IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 December 2022, accepted 14 January 2023, date of publication 23 January 2023, date of current version 31 January 2023.
Digital Object Identifier 10.1109/ACCESS.2023.3239302

RESEARCH ARTICLE

Adaptive Congestion Control Mechanism to Enhance TCP Performance in Cooperative IoV

TAPAS KUMAR MISHRA[1], KSHIRA SAGAR SAHOO[1,2], (Member, IEEE), MUHAMMAD BILAL[3], (Senior Member, IEEE), SAYED CHHATTAN SHAH[4], (Senior Member, IEEE), AND MANAS KUMAR MISHRA[5]
[1]Department of Computer Science Engineering, SRM University, Amaravati, Andhra Pradesh 522240, India
[2]Department of Computing Science, Umeå University, 901 87 Umeå, Sweden
[3]Department of Computer Engineering, Hankuk University of Foreign Studies, Yongin-si 17035, South Korea
[4]Department of Information and Communication Engineering, Hankuk University of Foreign Studies, Yongin-si 17035, South Korea
[5]Department of Computer Science, F. M. Autonomous College, Balasore, Odisha 756001, India

Corresponding authors: Muhammad Bilal (mbilal@hufs.ac.kr) and Sayed Chhattan Shah (shah@hufs.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant through the Ministry of Science and ICT Korea under Grant 2021R1F1A1045933, and in part by the Hankuk University of Foreign Studies Research Funds 2022.

ABSTRACT One of the main causes of energy consumption in Internet of Vehicles (IoV) networks is an ill-designed network congestion control protocol, which results in numerous packet drops, lower throughput, and increased packet retransmissions. In IoV network, the objective to increase network throughput can be achieved by minimizing packets re-transmission and optimizing bandwidth utilization. It has been observed that the congestion control mechanism (i.e., the congestion window) can plays a vital role in mitigating the aforementioned challenges. Thus, this paper present a cross-layer technique to controlling congestion in an IoV network based on throughput and buffer use. In the proposed approach, the receiver appends two bits in the acknowledgment (ACK) packet that describes the status of the buffer space and link utilization. The sender then uses this information to monitor congestion and limit the transmission of packets from the sender. The proposed model has been experimented extensively and the results demonstrate a significantly higher network performance percentage in terms of buffer utilization, link utilization, throughput, and packet loss.

INDEX TERMS IoT, IoV, congestion control, energy efficiency, AIMD, TCP, flow control.

I. INTRODUCTION

In, recent years, the Internet of Things (IoT) has been utilized to develop a wide range of applications across various industries. These applications include healthcare systems, agricultural systems, industrial applications, transportation systems, and more. The IoT technology enables the integration of physical devices, vehicles, buildings and other items—embedded with electronics, software, sensors and network connectivity—to enable these objects to collect and exchange data. The communication systems that depends upon battery power needs special focus on energy consumption. In IoT environment, the minimization of energy consumption pertaining required throughput of the communication system is utmost important. In both wired and wireless network system, there are various reasons for dropping a packet in the intermediate node of a connection that includes; congestion of the network, unreliable link connectivity, synchronization of the packet injection to the network by the sender, etc. The use of ad-hoc technology and the inception of smart-city networks using IoT intensify the complexity of the congestion control algorithms [1], [2], [3], [4], [5], [6], [7]. Further, internet connectivity to the vehicles which is called as **I**nternet **o**f **V**echicles (**IoV**) used to monitor the vehicle traffic, parking facility, and many more generate huge amount of data. But, the smart city network in association with IoV network can not neglect the data generated by vehicles as it contains very sensitive information which helps to reduce traffic jam in the road [8]. The congestion is simply defined as a higher injection of data or control packets to an intermediate node

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

than the forwarding packets by the node in a particular period. The stop and wait protocol, when deployed in the network, it does not lead to congestion as it takes care of one packet at a time. Thus, it gives poor performance when the transmission delay of a link is high as most of the time the entire link would be transmitting only one packet. This leads to the under-utilization of the bandwidth of the link. Hence, congestion control algorithms are deployed on the sender side to send more than one packet at a time [9] so that maximum bandwidth can be utilized. This is supported by sliding window protocols like Selective Repeat ARQ and Go-Back-N ARQ. Further, many modifications have been proposed to maintain the dynamic length of the window by different congestion control mechanisms like modified additive increase multiplicative decrease (mAIMD) [10], Transmission Control Protocol (TCP) mSACK [11], CUBIC [12]. In addition to this, some improved protocols are also developed for IoT [13], [14]. But, these approaches create congestion on the network due to a higher rate of injection of data packets to the network. Hence, it is very much essential to send more than one packet at a time to achieve higher throughput but at the same time it also important to monitor the status of the buffer at the receiver's node. Thus, it needs a mechanism that should consider both buffer status and link utilization of the intermediate node during any modification on the congestion window of the sender.

Further, it is clear that among various challenges of communication, congestion is the major one that affects the network performance [15]. In the field of smart-city, which is an advancement of ad-hoc and sensor networks, packet drop due to two fundamental causes. 1) congestion at the intermediate node, and 2) due to interference (i.e., link collision). TCP/IP protocol suite is the most used protocol in communication. However, it has some limitations in that it follows the layering structure of the OSI model strictly. Each layer is assigned a fixed task to control the communication. Moreover, it is experimented with and suggested by Fu et al. [15], Foukalas et al. [16] that throughput can be maximized by interacting variables of other layers which are beyond the limitations of the OSI model. The Cross-layer approach is the concept that enables one layer to share its variables with its non-adjacent layers. Many research articles including [17], [18] proved that the protocols have been improved the performance and achieve efficient resource allocation by using the cross-layer approach. Thus, the cross-layer optimization motivates the researchers for more attraction towards the improvement of the quality of service. It allows non-adjacent layers to access information of variables of other layers keeping the consistency of the protocol. There are many cross-layer approaches (like discussed in [18] and [19]) available in literature where they have designed and improved the quality of service of the communication system. Further, some researchers suggested to handle congestion control using hardware developments [20] and congestion control using cross layer approach [21].

Sharing information between two non-adjacent layers is implemented in two ways, 1) manager method and 2) non-manager method [15]. In the manager method, there exists a common virtual space to share information between layers. However, in the non-manager method, the two non-adjacent layers share information directly, without any intermediate controller. Here, we are interested to create a new interface that implements upward information flow. In this way, we allow the data link layer to provide its information to the transport layer on request to increase the throughput of communication which can be easily encoded in various models of TCP as described in [22], [23], and [24].

As congestion causes a great loss to the network in terms of throughput and energy consumption, researchers prefer to focus on the prevention of congestion. There are four major mechanisms to prevent network congestion as follows.

i **Network Scheduling**: It is the part of the procedure inside a node that is engaged to monitor the receiving and dispatching packets throughout the communication. Also, it takes the full responsibility for active queue management based on the requirement, i.e., ordering of packets, packet drop mechanisms, maintaining the priority of packets at the change of priority dynamically, etc.

ii **Detection of Congestion**: This is a very important mechanism to control congestion. The congestion control will be more effective if this can be detected very earlier. There exist several mechanisms towards detecting the mechanisms such as queue length-based congestion detection, throughput-based congestion detection, round trip time-based congestion detection.

iii **Congestion Notification**: It is the second step of congestion control mechanism. After congestion detection, it is the most important issue where we notify the congestion to the sender node. Two major techniques called implicit and explicit are well-known techniques for congestion notification. The implicit technique can reduce the overhead for data communication but sometimes explicit mechanism performs better than compared to the implicit mechanism.

iv **Congestion Avoidance Algorithm**: After detecting congestion at any node, the congestion avoidance algorithm runs at the source node. It mainly focuses on rate adjustment of the flow for a temporary solution at the transport layer. However, sometimes the source can divert packets to another direction if possible by keeping the same flow rate, which is maintained by modifying routing algorithms. Still, it can not get a better result as repeated congestion affect a lot towards the network throughput.

Still, several avoidance algorithms for congestion are followed at the sender side after detecting congestion. Here, we have discussed two specific approaches for congestion control, i.e., A) end-to-end congestion control, B) network-assisted congestion control.

End-to-end congestion control: In this approach, the network layer does not provide any explicit support to the transport layer. Further, the presence of congestion in the network are observed by the transport layer of the source based on network behavior; such as loss of packets, packet arrival delay, jitter, etc. The TCP end-to-end congestion control mechanism has three major components.

