IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 December 2022, accepted 17 January 2023, date of publication 14 February 2023, date of current version 15 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3244888

RESEARCH ARTICLE

A Cross-Layer Solution for Contention Control to Enhance TCP Performance in Wireless Ad-Hoc Networks

NOOR MAST[1], SHAFIULLAH KHAN[1,2], M. IRFAN UDDIN[1], YAZEED YASIN GHADI[3], HEND KHALID ALKAHTANI[4], AND SAMIH M. MOSTAFA[5]

[1]Institute of Computing, Kohat University of Science and Technology, Kohat 26000, Pakistan
[2]Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huai'an 233003, China
[3]Department of Computer Science, Al Ain University, Al Ain, United Arab Emirates
[4]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh 11671, Saudi Arabia
[5]Computer Science Department, Faculty of Computers and Information, South Valley University, Qena 83523, Egypt

Corresponding authors: Samih M. Mostafa (samih_montser@sci.svu.edu.eg) and Noor Mast (noor.mast@kust.edu.pk)

Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R384), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

ABSTRACT With the development of wireless technology, users not only have wireless access to the Internet, but this has also sparked the emergence of Wireless Ad-hoc Networks (WANETs); this promising networking paradigm has the potential to adopt the shape of new emergent networks such as the Internet of Things (IoT), Vehicular Ad-hoc Networks (VANET) and Wireless Sensor Networks (WSN). However, channel contention (CC) is one of the key reasons why the TCP performs poorly in WANETs. This paper presents a mechanism called Cross-layer Solution for Contention Control (CSCC) to enhance TCP performance in WANETs. Each node starts marking packets in the proposed mechanism when its CC level reaches a certain threshold. As a result, the source node adjusts the congestion window (cwnd) size to a good state to control the insertion ratio of packets into the network. To provide a fair share to each flow, the flow having a large cwnd is penalized more. Numerous simulations have been conducted across several topologies to clarify the performance of the suggested mechanism. The simulation findings show that, in the presence of the Ad-hoc On-demand Distance Vector (AODV) routing and Dynamic Source Routing (DSR) protocols, the proposed CSCC mechanism outperformed TCP NewReno in terms of throughput and fairness. In comparison to TCP NewReno, the suggested mechanism has fewer retransmitted packets.

INDEX TERMS TCP, wireless ad-hoc networks (WANETs), channel contention (CC), congestion window (cwnd), CSCC, IEE802.11.

I. INTRODUCTION

Transmission Control Protocol(TCP) [1] is a reliable acknowledgment-based transport layer protocol initially designed for wired networks. When TCP was in its early days, it faced congestion problems, which led to the integration of congestion algorithms [2], [3] for further advancement. Due to its reliability, TCP is widely utilized in numerous Internet applications such as email, remote access, and file transfer. Moreover, according to reports, TCP is used to carry up to 90% of all internet traffic [4], [5], making it a vital protocol that still needs work to be improved.

However, with the development of wireless technology, users not only have wireless access to the Internet, but this has also led to the emergence of Wireless Ad-hoc Networks (WANETs). These networks are beneficial when the infrastructure may not exist or may be too expensive. WANETs are built up of wireless nodes (as depicted in Fig. 1), where nodes are connected wirelessly to each other without using any access point, and every node performs the roles of both a host and a router. These networks can be quickly deployed anywhere at any time. This alluring networking paradigm has the

The associate editor coordinating the review of this manuscript and approving it for publication was Jose Saldana.

potential to adopt the shape of new emerging networks such as the Internet of Things (IoT), Wireless Sensor Networks (WSN), Vehicular Ad-hoc Networks (VANET) and Tactical Wireless Networks [5], [6]. The IEEE 802.11 standard [7] is a de-facto standard for accessing medium in WANET [8].

However, wireless networks have unique properties (such as dynamic topology, a shared medium, and a medium prone to errors) compared to wired networks. Due to this mismatch, TCP experiences difficulties on wireless networks and performs poorly, notably in WANETs [9]. TCP must overcome these difficulties to efficiently use wireless technologies by connecting to the Internet or creating a WANET.

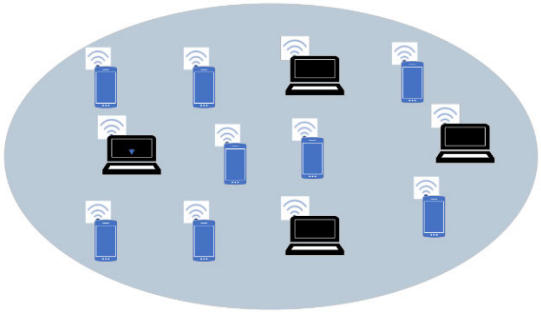

FIGURE 1. A scenario of a wireless Ad hoc network.

The research community works on various aspects to improve wireless networks' overall performance. For example, many routing protocols like [10], [11], [12], [13], [14], and [15] have been proposed to establish network paths effectively. But TCP's performance on wireless networks is still insufficient in the presence of such routing algorithms. Because there have been changes to the wireless technology's lower tiers of the communications stack without considering how those changes would affect the higher layers; therefore, wireless networks have a very different communication environment than cable ones. However, TCP fails to account for this shift and continues to operate as though a wireless network were wired. Furthermore, utilizing congestion algorithms in the event of losses not arising from network congestion is the primary problem for TCP in WANET. That is to say, TCP cannot tell the difference between losses due to congestion and losses due to the unique features of WANET [16], [17], [18].

For wired networks using TCP, buffer overflow is the most common cause of lost packets. But this assumption is false with WANETs because packet loss can be caused by channel contention (CC) or route failure [19]. Furthermore, if the WANET nodes can store more than ten packets in the buffer, then buffer overflow is rare. However, one of the most frequent causes of packet loss in WANETs is CC, resulting from the shared medium [20], [21]. Therefore, TCP must be aware of CC to respond accordingly and operate more effectively in WANETs.

To enhance TCP's functionality in WANETs, this paper presents a mechanism known as Cross-layer Solution for Contention Control (CSCC). In the proposed mechanism, packets are marked at each node once the CC at the MAC layer reaches a certain threshold. As a result, the source node adjusts the congestion window (cwnd) size to a good state to control the insertion ratio of packets into the network. To provide a fair share to each flow, the flow having a large cwnd is penalized more.

Here is how the rest of the paper is structured. Section II describes the Distributed Coordination Function of the IEEE 802.11 MAC protocol. Section III summarises the relevant literature. The proposed mechanism is described in detail, beginning with section IV. Results from simulation experiments of the suggested mechanism's performance are reported in Section V, and section VI offers the conclusion.

II. THE DISTRIBUTED COORDINATION FUNCTION

In the IEEE 802.11 MAC protocol, the Distributed Coordination Function (DCF) provides two mechanisms for accessing the medium. One of these mechanisms is the CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) used to share the medium among compatible devices, also known as the basic access mechanism. Before commencing a transmission in CSMA/CA, A node will check by sensing the medium that any other node within the range is sending data. The node transmits a frame if the medium is free for longer than the distributed inter-frame space (DIFS). Otherwise, the node defers transmission using the binary exponential backoff process to reduce the chance of packet collisions with packets sent by other nodes. The receiver sends an acknowledgement (ACK) to the sender when the frame is received. Otherwise, the sender schedules a retransmission of the frame. Additionally, the CSMA/CA algorithm requires that adjacent frame sequences have a minimum specified space between them.

