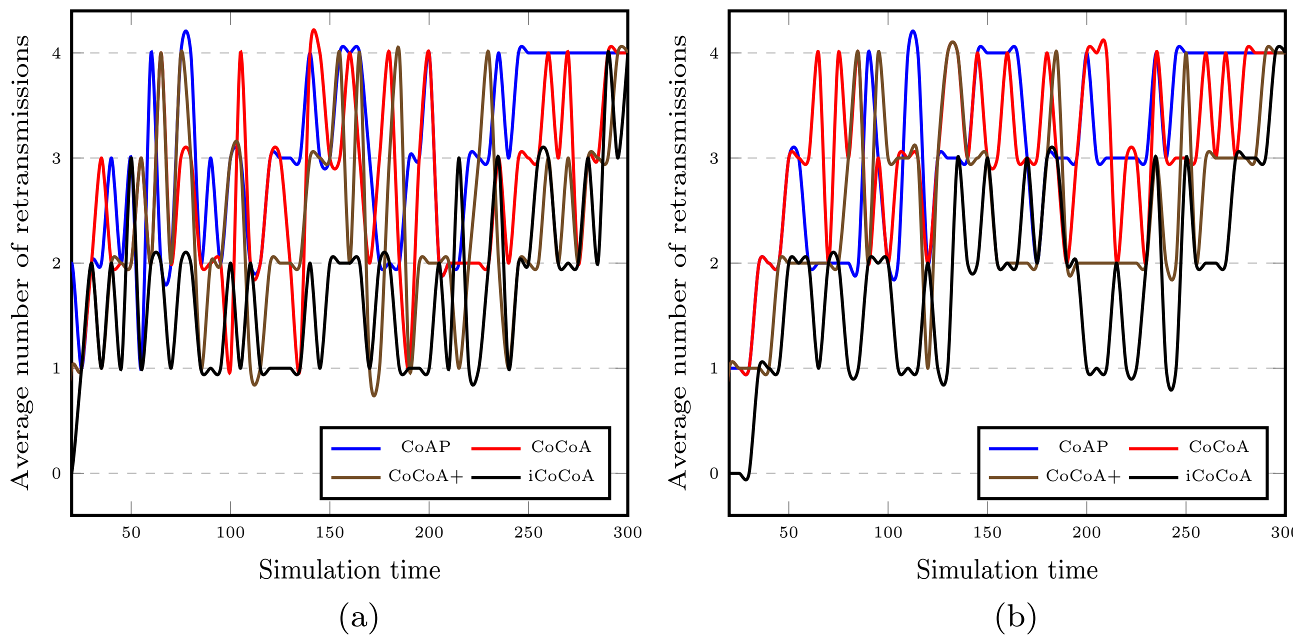

Fig. 5 Average Number of retransmissions in **a** continuous **b** burst Scenarios

5.2 Carried load per node

The carried load is an important parameter, and it shows the nodes' congestion levels during the data transmissions. The heavy carried load indicates the more chance to lead the congestion. The load may be increased due to ACKs for the successful and unsuccessful transmissions. The carried load of the first 50 sensor nodes during the simulation time $\mathcal{T} = 20S$ for both the Continuous and Burst scenarios in Fig. 6a, respectively. The iCoCoA results in less carried load over the default CoAP, CoCoA, CoCoA+ and EnCoCo-RED. We observe the better-carried load in the iCoCoA is because it avoids unnecessary data transmissions and ACKs. From Fig. 6a, we can also observe that the congestion is not equally shared among all the nodes in the network. Sometimes, the nodes far from the server are discarded earlier than the nodes closer to the server. In iCoCoA, it can be controlled to reduce the carried load of the ACKs, and overall it improves the performance by mitigating the congestion.

5.3 Packet delivery ratio (PDR)

The PDR is defined as the ratio of number of packets received at destination and the number of packets transmitted by the motes. It is directly proportional to σ and inversely proportional to the congestion degree. The PDR of the proposed method is computed using the Eq. (8). The data driven application such as IoT, reducing packet loss is very important.

Figure 7a, b show the comparisons of the proposed and existing methods concerning the PDR during the simulation runs for both continuous and burst scenarios, respectively. The percentage of PDR reduces gradually as the simulation time increases, and it happens because of occurring the congestion. The mote holds the packet until a specified deadline called RTO, and if it exceeds, the mote drops the packet. The iCoCoA outperforms compared with the existing approaches and gives the best PDR. It increased the PDR to approximately 10%, 6–8%, 3–7%, and 3–6% when compared with the CoAP, CoCoA, CoCoA+, and EnCoCo-RED, respectively. These improvements are achieved because of the proper estimation of RTO by considering past experiences. The iCoCoA still causes packet loss when there is no possibility of controlling the congestion and exceeds the buffer timeout. However, it is minimal when compared with the other existing approaches.

The simulations runs are tested with varying the payload size of the CoAP request, and responses for 2^5–2^{15} are shown in Fig. 8. The continuous and burst scenarios of the PDR with variable payload size are presented in Fig. 8a and Fig. 8b, respectively. From Fig. 8, we notice that the increasing payload is decreasing the PDR. Increasing the payload will also increase the carried load. It also affects the buffer occupancy of the packets. So, each retransmission of a packet is highly affected by the various performance metrics, including PDR.