1) **Initial rate and adjustment**: When the TCP connection begins, the value of the congestion window is usually assigned to 1. As the available bandwidth to the connection may be much larger than Maximum Segment Size (MSS) for each Round Trip Time (RTT). The TCP sender continues to increase its sending rate exponentially until a threshold value and then it increases additionally. After any packet loss notification from the receiver, the sender node runs the flow rate control mechanism.
2) **Flow rate control mechanism**: Additive Increase and Multiplicative Decrease (AIMD) is the most used rate control approach [3]. This is a combined approach of two parts of an algorithm, i.e., 1) increasing flow rate (Additive Increase) to achieve better throughput and 2) decreasing flow rate (Multiplicative Decrease) at detection of congestion. Additive increase manages the congestion window in such a way that the efficiency of the network will be maximized. Moreover, it attempts to enhance its throughput by increasing its congestion window size by a fixed additional factor. This part of the algorithm runs until there is no congestion notification instruction received at the sender side. Further, the multiplicative decrease approach reduces the congestion window when congestion notification arrives. This approach decreases the current value of the congestion window by multiplying the fixed decrease parameter after a loss event. As an example, it may take two-third of the current congestion window if the decrease parameter is 0.67. However, the congestion window reduces to two-third of the current size always but never allowed to drop below 1 after each packet loss.
3) **Reaction to timeout events**: After a timeout event, the TCP sender goes for a slow start phase, i.e., it sets the congestion window to 1 MSS (Maximum Segment Size) and grows exponentially until it reaches the fixed point. Let one half of the value it had before the timeout event. At that point, it grows linearly, as in the case of the packet drop phenomenon. TCP manages these challenges by maintaining a system variable called threshold (*ssthress*), which determines the window size at which the slow start will finish and congestion avoidance will begin.

Contributions: The contributions of this work are as follows.

1) The root cause of congestion in real-time systems is identified along with its behavior and variation of the congestion window.
2) A modified architecture is proposed to monitor link utilization and buffer utilization at intermediate using the cross-layer approach.
3) Algorithms for both sender and intermediate nodes are proposed to act proactively to avoid congestion.
4) The proposed model is simulated extensively with different values of buffer and link threshold parameters. It is compared with the existing models (mSACK and mAIMD).
5) Further, the model is analyzed and a report has been presented with the justification of link and buffer threshold supporting the proposed model.

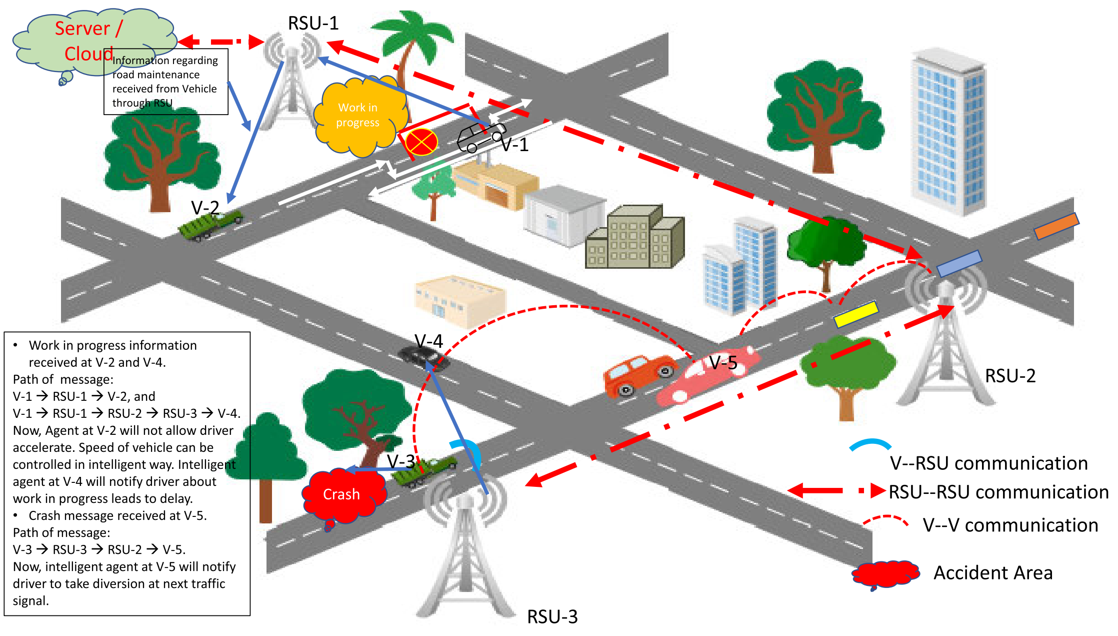
The rest of the paper is organized as follows. In section II, some related mechanisms are outlined. In section III the problem statement is defined explicitly. The proposed approach is discussed in section IV. In section V, the simulation results are presented. Finally, it is concluded with some future directions in section VI.

II. RELATED WORKS

Recently many models are proposed to update the congestion window. TCP-CUBIC [12] is the latest congestion control algorithm which is adopted by the browsers in MacOS in 2014 and windows in 2017. The model claims that it updates congestion window based on real time. In this model, it takes little bit time to reach the peak of the congestion window but it can stay more time in platue region. However, the RFC of the CUBIC model shows that it can not perform well in scenarios where end to end connection are short. It can perform well when round trip time (RTT) is high as it updates congestion window based on real time not on RTT based time. Thus, in the scenarios like smart cities where connections are monitored by the capacity of wide area networks (WAN) this CUBIC model of increasing window size will be not effective.

In [25], the authors proposed (TCP-WBQ) another active queue management based congestion control approach which identifies the congestion based packet drop and random packet drop in the network. It maintains a backlog queue in the intermediate router for each TCP and based on the length of the backlog queue with respect to the outgoing packets, it updates congestion window. It follows adoptive decrease and multiplicative increase of congestion window for efficient use of shared capacity of the link. However, it will not work for multipath TCP streamings with multiple TCP streams as backlog queue will not be able to differentiate such a dynamic environement with huge number of connections.

The authors in [26], proposed a multipath TCP that updates congestion window based on the feedback from the learning agent. Here, they proposed a Random Forest Regressing (RFR) method to predict throughput that maintains a tradeoff among loss rate, latency and sending rate. They propose an offline training at the TCP source that learns from the past history of the data packets that are sent from the source. However, we believe it is truely difficult to alarm the source node when there is a scenario that leads to congestion in the intermediate node. Thus, it will not be able to differentiate the random packet loss and congestion driven packet loss.



Information regarding road maintenance received from Vehicle through RSU

- Work in progress information received at V-2 and V-4.

Path of message:
V-1 → RSU-1 → V-2, and
V-1 → RSU-1 → RSU-2 → RSU-3 → V-4.
Now, Agent at V-2 will not allow driver accelerate. Speed of vehicle can be controlled in intelligent way. Intelligent agent at V-4 will notify driver about work in progress leads to delay.

- Crash message received at V-5.

Path of message:
V-3 → RSU-3 → RSU-2 → V-5.
Now, intelligent agent at V-5 will notify driver to take diversion at next traffic signal.

FIGURE 1. Communication among vehicle and road side units (Access points).

In EC-Elastic [27], the authors have proposed a varying parameter based TCP which changes congestion window by multiplication factor of parameters β_1 and β_2 which ranges from 0.5 to 0.9. However, it is not clear about the value of the parameters in different scenarios.

The TCP provides end-to-end service through a reliable connection over the Internet. Basically, it includes two primary phases: 1) slow start, and 2) congestion avoidance. Enhanced recovery from sporadic errors is provided by fast retransmission and fast recovery. Many congestion control protocols have been proposed earlier to handle congestion. In TCP, the AIMD flow control mechanism is used to handle packet loss because of actual network congestion.

TCP Tahoe [22], coarse-gain approach suggests that the TCP connection starts with a slow start. For each transmission, the suggested protocol increases the sender congestion window by 1. A packet loss or long round trip time is taken as a sign of congestion and prevention algorithms are used to handle it. However, due to a very slow increase in window size, it can not achieve maximum link utilization. TCP New Reno [28], enhances the congestion avoidance techniques by adding some intelligence over it, like fast retransmission and fast recovery. Fast retransmission and recovery phases eliminate about a part of the coarse gain by multiplying a predefined threshold variable and yields a 20% improvement in throughput when that variable is assigned to 0.5. But, when there exist multiple packet losses in one window, it doesn't perform well. TCP Bus [29] is able to detect multiple packet losses. It increases its fast recovery duration until all the data which were outstanding at the time it entered fast recovery is acknowledged. It takes one RTT to detect each packet's loss. When the acknowledgment for the fast retransmitted segment is received, only then it can deduce which segment is lost. Hence, loss of ACK packet, the RTT time is doubled and the sender can not take decision properly.

All the above-described protocols use cumulative acknowledgment (ACK) from the receiver. TCP Vegas [30] was proposed, where a congestion control mechanism based on the proactive measure. It doesn't depend solely on packet loss as a sign of congestion. It detects congestion before the packet loss occurs. It is totally based on a significant estimation of RTT, i.e., by keeping track of how long it takes for the ACK to get back. Vegas increases its congestion window exponentially only after every other RTT which are in time and it calculates the actual sending throughput to the expected. When the difference goes above a certain threshold, it exits slow start and enters congestion avoidance. The athors in [31] use selective ACK (SACK) in TCP. In this technique, the receiver identifies the segment of message for wihch ACK is acknowledged. Also, it identifies the selective segments are received. Then sender (after receiving ACK and SACK) has a picture of acknowledged segments and outstanding segments. Here, duplicate ACK is considered as packet loss, and SACK moves into the congestion avoidance phase by halving the congestion window. During this time, as no packet has been forwarded the link goes empty and as the congestion window reduces to half, bandwidth utilization goes low. All the above mechanisms consider that the only cause for packet loss is congestion.