On the other hand, the virtual carrier sensing method is an alternative to the basic access mechanism that calls for exchanging special RTS and CTS (Request to Send and Clear to Send) frames before transmitting actual data frames. With virtual carrier sensing, a sender will first send an RTS frame, and then, after a brief delay called short inter-frame space (SIFS), the receiving node will send a CTS frame in response. If the CTS frame is received after the RTS frame, the sender is free to send the data frame; otherwise, the transmission of the RTS frame is rescheduled.

When the medium is busy, both medium access mechanisms initiate the Binary Exponential Backoff (BEB) algorithm, where CW_{min} (minimum contention window) is the initial size of the contention window (CW) in the BEB algorithm. After that, the size of the CW is incremented exponentially for each unsuccessful transmission. However, the size of the CW cannot exceed the size of the CW_{max} (maximum contention window).

The IEEE 802.11 MAC protocol specifies that if the number of tries for transmitting a frame hits its maximum limit, then drop the frame and set CW to its minimum value. Also, reduce the value of CW to a minimum in the case of

successful transmission. Since the RTS and CTS frames in the virtual carrier sensing mechanism provide information about the time needed to transmit a frame. Any listening node can read this information and utilize it to update its NAV (Network Allocation Vector). Each node uses its NAV to estimate how long the medium will be busy and defers transmission accordingly.

From the above explanation, DCF utilized in non-QoS WLAN defines four components—Physical Carrier Sense, Virtual Carrier Sense, Random Back-off timings, and Inter-frame Spaces (IFS)—to guarantee that devices share the medium fairly.

DCF gives all users the same priority, so it is not considered the best mechanism for transmitting highly sensitive packets such as voice or video. Therefore, a mechanism was required to assign different priorities to different types of packets and allow higher-priority packets more access to the shared medium. It resulted in the development of Enhanced Distribution Coordination Access (EDCA), also called The Enhanced DCF.

EDCA assigns the highest priority to voice, followed by video, while best effort and background are placed in the third and fourth categories. So, for queueing packets, there are four Access Categories (ACs) in EDCA, and each AC has its respective queue and contention requirements.

The packets to be transmitted through EDCA faces two types of contention: internal contention among the ACs and external contention among nodes. While the packets to be transmitted through DCF only faces external contention. However, the proposed mechanism is not only in DCF; it can be adopted with EDCA.

III. RELATED WORK

The researchers have proposed various mechanisms for addressing the CC problem to improve TCP performance. We want to present a brief overview of these techniques here. One of these mechanisms is the Prioritized Packet Scheduling with Adaptive Backoff window (PPSAB) [22]. This mechanism calculates the retransmission probability (RP) based on the packet expiry time and the number of tries made for frame transmission. The frame with the shortest lifespan and the most transmission attempts is given the highest priority. Then each node modifies the CW using the RP and number of neighbor nodes. The weight components $\alpha 1$ and $\alpha 2$ are used to calculate $\mathrm{CW_{par}}$, and CW is then dynamically adjusted by equation (1).

$$\begin{aligned} \mathrm{CW_{par}} &= (\alpha 1 \times \text{Average\_Active\_Neighbors}) + (\alpha 2 \times \mathrm{RP}) \\ \mathrm{CW} &= \mathrm{CW_{max}} \times \mathrm{CW_{par}} \end{aligned} \tag{1}$$

A mechanism named Priority Contention Window Approach (PCWM) [23] is proposed by Chou et al. to tackle the CC problem. The MAC layer receives data from the network layer about the total and remaining hops in the PCWM mechanism. Then, using equation (2) provided, the value of CW is determined at each node along the path.

$$\begin{aligned} CW_{Max} &= 1024/2^{y}, \quad y = Max\,(0, (L - D - 5)) \\ CW_{Min} &= 1024/(2^{x} * 2^{(L-D)}), \quad x = Max\,(0, (5 - L)) \end{aligned}$$

L is the routing path's overall hop count, and D is the remaining hop counts in the Ad-hoc On-demand Distance Vector (AODV) routing table.

$$CW = CW_{Min} * 2^{n-1} \tag{2}$$

In equation (2) n is the number of attempts for data transmission, and $\mathrm{CW_{Min}} \leq \mathrm{CW} \leq \mathrm{CW_{Max}}$

A mechanism named Cooperative MAC protocol with Multi-Node Collision Avoidance (MNCA-CMAC) [24] is proposed by Shan Wu et al. to avoid collision among frames. This mechanism consists of three phases, i.e., (i) channel reservation phase, (ii) cooperative node selection phase and (iii) data transmission phase. The Cooperative RTS and Cooperative CTS (CCTS) packets are used for channel reservation. After receiving the CCTS packets, the sender awaits to receive the HTS (help-to-send) packets from the cooperating nodes, which identify the residual energy of a cooperative node. The sender waits for a predetermined interval to elapse or to receive a specified number of HTS packets. When either of these two conditions occurs, the sender sends an SEI (Selection End Indicator) packet to terminate the selection phase of the cooperative node. After that, the data packet is transmitted to the most cooperative node.

Some techniques are focused on generating a delayed/proxy ACK to minimize CC. Proxy Acknowledgement (PACK) [25] is one such mechanism; in this mechanism, a proxy node is nominated if the number of hops on the path surpasses a predefined threshold. The proxy node identifies missing packets and informs the source node by sending an ACK packet. As a result, the source node will retransmit the lost packets without waiting for a retransmission timeout. One of the delay ACK mechanisms is the TCP ACK Delay Window (TCP-ADW) [26]; this mechanism looks at the channel situation to determine the number of ACKs for increasing/decreasing the delay window. The receiver must provide an ACK and set the count variable to zero when the sum of all received packets reaches the delay window. If an out-of-order packet arrives, then immediately provide an ACK or if a packet fills a gap in the receiver's buffer.

Altman et al. proposed a mechanism called the Dynamic Delayed ACK (DDA) [27] based on RFC1122 [28] also belongs to the category of delay ACK. After receiving **d** packets (where **d**=2), RFC 1122 specifies a standard for sending an ACK. However, send an ACK if **d** packets are not received within a specific time. The value of **d** in DDA can be between 1 and 4. Initially, DDA creates an ACK on the arrival of each packet and then increases this number to four (**d**=4) based on the sequence number of a segment. Under this strategy, once **d** achieves the value of four, there is no way to bring it back down. To improve this idea, even more, a mechanism called TCP-DAA (TCP Dynamic Adaptive ACK) [29] is suggested.

This mechanism considers the channel situation for sending an ACK; if the channel is good, send an ACK after four packets or two packets. However, immediately send an ACK if an out-of-order packet arrives or a packet fills the gap in the receiver's buffer. In standard TCP, three duplicate ACKs are required for fast retransmission, whereas in TCP-DAA, two are needed.

The author says in [17] that TCP-DCA (TCP with Delayed Cumulative ACK) is a technique that aims to determine the delay windows based on the hop count. The ACK might be suspended for an entire cwnd using TCP-DCA if the number of hops on the path is less or equal to three. On paths with a hop count of more than three but less or equal to nine, the receiver delivers an ACK after five packets. However, send an ACK after three packets in the case of more than nine hops. In contrast, TCP-ADA (TCP with Adaptive Delayed Acknowledgement) [30] is a technique; according to this approach, to avoid contention and collision, the best solution is to generate an ACK for a single cwnd. In the Contention-based Path Selection (COPAS) [31] mechanism, a different approach has been adopted than delay ACK techniques. The COPAS uses different routes for forwarding the data and ACK packets. After that, continuously monitors the paths for contention. A less-contended route is selected as soon as the traffic on a path exceeds a certain level.