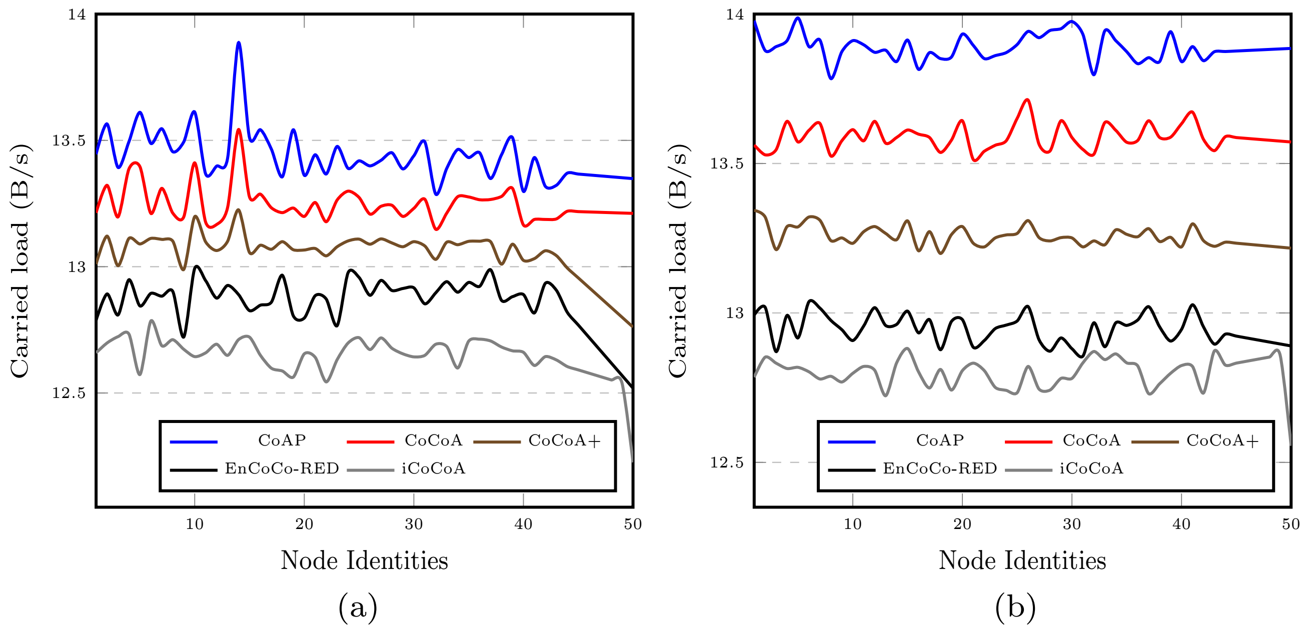

Fig. 6 Carried load per node at simulation time is 10 s **a** continuous **b** burst

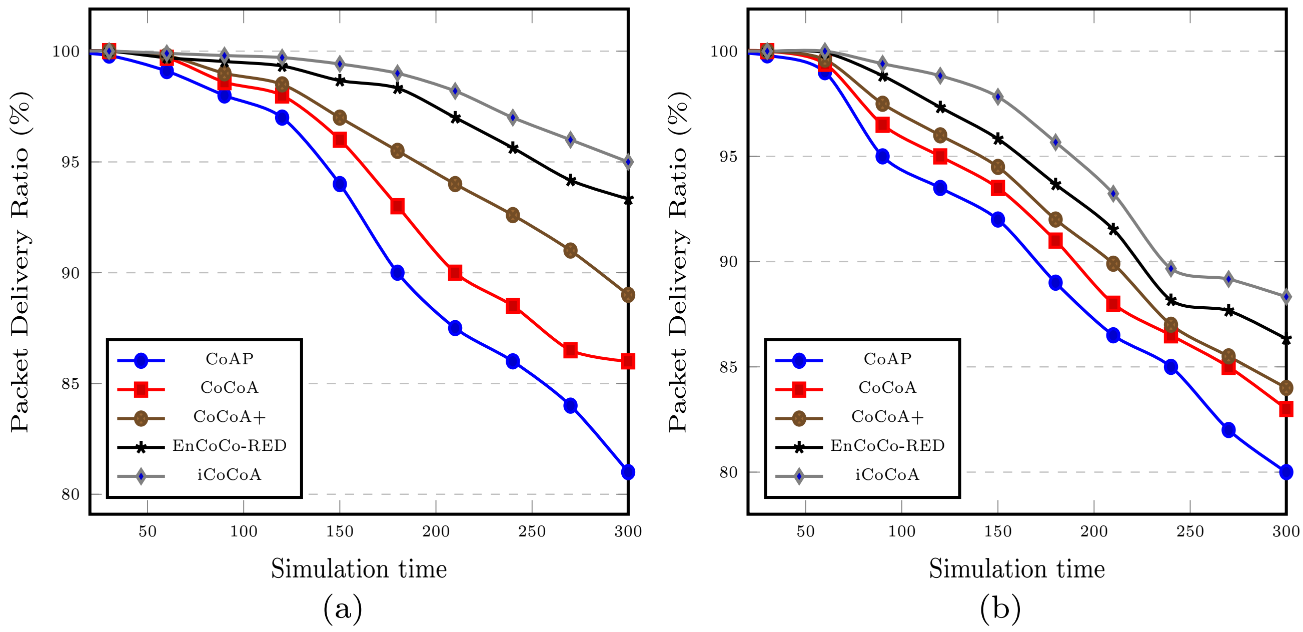

Fig. 7 Packet Delivery Ratio by varying the Simulation time **a** continuous **b** burst

