A Hierarchical Congestion Control Method in Clustered Internet of Things

Sadaf Mokhtari
Islamic Azad University of Dezful
Hamid Barati (hbarati@iaud.ac.ir)
Islamic Azad University of Dezful https://orcid.org/0000-0003-0028-3568
Alli Barati
Islamic Azad University of Dezful

Research Article

Keywords: Internet of things (IoT), congestion control, congestion score (CS), Finite state machines, Buffer empty space (BES)

Posted Date: November 11th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-679887/v1

Version of Record: A version of this preprint was published at The Journal of Supercomputing on February 21st, 2022. See the published version at https://doi.org/10.1007/s11227-022-04340-7.

Noname manuscript No.
(will be inserted by the editor)

A Hierarchical Congestion Control Method in Clustered Internet of Things

Sadaf Mokhtari · Hamid Barati · Ali Barati

Received: date / Accepted: date

Abstract Internet of things (IoT) is a modern technology where data can be transmitted to any things (human, animal, or object) over communications networks, whether Internet or intranet. Congestion occurs when the input data rate to the node, higher than the output data rate of node. Congestion control in computer network modulates traffic entry into a network in order to avoid congestive. This paper suggests a method for congestion control in the internet of things in two phases. The first phase is intra-cluster congestion control, which uses two parameters congestion score (CS) and buffer empty space (BES) to congestion avoidance. In this phase based on these two parameters, 9 states are defined to determine the congestion status of each node, and based on these 9 states the appropriate decision is made to the node. The second phase is inter-clusters congestion control. In this phase, after determined cluster head priority, the parameters of back off timer (BFT), waiting time to receive acknowledgment ($WTTR_{ACK}$), sequence number (SEQ) and retransmission counter (RC) are used for congestion control. The proposed congestion control method is simulated by NS-2 software. A comparison between the performance of proposed method and conventional methods shows that applying proposed method results in a significant improvement in average congestion score (CS), packet lost rate, energy consumption and end-to-end delay.

S. Mokhtari
Department of Computer Engineering, Dezfoul Branch, Islamic Azad University, Dezfoul, Iran
E-mail: smokhtari@iaud.ac.ir

H. Barati
Department of Computer Engineering, Dezfoul Branch, Islamic Azad University, Dezfoul, Iran
E-mail: hbarati@iaud.ac.ir

A. Barati
Department of Computer Engineering, Dezfoul Branch, Islamic Azad University, Dezfoul, Iran
E-mail: abarati@iaud.ac.ir

Keywords Internet of things (IoT) · congestion control · congestion score (CS) · Finite state machines · Buffer empty space (BES)

1 Introduction

The Internet of Things is a concept that is difficult to give a precise definition of it. The common point of all definitions lies in the notion that in the Internet, data were generated by users, but in the Internet of Things, data is generated by things. The Internet of things described the universal of things in which everything is intelligently connected and communicating. In other words, with the Internet of things, the physical world becomes a huge information system [1], [2]. IoT is a modern technology where data can be transmitted to any things (human, animal, or object) over communications networks, whether Internet or intranet. The Internet of Things has many benefits such as automation communications, control of monitoring information, time savings, money savings, automation of daily tasks and better monitoring of devices [3], [4]. The IoT available services have increased its applications that can change the lifestyle as well as improve the quality of life. Applications of the Internet of things in various fields include smart home, smart city, health care, military, business and smart agriculture [5], [6]. Internet of Things has its own set of challenges, like any other technology, including security and privacy, standardization, over-reliance on technology, lack of jobs, the digital divide, environmental impacts, and congestion control [6].

Congestion control when the input data to a node exceeds its capacity. When sent a large number of packets to a part of the network, congestion occurs and its performance is reduces [7]. Congestion control in computer networks is meant to avoidance the data from being overloaded in the node and thus the Packets loss. The congestion in the network reduces overall network quality and increases packet lost [8]. The solutions offered to congestion avoidance are based on the data transmitter rate reduction. The idea behind congestion control mechanisms originates from the point of network bandwidth, node processing ability, server capacities, channel capacity, flow of the link, number and size of distinct flow and channel reliability [9]. Achieving high throughput in network bottlenecks, fair allocation in the shortest time, maintaining and increasing responsiveness, compatibility with older protocols widely used are the most important congestion control objectives [10]. Congestion control methods are divided into the following three categories. Based on the type and size of feedback received from the network, based on the ability to implement in the network, based on the type of improvement or dedicated performance [10], [11]. In general, there are two approaches of resource creation and demand reduction for congestion control. In the approach of creating new resources, adding new resources to control and congestion avoidance is suggested. In the demand reduction approach, the amount of resources available will be shared among service users, and by sending explicit or implicit messages to message

senders, the allowed rate for sending data is notified [11], [12].

In this paper proposed a congestion control method in the clustered Internet of things network . In the proposed method, the Internet of things network is considered clustered and done in two phase, intra-cluster and inter-clusters. In intra-cluster congestion control, for each node 9 states are defined based on two parameters of the node's congestion score (CS), and buffer empty space (BES). In this phase the appropriate decision for the node is made based on each of the occurrences state. In inter-clusters congestion control, we consider four parameters: waiting time to receive acknowledgment ($WTTR_{ACK}$), back off timer (BFT), sequence number (SEQ) and retransmission counter (RC). Waiting time to receive acknowledgment (WTT_{RACK}) depends on the number of hops each packet takes to reach its destination and the number of retransmission of clusters head's packets. The main purpose of estimating the waiting time to receive acknowledgment ($WTTR_{ACK}$) for congestion control is to reduce packet lost and packets delay. The proposed method defines a backoff timer (BFT) between consecutive transmissions. In fact, backoff timer is time interval between two transmissions that the sender node does not sending any packet during this interval. The sequence number (SEQ) depends on the cluster head's priority and the retransmission counter (RC) depends on the sequence number (SEQ). Using these parameters, we try to inter-clusters congestion avoidance. The innovations of the proposed method are:
• Use the combination of parameters congestion score (CS) and buffer empty space (BES) to determine the states of each node and appropriate decisions are made based on the states of node in the first Phase.
• In the second phase, Cluster heads prioritize based on their data type so that important data is more likely to reach its destination.
• Using a combination of parameters such as waiting time to receive acknowedgment ($WTTR_{ACK}$), back off timer (BFT), sequence number (SEQ) and retransmission counter (RC) to send data to each cluster head based on their priority in the second phase.

The rest of the paper is organized as follows: Section 2 represents an overview of the related works. In Section 3, the details of the proposed method, which is in two phase intra-cluster and inter-clusters to congestion control in the Internet of Things, are presented. Then we provide the simulation of the proposed protocol and the analysis of its performance in Section 4. Finally, Section 5 concludes this paper.

2 RELATED WORKS

TCP congestion control with MDP algorithm for internet of things over heterogeneous network provided by Toprasert and Lilakiataskun[3]. This method is designed to describe TCP congestion control algorithm and is called TCP Siam which increases congestion avoidance. Useful output, Useful round trip

time and use in heterogeneous networks are the advantages of this method. This model uses the Markov decision-making process to estimate the Round Trip Time (RTT). This model is sufficient to handle congestion window (cWnd) values in TCP and strong incremental mode. In this method, three phases of congestion control, Markov Decision Process Model and Friendliness and Fairness Compatibility are defined. Congestion control mode is a transfer mode from TCP connection. In the Markov decision-making phase, the MDP algorithm improves congestion status by calculating the probability of transfer status, reward and discount factor. In the third phase, TCP is designed to share bandwidth between streams and others using other congestion control TCP protocols.

Poddar et al. [7] presented a congestion control method for IoT using a channel trust based approach. This method uses the node congestion and reliability criteria to determine whether the channel is reliable for packet transfer from source to destination. A priority plan has been chosen to determine a faster response time in emergency cases. Each node has a number of priority queues of equal size for the two data types. Queue schedulers are provided to provide different traffic with different priorities than priority queues. The proposed method is divided into three stages: priority setting, node level congestion size, and channel reliability estimation. In step one, the node with the maximum number of links to neighboring nodes in the network is designated as monitoring node and assigns heterogeneous traffic data priorities. In the second step, the scheduling rate and the sending rate are used to determine the node surface congestion. In the third step, channel validation is used to decide whether the channel selected is safe to transfer packets securely to the next node.

Rathood et al. [10] presented a new congestion control algorithm called CoCoA++ to address the issue of network congestion in Internet of Things (IoT). The method uses delay gradient (DG) to obtain a better measurement of network congestion and implements a probable back-off to counter congestion. The delay gradients and the probability backoff factor are integrated with Constrained Application Protocol (CoAP). The main goal of the CoCoA++ algorithm is to provide better end-to-end network congestion estimation using the benefits of CDG. delay gradients give a more accurate measure of congestion and the Retransmission Time Out (RTO) is reduced significantly, thereby leading to less delays and high packet sending rates.