The NRED (Neighbourhood Random Early Detection) [32] technique, which is based on [33], is proposed by Xu et al. to alleviate the effect of unfairness in WANETs. In the NRED algorithm, a distributed neighborhood queue strategy is adopted, in which all neighbor nodes' queues are aggregated so that every node holds a piece of the distributed queue. To determine the size of the distributed queue, each node monitors channel utilization; the packet dropping/marking probability is determined based on channel utilisation.

Fu et al. have proposed a mechanism called Link layer RED+AP (LRED+AP) [20]; this mechanism selects two thresholds (i.e., maximum and minimum) to manage CC. The packets are dropped at the maximum threshold and added extra time to back-off time at the minimum threshold level. The additional time is equal to the transmission time of the preceding packet. Thus, the spare time depends on the size of the last transmitted packet.

Cross-layer congestion control (C^3TCP) [34] has been proposed to deal with network congestion to improve TCP performance. This mechanism minimizes the data injected into the network for congestion avoidance. Therefore, the bandwidth and the delay experienced by each link are evaluated at each node. The collected data are inserted into the MAC header's option field. After collecting the bandwidth and link delay information at the first node, the next node compares its bandwidth with the bandwidth of the previous node and chooses the smallest one. However, the delay on the current link will be added to the previous delay. The process mentioned above is repeated on each intermediate node. As a result, when the destination node receives a data packet, it contains the minimum available bandwidth on the path. It will also include the link delay information for the entire route. After that, the bandwidth and link delay information is communicated to the source nod in the ACK packet for transmission rate adjustment.

The Wireless Contention Control Protocol (WCCP) [19] uses channel busyness to identify the network utilization and congestion status. Moreover, it allocates resources to a flow based on available bandwidth. WCCP replaces the TCP's window technique with a rate-based algorithm. As a result, WCCP introduces two modules: one at the transport layer and the second between the network and MAC layer, to check and, if required, alter the value of the feedback field in the TCP's packet. The source node adjusts its transmission rate based on the value of the feedback field.

Hamadani proposed a solution to address the problem of intra-flow instability called TCP ConTention Control (TCTC) [35]. The leading cause of intra-flow instability, according to [35], is transmitting more data to the network. In the proposed solution, the destination node monitors the amount of throughput achieved for a fixed interval of time and the level of contention in this interval. The receiver node decides the appropriate amount of data to be emitted by the sender node based on the information collected within the fixed interval to achieve high throughput and reduce the delay on each connection. After presenting a summary of the proposals suggested by the research community, the following section offers the proposed CSCC mechanism.

TABLE 1. presents a summary of the proposals discussed as related work. Several proposals are based on delayed ACKs, and all these proposals have a limitation in common: excessive ACK delays can upset TCP's round-trip time and packet clocking algorithms. In contrast, mechanisms such as PACK violate the end-to-end connection handling mechanism of TCP. Moreover, failure of the node responsible for proxy ACK may lead to more poor performance if the destination node demands retransmission.

In PPSAB, estimating active nodes is challenging, and the wrong estimation can affect the algorithm decision. While in MNCA-CMAC, after the CCTS packets, the exchange of the HTS packets is also required to determine the cooperative nodes. So, it is an extra burden on the shared medium. On the other hand, in WANET, the probability of route failure increases with high mobility, and the routing's overhead will also increase. Therefore, COPAS, which maintains two routes, is unsuitable in a high-mobility environment.

Moreover, the extra delay at the MAC layer in the case of LRED+AP is also a reason for low throughput and retransmission may occur. None of those mentioned above mechanisms enable TCP to distinguish that the network is congested or has high contention. However, the proposed mechanism sends a contention notification to the sender that the network has contended, and the sender can take appropriate action in the case of contention..

TABLE 1. A summary of the related work.

Mechanism	Layers Involved	Problem to Handle	Routing Protocol used	Brief Description	Simulation Environment
PPSAB	N & MAC	Channel contention	Not mentioned	Adjust the CW based on the number of active nodes and the retransmission probability (RP).	Mobile
PCWM	N & MAC	Channel contention	AODV	To adjust CW of a node based on the total hop count and remaining hop count in the routing.	Mobile
MNCA-CMAC	MAC	Channel contention	Not mentioned	Calculates the contention waiting time based on the network's probability of channel gains.	Static
PACK	T	Channel contention	AODV	A proxy node is nominated if the number of hops on the path surpasses a predefined threshold.	Mobile
DDA	T	Channel contention	AODV	Suggests the DACK for one to four packets based on the sequence number.	Static
TCP-ADW	T	Channel contention	AODV	To estimate the channel situation and set the delay window based on cwnd, packet loss and packets' inter-arrival times.	Mobile
TCP-DAA	T	To minimize retransmission due to contention	Not mentioned	Delay ACK for a maximum of 4 packets, based on the inter-arrival time of packets to estimate channel condition.	Static
TCP-DCA	T	Channel contention	AODV and DSR	To delay ACK based on hop count.	Mobile
TCP-ADA	T	Channel contention	DSR	Suggested one ACK for the whole cwnd.	Mobile
COPAS	N	Channel contention	DSR	Forward Data and ACK packets on different routes based on Backoff time.	Static
NRED	MAC	Unfairness	AODV	Extends the RED concept to the distributed neighborhood queue, and packet dropping probability is based on channel usage.	Static
LRED+AP	MAC	Channel contention	Manual routing	Specifies two thresholds; packets are dropped at the maximum threshold and add a delay at the minimum threshold.	Static
C^3TCP)	T & MAC	Congestion	Not mentioned	Adjusting transmission rate based on bandwidth and delay	Static
WCCP	T & MAC	Unfairness due to contention	AODV	Adjust the sending rate based on Channel Usage. It replaces the window algorithm with a rate control algorithm.	Static
TCTC	T & MAC	Congestion	DSR	Adjust the data rate based on the achieved throughput and contention delay experienced by packets in a specific interval.	Static

IV. PROPOSED SOLUTION

In the proposed mechanism, each node counts the number of attempts to access the medium for transmission at the MAC layer. After that, each node calculates the Weighted Moving Average (WMA) of the number of tries (as explained in subsection A of Section IV) to estimate the CC and react accordingly. When the WMA reaches a specific threshold CC_{Thresh} (Channel Contention Threshold), the MAC layer will set the CC status ON. As a result, the node will start marking packets to inform the TCP's source node about contention (Subsection B of Section IV describes how to notify the source node). On receipt of CC notification, the source node adjusts its transmission rate to control contention.