In the protocols embed with AIMD perceives a saw-tooth pattern in its congestion window which decreases the average length of congestion window. As a result, throughput drops dramatically due to under-utilization of bandwidth. Thus, it is observed that sender's flow control mechanism triggers and reduces the congestion window size even though there is no congestion. The reason is that congestion control algorithms work in the transport layer as a part of TCP where actual link utilization status can be found at the Media Access Control (MAC) protocol of the data link layer. Thus, there is a need of making a bridge between the MAC of the data link layer, and the TCP of the transport layer which would be called a cross-layer approach.

There are two popular approaches for improving the performance of TCP using a cross-layer mechanism in the wireless network [32], i.e., a link-layer approach and Explicit Congestion Notification (ECN) approach like [33].

In the link-layer approach, coding and ARQ schemes are used to retransmit packets that are lost due to variation in link state. This strategy works because the time required for recovering from packet error at the link-layer is much smaller than the time in which TCP receives feedback about the packet. The latter approach is based on the addition of the ECN bit in the TCP header. Whenever there is congestion at the router, the router sets the ECN bit in the TCP header of the packet. When the marked packet reaches the destination, the destination becomes aware of the congestion in the network. The destination then explicitly informs the source of congestion so that it may reduce its transmission rate.

Ad-hoc networks also have some tough challenges to TCP, because of dynamic aspects [34]. Some congestion control proposals such as BBR [35], TCP-ELFN [36], TCP-BUS by [29] use the feedback information from the network layer to signal non-congestion related cause of packet loss. These feedback approaches help TCP to distinguish true congestion and other problems such as channel errors, link contention, and route failure. Masri et al. [37] have mentioned that the special feedback is received from the receiver that helps to regulate the window size of the sender. It increases/decreases the congestion window according to the level of congestion at the intermediate nodes in wireless mesh networks. However, the above protocols would fail when the transmission link experiences a long delay where nodes are deployed in long distances. There may be a higher number of packets at transit till the sender receives an explicit notification that the link is congested. Thus, the packets in transit are dropped.

The authors in [38] have extended the concept of [39]. Where they have adopted the flow rate by considering the queue length of the buffer of the neighboring nodes. However, this technique will attain higher local overhead as all neighboring nodes will explicitly send their buffer status. ECN approach is extended by [40] where they have enabled a router to mark packets and the receiver to signal congestion to the sender without trusting the sender whether it has responded to congestion or not. However, this mechanism does not consider the global channel scenario. It only percepts the receiving channel strength and updates the congestion window without considering the sending channel strength of the receiver. Thus congestion control is still an open issue for all types of networks.

TABLE 1. Acronyms and descriptions.

Terms	Acronyms
$Link_{status}$	Percentage of channel currently used
$Buffer_{status}$	Percentage of buffer currently used
f_p	Packet forwarding counter
f_p^t	Packet forwarded by time instance I^t
$Channel\_capacity$	bandwidth of channel
$ssthress\_buffer$	Threshold of buffer
$ssthress\_link$	Threshold of link
$CWND_{(t)}$	Congestion window length at time t
LU^t	Link utilization at time t
I^t	Time instance at time t

III. MOTIVATION

A. PROBLEM STATEMENT

In this paper, we have focused a scenario of smart city network with vehicles connected with internet (IoV) network shown in Fig. 1. Here, vehicles communicate with other vehicles and road-side units (RSU) regarding accident and other information like traffic jam. At the same time, vehicles can get the information from RSUs regarding the status of the upcoming roads for a smooth and faster transportation system. Further, RSUs communicate themselves regarding this important information to improve the overall transportation system, parking system of the city. Also, the information is supposed to be stored in the central server after filtering the data which may be used to predict the traffic behaviour of the individual roads and cities using machine learning algorithms. Thus, the network system experiences a sudden rise of data packets when there will be an accident and city is too much crowd. Hence, it is very much essential to transmit the data packets in the peak hours successfully with maximum throughput, which requires an efficient congestion detection and control approach.

In all the above congestion detection mechanisms, it is observed that the congestion avoidance techniques start only after the detection of congestion. However, the existing mechanisms detect congestion by packet loss at any intermediate node or receiving ACK after time out. But, these events happen after occurrence of congestion although some approaches monitor on the delay of receiving ACK and buffer occupancy. Thus, there is a need for triggering notification by the intermediate node which is experiencing high forwarding rate and buffer usage beyond a certain threshold. This scenario is presented in Fig. 3 where the intermediate node experiences packet loss due to buffer whelming. However, some algorithms also implement the congestion prevention mechanism. Still, it is seen that the performance of the TCP is low as the congestion due to buffer full is not successfully controlled, which motivates us to design such an algorithm that can overcome this problem.

B. IMPORTANT ISSUES IN EXISTING CONGESTION CONTROL MECHANISM

The explicit congestion control notification (ECN) provides a robust model for congestion control. ECN notifies sender node regarding the status of the congestion in the path explicitly. It uses two bits in the IP header so that the intermediate node acknowledges receiver regarding congestion. After receiving the existing of congestion in the path, the receiver enables ECN-Echo (ECE) bit and echo backs the sender that congestion exists in the path. Thus, the sender reduces the congestion window size and acknowledges the receiver by enabling the congestion-window-reduced (CWR) bit in the next data packet. Here, the congestion is identified by an intermediate node of the path but the information is propagated to the receiver and again it echos back to the sender by the transport later protocol. Thus, it takes at least one round-trip-time to react towards congestion control. To act proactively towards congestion control we propose a new mechanism here. In the proposed model, the intermediate node directly acknowledges the sender about the congestion in the network and the sender node reduces the window size.

IV. PROPOSED APPROACH

Considering the above issues and challenges, we propose a novel congestion prevention and avoidance algorithm. As we have discussed that congestion window size is being regulated based on buffer status and link utilization, the congestion status is being notified to the sender by the receiver based on its own bandwidth utilization and buffer status using cross layer approach.

A. SYSTEM AND NETWORK SETTINGS

In the proposed model, various end points and intermediate nodes as considered. End points are the sources and destinations of the data packets which such as vehicles, smart camera of the traffic points, server or clouds. The intermediate nodes helps to forward the packets to the needful destinations. Here, the RSUs and the vechicles act as intermediate nodes. RSUs installed on the city are connected to each other in mesh topology when they are in range of communications, where some of the RSUs can send the crucial information to the server. All vechiles in the transmission range of the RSU are connected in star topology. A vehicle is supposed to broadcast message when the message is either accidental or the vechile is not in the range of an RSU, which sould be considered as high priority message. The duplicate message received at any RSU are to be dropped there only. In our model, we have considered that an RSU can connect with maximum 1000 vechicles which turns to number of connections. However, we have not considered mobility speed, priority of the packets and security aspects in our model. We have focused only on congestion control aspects.

Algorithm 1 Algorithm for Receiving an ACK Packet at Intermediate Node

```
1  Calculate $Link_{status}$ as $\frac{f_p \text{in previous unit of time}}{Channel\_capacity}$ ;
2  Calculate $Buffer_{status}$ as $\frac{buffer\_length}{max\_buffer}$ ;
3  if $Buffer_{status} > ssthress\_buffer$ and $BUS = 0$ then
4      Assign $BUS \leftarrow 1$ ;
5  else
6      do not update $BUS$ bit of ACK packet ;
7  end
8  if $Link_{status} > ssthress\_link$ and $LIS = 0$ then
9      Assign $LIS \leftarrow 1$ ;
10 else
11     do not update $LIS$ bit of ACK packet ;
12 end
```

B. PROPOSED CONGESTION CONTROL MECHANISM

In this approach, an explicit throughput and buffer status notification-based mechanism is proposed instead of explicit loss notification. The throughput and buffer status notification is sent from the receiver (intermediate) node to the sender node. The status of the buffer at the transport layer is provided to the data-link layer. In this approach, we have introduced two extra bits from the reserved bits in the TCP packet header to acknowledge the buffer status and link utilization. In the existing congestion control algorithm follows the ELN technique that modifies two bits in the IP header of the data packet and the data packet is received at the destination node. After receiving the data packet at the destination node, it generates an ACK packet by modifying two bits of the TCP header and sending it to the source node. Thus in the existing congestion control algorithm, the sender takes too long to reduce the congestion window. However, in the proposed approach, the intermediate node modifies the BUS and LIS bit of TCP header of the next ACK packet traveling towards the source node so that it can reduce the congestion window proactively. These two bits play a vital role in slow start, loss recovery, and congestion prevention algorithm at the sender side. The sender initiates the congestion prevention algorithm after receiving this notification and updates the congestion window as additive increase, multiplicative increase, subtractive decrease, or multiplicative decrease when required. Thus the proposed approach avoids congestion due to packet loss caused by buffer overwhelming. However, the packet loss due to interference is not the scope of this research. The proposed mechanism using the cross-layer approach performs better as compared to traditional approaches.