Ancillotti and Bruno [13] have compared COAP and COCOA+ congestion control mechanisms for different scenarios of IoT application. CoAP is a web transfer protocol that provides basic RESTful services for IoT devices. CoAP uses a reload time (RTO) technique to identify packet lost. In fact, when a confirmation message is sent, a timer is set for the RTO value, If the timer expires and the sender has not yet received confirmation from the destination, the lost message is assumed and the CoAP message is retransmitted. CoCoA+ is a simple CoAP_CC enhancement that makes CoAP congestion control more responsive to network conditions.

Hassan et al. [14] proposed the congestion control Method in CoAP. In this method, the proposed solution involves an adaptive mechanism for sending node data. This adaptive mechanism mainly depends on the traffic priority class and the loss rate. The proposed method has three stages: traffic priority allocation, adaptive RTO and adaptive pause timer. In the first stage, data traffic is divided into high priority traffic and low priority traffic. Second, the RTO value is calculated using two estimators. In the third step, the back off timer for default Coap increases mainly at the exponential level.

Bolettieri et al. [15] proposed precise congestion control algorithm for CoAP (PCoCoA). The proposed method guarantees precise COAP congestion control and delays comparable to COAP and COCOA+. This approach proposes two methods for congestion control: introducing a specific COAP option and modify the Message ID semantic. In the first case, a new CoAP option is called the Transmission Counter (TC). This option is used to link any ACK message to its corresponding transmitted CON message. An alternative solution is to avoid overloading the message ID. To support this approach, MID can be divided into a 14-bit ID and a 2-bit sequence number. The second element is used to reduce fake retransmission.

Hamimi et al.[16] proposed the congestion control mechanism for internet of things (IoT) paradigm. The purpose of this approach is to reduce the number of closed losses during transmission and minimize the number of visits to reduce the corresponding cost, energy and response time. The proposed method is divided into two phase: cluster head selection and package discarding. Hence, the presence of a cluster head reduces the loading in the sink node, while improving the performance of applications. In the second part of the cluster, the process of packet destruction is performed by each node. The purpose of the deletion process is to reduce the number of data packets that are sent to the next cluster head.

CoAP congestion control in the Internet of Things provided by Betzler et al. [17]. CoAP defines four types of messages: confirmable (CON), non-confirmable (NON), reset (RST), and acknowledgment (ACK) messages. CoCoA offers a congestion control solution that reduces the conservative message rate limitations of the basic CoAP specifications while ensuring that protocol operations are secure. CoCoA includes three main components: adaptive RTO calculation, a variable back off factor (VBF), and RTO aging.

Betzler et al. [18] proposed an advanced congestion control mechanism for coap (Cocoa+). CoAP with the main congestion control (CC) mechanism resolves this important issue. CoCoA+ includes the following suggestions to address when using CoCoA as a congestion control mechanism: 1) A modification of the weak estimator calculations to reduce the impact of $\mathrm{RTT}_{\mathrm{weak}}$ variations and its impact on RTO overall. 2) Replacement of BEB used for remobilization by a variable return factor (VBF). 3) New approach for large amounts of $\mathrm{RTO}_{\mathrm{overall\ age}}$.

Bhalerao et al. [19], proposed the improve congestion control in CoAP protocol. The CoCoA 4-State-Strong method is presented to analyze and improve congestion control in the CoAP. Depending on the number of times a

package is resent, each CoAP transaction is considered in one of four modes: 1, 2, 3, 4. The transaction starts in Mode 1 and each time a packet is resubmitted, its status increases by one. Each time a packet is successfully transmitted and verified within a specified time, its status decreases by one. This allows you to adjust the return parameters accordingly.
A summary and comparison of the mentioned algorithms presents in table 1.

3 The proposed method

When the number of sent packets exceeds the capacity of the network, congestion occurs and as a result the sent packets are either deleted or delayed. Congestion control in the internet of things, like any other subject, has its own objectives, such as increasing packet delivery rates, reducing energy consumption, fair allocation in the shortest possible time and most effective way, maintaining and enhancing accountability and reducing the time of responsiveness. In the proposed method for achieving these goals, it is assumed that the network is clustered and congestion control is performed in two phase intra-cluster and inter-clusters. In the first phase based on two parameters congestion score (CS) and buffer empty space (BES), the congestion state of each cluster member is specified and the appropriate transfer rate for each node is determined according to node congestion state. In the second phase, the inter-clusters congestion control is based on the cluster head priority. The phases of the proposed method are described at follows. The notations used in the proposed method are listed in table 2.

3.1 First phase: Intra-cluster congestion control

The main purpose of this phase is control and provides a solution to congestion avoidance in the cluster head. In this section, two parameters of congestion score (CS) and buffer empty space (BES) are used to control and provide a solution to congestion Avoidance. According to these parameters 9 states may occur for each cluster member node. Based on these 9 states, the appropriate decision for data sending rate for each cluster member's node is determined. According to the states that may occur for each node, the state space for the finite state machine is determined. The steps of the first phase are as follows.

Step 1: Calculate congestion score (CS)
CS determine the current congestion score in cluster member's node and calculated by Equation (1).

$$CS_{node\,i} = \left(1 - \frac{r^i_{for}}{r^i_{in}}\right) + \left(1 - \frac{N^i_{ACK}}{N^i_{source\_send}}\right) \tag{1}$$

$$r^i_{in} = R^i_s + R^i_{tr}, \tag{2}$$

Table 1 Summary and comparison of related works

References	Approach	Advantage	Disadvantage
Toprasert and Lilaki-ataskun[3]	TCP congestion control with MDP algorithm for IoT over heterogeneous network	Increase congestion avoidance, improved performance when packets are lost	Low performance in high window size and real time network
Poddar et al. [7]	Congestion control for IoT using channel trust based approach	Efficient congestion control and maintains an optimal data rate during packet transmission	Loss of any hop data would cause more wastage of network resources than node data
CoCoA++ [9]	Delay gradient based congestion control for internet of things	Determination accurate level of congestion, reduction RTO with minimal delays and high packet sending rates	The design of CDG is tailored to work alongside TCP
Ancillotti and Bruno [12]	Comparison of CoAP and CoCoA+ congestion control mechanisms for different IoT application scenarios	Using of stop and wait method in coap and cocoa	Cocoa poor performance in networks with burst traffic or small RTTs, unnecessary retransmission
Hassan et al. [13]	Adaptive congestion control mechanism in CoAP application protocol for internet of things (IoT)	Control of transmission of applied data effectively	Decreasing in data transmission performance
pCoCoA [14]	A precise congestion control algorithm for CoAP	pCoCoA reduces retransmissions and the ability to work in bursty traffic occurrence-based scenarios	Some of the fixed values used in this scheme not are suitable
PDNC [15]	Congestion control mechanism for internet of things (IOT) paradigm	Increase the quality of service, retransmission time decrease	The average queuing delay is high
CoAp [16]	CoAP congestion control for the internet of things (IOT)	Increase the utilization, proper retransmission	RTT estimates are quite unreliable
CoCoA+ [17]	An advanced congestion control mechanism for CoAP	Higher degree of reliability, lower delay, resilient against sudden changes of network traffic	Requires to confirmable transfer, exponential backoff
CoCoA [18]	An analysis and improvement of congestion control in the CoAP internet-of-things protocol	Analyzing and improve CoCoA constraints	More retransmissions

Table 2 Notations

Notation	Description
$CS_{node\,i}$	Congestion score of node i
$r^i{}_{for}$	Forwarding packet rate of node i
$r^i{}_{in}$	Input packets rate to node i
$N^i{}_{ACK}$	The number of acknowledgment received by node i
$N^i{}_{source\_send}$	The number of packets generated by the node i
$R^i{}_S$	The source traffic rate
$R^i{}_{tr}$	The transition traffic rates
$BES_{\text{node i}}$	Node's buffer empty space i
$N^i{}_Q$	The number of packets in the node's buffer i
$N^i{}_T$	The buffer size of node i
$T_{reference}$	Reference time
$WTTR_{ACK}$	Waiting time to receive acknowledgment
BFT	Back off timer
SEQ	Sequence number
$SEQ_{reference}$	Reference sequence number
RC	Retransmission counter

Where $r^i{}_{in}$, $r^i{}_{for}$, $N^i{}_{ACK}$, and $N^i{}_{source\_send}$ indicates the input packets rate to node i, the forwarding packets rate of node i, the number of acknowledgment received by node i, and the number of source sent packets (the number of packets generated by the node i itself), respectively. According to equation (2) $r^i{}_{in}$ of the sum of $R^i{}_s$ which is the source traffic rate (number of packets generated by node i) and $R^i{}_{tr}$ which is the transition traffic rate (number of packets that node i receives from other nodes) is obtained.
According to equation (3), the congestion score in each node has three levels.

$$CS_{node\,i} = \begin{cases} congestion\ dosen't\ happen & if\ 0 \leq \mathrm{CS}_{node\,i} < 0.5 \\ on\ the\ threshold\ of\ congestion & if\ 0.5 \leq \mathrm{CS}_{node\,i} < 1 \\ congestion\ happens & if\ 1 \leq CS_{node\,i} \leq 2 \end{cases} \tag{3}$$

$CS_{node\,i}$, is never negative because it is never $N^i{}_{ACK} > N^i{}_{source\_send}$ and $r^i{}_{for} > r^i{}_{in}$. The conditions under which the node congestion scores may be in one of the three above levels are described below.
• When CS is between 0 and 1, node has not congestion, But it is divided into two levels for better understanding. If CS is between 0 to 0.5, the node has not congestion and if CS is 0.5 to 1, the node is on the threshold of congestion.