Marking packets is more effective than dropping packets. Because when the MAC layer fails to transmit a frame, it is dropped and wrongly notifies the network layer that the path is unavailable. The network layer then initiates an unnecessary route recovery process [36]. The proposed mechanism attempts to control packet drop due to CC to save the time

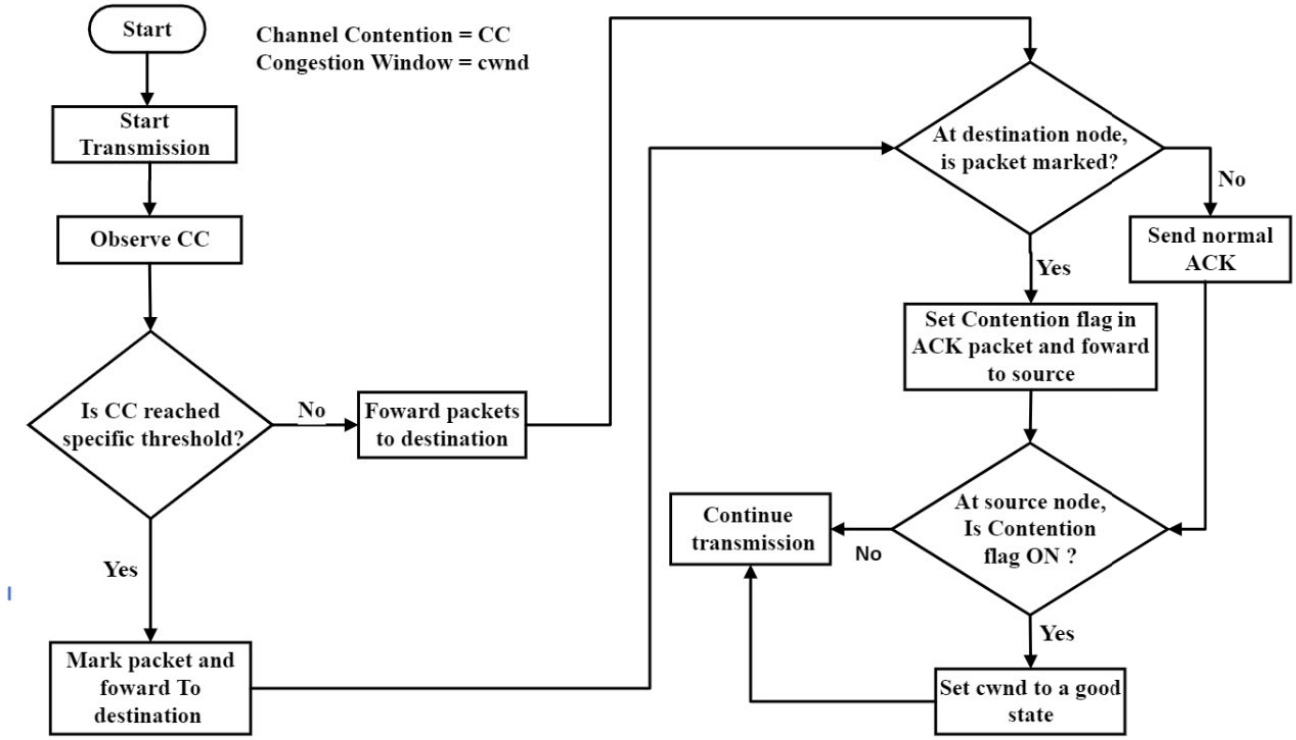

FIGURE 2. Flowchart of a cross-layer solution for contention control (CSCC) to Enhance TCP performance in WANET.

TABLE 2. Computing the weighted moving average (WMA).

Packets Transmission Sequence	Node A		Node B	
	WMA	Transmission Attempts	WMA	Transmission Attempts
	Initially A_{WMA}=0		Initially BWMA=0	
1	A_{WMA}= (0.55 x 1) + (1-0.55) x 0 = 0.55 + 0 = **0.55**	1	B_{WMA}= (0.55 x 2) + (1-0.55) x 0 =1.1 + 0 =**1.1**	2
2	A_{WMA}= (0.55 x 2) + (1-0.55) x0.55 =1.1 + 0.2475 =**1.3475**	2	B_{WMA}= (0.55 x 3) + (1-0.55) x 1.1 =1.65 +0.495 =**2.145**	3
3	A_{WMA}= (0.55 x 1) + (1-0.55) x1.3475 = 0.55 + 606375 =**1.156375**	1	B_{WMA}= (0.55 x 3) + (1-0.55) x 2.145 =1.65 + 0.96525 =**2.61525**	3
4	**2.1704**	3	**2.2769**	2
5	**2.0767**	2	**2.1246**	2
6	**1.4845**	1	**1.5061**	1

the network layer searches for a new path due to a wrong notification of route failure.

The problem of contention and congestion occurs because of the greedy behaviour of TCP. However, in WANETs, congestion control is often turned on due to MAC layer losses and not buffer overflow. The proposed mechanism enables the TCP to distinguish congestion and contention losses and react accordingly. The CC leads to the problem of unfairness as well. So, in the case of CC, it makes sense to impose a higher penalty on flows with a larger cwnd. Therefore, the proposed mechanism adjusts the cwnd size to a good state (good state is explained in subsection C of section IV) to make fair and efficient use of channel resources. Fig. 2 shows the flowchart of the suggested mechanism.

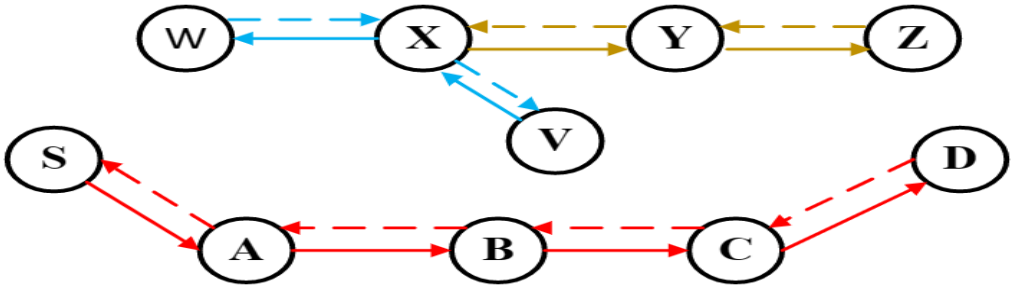

FIGURE 3. A ten nodes scenario of WANET.

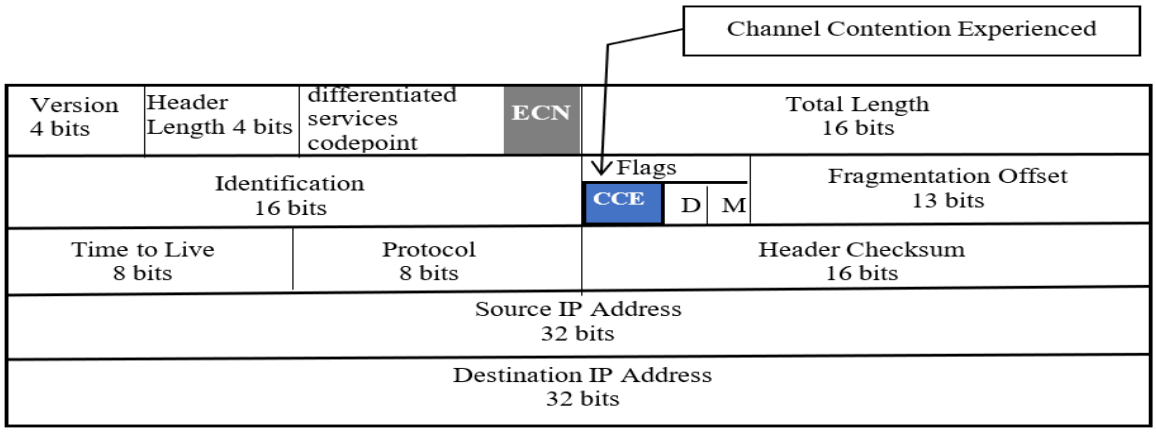

FIGURE 4. IP header with suggested modification.