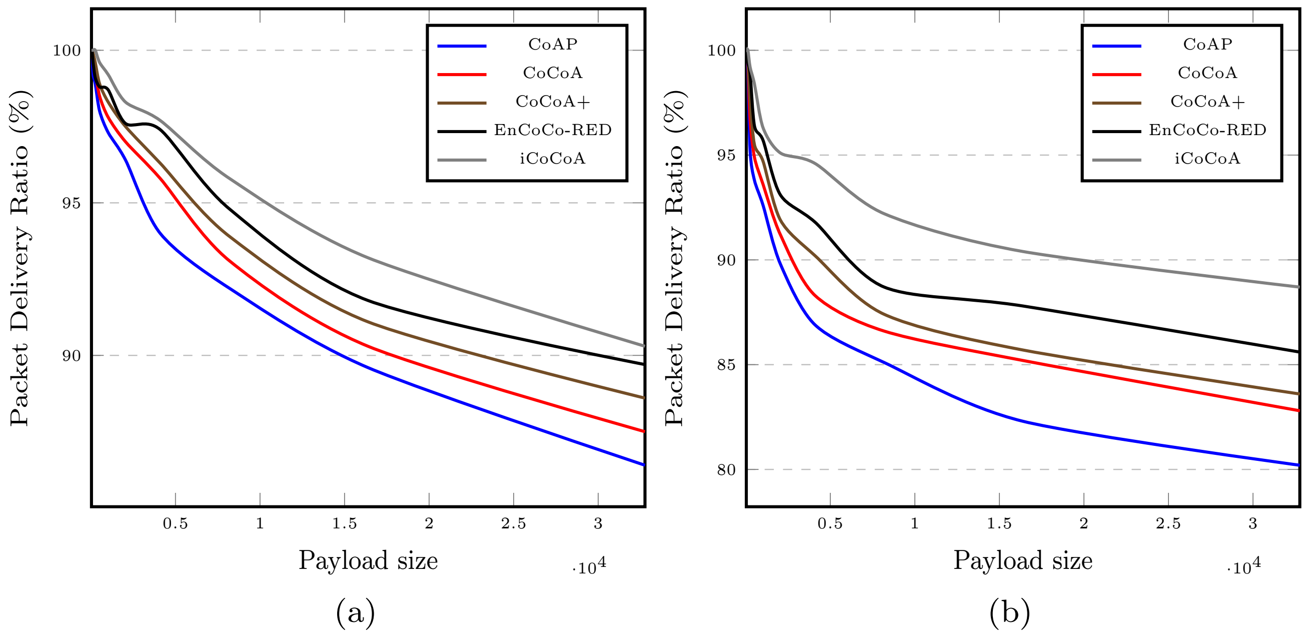

Fig. 8 Packet Delivery Ratio by varying the Payload size **a** continuous **b** burst

5.4 Throughput

The throughput (σ) of the iCoCoA is computed using the Eq. (10). It depends on the various parameters which are described in the Sect. 3. The congestion and σ are inversely proportional to each other, it means decreasing the congestion automatically improves the throughput. Figure 9, shows the comparison of the percentage of the throughput during the simulation runs and we plot up to 300 s.

Figure 9a shows the comparison of throughput in continuous scenario and Fig. 9b represents burst strategy. From there, we observe that the throughput of the system decreases gradually in both the scenarios. Initially, the retransmissions of the iCoCoA are similar to the existing