1. Conditions where the node has not congestion. ($0 \leq CS_{node\,i} < 0.5$)

a. If number of forwarding packets by the node is equal to the number of input packets to the node ($r^i{}_{for}$==$r^i{}_{in}$) and the number of acknowledgment messages received by the node is equal to the number of source sent packets ($N^i{}_{ACK}$==$N^i{}_{source\_send}$) the congestion score is zero ($CS_{node\,i}$==0) and the node has not congestion.

b. If the number of acknowledgment messages received by the node is more than half the number of source sent packets ($N^i{}_{ACK} > \frac{1}{2}\ N^i{}_{source\_send}$) and the number of forwarding packets by the node with the number of input packets to the node, be equal ($r^i{}_{for} == r^i{}_{in}$) the congestion score is less than 0.5. ($CS_{node\,i} < 0.5$)

c. If the number of forwarding packets by the node is more than half the number of input packets to the node ($r^i{}_{for} > \frac{1}{2}\ r^i{}_{in}$) and the number of acknowledgment messages received by the node is equal to number of source sent packets ($N^i{}_{ACK} == N^i{}_{source\_send}$) the congestion score is less than 0.5. ($CS_{node\,i} < 0.5$)

2. Conditions where the node is on the threshold of congestion. ($0.5 \leq CS_{node\,i} < 1$)

a. If the number of acknowledgment messages received by the node is equal to half the number of source sent packets ($N^i{}_{ACK} == \frac{1}{2}\ N^i{}_{source\_send}$) and the number of forwarding packets by the node with the number of input packets to the node be equal ($r^i{}_{for} == r^i{}_{in}$) the congestion score is 0.5. ($CS_{node\,i} == 0.5$)

b. If the number of forwarding packets by the node is equal to half the number of input packets to the node ($r^i{}_{for} == \frac{1}{2}\ r^i{}_{in}$) and the number of acknowledgment messages received by the node is equal to the number of source sent packets ($N^i{}_{ACK} == N^i{}_{source\_send}$) the congestion score is 0.5. ($CS_{node\,i} == 0.5$)

c. If the number of forwarding packets by the node is less than half the number of input packets to the node ($r^i{}_{for} < \frac{1}{2}\ r^i{}_{in}$) and the number of acknowledgment messages received by the node is equal to the number of source sent packets ($N^i{}_{ACK} == N^i{}_{source\_send}$) the congestion score is more than 0.5. ($0.5 < CS_{node\,i} < 1$)

d. If the number of acknowledgment messages received by the node is less than half the number of source sent packets ($N^i{}_{ACK} < \frac{1}{2}\ N^i{}_{source\_send}$) and the number of forwarding packets by the node with the number of input packets to the node be equal ($r^i{}_{for} == r^i{}_{in}$) the congestion score is more than 0.5. ($0.5 < CS_{node\,i} < 1$)

• Conditions where the node has congestion. ($1 \leq CS_{node\,i} \leq 2$)

1. If the number of forwarding packets by the node is zero ($r^i{}_{for} == 0$) and the number of acknowledgment messages received by the node is equal to the number of source sent packets ($N^i{}_{ACK} == N^i{}_{source\_send}$) the congestion score is one. ($CS_{node\,i} == 1$)

2. If the number of acknowledgment messages received by the node is zero ($N^i{}_{ACK} == 0$) and the number of forwarding packets by the node with the number of input packets to the node be equal ($r^i{}_{for} == r^i{}_{in}$) the congestion

score is one. ($CS_{node\,i}$==1)

3. If the number of acknowledgment messages received by the node is half the number of source sent packets ($N^i{}_{ACK}$==$\frac{1}{2}$ $N^i{}_{source\_send}$) and the number of forwarding packets by the node with half the number of input packets to the node be equal ($r^i{}_{for}$==$\frac{1}{2}$ $r^i{}_{in}$) the congestion score is one. ($CS_{node\,i}$==1)

4. If the number of acknowledgment messages received by the node is less than half the number of source sent packets ($N^i{}_{ACK} < \frac{1}{2}$ $N^i{}_{source\_send}$) and the number of forwarding packets by the node is zero ($r^i{}_{for}$==0) the congestion score is more than one. ($1 < CS_{node\,i} < 2$)

5. If the number of acknowledgment messages received by the node is half the number of source sent packets ($N^i{}_{ACK}$==$\frac{1}{2}$ $N^i{}_{source\_send}$) and the number of forwarding packets by the node is zero ($r^i{}_{for}$==0) the congestion score is more than one. ($1 < CS_{node\,i} < 2$)

6. If the number of acknowledgment messages received by the node is more than half the number of source sent packets ($N^i{}_{ACK} > \frac{1}{2}$ $N^i{}_{source\_send}$) and the number of forwarding packets by the node is zero ($r^i{}_{for}$==0) the congestion score is more than one. ($1 < CS_{node\,i} < 2$).

7. If the number of forwarding packets by the node is less than half the number of input packets to the node ($r^i{}_{for} < \frac{1}{2}$ $r^i{}_{in}$) and the number of acknowledgment messages received by the node is zero ($N^i{}_{ACK}$==0) the congestion score is more than one. ($1 < CS_{node\,i} < 2$)

8. If the number of forwarding packets by the node is half the number of input packets to the node ($r^i{}_{for}$==$\frac{1}{2}$ $r^i{}_{in}$) and the number of acknowledgment messages received by the node is zero ($N^i{}_{ACK}$=0) the congestion score is more than one. ($1 < CS_{node\,i} < 2$)

9. If the number of forwarding packets by the node is more than half the number of input packets to the node ($r^i{}_{for} > \frac{1}{2}$ $r^i{}_{in}$) and the number of acknowledgment messages received by the node is zero ($N^i{}_{ACK}$==0) the congestion score is more than one. ($1 < CS_{node\,i} < 2$)

10. If the number of forwarding packets by the node is zero ($r^i{}_{for}$==0) and the number of acknowledgment messages received by the node is zero ($N^i{}_{ACK}$=0) the congestion score is two. ($CS_{node\,i}$==2)

In summary, the conditions described for the node's congestion are shown in table 3. Fig. 1 shows the congestion score in the node.