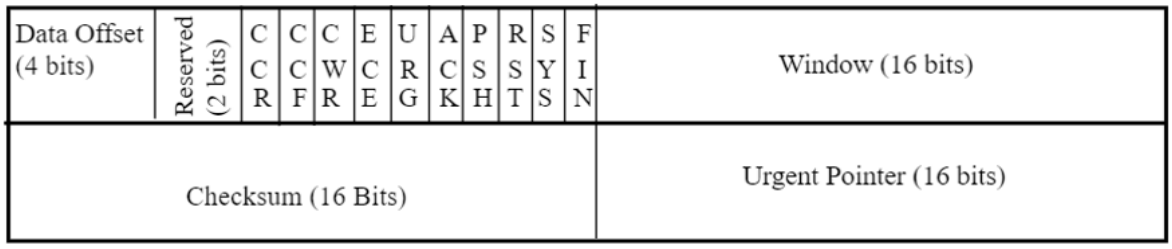

FIGURE 5. A portion of the TCP header with suggested modification.

A. COMPUTING THE WEIGHTED MOVING AVERAGE (WMA)

Whereas accessing the medium for transmission, the increase/decrease in the number of attempts means the node has potentially identified an increase/decrease in the CC. Therefore, to estimate the CC, each node obtains the WMA of the number of tries made to transmit a frame. Suppose the WMA is denoted by Å. Suppose again that R_{Att} is the number of attempts made by a node N transmitting a frame. Then at the end of every successful/unsuccessful transmission, a WMA is computed according to equation (3) [37] to reflect increases/decreases in the contention.

$$\left\{\begin{array}{l} \text{Å}_{n+1} = \alpha R_{att} + (1-\alpha)\,\text{Å}_n, 0 < \alpha < 1 \\ \qquad\qquad if\ transmission\ is\ successful \\ \text{Å} = CC_{Thresh}\ Otherwise \end{array}\right\} \quad (3)$$

The value of α is constant and must be chosen very carefully; the value for α must be selected such that it does not reveal contention early. Otherwise, TCP will reduce the cwnd size unnecessarily. On the other hand, conflict reflection would not even need to be long enough to allow cwnd to grow to a larger size. Both cases lead to the poor performance of the network.

When $\text{Å} \geq \text{CC}_{\text{Thresh}}$, the MAC layer sets the contention status ON. After that, the concerned node starts to mark packets to inform the source node about the medium contention. On receiving the contention notification, the source node adjusts its cwnd size to a good state, as explained in Subsection C of Section IV. Whereas algorithm 1 shows how to observe and set the CC status.

To illustrate the WMA computing process with an example, see Fig. 3, where the solid line represents the direction

of data packets, such that source S is sending data packets to destination D. Similarly dotted line represents the direction of ACK packets between the endpoints. So, in Fig. 3, three data flows are employed to transfer data packets; one flow is from S to D, the second is from V to W, and the third is from X to Z.

On the path S-A-B-C-D, the A_{WMA} and B_{WMA} denote the WMAs at nodes A and B, respectively. The initial value of WMA at each node is zero, as shown in TABLE 2, and the weight factor (α) value is 0.55. Suppose, at first-time, Node A after one attempt and Node B after two attempts transmit a frame. Furthermore, how the WMA will be computed at nodes A and B, in this case, is shown in TABLE 2. The computed values for AWMA and BWMA are 0.55 and 1.1, respectively. It is also shown in TABLE 2 how WMA will be calculated at Nodes A and B after transmitting the second and third packets. The computed WMA in the case of the fourth, fifth and sixth packets is also listed. Looking at TABLE 2, after transmitting the third packet, the WMA value at Node B is 2.61525, suppose it is greater than the CCThresh. So, the MAC layer of Node B will notify that the channel has contended. As a result, the network layer of Node B will start packet marking to inform the source node.

Algorithm 1 Observing Channel Contention

Initialization
{
    Å = 0
    R_{Att} = 0
}
Counts Number of R_{Att}
IF (Transmission successful == True)
    Å = $(1-\alpha)$ $Å_{Retry} + \alpha R_{Att}$
Else
    Å = CC_{Thresh}
Endif
IF(Å $\geq$ CC_{Thresh})
    Contention Status ON
Else
    Contention Status OFF
Endif

B. CHANNEL CONTENTION NOTIFICATION

Changes have been suggested in the IP (Internet Protocol) header for informing the source node about contention on the path. The IP header has a reserved field; the proposed mechanism uses this field to mark packets. Suppose the name of this field is CCE (Channel Contention Experienced), as shown in Fig. 4. When the MAC layer notifies that the contention has occurred, the network layer starts marking packets using the CCE field, as given in algorithm 2.

As clear from the literature, the ECN [38] mechanism has been proposed to inform the source node about the congestion or queue status. The ECN mechanism uses the ECN field in the IP header to mark packets in the case of congestion, as shown in Fig. 4. So, the proposed mechanism and ECN mechanisms can be implemented together. As a result, the TCP's source will be able to differentiate between congestion and CC losses and react accordingly.

In the TCP header, there are eight control bits. The suggested mechanism introduces two new control bits called CCF (Channel Contention Flag) and CCR (Channel Contention Responded), as shown in Fig. 5. When a packet arrives at the destination node with the CCE field ON. The destination node sets the value of the CCF field to one in the ACK packet to inform the source node about contention. On receiving the ACK packet with the CCF field ON, the response of the source node is explained in subsection C of section IV.

Algorithm 2 Marking Packets to Inform Source Node

//At Intermediate Node
IF (Contention Status ON)
    Set CCE = 1
Endif
//At Destination Node
Sending ACK
IF (CCE == 1)
    IN ACK Header Set CCF=1
Endif

C. RESPONSE OF SOURCE NODE TO MARKED PACKETS

To control contention and provide fairness among data flows, the source node adjusts the size of the cwnd to a good state when receiving an ACK packet where the CCF field has one value. To explain a good state, suppose there is a TCP flow with an initial cwnd size of one. Its size hits 128 after some time without receiving a channel contention notification. Now all sizes of cwnd that falls below 128 are good states. Suppose again, when TCP's cwnd size is 128, the source node receives a channel contention notification. Then cwnd size will adopt a value below 128 (in this case, all values below 128 represent good states) according to algorithm 3. Furthermore, a flow with a cwnd size is less than the slow start threshold (ssthresh) is considered a flow with a small cwnd; otherwise, it is a flow with a large cwnd. Algorithm 3 shows how to adopt a good state and sets the value of the CCR field to one to inform the destination node that the cwnd has been reduced. Moreover, if the TCP source receives a CCF notification in a good state before the expiry of one round trip time, the TCP source should ignore the succeeding CCF.