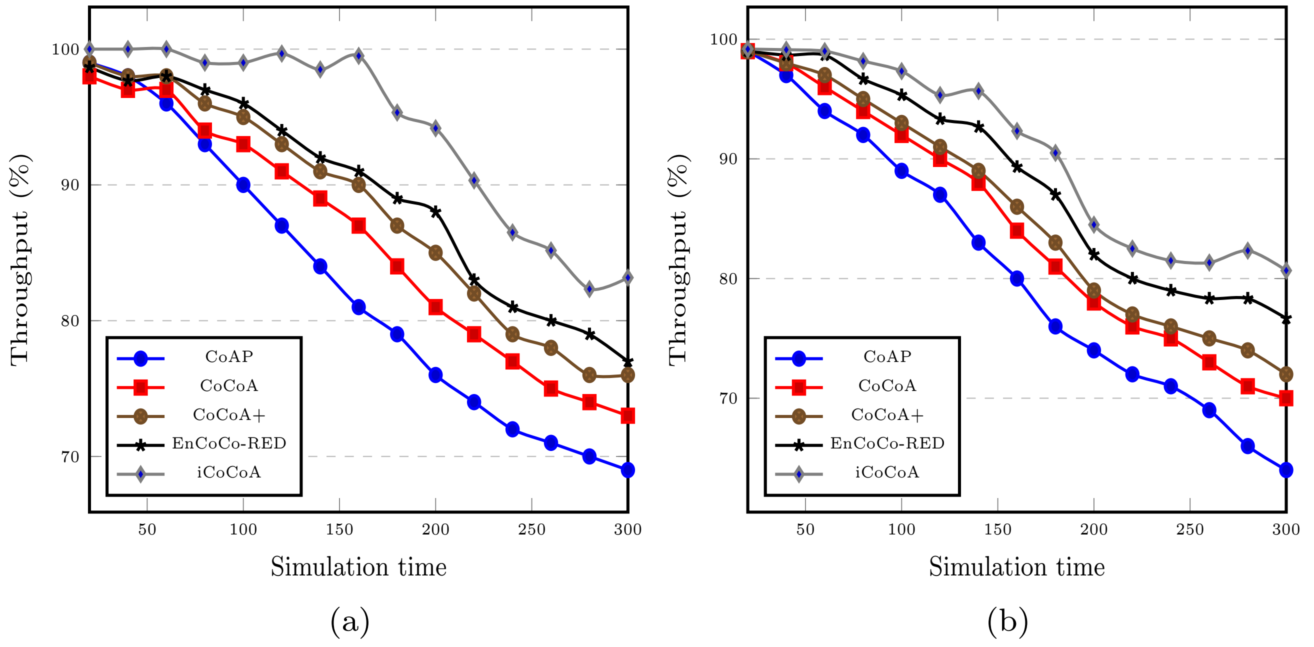

Fig. 9 Throughput vs. Simulation time **a** continuous **b** burst

approaches because of the arbitrary decisions. Slowly, the proposed approach increased its throughput because of handling congestion based on the experiences. It also avoids unnecessary frequent retransmissions to reduce network traffic and channel overflows. Hence, the iCoCoA increases approximately 10–15%, 5–7%, 3–5%, , and 2–5% of the throughput when compared with CoAP, CoCoA, CoCoA+, and EnCoCo-RED, respectively, for both continuous and burst scenarios varying the simulation time. The throughput of the proposed and existing CoAP, CoCoA and CoCoA+ are presented in Fig. 10 by varying the payload size. Even when the payload size increases the iCoCoA is performing better than the other existing algorithms. The improvement of the iCoCoA is 11–14% better

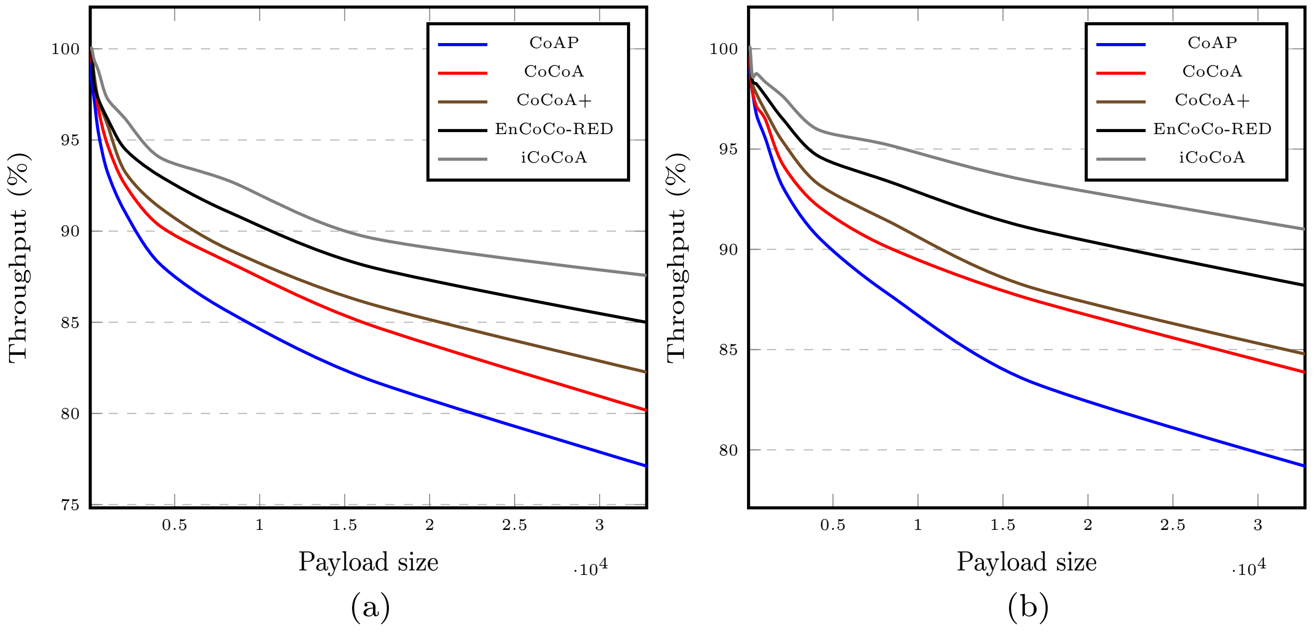

Fig. 10 Throughput vs. Payload size **a** continuous **b** burst

than CoAP, 4–7% than CoCoA, 2–5% than CoCoA+, and 2–4% than EnCoCo-RED.