Step 2: Calculate the Buffer Empty Space (BES)

Table 3 Different conditions of node congestion

Node's congestion interval or level		N^i_{for}/N^i_{in}	$N^i_{ACK}/N^i_{send\_source}$	$CS_{node\ i}$
$(0 \leq CS_{node\ i} < 1)$ The node is without congestion and is divided into two levels for better understanding	The node is without congestion (NWC) $(0 \leq CS_{node\ i} < 0.5)$	$(r^i_{for} == r^i_{in})$	$(N^i_{ACK} == N^i_{source\_send})$	0
		$(r^i_{for} == r^i_{in})$	$(N^i_{ACK} > \frac{1}{2}N^i_{source\_send})$	$CS_{node\ i} < 0.5$
		$(r^i_{for} > \frac{1}{2}r^i_{in})$	$(N^i_{ACK} == N^i_{source\_send})$	
	The node is on the threshold of congestion (NTC) $(0.5 \leq CS_{node\ i} < 1)$	$(r^i_{for} == r^i_{in})$	$(N^i_{ACK} == \frac{1}{2}N^i_{source\_send})$	$CS_{node\ i} == 0.5$
		$(r^i_{for} == \frac{1}{2}r^i_{in})$	$(N^i_{ACK} == N^i_{source\_send})$	
		$(r^i_{for} == r^i_{in})$	$(N^i_{ACK} < \frac{1}{2}N^i_{source\_send})$	$CS_{node\ i} > 0.5$
		$(r^i_{for} < \frac{1}{2}r^i_{in})$	$(N^i_{ACK} == N^i_{source\_send})$	
The node have congestion $(1 \leq CS_{node\ i} \leq 2)$		$(r^i_{for} == 0)$	$(N^i_{ACK} == N^i_{source\_send})$	$CS_{node\ i} == 1$
		$(r^i_{for} == r^i_{in})$	$(N^i_{ACK} == 0)$	
		$(r^i_{for} == \frac{1}{2}r^i_{in})$	$(N^i_{ACK} == \frac{1}{2}N^i_{source\_send})$	
		$(r^i_{for} == 0)$	$(N^i_{ACK} < \frac{1}{2}N^i_{source\_send})$	$CS_{node\ i} > 1$
			$(N^i_{ACK} == \frac{1}{2}N^i_{source\_send})$	
			$(N^i_{ACK} > \frac{1}{2}N^i_{source\_send})$	
		$(r^i_{for} < \frac{1}{2}r^i_{in})$	$(N^i_{ACK} == 0)$	
		$(r^i_{for} == \frac{1}{2}r^i_{in})$		
		$(r^i_{for} > \frac{1}{2}r^i_{in})$		
		$(r^i_{for} == 0)$	$(N_{ACK}i == 0)$	$CS_{node\ i} == 2$

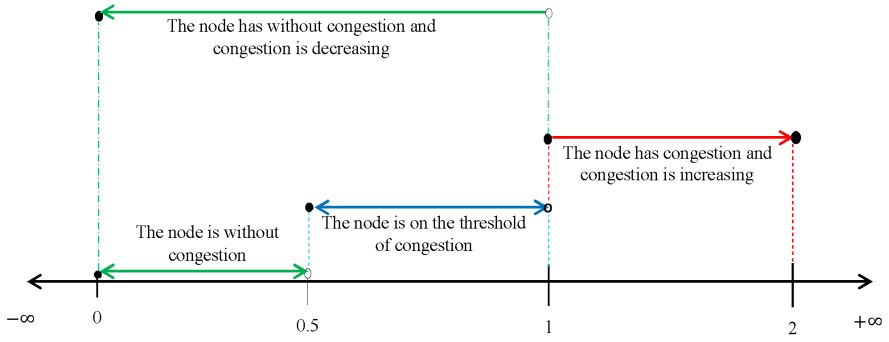

Fig. 1 Diagram of the congestion score in a node

If two nodes a and b are neighbors and node a intends send a packet to node b, node b must have sufficient buffer space to store packets from node a, BES is the buffer empty space in the node that calculated by equation (4).

$$BES_{node\ i} = \left(1 - \frac{N^i_Q}{N^i_T}\right) \times 100 \tag{4}$$

Where $N^i{}_Q$ and $N^i{}_T$ indicates the number of packets in the node's buffer i and the buffer size in node i.

• If $N^i{}_Q$==$N^i{}_T$, the buffer is completely full and $BES_{node\ i}$ is zero percent and this is the worst case for a node's buffer.

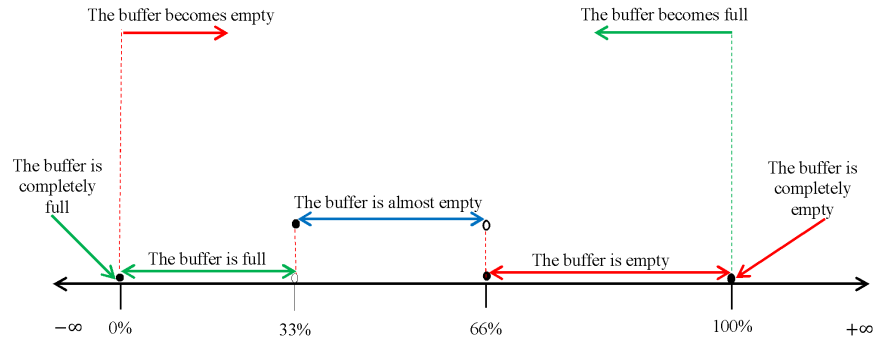

Fig. 2 Diagram of the buffer empty space in a node

• If $N^i{}_Q$==0, the buffer is completely empty and $BES_{node\,i}$ is %100.
• When $N^i{}_Q < N^i{}_T$ and $0 < BES_{node\,i}$<100, according to equation (5), the buffer empty space in each node is divided into three levels.

Fig. 2 shows conditions of buffer empty space in each node.

$$BES_{node\,i} = \begin{cases} buffer\ is\ empty & if\ \%66 \leq BES_{node\,i} \leq \%100 \\ buffer\ is\ almost\ empty & if\ \%33 \leq BES_{node\,i} < \%66 \\ buffer\ is\ full & if\ \%0 \leq BES_{node\,i} < \%33 \end{cases} \quad (5)$$

Step 3: Determine of the states of each node

After calculating the congestion score (CS) and the buffer empty space (BES), based on CS and BES nine states are define for each node. When each of the states occurs, the node sends a notification to the cluster head to announce its status. According to the received notification, cluster head warns node's neighbors that decrease or increase the data transmission rate so that congestion avoidance in the node. This method leads to control and congestion avoidance in the cluster member nodes. The nine states for each node are shown in table 4. The command that the cluster head announce based on received notification message to node's neighbors is shown in table 5.

• The worst state that can happen to a node is when it has congestion and the buffer is full, in this case it sends a notification message to the cluster head and then cluster head announces to node's neighbors to sharp decrease transmission rate to this node.
• If the node has congestion and the buffer is empty or node has congestion and the buffer is almost empty or node is on the threshold of congestion and the buffer is full, node sends a notification message to its cluster head requesting a reduction in transmission rates, and then cluster head announces to node's neighbors to decrease the transmission rate to this node.
• If the Node is on the threshold of congestion and the buffer is empty or node is on the threshold of congestion and the buffer is almost empty or node has

Table 4 Congestion states that may occur for each node

Buffer Empty Space (BES)	Congestion Score (CS)
Buffer is empty (BE)	Without congestion
Buffer is almost empty (BAE)	Without congestion
Buffer is full (BF)	Without congestion
Buffer is empty (BE)	On the threshold of congestion
Buffer is almost empty (BAE)	On the threshold of congestion
Buffer is full (BF)	On the threshold of congestion
Buffer is empty (BE)	Congested
Buffer is almost empty (BAE)	Congested
Buffer is full (BF)	Congested

Table 5 Cluster head's command to neighboring nodes

BES \ CS	Congested	On the threshold of congestion	Without congestion
Buffer is empty (BE)	Transmission rate ecrease (TRD)	Remains at the current state	Sharp increase in transmission rates (SITR)
Buffer is almost empty (BAE)	Transmission rate ecrease (TRD)	Remains at the current state	Transmission rate increase (TRI)
Buffer is full (BF)	Sharp decrease in transmission rates (SDTR)	Transmission rate ecrease (TRD)	Remains at the current state

not congestion and the buffer is full, its status will remain unchanged and does not send the notification message to its cluster head.
• If the node has not congestion and the buffer is almost empty, it sends a notification message to its cluster head and the cluster head announces to node's neighbors they can increase the transmission rate.
• If the node has not congestion and the buffer is empty, it sends a notification message to its cluster head and the cluster head announces to node's neighbors they can sharp increase transmission rate.

Table 6 shows the transition conditions of different states of the finite state machine. Fig. 3 shows the finite state machine for nine states that may occur for a node. Fig. 4 shows Overall flowchart of the first phase of the proposed method.