D. SELECTION OF VALUE FOR ALPHA (α)

The WMA given by equation (3) is a recursive function, and one can write it in terms of older weights, as provided by equation (4). Expanding equation (4) to its older value will continue until it reaches the base term $Å_0$. So, the recursive property of WMA implies that it calculates the value of the current state using the prior observation. The only choice a WMA user must make is the parameter alpha (α) selection,

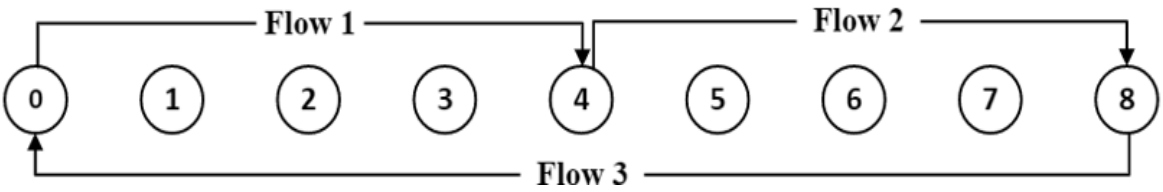

FIGURE 6. String topology of 9 nodes and three flows of TCP.

TABLE 3. Results achieved on a string topology of 9 nodes.

Values assigned to alpha (α).	Throughput (Flow 1)	Throughput (Flow 2)	Throughput (Flow 3)	Total Throughput (Kbps)	Fairness Index
0.40	130.51	159.43	25.17	315.11	0.768
0.45	121.11	168.73	43.15	332.99	0.821
0.50	140.67	176.56	37.31	354.54	0.800
0.55	123.63	169.66	63.70	356.99	0.883
0.60	119.41	160.65	34.27	314.33	0.799

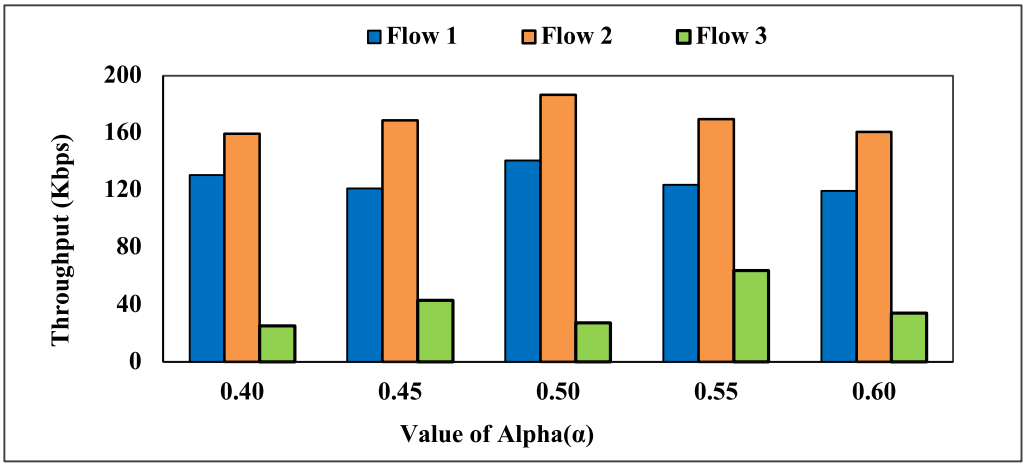

FIGURE 7. The throughput achieved on the nine nodes string.

which determines how significant the recent observation is in the WMA's computation.

Some simulation experiments have been conducted to determine the value of alpha (α) for efficient utilization of the network resources. Therefore, the performance of the proposed mechanism was analyzed in the string topology of 9 nodes depicted in Fig. 6. The values assigned to alpha(α) are 0.40, 0.45, 0.50, 0.55 and 0.60. The number of TCP flows was 3, each with a payload of 1460 bytes.

$$\begin{cases} \text{Å}_{n+1} = \alpha R_{att} + (1-\alpha)\,\text{Å}_n, \\ \quad 0 < \alpha < 1 \\ \textit{if transmission is successful} \\ \text{Å}_{n+1} = \alpha R_{att} + (1-\alpha) \\ \quad * \begin{pmatrix} \alpha R_{att-1} + \\ (1-\alpha)\,\text{Å}_{n-1} \end{pmatrix} \\ \text{Å}_{n+1} = \alpha R_{att} + (1-\alpha) \\ \quad *(\alpha R_{att-1} + (1-\alpha) \\ * \left(\alpha R_{att-2} + (1-\alpha)\,\text{Å}_{n-2}\right) \end{cases} \tag{4}$$

Conducting the simulation experiments, the throughput achieved by the proposed mechanism is shown in Fig. 7, and Fig. 8 illustrates the fairness indexes achieved in each case. For further detail, look at TABLE 3.

Looking at the results illustrated in Fig. 7 and Fig. 8 and listed in TABLE 3, high throughput and more fairness have been obtained by assigning a value of 0.55 to alpha(α). The closest results were achieved when a value of 0.50 was assigned to alpha(α); however, the best results were achieved when a weight of 0.55 was used for alpha. Therefore, during further simulation experiments, the value of 0.55 was used.

V. PERFORMANCE EVALUATION

Using the network simulator NS2.35 [39], multi-hop wireless simulation experiments were conducted to verify the proposed mechanism's performance against TCP NewReno. However, implementing the proposed algorithm, the most modified files are tcp.h, tcp.cc and tcp-newreno.cc at the transport layer. While at the MAC layer, the files called mac-802.h and mac-802.cc were modified to measure the channel usage and declare whether the channel has contended. During simulation experiments, in each scenario, each node's transmission range and sensing ranges were 250 and 550 meters,

Algorithm 3 Adjusting the Size of Cwnd to a Good State

// on Receipt of ACK packet
ACK packet received
IF (CCF is ON)
  IF ($cwnd \leq ssThresh$)
    IF ($cwnd \leq \frac{1}{2}ssThresh$)
      $cwnd = cwnd$
    Else
      $cwnd = \frac{3}{4}ssThresh$
    Endif
  Else IF ($cwnd > ssthresh$)
    IF ($\frac{cwnd}{2} \leq ssThresh$)
      $cwnd = cwnd/2$
    Else
      $cwnd = ssThresh$
    Endif
  Endif
  Set CCR = 1
Endif

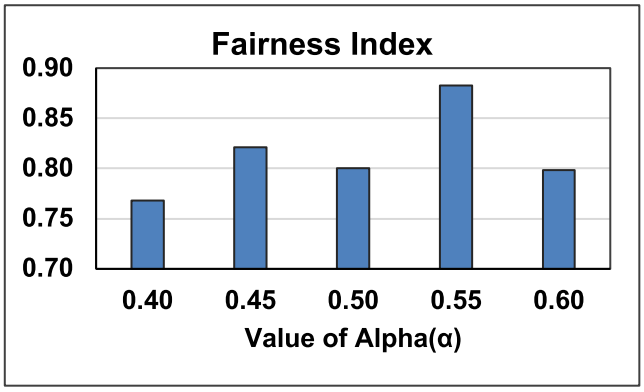

FIGURE 8. The fairness indexes achieved with different values of alpha.

respectively, and the data transfer rate was assumed to be 2Mbps. TCP packet in each case had a size of 1460 bytes. For each scenario, the simulation lasted 300 seconds. Each scenario's results are based on an average of 15 runs.

String topology was considered during the simulation to determine the effect of the increasing number of hops. Then a grid topology and a more realistic random topology were evaluated with a growing number of flows. Throughput and flow fairness criteria were chosen for the performance study, and simulation tests were conducted with 95% confidence. The quantity of retransmitted packets is also used as a performance indicator. TCP retransmits packets for two reasons: (i) when any packet loss is detected or (ii) when a retransmission timeout occurs. As a result, if an algorithm has a low number of retransmissions, it also has a low number of retransmission timeouts and dropped packets.