5.5 Fairness index of congestion

The fairness index of the congestion estimation (F_σ) determines the equal share of bottleneck of congestion among the network. There are several methods to calculate fairness index (HoBfeld et al. 2017), whereas we follow Donta et al. (2021) to compute F_σ. The F_σ value ranges $0 \leq F \leq 1$, whereas the higher value of the F_σ shows that the maximum fairness and vice versa. The F_σ of the given network is computed as follows:

$$F_\sigma = 1 - \left(\frac{2 \times S_\sigma}{H_\sigma - L_\sigma} \right) \tag{33}$$

where S_σ indicates the standard deviation (SD) of the throughput during the total simulation time T, L_σ and H_σ denote the minimum and maximum throughput during T. The lower the SD, the higher the F_σ value and vice versa. The Eq. (34) shows the calculation of the SD of the throughput. The value of t is updated as $t = t + k$, where the value of k is same as Eq. (20).

$$S_\sigma = \sqrt{\frac{\sum_{t=1}^{T} (\sigma_t - \sigma)^2}{T}} \tag{34}$$

The F_σ of the congestion for the CoAP, CoCoA, CoCoA+, EnCoCo-RED and iCoCoA for the running example of continuous scenario are 0.8464, 0.8663, 0.8781, 0.8816 and 0.9006, respectively. Whereas for the burst network the F_σ is 0.8594, 0.8776, 0.8846, 0.8898 and 0.9009 for CoAP, CoCoA, CoCoA, EnCoCo-RED and iCoCoA, respectively. The proposed iCoCoA achieves the higher F_σ compared with the existing strategies. The higher F_σ indicates the equal share of the bottleneck in the congestion among the motes. The iCoCoA achieves the best F_σ over existing methods due to choosing efficient and dynamic RTOs by avoiding frequent retransmissions. The proposed method decides these RTOs based on the experience from the previous congestion cases.

5.6 Average energy consumption

The energy is one of the primary constraints for the IoT devices, because it operate with a low powered battery and continuous monitoring. In this work, we compute the EC of a device using the Eq. (2) and the average energy consumption (E) is determined as shown in Eq. (7). As we discussed in Sect. 3, more energy is consumed for data transmissions. The congestion decreases the E while increasing the number of retransmissions.

The comparison results of the proposed and existing methods are presented in Figs. 11 and 12. From the Fig. 11a, the E of the continuous scenario decreased approximately 8%, 7%, 5%, and 2–5% when compared with the CoAP, CoCoA, CoCoA+, and EnCoCo-RED,

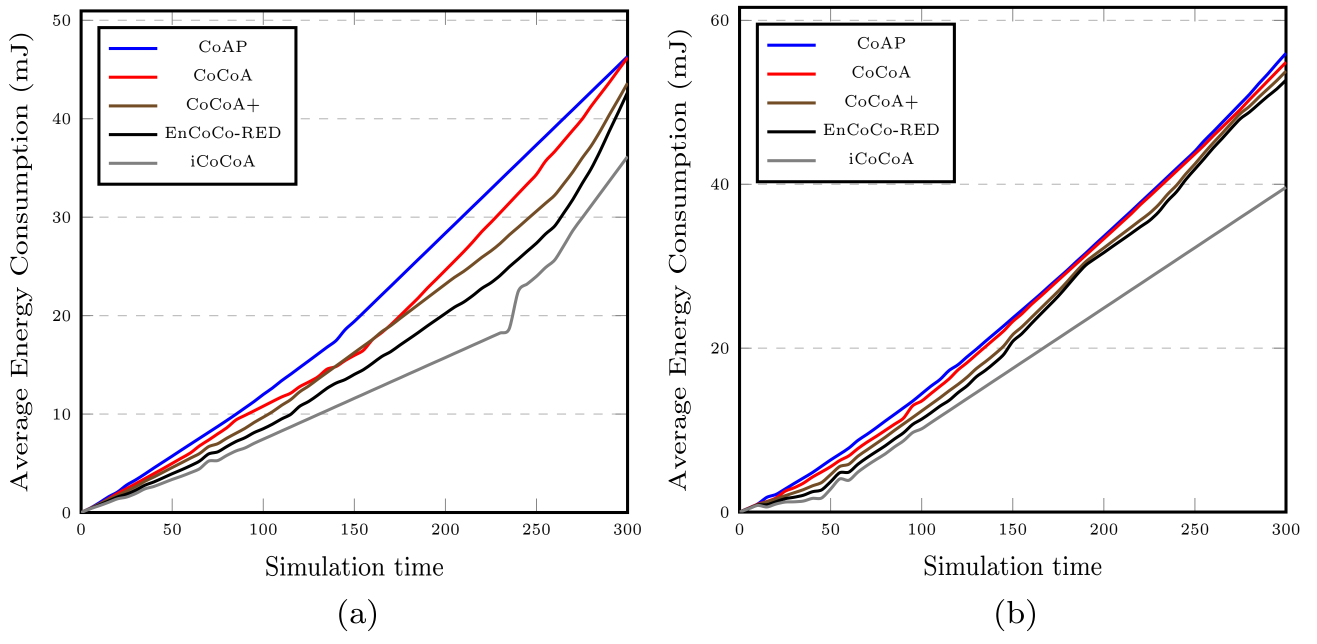

Fig. 11 Average Energy Consumption vs. Simulation Time **a** continuous **b** burst