1) CROSS-LAYER APPROACH

The data link layer has link status and TCP layer has the congestion window information for queuing packets. The two layers will share information using cross layer approach. So, the data link layer uses two reserve bits of TCP layer to update

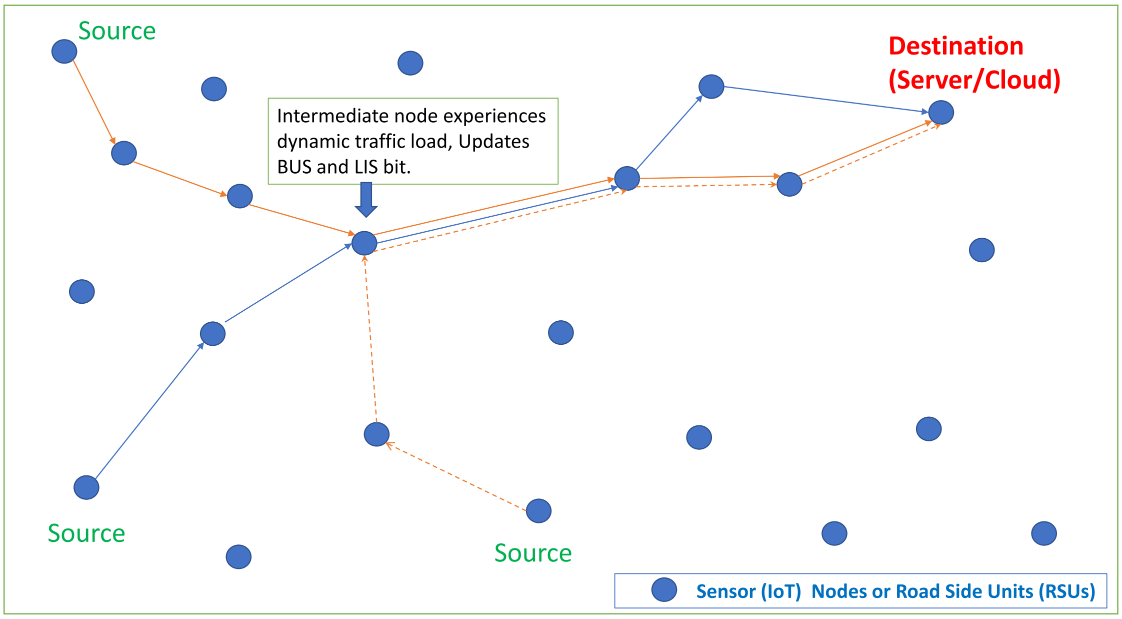

Source
Destination (Server/Cloud)
Intermediate node experiences dynamic traffic load, Updates BUS and LIS bit.
Source
Source
Sensor (IoT) Nodes or Road Side Units (RSUs)

FIGURE 2. Intermediate node experiences dynamic traffic load, updates BUS and LIS bit to all source nodes.

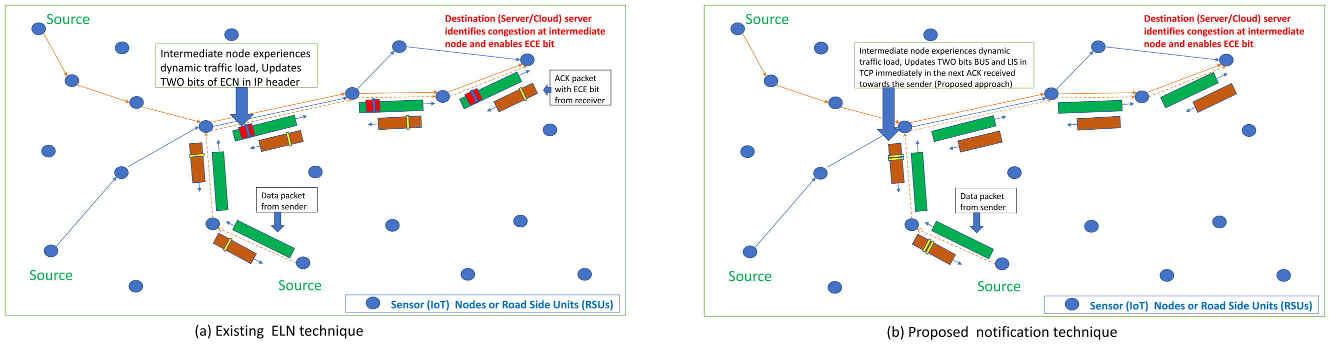

Source
Destination (Server/Cloud) server identifies congestion at intermediate node and enables ECE bit
Intermediate node experiences dynamic traffic load, Updates TWO bits of ECN in IP header
ACK packet with ECE bit from receiver
Data packet from sender
Source
Source
Sensor (IoT) Nodes or Road Side Units (RSUs)
(a) Existing ELN technique
Source
Destination (Server/Cloud) server identifies congestion at intermediate node and enables ECE bit
Intermediate node experiences dynamic traffic load, Updates TWO bits BUS and LIS in TCP immediately in the next ACK received towards the sender (Proposed approach)
Data packet from sender
Source
Source
Sensor (IoT) Nodes or Road Side Units (RSUs)
(b) Proposed notification technique

FIGURE 3. Intermediate node experiences dynamic traffic load, updates BUS and LIS bit to all source nodes.

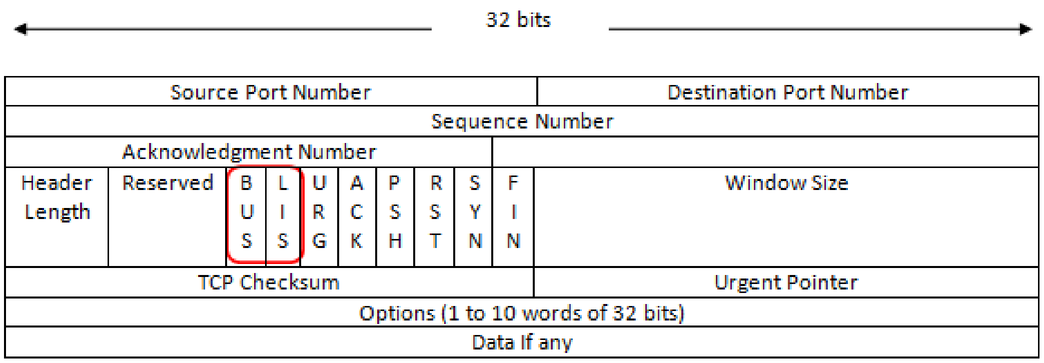


FIGURE 4. Modified TCP header format with LIS and BUS bits.

buffer and link utilisation. Thus, these two bits would be borrowed from the reserved six bits of the TCP header which would cost no overhead to the TCP packet as shown in Fig. 4. Thus, buffer status is represented by one bit (BUS) and link utilization status would be represented by one bit (LIS). The intermediate node sets two bits always in explicit ACK packet or by piggybacking the ACK packets using selective repeat ARQ sliding window congestion control protocol. It does not

Algorithm 2 Algorithm for Updating *cwnd* at Source Node

```
if no timeout and no dupACK then
    if BUS=0 and LIS=0 then
        $CWND_{(t+1)} \leftarrow 2 * CWND_{(t)} + 1$ ;
    else if BUS=0, LIS=1 then
        $CWND_{(t+1)} \leftarrow CWND_{(t)} + \alpha$ ;
    else if BUS=1, LIS=1 then
        $CWND_{(t+1)} \leftarrow CWND_{(t)} - \gamma$ ;
    end
else
    if dupACK received then
        $CWND_{(t+1)} \leftarrow CWND_{(t)} / 2$ ;
    else
        Retransmit packets in buffer ;
    end
end
```

cost any overhead to the IoV network. The network layer calculates the link status periodically which is the portion of packets sent over channel capacity. When the intermediate node achieves expected use of channel, it accesses the TCP header and updates LIS bit to 1. Thus, it acknowledges to the senders via ACK packets by updating the TCP header. In addition to this, it also computes buffer status which is the fraction of current buffer occupancy over maximum buffer length. When the intermediate node experiences more use of buffer than its threshold, it anticipates congestion. The intermediate node updates the sender node by updating the BUS bit of the TCP header. The detailed process is presented in the algorithm 1. Buffer status information and link status information are not available in TCP header. Here, Cross layer approach is used to access the TCP header information at data link layer and network layer. Thus, the source node regulates its packet penetration to the IoV network using algorithm 2. The different acronyms used are described in Table 1 for better readability.

C. CONGESTION PREVENTION

The congestion control algorithm of the proposed approach has three phases; i) slow start (Initial rate and adjustment), ii) Congestion Prevention (Flow rate control mechanism) iii) Early recovery (Congestion avoidance on packet drop). In each phase, the sender regulates the congestion window size differently using equation 1. These three phases are explained as follows.