The AODV [11] and Dynamic Source Routing (DSR) [10] routing protocols were employed to establish the routes. DSR and AODV are on-demand routing protocols, i.e., a path is kept around for as long as it is essential. The DSR uses source routing in which the sender of a packet determines the complete sequence of the nodes through which the packet must pass. But in AODV, each node has a routing table that it uses to decide where to forward packets. TABLE 5 provides a detailed description of the simulation parameters used for the experiments.

TABLE 4. Values of parameters used in the simulation.

Parameters	Value
Simulation Time	300 Second
Topologies and number of nodes	A string of 16 nodes
	13x13 Grid
	Random topology in a 1000 x 1000 meters area with 100 nodes
Routing protocols	AODV and DSR
Transmission Range	250m
Data rate	2Mbps
Queue size	20 packets
Packet size	1460 Bytes
Slot time	20 µs
SIFS	10 µs

TABLE 5. Confidence intervals computed for throughput on a string topology with AODV.

	TCP NewReno			CSCC		
Number of Hops	Throughput	95% Confidence Interval		Throughput	95% Confidence Interval	
		Lower Bound	Upper Bound		Lower Bound	Upper Bound
3	327.59	325.92	329.26	363.58	361.81	365.35
4	247.01	245.71	248.31	283.60	282.73	284.46
5	205.45	203.93	206.98	241.66	240.87	242.46
6	180.01	178.51	181.51	220.25	219.02	221.47
7	155.45	146.29	164.60	201.60	196.87	206.34
8	139.88	132.22	147.54	188.20	178.29	198.11
9	131.53	126.22	136.84	177.77	164.67	190.87
10	122.33	117.08	127.58	168.01	162.17	173.85
11	116.07	109.23	122.91	160.52	150.65	170.39
12	106.99	99.87	114.11	153.79	148.03	159.56
13	99.53	91.96	107.10	147.80	138.51	157.09
14	93.80	88.20	99.40	142.46	134.55	150.36
15	89.11	82.13	96.09	139.40	133.84	144.96

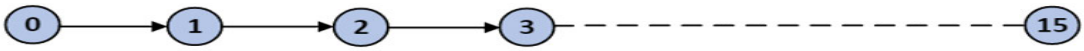

FIGURE 9. String topology of 16 nodes.

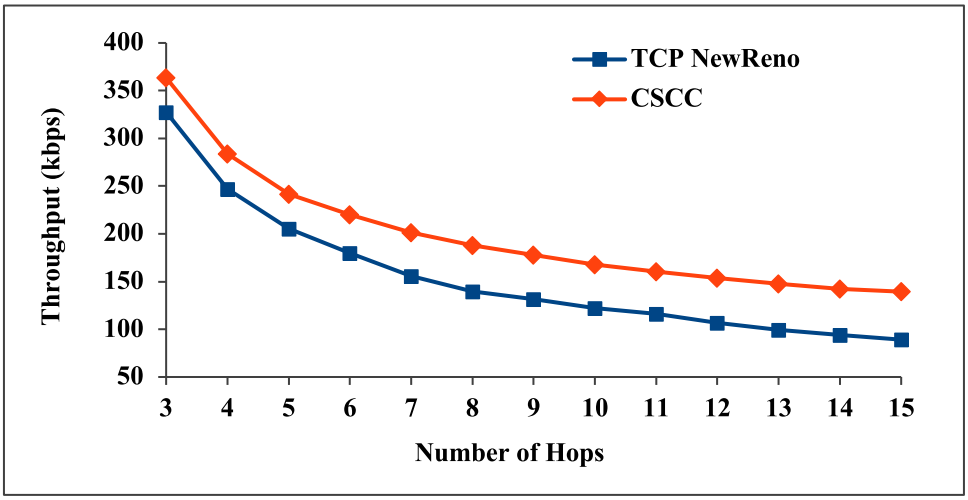

FIGURE 10. Throughput achieved in the string topology with AODV.

TABLE 6. Confidence intervals computed for throughput on a string topology with DSR.

Number of Hops	TCP NewReno			CSCC		
	Throughput	95% Confidence Interval		Throughput	95% Confidence Interval	
		Lower Bound	Upper Bound		Lower Bound	Upper Bound
3	230.79	227.00	234.58	259.83	256.41	263.25
4	195.33	191.95	198.71	227.97	221.70	234.23
5	166.82	164.54	169.10	201.54	196.51	206.58
6	142.13	138.93	145.32	176.72	172.75	180.68
7	128.48	123.17	133.78	164.84	159.67	170.01
8	119.47	115.85	123.10	161.61	158.07	165.16
9	114.47	111.35	117.59	159.51	153.72	165.31
10	112.26	108.67	115.86	158.86	155.91	161.81
11	107.98	104.91	111.05	155.74	152.29	159.19
12	105.08	100.75	109.40	150.70	147.59	153.80
13	102.61	99.30	105.92	151.84	147.66	156.03
14	98.46	96.35	100.58	145.90	143.19	148.61
15	92.38	89.61	95.15	142.92	138.82	147.03

A. STRING TOPOLOGY

To analyze how an increasing number of hops affects the performance of the proposed mechanism, the multi-hop simulations were performed in a string topology of 16 nodes. The path length of a minimum of three and a maximum of 15 hops was considered. The distance between the adjacent nodes was set at 200 meters. The graphical representation of this topology is shown in Fig. 9. In the first case considered, a connection has been established between node 0 and node 3 to transfer the data, where node 0 and node 3 act as the source and destination nodes, respectively, which are not in the direct transmission range of each other.

In the second case, node 0 and node 4 are considered to act as the source and destination nodes, respectively, whereas nodes 1, 2 and 3 are intermediate nodes that forward packets between node 0 and node 4. This way, the transmission between node 0 and node 5, then node 6 and 7 up to node 15, was considered.

In each case considered for the string topology, the throughput achieved with TCP NewReno and the CSCC mechanism is depicted in Fig. 10 and Fig. 11; it is clear from these figures that the performance of the CSCC mechanism is more satisfactory than that of TCP NewReno in terms of throughput. The suggested mechanism achieved high throughput over TCP NewReno as the number of hops increased, ranging from 10.99% to 56.43% and a 12.58% to 54.71% in the presence of AODV and DSR, respectively. The 95% confidence intervals computed for the achieved throughput for the string topology cases considered with AODV and DSR are given in Tables 6 and 7, respectively.

It is clear from Fig. 12 and Fig. 13, illustrating the number of retransmitted packets, that the CSCC mechanism is transmitting fewer packets than TCP NewReno and achieving high throughput because there is lower contention on the channel. A reduced number of retransmissions

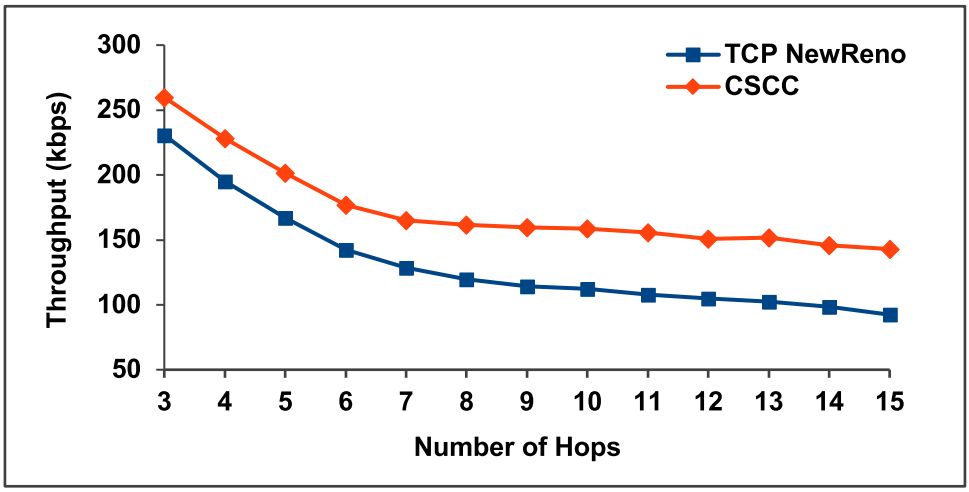

FIGURE 11. Throughput achieved in the string topology with DSR.