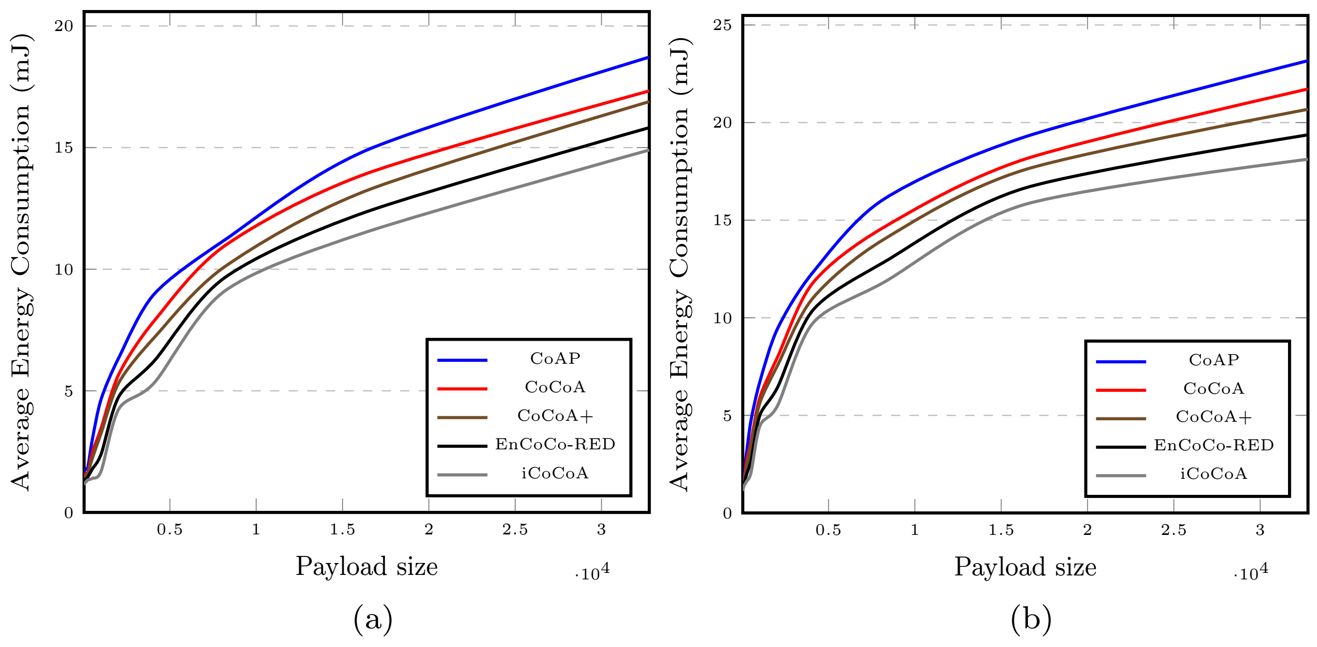

Fig. 12 Average Energy Consumption by varying the payload size **a** continuous **b** burst

respectively during the increasing of simulation time. Similarly, Fig. 11a shows the growth in the burst scenario approximately 9%, 7%, 4–6%, and 2–5% when compared with the CoAP, CoCoA, CoCoA+, and EnCoCo-RED, respectively. The average EC of the proposed and existing methods by varying the payload size is tested and plotted in Fig. 12 for both the scenarios. From Fig. 12a, we notice that the energy consumption of the proposed work is less compared to the CoAP, CoCoA, CoCoA+, and EnCoCo-RED, approximately 15%, 11% and 9%, and 8%, respectively. Similarly, in the burst scenario, the performance improvement of the proposed iCoCoA is approximately 11% than CoAP, 8% than CoCoA, 7% than CoCoA+, and 4–7% than EnCoCo-RED protocols. The EC of the proposed method outperformed because of eliminating the unnecessary frequent retransmission in the network through RTO's proper estimation. iCoCoA also considers the packet drop scenario for heavy traffic when there is no possibility of retransmission within the deadline.

5.7 Fairness index of energy consumption

The fairness index of EC (F_e) determines the equal share of bottleneck of EC among all the devices in the network. The F_e computation is similar as shown in section 5.5. The F_e of the given network's EC is shown in Eq. (35), which is similar when compared with Eq. (33) but varies in terms of input data as shown below:

$$F_e = 1 - \left(\frac{2 \times S_e}{H_e - L_e} \right) \tag{35}$$

where S_e indicates the SD of the average EC of the motes computed using Eq. (36), $L_e = \min\{S_e\}$ and $H_e = \max\{S_e\}$ in the network.

$$S_e = \frac{1}{n} \sqrt{\sum_{i=1}^{n} \left(E_i - E\right)^2} \tag{36}$$

We examine the F_e of the existing and proposed algorithms in continuous and burst scenarios. The F_e of CoAP, CoCoA, CoCoA+, EnCoCo-RED and iCoCoA for the continuous scenario is approximately 0.5309, 0.5521, 0.593, 0.6011, and 0.6209, respectively. The F_e of CoAP, CoCoA, CoCoA+, EnCoCo-RED and iCoCoA for the burst scenario is approximately 0.4389, 0.4379, 0.4433, 0.4612 and 0.5426, respectively. The F_e of the iCoCoA is always higher when compared with the existing approaches. The higher F_e indicates the equal share of the bottleneck for the motes EC in the network. The proposed iCoCoA achieves the best F_e over existing methods due to avoiding unnecessary packet retransmissions.