3.2 Second phase: Inter-clusters congestion control

The main purpose of this phase is to congestion control inter the cluster heads. After collecting and sending data by the cluster members to the cluster head, data is send inter-cluster heads to reach the destination. Due to factors

Table 6 finite state machine transition conditions

The node has congestion (NC), Buffer is empty (BE)	$P_1 \begin{cases} 1 \leq CS_{node\,i} \leq 2 \\ 66 \leq BES_{node\,i} \leq 100 \end{cases}$
The node has congestion (NC), Buffer is almost empty (BAE)	$P_2 \begin{cases} 1 \leq CS_{node\,i} \leq 2 \\ 33 \leq BES_{node\,i} < 66 \end{cases}$
The node has congestion (NC), Buffer is full (BF)	$P_3 \begin{cases} 1 \leq CS_{node\,i} \leq 2 \\ 0 \leq BES_{node\,i} < 33 \end{cases}$
The node is on the threshold of congestion (NTC), Buffer is empty (BE)	$P_4 \begin{cases} 0.5 \leq CS_{node\,i} < 1 \\ 66 \leq BES_{node\,i} \leq 100 \end{cases}$
The node is on the threshold of congestion (NTC), Buffer is almost empty (BAE)	$P_5 \begin{cases} 0.5 \leq CS_{node\,i} < 1 \\ 33 \leq BES_{node\,i} < 66 \end{cases}$
The node is on the threshold of congestion (NTC), Buffer is full (BF)	$P_6 \begin{cases} 0.5 \leq CS_{node\,i} < 1 \\ 0 \leq BES_{node\,i} < 33 \end{cases}$
The node has not congestion (NNC), Buffer is empty (BE)	$P_7 \begin{cases} 0 \leq CS_{node\,i} < 0.5 \\ 66 \leq BES_{node\,i} \leq 100 \end{cases}$
The node has not congestion (NNC), Buffer is almost empty (BAE)	$P_8 \begin{cases} 0 \leq CS_{node\,i} < 0.5 \\ 33 \leq BES_{node\,i} < 66 \end{cases}$
The node has not congestion (NNC), Buffer is full (BF)	$P_9 \begin{cases} 0 \leq CS_{node\,i} < 0.5 \\ 0 \leq BES_{node\,i} < 33 \end{cases}$

such as excess transmission rate, inaccurate routing, inadequate bandwidth and low queue lengths can occur congestion. In this phase, controlling and congestion avoidance inter-cluster heads based on cluster heads priority, calculating waiting time to receive acknowledgment ($WTTR_{ACK}$), back off timer (BFT), sequence number (SEQ) and retransmission counter (RC) for data sent to cluster head is done. The steps of the second phase are as follows.

Step 1: Cluster head priority

In this step, the cluster heads are divided into five categories and each is given priority (P) according to the type and importance of their services as shown in table 7. High priority is given to cluster heads that have critical applications and services. Devices such as heart beats sensor, blood pressure sensor, body temperature sensor, monitoring sensor and fire alarming sensor have critical applications and services. There are also cluster heads that their data have less important such as refrigerators, smart TVs, washing machines,

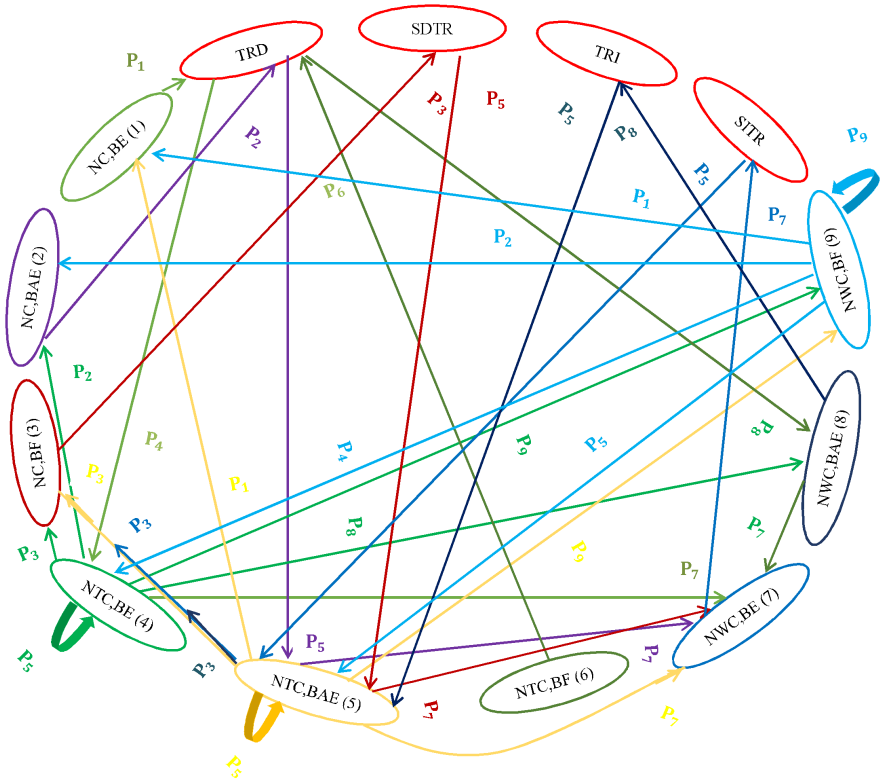

Fig. 3 Finite state machine for each node in the first phase

Table 7 The cluster head priority for sending data

Priority (P)	Cluster head
5	Cluster head with too high priority for sending data
4	Cluster head with High priority for sending data
3	Cluster head with normal priority for sending data
2	Cluster head with low priority for sending data
1	Cluster head with too low priority for sending data

and room humidity sensors.

According to cluster head's priorities, a back off timer (BFT) is set for them. The cluster head with higher priority has a shorter back off timer (BFT), because its data is critical and should arrive sooner than the cluster head with lower priority to destination.

Step 2: Send data to the destination by the cluster heads

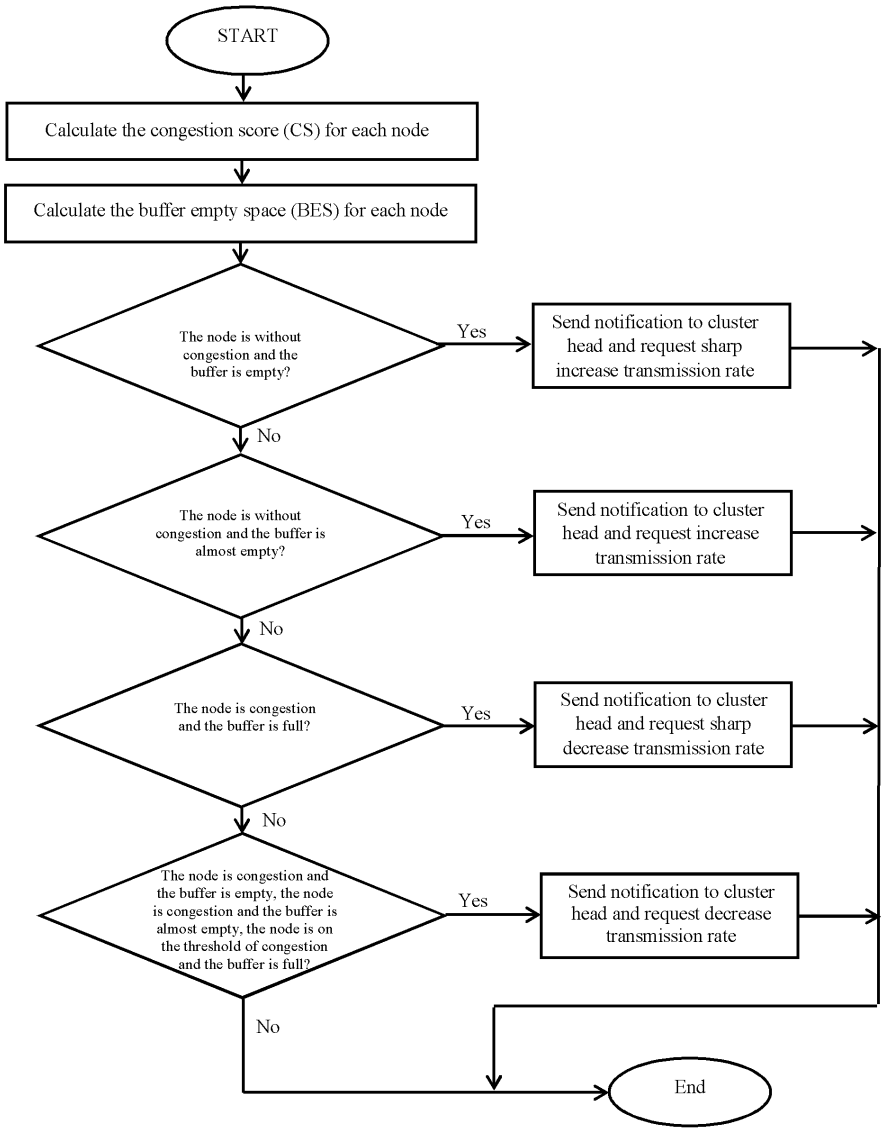

Fig. 4 The flowchart of intra-cluster congestion control

After determining cluster head priority, data should be sent to the destination by cluster heads. The cluster head with highest priority starts sending packets and wait for $T_{reference}$ in first transmission. (RC=0, $WTTR_{ACK}$=$T_{reference}$); If the acknowledgment message (ACK) is received from the destination and the cluster head has other packet to send, it will start sending them. Otherwise if it did not receive the acknowledgment message of the sent packet, calculates the waiting time to receive acknowledgment ($WTTR_{ACK}$), back off timer (BFT),

sequence number (SEQ) and retransmission counter (RC) for cluster head and resend the packet. The cluster head waits for $WTTR_{ACK}$ to receive ACK. In each resend, if the ACK is received and the cluster head has other packets to send, it will be sending them. But if the ACK is not received in any of the re-sending and the number of specified re-sending expires, that packet will be deleted. This process continues until all data is sent to the destination. Then the other cluster heads sending their data to the destination in order of priority.