1) **slow start**: Basically, it is the initial phase where the link is underutilized. Thus, to achieve the maximum throughput the sliding window grows exponentially till the link can be utilized at its maximum. When the sender receives notification from the receiver that it has achieved a predefined threshold of link utilization, the sender enters phase two for congestion prevention. In this proposed mechanism *ssthress_link* is taken as 50% keeping the sudden rise of data packets at the intermediate nodes. It is observed that 50% performs better result when it is tested with different settings with 50% and 60%. Thus, to maintain a dynamic update in this *ssthress_link*, machine learning algorithms can be used based on various time stamps, traffic congestion in the city, weather conditions, etc.
2) **Congestion Prevention**: In this phase, the sender follows Additive Increase and Subtractive Decrease (AISD) approach and tries to maintain consistency in throughput. Moreover, the congestion window is increased by an additive parameter α enhance link utilization that will achieve better throughput. Thus, it slowly regulates the congestion window such that it utilizes the link 100% and goes for buffering packets. Further, when it utilizes 30% of the buffer it decreases the congestion window so that it will not affect round trip time (RTT) but it can sustain a little bit of time to maintain link utilization when the window size is reduced to half due to packet drop. Thus, *ssthress_buffer* is set as 30% considering various settings and their performance.
3) **Early recovery**: In this phase, the congestion window of the sender is reduced to half when it receives three duplicate ACKs like mSACK. It is assumed that the cause of packet drop is congestion when three duplicate ACKs are received. Otherwise, after receiving the first and second ACK, the sender retransmits the packet. The intermediate node sets the LIS bit as 0 if bandwidth is under-utilized. In such cases, the buffer utilization status is also 0. So, the source nodes increase their window size by double of their existing window size. Thus, it achieves proper bandwidth in a very short time. The intermediate node updates the status of its own buffer and link to the source nodes individually. Based on the notifications received from the intermediate nodes, the source nodes update their window size. In case of under-utilization of buffer and proper utilization of bandwidth, the window size increases slowly (additive increases). In case of under-utilization of bandwidth, the window size increases rapidly to use the bandwidth. However, the window size of the source decreases slowly when buffer and bandwidth is properly utilizes. After a threshold of buffer utilization exceeds, the window size decreases slowly to maintain a proper utilization of buffer and to avoid packet drops due to buffer overwhelming.

$$CWND_{(t+1)} = \begin{cases} 2 * CWND_{(t)} + 1, & \text{if BUS=0, LIS=0} \\ CWND_{(t)} + \alpha, & \text{if BUS=0, LIS=1} \\ CWND_{(t)} - \gamma, & \text{if BUS=1, LIS=1} \\ CWND_{(t)}/2, & \text{if duplicate ACK} \end{cases} \tag{1}$$

$$LIS = \begin{cases} 1, & \text{Link utilization is more than 60\%} \\ 0, & \text{Otherwise} \end{cases} \tag{2}$$

$$BUS = \begin{cases} 1, & \text{Buffer utilization is more than 50\%} \\ 0, & \text{Otherwise} \end{cases} \tag{3}$$

D. CONGESTION AVOIDANCE

In this phase, the sender detects the actual cause of congestion. A multiplicative decrease scheme is followed if it is interpreted that packet drop is due to buffering overwhelming. According to the proposed approach, the IoV network will not experience any congestion in ideal scenarios. However, in dynamic scenarios where the number of TCP connections varies to a large extent, the intermediate node experience a sudden change in link utilization, and buffer status will change accordingly. Thus, the intermediate node will not able to convey the buffer and link status to the sender node instantly, and the sender node will go on sending more packets. Further, the sender does not change its congestion window if the ACK packet from the receiver does not reach the sender. Thus, the outstanding packets at the current sliding window are re-transmitted and cause packet drop.

V. SIMULATION RESULTS AND PERFORMANCE ANALYSIS

During simulation, various aspects of the smart city environment are taken into consideration. Further, the parameters taken here for simulation are aligned with the parameters of the routers available which are shown in Table 2.

TABLE 2. Parameter settings at a glance.

Parameters	Value
Transport Protocol	TCP NewReno
Congestion Control Mechanisms	mAIMD, mSACK, Proposed
Sliding Window Protocol	Selective Repeat ARQ
Packet size	30 bytes
Interface Queue Length	300 packets
Number of TCP Connections	1000
Interface Queue Type	DropTail
Simulation time (in seconds)	1000–5000
Channel bandwidth	10 MHz
Mac Layer	IEEE 802.11p
Communication Range	upto 1000 m
Propagation Model	Nakagami model

The proposed work mainly focused on the congestion window size of the sender, link utilization status, and buffer status of the intermediate node. According to the availability of the buffer space and utilization of the channel, the congestion window of the sender is regulated. The main objective of this approach is to prevent congestion by regulating packet transmission at the sender side. Here, we have assumed that vehicles can communicate with RSUs without loss. The model is simulated extensively to investigate the efficiency of the model in terms of throughput, packet success rate, the average length of buffer occupancy, and variation of congestion window size of the sender.

A. SIMULATION PARAMETERS

Basic payload (M_{PL}) generated by a sensor in each round is predefined, i.e., 120 bytes [41]. Thus, number of packets to be sent in a round depends on the size of the packet. The packet size varies from 120, 60, 40, 30, 24 and 20 bytes based on the predefined setup. Hence, size of the data packet (M_P) including 8 bytes of header ($M_H = 8$) is 28 bytes to 128 bytes, i.e., $M_P = M_{PL} + M_H$. The number of connections in transportation system which is supposed to be smart enough due to its real time update of the scenario may reach 1000 in peak hours. If we consider a router which works as an intermediate node between router-router, router-server, or RSU-server, huge number of connections are expected in the intermediate node. Further, the TCP header has 16 bit of port address which can support up to 10^5 connections [9]. Here, the interface queue length (Queue size) is taken as 300 as a standard size. The queue size will vary if the router have enough memory. In order to monitor the congestion window size of the sender, the model is simulated for different round trip times (RTT), i.e., 1000 to 4000 times. Also, the model keeps an eye on the buffer occupancy of the intermediate node in different RTTs and their overwhelming frequencies. The *ssthresh_buffer* is experimented with different values (such as 0.3, 0.4, 0.5, 0.6, and 0.7). As the results were vary closed for 0.3, 0.4 and 0.5, so for presentation purpose, we kept 0.3 in the paper. Similarly, the throughput shows closed results when threshold of link utilization is 0.6 and 0.7, so we kept 0.6 in the paper for presentation purpose. The simulation results are analyzed rigorously and found that the proposed approach shows better results in terms of buffer utilization and throughput as compared with the other congestion control approaches (mAIMD and mSACK) which are presented below.

B. EVALUATION PARAMETERS

The model is evaluated based on various parameters that include packet success ratio (PSR), throughput in terms of link utilization, average length of buffer space occupied, variation of congestion window size. PSR is calculated for the intermediate node, where it is the ratio of the number of packets successfully forwarded to the number of packets received by the intermediate node using equation 4. Buffer space is also monitored at the intermediate node. In each acknowledgment packet sent to the sender, the intermediate node keeps track of the number of packets available at buffer. Further, link utilization is calculated using equation 5. It is the ratio of the average number of bits forwarded in a unit of time to the channel capacity. That represents time taken to transmit a block of packets packet to the total time it takes to transmit, idle time and to receive the acknowledgement of packets.

$$PSR\% = \frac{f_p}{f_r} \tag{4}$$

$$LU^t(in\%) = \frac{f_p^t \, * \text{ Packet size} * 8}{(I^t - I^{t-1}) * \beta} \tag{5}$$

where, f_p^t represents number of packets forwarded by time instance t(I^t). f_p represents the total number of packets forwarded during the simulation time. f_r represents number of packets received, β is bandwidth of the link.

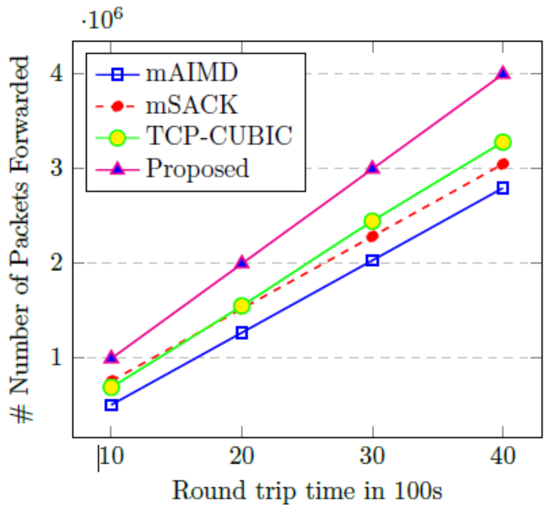


FIGURE 5. Throughput with respect to number of packets forwarded successfully.