5.8 Discussion

While, the proposed iCoCoA predicts and controls the congestion efficiently, some of the pitfalls are still possible in iCoCoA. Here, we listed some of the limitations and possible alternate solutions to overcome them. The agent is trained after a few iterations of the network, so it is not possible to avoid the congestion in the initial stages of the simulation. However, it does not affect the system because the traffic is not so high up to few iterations. Due to this, the possibility of congestion occurrence is very low. If the unanticipated congestion appears in the network, iCoCoA selects a random action to generate RTO values for controlling it. Another limitation is the memory requirements to store the experiences in constrained IoT devices for training. This limitation is overcome by limiting a set of few most recent experiences instead of maintaining all the outcomes of the agent.

The time taken to train and provide the results by the agent is overcome by limiting the amount of the training dataset. Further, we can still reduce the computational load by using frame-skipping method used in Mnih et al. (2015) for Atari game. Furthermore, even the changes in the network conditions are dynamic, but for some of the cases, these changes are not effected on the network features. With these features, the agent provides an action based on past experiences (available in ERB) without further training. The required computational resources for the agent also depend on the number of CONV or Pooling layers used for the agent. Deciding the number of layers in the agent for achieving the best result is a challenging task. However, in Lippmann (1987), the author proved that the two hidden layers are sufficient for the efficient classification. So, the proposed method also uses only two CONV layer to limit the computational overhead. So, we invent an efficient congestion prediction techniques within the protocol.

6 Conclusion

IoT is connected with a large number of devices, and these are exchanging their data continuously among them by interacting with the environment. These data transmissions slowly increase the traffic and lead to congestion in the network. It causes unnecessary retransmissions, thus it degrades the performance of the IoT such as throughput, PDR, energy consumption, packet loss etc. CoAP is an application layer protocol, which use to control the congestion in a static environment, but it is not complete relief from the congestion. In this article, we proposed an intelligent congestion control algorithm named iCoCoA for CoAP using deep reinforcement learning approach to predict and control the congestion in the dynamic environments. This iCoCoA extracts the various network features and produces RTO values dynamically using DRL agent to avoid unnecessary frequent retransmissions. It also confirms the possibility of retransmission of a packet within certain amount of time or it will drop the packet. The performance of the iCoCoA is substantiated using Contiki v3.0 with cooja simulator in continuous and burst environments, and it outperforms the existing CoAP, CoCoA, and CoCoA+ algorithms. As a further work, area of research in CoAP is to and avoid the congestion before it occurs in the network so that it completely avoids the retransmissions by choosing alternative decisions.

Acknowledgements This work was supported by the Archimedes Foundation under the Dora plus Grant 11-15/OO/11476 and SERB, India, through grant CRG/2021/003888. We also thank financial support to UoH-IoE by MHRD, India (F11/9/2019-U3(A)).

Data availability statement All data generated or analysed during this study are generated randomly during the simulation. The details about data generation is included in this article.

Declarations

Conflict of interest There are no potential conflicts of interest.

References

Agyemang JO, Kponyo JJ, Gadze JD, Nunoo-Mensah H, Yu D (2022) Lightweight messaging protocol for internet of things devices. Technologies 10(1):21

Aimtongkham P, Horkaew P, So-In C (2021) An enhanced CoAP scheme using fuzzy logic with adaptive timeout for IoT congestion control. IEEE Access 9:58967–58981

Akpakwu GA, Hancke GP, Abu-Mahfouz AM (2020) CACC: context-aware congestion control approach for lightweight CoAP/UDP-based internet of things traffic. Trans Emerg Telecommun Technol 31(2):e3822

Betzler A, Gomez C, Demirkol I, Paradells J (2013) Congestion control in reliable CoAP communication. In: Proceedings of the 16th ACM International Conference on Modeling, analysis & simulation of wireless and mobile systems. ACM, pp 365–372

Betzler A, Gomez C, Demirkol I, Paradells J (2015) CoCoA+: an advanced congestion control mechanism for CoAP. Ad Hoc Netw 33:126–139

Betzler A, Gomez C, Demirkol I, Paradells J (2016a) CoAP congestion control for the Internet of Things. IEEE Commun Mag 54(7):154–160

Betzler A, Isern J, Gomez C, Demirkol I, Paradells J (2016b) Experimental evaluation of congestion control for CoAP communications without end-to-end reliability. Ad Hoc Nets 52:183–194