Step 3: Calculate the SEQ and RC

To prevent extra overhead in data transfer and congestion control, a parameter called sequence number (SEQ) is used to send data based on the cluster head priority. The cluster head's data with higher priority is assigned sequence number with more bits number and the cluster head's data with lower priority is assigned sequence number with less bits number. Therefore, the number of retransmission for cluster head with higher priority is more than the cluster heads with lower priority. The sequence number (SEQ) and the retransmission counter (RC) are calculated by equation (6).

$$\begin{cases} SEQ = \lceil \log_2^P \rceil * SEQ_{reference} \\ SEQ_{reference} = 2bit \\ RC \leq 2^{SEQ} \end{cases} \tag{6}$$

Step 4: Calculate the back off timer (BFT)

The back off timer (BFT) is the time interval between the two transmissions. in other words cluster head wait in a time interval after each sending data to destination and does not send any data. After the back off timer (BFT), it will start sending data to the destination again. The back off timer (BFT) for each cluster head based on the cluster head priority and calculated by equation (7).

$$BFT = \left(\frac{1}{P} * T_{reference} \right) \tag{7}$$

Cluster head with higher priority will have less back off timer (BFT), because their data is of more importance and must reach its destination earlier. The back off timer (BFT) for cluster head with lower priority is higher, because their data is less important. Priority and back off timer (BFT) are inversely related.

Step 5: Calculate the waiting time to receive acknowledgment ($WTTR_{ACK}$)

The waiting time to receive acknowledgment ($WTTR_{ACK}$) is the time interval that the node waits to receive acknowledgment (ACK) that it sends from the destination. At this step, waiting time to receive acknowledgment ($WTTR_{ACK}$) based on the number of hops that packet takes to reach its destination and the number of retransmission of cluster head's packet. The waiting time to receive acknowledgment is calculated by equation (8). The higher the number of retransmission and the number of hop the longer the waiting time to receiving ACK ($WTTR_{ACK}$). According to equation (6), as

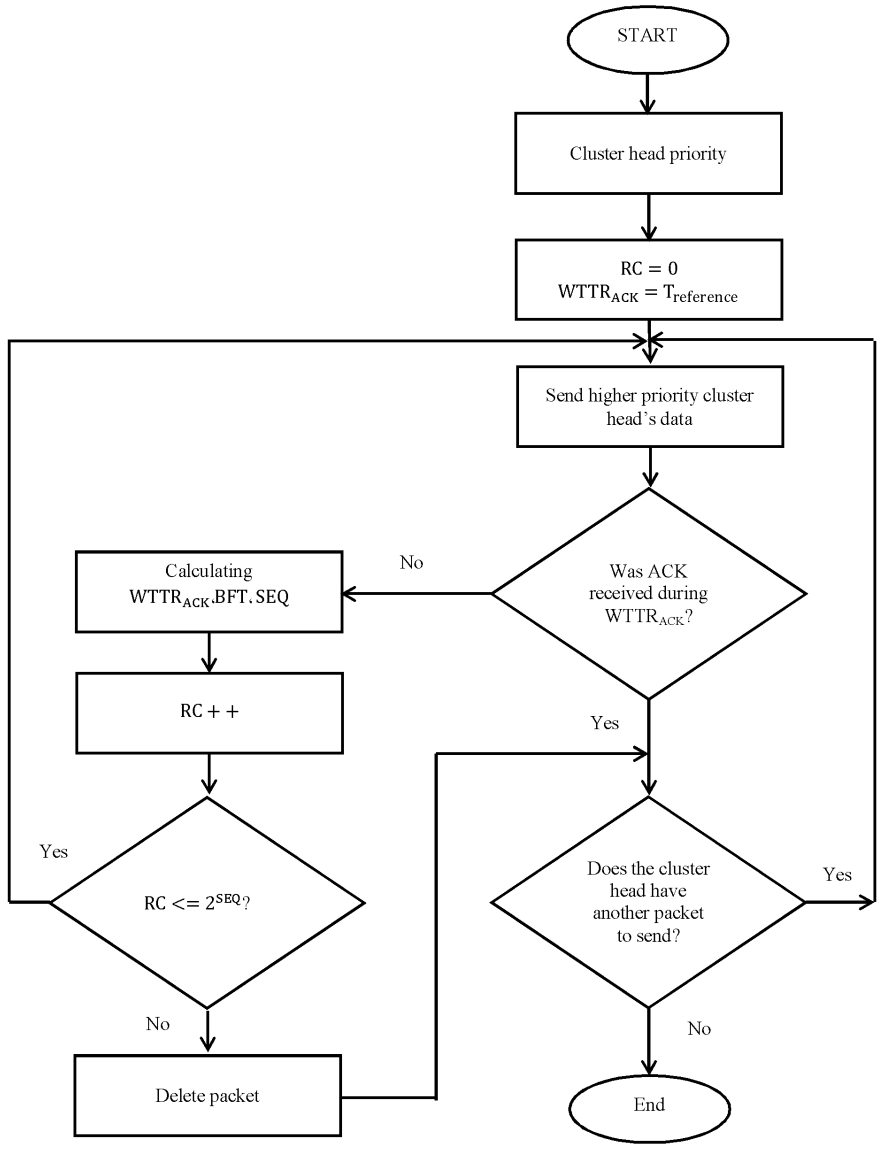

Fig. 5 The flowchart of inter-clusters congestion control

the number of retransmission is higher for high priority cluster heads, so $WTTR_{ACK}$ is higher for high priority cluster heads.

$$WTTR_{ACK} = (HOP_{count} * T_{reference}) * \lceil \log_2^{RC} \rceil \tag{8}$$

Fig. 5 shows Overall flowchart of the second phase of the proposed method.

Table 8 Simulation Parameters

Value	Parameter
$400 \times 400m^2$	Network's size
200	Number of network nodes
2 J	Initial Energy
Mac/802-15.4	Mac Layer Protocol
Random	Node distribution model
512 byte	Packets size
600 s	Simulation time

In the second phase, the cluster heads are prioritized according to their data type into five categories, and the cluster head's with highest priority starts sending its packets to the destination. In the first send if the acknowledgment message is received from the destination and the cluster head has other packet to send, it will sending them. But if it did not receive the acknowledgment message of the transmitted packet, it calculates waiting time to receive acknowledgment ($WTTR_{ACK}$), back off timer (BFT), sequence number (SEQ) and retransmission counter (RC) , and the cluster head re-sends the same packet in the $WTTR_{ACK}$ time interval. If cluster head receives acknowledgment message in one of re-sending and the cluster head has other packet to send, it will be send. But if in none of the re-sending does not receive the acknowledgment message from the destination, it deletes the packet. If cluster head has other packet, it will sends all its data to the destination. Then cluster heads send its data based on priority.

4 Simulation and result

The proposed method is simulated by network simulator NS2. In order to evaluate the performance of the proposed method in the internet of things (IoT) network, the results of the simulation are compared with two methods Poddar et al. [7] and CoCoA [18]. The simulation parameters are shown in table 8. The results of the simulation are investigated in terms of parameters such as congestion score (CS), packet lost rate, energy consumption and end-to-end delay, which are described below.