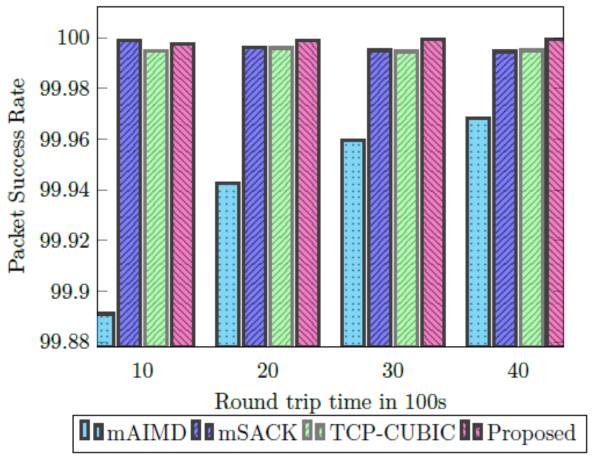


FIGURE 6. Efficiency in terms of packet success rate.

C. THROUGHPUT ANALYSIS

In Fig. 5, the throughput of the proposed approach is presented with a comparison with the mSACK, mAIMD and TCP-CUBIC approaches. After a number of experiments, the average throughput is taken into analysis and presented in the above figure. It is observed that the mSACK transmits slightly more packets than mAIMD approach. In the same experiment settings, the proposed model achieves close to 33% better results than mSACK and close to 90% better results than mAIMD in terms of buffer usage. But in another perspective, it is observed that mAIMD approach achieves this throughput by less number of attempts, i.e., packet service ratio (PSR) of mAIMD is significantly higher than mSACK. The main cause behind less PSR of mSACK approach is that more packet loss occurs in the case of mSACK than mAIMD. Fig. 6 presents the percentage (%) of packets successfully sent in different RTTs. Thus, minimum packet loss achieves better PSR. In all the experiments it is observed that the mAIMD approach provides a little bit less PSR but all models have PSR close to 99% as all the models follow the early recovery model of TCP NewReno. But, due to the early recovery model, at each instance of packet drop, the congestion avoidance algorithm reduces the congestion window to half. Thus, after reducing the window size it takes a little bit of time to attain the optimal window size and leads to under-utilization of bandwidth. Considering the *ssthress_link*, it shows that notification raised at 50% ocupancy of of bandwidth yields better result as compared to 60% and more as shown in Fig. 7 (a) and Fig. 8 (a). Further, it shows that packet transmission rate decreases when buffer threshold range increases which is clearly visible in Fig. 7(b) and Fig. 8(b). However, the latest model TCP-CUBIC can not perform well as it takes much time to reach the plateau level of the congestion window. Further, the short length between the source and the destination in transportation systems networks, the fluctuation of the number of connections are very high. Thus, the real time monitoring of the congestion window using the cubic model can not perform well which was also suggested by the TCP-CUBIC model.

D. BUFFER OCCUPANCY ANALYSIS

All three models follow the same approach in their initial phase (slow start) by which they try to attain the maximum utilization of the link capacity. However, the existing two approaches can not sustain due to their aggressive nature in increasing the congestion window. However, the proposed approach follows an additive increase and subtractive decrease (AISD) approach to maintain the link utilization by receiving the link utilization and buffer utilization status from the intermediate (immediate successor) node. Thus, the proposed model maintains a consistency in its congestion window size to maintain 30% utilization of the buffer. When the sender node receives a notification about the threshold of buffer utilization, then the sender follows a subtractive decrease mechanism to reduce the window size by a fixed chunk (here we have taken 2). Thus, the proposed model maintains to utilize buffer space close to 30% which is shown in Fig. 9. It is observed that *ssthress_buffer* notification affects both in consistency of the congestion window and packet transmission rate. The experiment result shows that notification of BUS bit at 30% buffer outperforms than other combinations. It is observed that the TCP-CUBIC model can use a substantial portion of the buffer. Because, it does not reduce the congestion window on round trip time basis. The window size in CUBIC model are updated on real time basis. Thus, as the environment is dynamic in nature considering the number of connections and huge number of connections are there as compared to the bandwidth capacity of the network, the CUBIC model can not perform well.

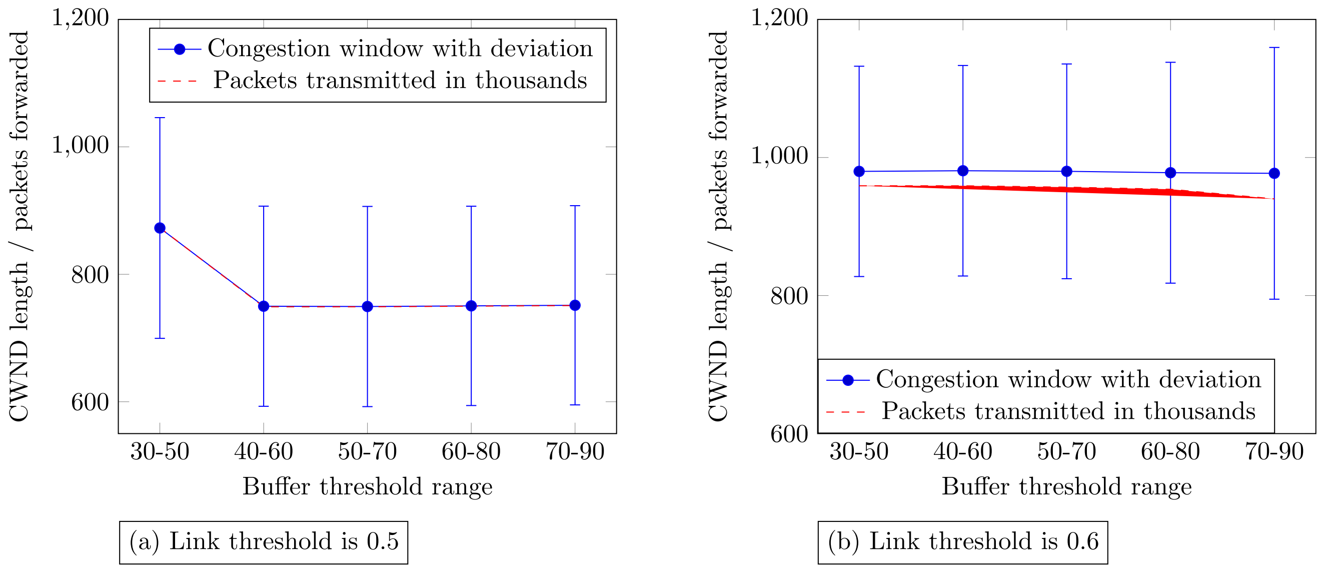


FIGURE 7. Threshold analysis of buffer and bandwidth. Buffer threshold range is taken as 20.

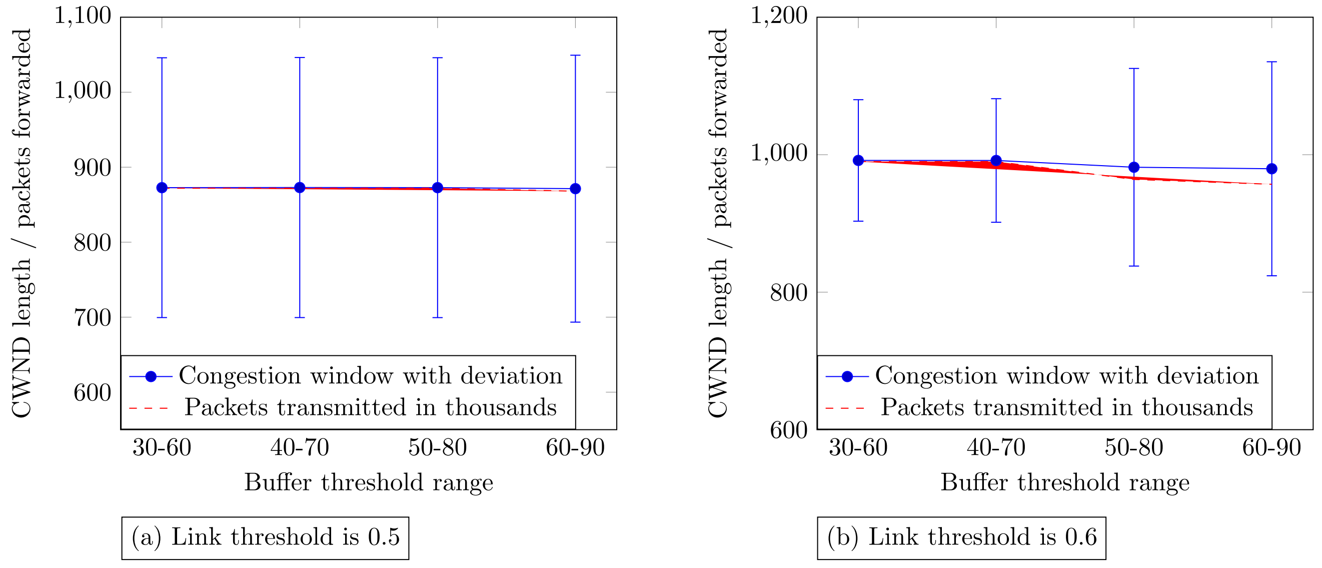


FIGURE 8. Threshold analysis of buffer and bandwidth. Buffer threshold range is taken as 30.