Bolettieri S, Tanganelli G, Vallati C, Mingozzi E (2018) pCoCoA: a precise congestion control algorithm for CoAP. Ad Hoc Netw 80:116–129

Bormann C, Castellani AP, Shelby Z (2012) CoAP: an application protocol for billions of tiny internet nodes. IEEE Internet Comput 2:62–67

Demir AK, Abut F (2020) mlCoCoA: a machine learning-based congestion control for CoAP. Turk J Electr Eng Comput sci 28(5):1–20

Donta PK, Amgoth T, Annavarapu CSR (2020) Congestion-aware data acquisition with q-learning for wireless sensor networks. In: 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). IEEE, pp 1–6

Donta PK, Amgoth T, Annavarapu CSR (2021) An extended aco-based mobile sink path determination in wireless sensor networks. J Ambient Intell Humaniz Comput 12(10):8991–9006

Donta PK, Srirama SN, Amgoth T, Annavarapu CSR (2022) Survey on recent advances in iot application layer protocols and machine learning scope for research directions. Digit Commun Netw 8(5):727–744

HoBfeld T, Skorin-Kapov L, Heegaard PE, Varela M (2017) Definition of QoE fairness in shared systems. IEEE Commun Lett 21(1):184–187

Jamshed MA, Ali K, Abbasi QH, Imran MA, Ur-Rehman M (2022) Challenges, applications and future of wireless sensors in internet of things: a review. IEEE Sens J 22(6):5482–5494

Jay N, Rotman N, Godfrey B, Schapira M, Tamar A (2019) A deep reinforcement learning perspective on internet congestion control. In: International Conference on machine learning, pp 3050–3059

Kaur N, Sood SK (2017) An energy-efficient architecture for the internet of things. IEEE Syst J 11(2):796–805

Kim M, Lee S, Khan MTR, Seo J, Bae Y, Jeong Y, Kim D (2019) A new CoAP congestion control scheme using message loss feedback for IoUT. In: Proceedings of the 34th ACM/SIGAPP Symposium on applied computing. SAC '19. ACM, pp 2385–2390

Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90

Lippmann R (1987) An introduction to computing with neural nets. IEEE ASSP Mag 4(2):4–22

Mahajan N, Chauhan A, Kumar H, Kaushal S, Sangaiah AK (2022) A deep learning approach to detection and mitigation of distributed denial of service attacks in high availability intelligent transport systems. Mob Netw Appl 20:1–21

Martinez B, Monton M, Vilajosana I, Prades JD (2015) The power of models: Modeling power consumption for IoT devices. IEEE Sens J 15(10):5777–5789

Mišić J, Ali MZ, Mišić VB (2018) Architecture for IoT domain with CoAP observe feature. IEEE Internet Things J 5(2):1196–1205

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529

Nie X, Zhao Y, Li Z, Chen G, Sui K, Zhang J, Ye Z, Pei D (2019) Dynamic TCP initial windows and congestion control schemes through reinforcement learning. IEEE J Sel Areas Commun 37(6):1231–1247

Praveen Kumar D, Tarachand A, Rao ACS (2019) Machine learning algorithms for wireless sensor networks: a survey. Inf Fusion 49:1–25

Rathod V, Jeppu N, Sastry S, Singala S, Tahiliani MP (2019) CoCoA++: delay gradient based congestion control for Internet of Things. Future Gener Comput Syst 100:1053–1072

Salkuti SR (2018) Congestion management using optimal transmission switching. IEEE Syst J 12(4):3555–3564

Sandell M, Raza U (2019) Application layer coding for IoT: benefits, limitations, and implementation aspects. IEEE Syst J 13(1):554–561

Sangaiah AK, Ramamoorthi JS, Rodrigues JJ, Rahman MA, Muhammad G, Alrashoud M (2020) LACCVoV: linear adaptive congestion control with optimization of data dissemination model in vehicle-to-vehicle communication. IEEE Trans Intell Transp Syst 22(8):5319–5328

Sargent M, Allman M, Paxson V (2011) Computing TCP's retransmission timer. Computing

Sun X, Ansari N (2018) Traffic load balancing among brokers at the IoT application layer. IEEE Trans Netw Serv Manag 15(1):489–502

Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press

Suwannapong C, Khunboa C (2019) Congestion control in CoAP observe group communication. Sensors 19(15):3433

Suwannapong C, Khunboa C (2021) EnCoCo-RED: enhanced congestion control mechanism for CoAP observe group communication. Ad Hoc Netw 112:102377

Uroz D, Rodríguez RJ (2022) Characterization and evaluation of IoT protocols for data exfiltration. IEEE Internet of Things J 9(19):19062–19072

Xiao K, Mao S, Tugnait JK (2019) TCP-Drinc: smart congestion control based on deep reinforcement learning. IEEE Access 7:11892–11904

Yadav RK, Singh N, Piyush P (2020) Genetic CoCoA++: genetic algorithm based congestion control in CoAP. In: 2020 4th International Conference on intelligent computing and control systems (ICICCS). IEEE, pp 808–813

Zhang S, You X, Zhang P, Huang M, Li S (2022) A UCB-based dynamic CoAP mode selection algorithm in distribution IoT. Alex Eng J 61(1):719–727