Network congestion occurs due to an increase in the rate of input data to the node relative to the rate of output data of the node. Network's congestion reduces quality of service (QOS). Congestion increases the delay and packet lost. Fig. 6 shows the network's congestion score (CS) in the proposed method compared to other methods. As can be seen in the fig. 6, the proposed method imposes less congestion on the network than other methods. In the proposed method to intra-cluster congestion control uses the node's congestion score (CS) parameter, which is a hybrid parameter that is calculated based on the number of received packets, the number of sent packets as well as the number of received $ACKs$. Also another effective parameter used in the intra-cluster communication and has a great impact on congestion control is the buffer empty

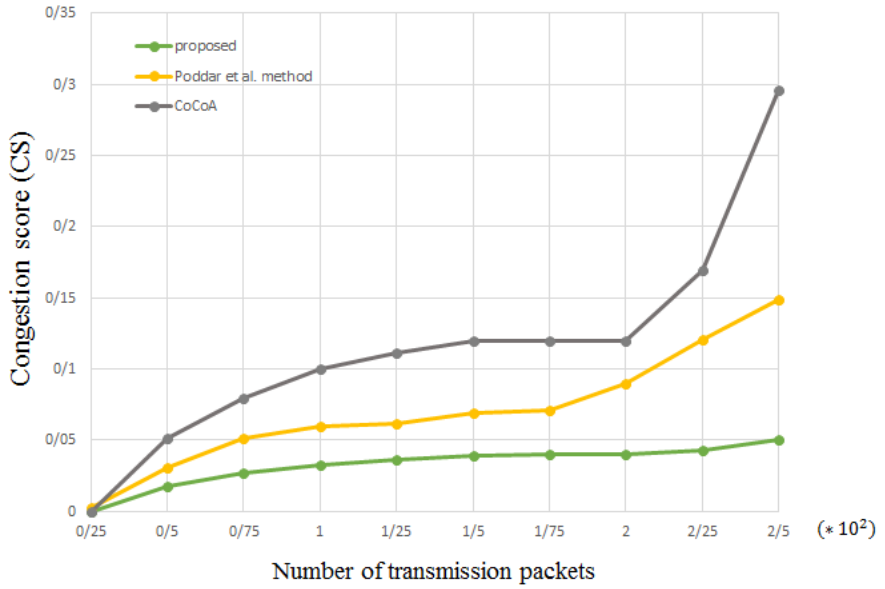

Fig. 6 Congestion score (CS)

space (BES) of each node. Considering the two parameters, the congestion score (CS) and the buffer empty space (BES) for the nodes different states defined and based on the defined state, the appropriate decision is made to congestion avoidance in the network. In the second phase, the number of packets retransmission is according to their priority, so that more important data is sent first. As shown in the fig. 6, in the proposed method network's congestion score (CS) is lower than other methods, and the congestion score (CS) in the proposed method has decreased with increasing number of sent packets.

Packet lost rate shows the ratio of the number of packets lost to the total number of sent packets. Network's congestion is one of the factors preventing packets to reaching their destination. The proposed method tries to minimize the number of packets lost by avoiding congestion. Packet lost occurs when one or more sent packet through the network's nodes cannot reach their destination. The packet lost rate in the proposed method compared to the other methods is shown in fig. 7. As shown in fig. 7, the proposed method prevents packet lost and thus keeps the packet lost rate at a lower level than the other methods. In addition, considering the buffer empty space (BES) parameter to sending packets in first phase cause reduces the number of packets lost due to the buffer being full. The proposed method also considers the back off timer (BFT) and the number of appropriate retransmission for packets of higher importance in second phase. As a result, the proposed method performs better in controlling packets lost and reducing packet lost rate.

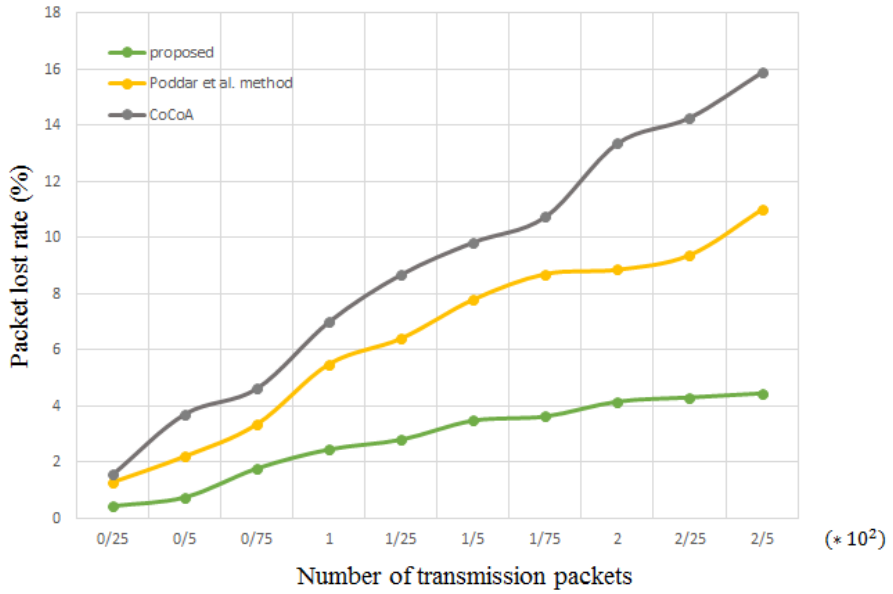

Fig. 7 Packet lost rate

Fig. 8 shows the energy consumption of the proposed method compared to other methods. Increasing the number of packets in the network increases the probability of congestion. In case of congestion, the probability of lost of packet is increased, which causes the packet lost to be retransmission from the source, thus increasing the energy consumption. In order to reduce packet lost, the proposed method is performed at two phase intra-cluster and inter-clusters. Intra-cluster congestion control is using appropriate mechanisms, investigating node conditions and making appropriate decisions according to node conditions. In order to inter-clusters congestion control, appropriate mechanism is provided for the data retransmission according to cluster head priority. Therefore, it is suitable to avoid congestion and increase energy consumption between cluster member nodes as well as cluster heads. As shown in the fig. 8, the energy consumption in the network increases as the packets increase and increasing the congestion. However, due to congestion control and the reduction of packet lost rates in the proposed method, as well as the determination of the number of times appropriate to packets retransmission, the energy consumption in the proposed method is lower.

End-to-end delay is the time required between the sent packets from the source node to the destination node. Congestion and delay are directly related and congestion increase results in more delay, Because congestion on the one hand causes delay in data transmission and on the other hand causes packets lost and data retransmission in the network. Retransmission causes delays on the network. Fig. 9 shows the end-to-end delay of the proposed method compared to other methods. The proposed method minimizes packet

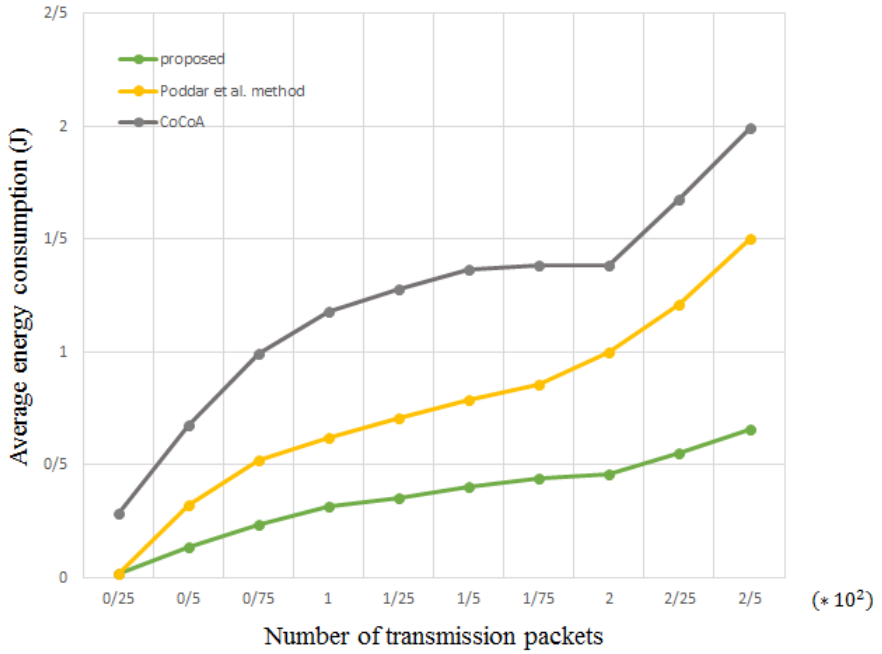

Fig. 8 Average energy consumption

lost as much as possible and reduces the number of packets retransmission in the network. In the inter-clusters phase, data with higher priority is send faster and more frequently. Therefore, the delay in sending these packets is also reduced. The proposed method is also suitable for real-time applications because important data is transmission faster and, if lost, retransmission with more iterations. In general, the proposed method imposes less delay on the network than other methods.