E. CONGESTION WINDOW LENGTH ANALYSIS

Furthermore, the length of the congestion window is also minutely monitored in our experiments. It is observed that the length of the congestion window in the proposed model maintains consistency, which is shown in Fig. 10. As we have discussed earlier, the congestion window reduces to half when there is a packet loss in both mAIMD and mSACK models. Apart from the early recovery phase, the congestion window is never reduced in mAIMD and mSACK model rather it always grows either exponentially in the slow start phase or linearly after attaining threshold. However, just after early recovery, the existing models increase their congestion window linearly. Thus, it takes too much time to attain the optimal congestion window size again. Thus, the congestion window size varies frequently and leads to less throughput. The TCP-CUBIC varies at each interval of time as the environment is much dynamic. But, it shows equivalent variation of the window length as compared to our model and the cubic model can not reach better window size as it takes time to update. Thus, the CUBIC model under-utilises the link capacity as it can not update the window size proactively. The growth of window size is shown in Fig. 11. It shows that the proposed model performs better than TCP-CUBIC. As the CUBIC model updates window size on real time basis, it takes more time to reach at optimal level as compared to RTT based models. Thus, the scenarios where number of

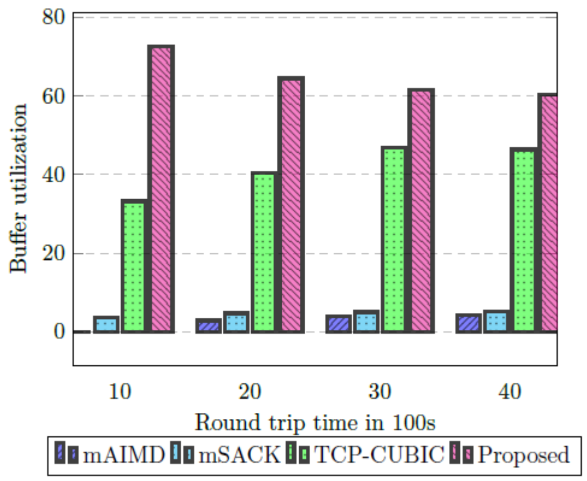


FIGURE 9. Average buffer utilization by the nodes (average packets occupied out of 300).

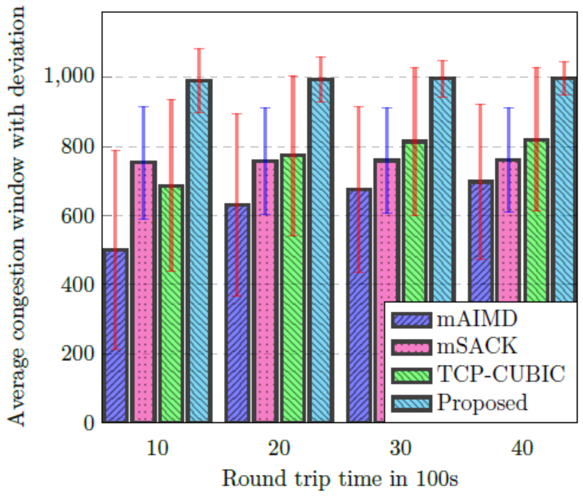


FIGURE 10. Average length of congestion window with variations at sender nodes.

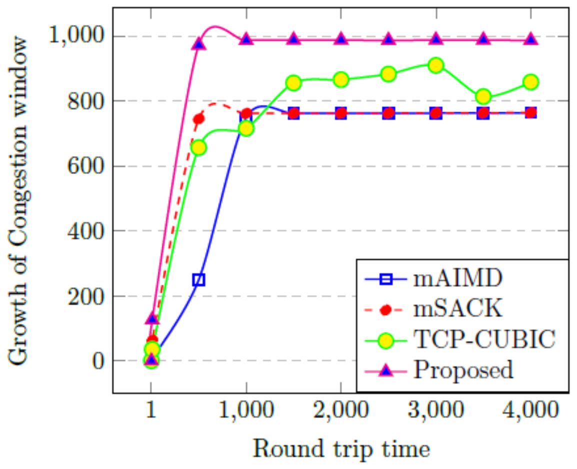


FIGURE 11. Growth of congestion window and convergence.

hops are less RTT based models outperform than CUBIC model. Further, the scenarios where number of connections are varying frequently CUBIC model will sustain very less time in the platue level. However, the proposed model will work effectively in all scenarios.

However, the proposed approach follows an additive increase and subtractive decrease of congestion window after attaining the initial threshold of bandwidth utilization. But, it follows the same steps to achieve the initial threshold. Thus, the proposed model tries to attain optimal congestion window size, which helps to achieve maximum throughput by utilizing the link capacity at its maximum level. Thus, the consistency in maintaining optimal window size makes the proposed model efficient than the existing models. Analysing the experimented results which are shown in Fig. 7 and Fig. 8, it is realised that length of average congestion window and its variation is high in case of threshold notification at higher range. Further, average length of congestion window and its variation is less in case of lower range of notifications along with lower packet transmission rate. Thus, a balancing of thresholds are required to maintain higher transmission rate and lower deviation of congestion window. According to our experiment, it is suggested that *ssthress_link* as 50% and *ssthress_buffer* as 30%-50% gives better result.

VI. CONCLUSION

In this paper, we have presented a novel congestion control approach that will be useful in smart-city networks and transportation system of vehicular networks to monitor traffic management and parking management. The proposed model mainly focuses to prevent congestion in IoV network using the link and buffer utilization status of the intermediate node. It reduces the risk of congestion at any intermediate node and saves energy by reducing packet drop due to buffer overwhelming. The proposed approach sends link utilization status and buffer utilization status to the predecessor node by adding two extra bits in the acknowledgment packet which will prevent congestion proactively in IoT environments. The proposed model achieves better results as compared to mSACK and MAIMD approach in terms of throughput and packet success rate. In the future, this approach may be used in other network environments like WSNs with health care systems, industrial security/alarm systems, and Industrial machinery monitoring systems, etc. where the data transmission link experiences higher transmission delay. Further, message flooding stands as an issue in V-V communication. It can be minimized by tagging the messages with message ID and location ID. So that all incoming duplicate messages will be dropped at any of the receiver node. Hence, tagging messages with ID is an open issue for vehicular networks and transportation system.

REFERENCES

[1] C. Sergiou, P. Antoniou, and V. Vassiliou, "A comprehensive survey of congestion control protocols in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1839–1859, 4th Quart., 2014.

[2] R. Al-Saadi, G. Armitage, J. But, and P. Branch, "A survey of delay-based and hybrid TCP congestion control algorithms," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3609–3638, 4th Quart., 2019.

[3] M. J. A. Jude, V. C. Diniesh, and M. Shivaranjani, "Throughput stability and flow fairness enhancement of TCP traffic in multi-hop wireless networks," *Wireless Netw.*, vol. 26, no. 6, pp. 4689–4704, Aug. 2020.

[4] S. Jamali, M. M. Talebi, and R. Fotohi, "Congestion control in high-speed networks using the probabilistic estimation approach," *Int. J. Commun. Syst.*, vol. 34, no. 7, p. e4766, May 2021.

[5] H. Haile, K.-J. Grinnemo, S. Ferlin, P. Hurtig, and A. Brunstrom, "End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks," *Comput. Netw.*, vol. 186, Feb. 2021, Art. no. 107692.

[6] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1146–1159, Nov. 2019.

[7] S. R. Pokhrel and J. Choi, "Improving TCP performance over WiFi for Internet of Vehicles: A federated learning approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6798–6802, Jun. 2020.

[8] S. S. Shah, A. W. Malik, A. U. Rahman, S. Iqbal, and S. U. Khan, "Time barrier-based emergency message dissemination in vehicular ad-hoc networks," *IEEE Access*, vol. 7, pp. 16494–16503, 2019.

[9] A. Minakhmetov, C. Ware, and L. Iannone, "TCP congestion control in datacenter optical packet networks on hybrid switches," *J. Opt. Commun. Netw.*, vol. 10, no. 7, pp. 71–81, Jul. 2018.

[10] P. J. Argibay-Losada, G. Sahin, K. Nozhnina, and C. Qiao, "Transport-layer control to increase throughput in bufferless optical packet-switching networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 8, no. 12, pp. 947–961, Dec. 2016.

[11] E. Blanton, M. Allman, K. Fall, and L. Wang. *A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP*, document RFC3517, RFC Editor, USA, Apr. 2003, doi: 10.17487/RFC3517.

[12] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operat. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[13] C. Lim, "Improving congestion control of TCP for constrained IoT networks," *Sensors*, vol. 20, no. 17, p. 4774, Aug. 2020.

[14] L. P. Verma and M. Kumar, "An IoT based congestion control algorithm," *Internet Things*, vol. 9, Mar. 2020, Art. no. 100157.