5 Conclusion

In this paper proposed a congestion control method on the clustered Internet of things. The proposed method consists of two phases. The first phase is intra-cluster congestion control and uses two parameters of the congestion score (CS) and buffer empty space (BES). The second phase is inter-clusters congestion control. In this phase, several parameters are used to congestion control, such as back off timer (BFT), Waiting time to receive acknowledgment ($WTTR_{ACK}$), sequence number (SEQ), and retransmission counter (RC). Unlike most existing methods that focus on only one criterion; this method focuses on several criteria and examines them. Simulations show that the proposed method performs better in terms of criteria such as congestion score (CS), packet lost rate, or energy consumption and end-to-end delay. The simulation

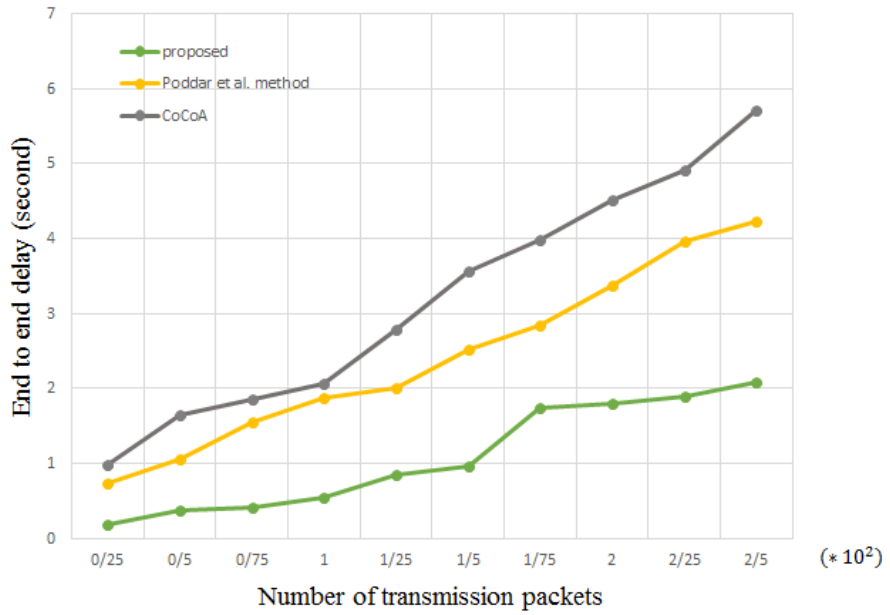

Fig. 9 End to end delay

results show, proposed method has less congestion than the other methods.

References

1. Anani, W., Ouda, A., & Hamou, A. (2019, May). A Survey Of Wireless Communications for IoT Echo-Systems. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)* (pp. 1-6). IEEE.
2. Din, I. U., Guizani, M., Kim, B. S., Hassan, S., & Khan, M. K. (2018). Trust management techniques for the Internet of Things: A survey. *IEEE Access, 7*, 29763-29787.
3. Sabry, S. S., Qarabash, N. A., & Obaid, H. S. (2019, March). The Road to the Internet of Things: a Survey. In *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)* (pp. 290-296). IEEE.
4. Zhang, L., Liang, Y. C., & Xiao, M. (2018). Spectrum sharing for Internet of Things: A survey. *IEEE Wireless Communications, 26*(3), 132-139.
5. Li, S., Peng, S., Liao, X., Zhu, P., & Peng, Y. (2007, May). A Framework for Congestion Control for Reliable Data Delivery in Wireless Sensor Networks. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management* (pp. 793-796). IEEE.
6. Mishra, N., Verma, L. P., Srivastava, P. K., & Gupta, A. (2018). An analysis of IoT congestion control policies. *Procedia computer science, 132*, 444-450.
7. Poddar, M., Chaki, R., & Pal, D. (2018, September). Congestion Control for IoT Using Channel Trust Based Approach. In *IFIP International Conference on Computer Information Systems and Industrial Management* (pp. 392-404). Springer, Cham.
8. Naghibi, M., & Barati, H. (2020). EGRPM: Energy efficient geographic routing protocol based on mobile sink in wireless sensor networks. *Sustainable Computing: Informatics and Systems,* 25, 100377.
9. Qazi, I. A., & Znati, T. (2011). On the design of load factor based congestion control protocols for next-generation networks. *Computer networks, 55*(1), 45-60.

10. Rathod, V., Jeppu, N., Sastry, S., Singala, S., & Tahiliani, M. P. (2019). CoCoA++: Delay gradient based congestion control for Internet of Things. *Future Generation Computer Systems, 100*, 1053-1072.
11. Kim, J. E., Boulos, G., Yackovich, J., Barth, T., Beckel, C., & Mosse, D. (2012, June). Seamless integration of heterogeneous devices and access control in smart homes. In *2012 Eighth International Conference on Intelligent Environments* (pp. 206-213). IEEE.
12. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems, 29*(7), 1645-1660.
13. Ancillotti, E., & Bruno, R. (2017, July). Comparison of CoAP and CoCoA+ congestion control mechanisms for different IoT application scenarios. In *2017 IEEE Symposium on Computers and Communications (ISCC)* (pp. 1186-1192). IEEE.
14. Hassan, R., Jubair, A. M., Azmi, K., & Bakar, A. (2016, December). Adaptive congestion control mechanism in CoAP application protocol for internet of things (IoT). In *2016 International Conference on Signal Processing and Communication (ICSC)* (pp. 121-125). IEEE.
15. Bolettieri, S., Tanganelli, G., Vallati, C., & Mingozzi, E. (2018). pCoCoA: A precise congestion control algorithm for CoAP. *Ad Hoc Networks, 80*, 116-129.
16. Halim, N. H. B., Yaakob, N. B., & Isa, A. B. A. M. (2016, August). Congestion control mechanism for Internet-of-Things (IOT) paradigm. In *2016 3rd International Conference on Electronic Design (ICED)* (pp. 337-341). IEEE.
17. Betzler, A., Gomez, C., Demirkol, I., & Paradells, J. (2016). CoAP congestion control for the internet of things. *IEEE Communications Magazine, 54*(7), 154-160.
18. Betzler, A., Gomez, C., Demirkol, I., & Paradells, J. (2015). CoCoA+: An advanced congestion control mechanism for CoAP. *Ad Hoc Networks, 33*, 126-139.
19. Bhalerao, R., Subramanian, S. S., & Pasquale, J. (2016, January). An analysis and improvement of congestion control in the CoAP Internet-of-Things protocol. In *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (pp. 889-894). IEEE.
20. Ashton, K. (2009). That 'internet of things' thing. *RFID journal, 22*(7), 97-114.
21. Afanasyev, A., Tilley, N., Reiher, P., & Kleinrock, L. (2010). Host-to-host congestion control for TCP. *IEEE Communications surveys & tutorials, 12*(3), 304-342.
22. Brakmo, L. S., O'Malley, S. W., & Peterson, L. L. (1994, October). TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of the conference on Communications architectures, protocols and applications* (pp. 24-35).
23. Baronti, P., Pillai, P., Chook, V. W., Chessa, S., Gotta, A., & Hu, Y. F. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards *Computer communications, 30*(7), 1655-1695.
24. Chaudhary, S., Johari, R., Bhatia, R., Gupta, K., & Bhatnagar, A. (2019, April). Craiot: Concept, review and application (s) of iot. In *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)* (pp. 1-4). IEEE.
25. Gonsai, D. A., Goswami, B., & Kar, U. (2013). Evolution of Congestion Control Mechanisms for TCP and Non TCP Protocols. *Matrix Academic International Online Journal Of Engineering And Technology*.
26. Hatamian, M., & Barati, H. (2015, July). Priority-based congestion control mechanism for wireless sensor networks using fuzzy logic. In *2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-5). IEEE.
27. Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., & Sikdar, B. (2019). A survey on IoT security: application areas, security threats, and solution architectures. *IEEE Access, 7*, 82721-82743.
28. Lee, J. J., Kim, K. T., & Youn, H. Y. (2016). Enhancement of congestion control of constrained application protocol/congestion control/advanced for Internet of Things environment. *International Journal of Distributed Sensor Networks, 12*(11), 1550147716676274.
29. Mishra, N., Verma, L. P., Srivastava, P. K., & Gupta, A. (2018). An analysis of IoT congestion control policies. *Procedia computer science, 132*, 444-450.
30. Mohammadian, H. D. (2019, April). IoT–a Solution for Educational Management Challenges. In *2019 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1400-1406). IEEE.

31. Sinche, S., Raposo, D., Armando, N., Rodrigues, A., Boavida, F., Pereira, V., & Silva, J. S. (2019). A Survey of IoT Management Protocols and Frameworks. *IEEE Communications Surveys & Tutorials*.
32. Toprasert, T., & Lilakiataskun, W. (2017, September). TCP congestion control with MDP algorithm for IoT over heterogeneous network. In *2017 17th International Symposium on Communications and Information Technologies (ISCIT)* (pp. 1-5). IEEE.
33. Whitmore, A., Agarwal, A., & Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. *Information Systems Frontiers, 17*(2), 261-274.
34. Wang, J., Wen, J., Zhang, J., Xiong, Z., & Han, Y. (2016). TCP-FIT: An improved TCP algorithm for heterogeneous networks. *Journal of Network and Computer Applications, 71*, 167-180.