[15] B. Fu, Y. Xiao, H. J. Deng, and H. Zeng, "A survey of cross-layer designs in wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 110–126, 1st Quart., 2014.

[16] F. Foukalas, V. Gazis, and N. Alonistioti, "Cross-layer design proposals for wireless mobile networks: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 70–85, 1st Quart., 2008.

[17] C. Lochert, B. Scheuermann, and M. Mauve, "A survey on congestion control for mobile ad hoc networks," *Wireless Commun. Mobile Comput.*, vol. 7, no. 5, pp. 655–676, 2007.

[18] R. Jurdak, *Wireless Ad Hoc and Sensor Networks: A Cross-Layer Design Perspective*. Cham, Switzerland: Springer, 2007.

[19] J. Camp and E. Knightly, "Modulation rate adaptation in urban and vehicular environments: Cross-layer implementation and experimental evaluation," *IEEE/ACM Trans. Netw.*, vol. 18, no. 6, pp. 1949–1962, Dec. 2010.

[20] C. Sau, T. Fanni, C. Rubattu, L. Raffo, and F. Palumbo, "The multi-dataflow composer tool: An open-source tool suite for optimized coarse-grain reconfigurable hardware accelerators and platform design," *Microprocessors Microsyst.*, vol. 80, Jan. 2021, Art. no. 103326.

[21] M. A. Hanif and M. Shafique, "A cross-layer approach towards developing efficient embedded deep learning systems," *Microprocessors Microsyst.*, vol. 88, Feb. 2022, Art. no. 103609.

[22] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[23] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. 1st Annu. Int. Conf. Mobile Comput. Netw.*, 1995, pp. 2–11.

[24] B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK," *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 959–971, Dec. 2003.

[25] J. Tang, Y. Jiang, X. Dai, X. Liang, and Y. Fu, "TCP-WBQ: A backlog-queue-based congestion control mechanism for heterogeneous wireless networks," *Sci. Rep.*, vol. 12, no. 1, pp. 1–17, Mar. 2022.

[26] X. Ji, B. Han, C. Xu, C. Song, and J. Su, "Adaptive QoS-aware multipath congestion control for live streaming," *Comput. Netw.*, vol. 220, Jan. 2023, Art. no. 109470.

[27] A. El-Bakkouchi, M. E. Ghazi, A. Bouayad, M. Fattah, and M. E. Bekkali, "EC-elastic an explicit congestion control mechanism for named data networking," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 11, pp. 1–10, 2021.

[28] A. Gurtov, T. Henderson, S. Floyd, and Y. Nishida. *The NewReno Modification to TCP's Fast Recovery AlgorithmThe NewReno Modification to TCP's Fast Recovery Algorithm*, RFC Editor, Apr. 2012, p. 16. [Online]. Available: https://www.rfc-editor.org/info/rfc6582, doi: 10.17487/RFC6582.

[29] D. Kim, C.-K. Toh, and Y. Choi, "TCP-BuS: Improving TCP performance in wireless ad hoc networks," *J. Commun. Netw.*, vol. 3, no. 2, pp. 1–12, Jun. 2001.

[30] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[31] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. *TCP Selective Acknowledgment Options*, document RFC 2018, 1996. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2018.html

[32] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross-layer design," *IEEE Wireless Commun.*, vol. 12, no. 1, pp. 3–11, Feb. 2005.

[33] S. Floyd, "TCP and explicit congestion notification," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, pp. 8–23, Oct. 1994.

[34] B. C. Sreenivas, G. C. B. Prakash, and K. V. Ramakrishnan, "L2DB-TCP: An adaptive congestion control technique for MANET based on link layer measurements," in *Proc. 3rd IEEE Int. Advance Comput. Conf. (IACC)*, Feb. 2013, pp. 1086–1093.

[35] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[36] J. P. Monks, P. Sinha, and V. Bharghavan, "Limitations of TCP-ELFN for ad hoc networks," in *Proc. MOMUC*, 2000, pp. 1–15.

[37] A. E. Mastri, A. Sardouk, L. Khoukhi, A. Hafid, and D. Gaiti, "Neighborhood-aware and overhead-free congestion control for IEEE 802.11 wireless mesh networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 10, pp. 5878–5892, Aug. 2014.

[38] K. M. Saleem, S. Waris, I. Ali, M. H. Anisi, and M. I. Khan, "Mitigation of packet loss using data rate adaptation scheme in MANETs," *Mobile Netw. Appl.*, vol. 23, no. 5, pp. 1141–1150, Dec. 2018.

[39] T. K. Mishra and S. Tripathi, "Explicit throughput and buffer notification based congestion control: A cross layer approach," in *Proc. 8th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2015, pp. 493–497.

[40] S. Bommisetti, B. Annappa, and M. P. Tahiliani, "Extended ECN mechanism to mitigate ECN-based attacks," in *Proc. Int. Conf. Control, Instrum., Commun. Comput. Technol. (ICCICCT)*, Jul. 2014, pp. 1105–1110.

[41] S. Kurt, H. U. Yildiz, M. Yigit, B. Tavli, and V. C. Gungor, "Packet size optimization in wireless sensor networks for smart grid applications," *IEEE Trans. Ind. Electron.*, vol. 64, no. 3, pp. 2392–2401, Mar. 2017.

TAPAS KUMAR MISHRA received the Ph.D. degree from IIT (ISM) Dhanbad, India, in 2017. He is currently working as an Assistant Professor with the Department of CSE, SRM University, Amaravati, India. His research interests include WSN, the IoT, network modeling, and AI.

KSHIRA SAGAR SAHOO (Member, IEEE) received the M.Tech. degree from IIT, Kharagpur, India, in 2014, and the Ph.D. degree from the Department of CSE, NIT, Rourkela, India, in 2019. He is currently working as a Postdoctoral Fellow with the Department of Computing Science, Umeå University, Umeå, Sweden. He has more than 70 research papers in top tier journals and conferences. His research interests include SDN, edge computing, the IoT, ML, and distributed computing. He is a member of IETE.

MUHAMMAD BILAL (Senior Member, IEEE) received the Ph.D. degree in information and communication network engineering from the School of Electronics and Telecommunications Research Institute (ETRI), Korea University of Science and Technology, in 2017. From 2017 to 2018, he was with Korea University, where he was a Postdoctoral Research Fellow with the Smart Quantum Communication Center. In 2018, he joined the Hankuk University of Foreign Studies, South Korea, where he is currently working as an Associate Professor with the Division of Computer and Electronic Systems Engineering. He is the author/coauthor of more than 100 articles published in renowned journals, including IEEE Transactions on Intelligent Transportation Systems, IEEE Internet of Things Journal, IEEE Systems Journal, IEEE Transactions on Industrial Informatics, IEEE Transactions on Network and Service Management, and IEEE Transactions on Network Science and Engineering, one book editorship, two book chapters, seven published proceedings articles, three issued U.S. patents, and six Korean patents. His research interests include network optimization, cyber security, the Internet of Things, vehicular networks, information-centric networking, digital twin, artificial intelligence, and cloud/fog computing. He has served as a Technical Program Committee Member for many international conferences, including the IEEE VTC, the IEEE ICC, ACM SigCom, and the IEEE CCNC. He is also an Editorial Board Member of IEEE Transactions on Intelligent Transportation Systems, IEEE Future Directions in Technology, Policy, and Ethics Newsletter, *Alexandria Engineering Journal* (Elsevier), *Computer Systems Science and Engineering*, *Intelligent Automation and Soft Computing*, *Frontiers in Communications and Networks*, *Frontiers in the Internet of Things*, and the Co-Editor-in-Chief of the *International Journal of Smart Vehicles and Smart Transportation*.

SAYED CHHATTAN SHAH (Senior Member, IEEE) received the M.S. degree in computer science from the National University of Computer and Emerging Sciences, in 2008, and the Ph.D. degree in computer science from Korea University, in 2012.

He held regular and visiting faculty positions with the Seoul National University of Science and Technology, Korea University, Dongguk University, Hamdard University, and Isra University. He is currently an Associate Professor of computer science with the Department of Information Communications Engineering, Hankuk University of Foreign Studies (HUFS), South Korea. He is also the Director of the Mobile Grid and Cloud Computing Laboratory. Before joining HUFS, he was a Senior Researcher with the Electronics and Telecommunications Research Institute, South Korea, and an Engineer with the National Engineering and Scientific Commission, Pakistan. He has also received funding for his research from numerous government agencies and foundations. His research interests include distributed computing systems and the Internet of Things. He is a member of the IEEE Communications Society, the IEEE Internet of Things Community, the IEEE Smart Cities Community, the International Telecommunication Union, and the European Research Council. He was the conference chair and on the program committees of various international conferences. He is also an Associate Editor of IEEE Access, *Information Processing Systems*, and *Intelligent Automation and Soft Computing*.

MANAS KUMAR MISHRA received the M.Tech. degree in computer science from Utakal University. He is currently working as an Assistant Professor with the Department of CSE, F. M. Autonomous College, Odisha. His research interests include WSN and distributed systems.