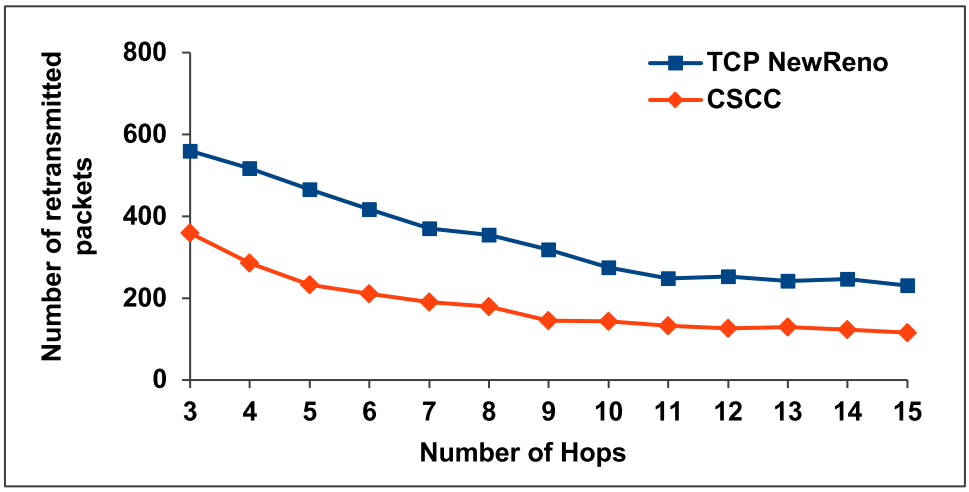

FIGURE 12. The number of retransmitted packets in the string topology with AODV.

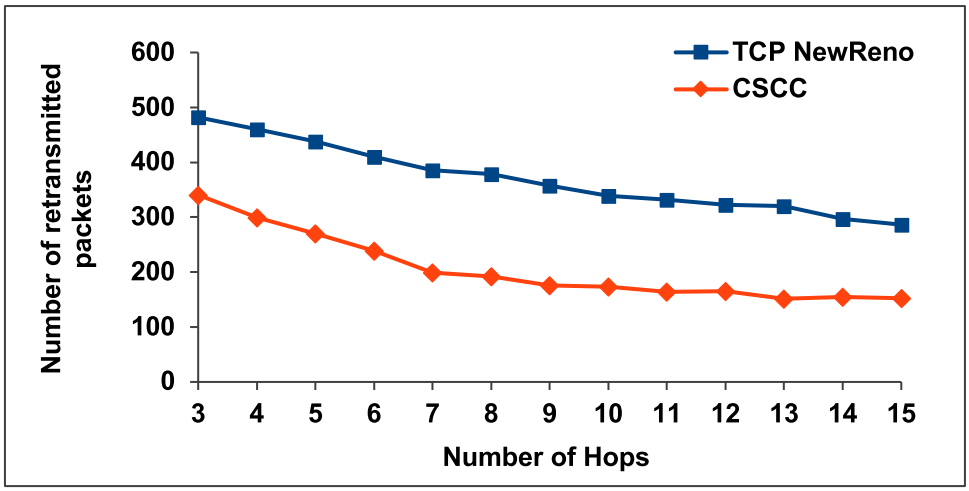

FIGURE 13. The number of retransmitted packets in the string topology with DSR.

means an improved utilization of network resources. Thus, the proposed mechanism handles contention more efficiently.

B. GRID TOPOLOGY

This subsection reports the results of the simulation analysis of the CSCC mechanism on a grid topology against TCP

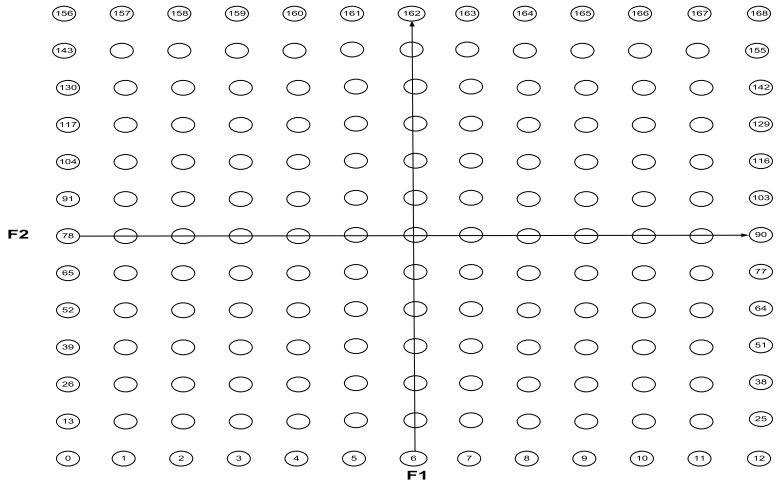

FIGURE 14. 13 × 13 grid with two flows.

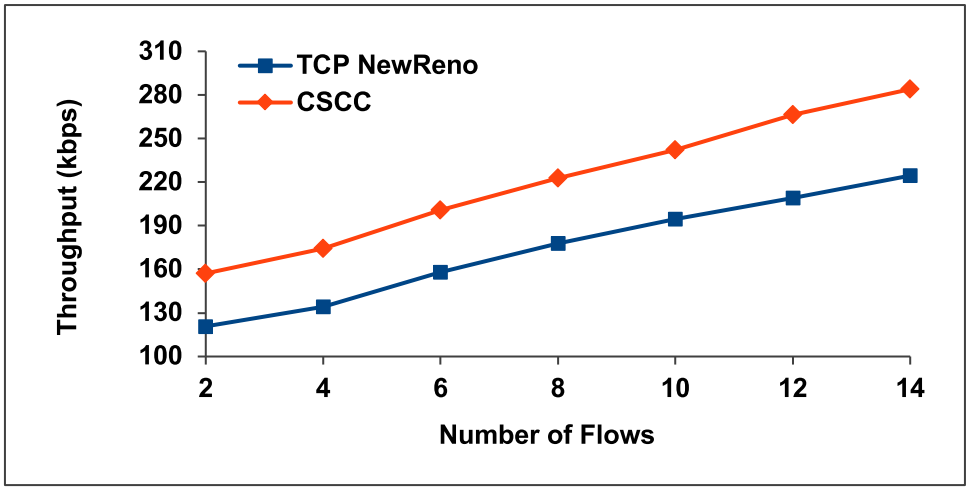

FIGURE 15. Throughput achieved on the 13 × 13 grid topology with AODV.

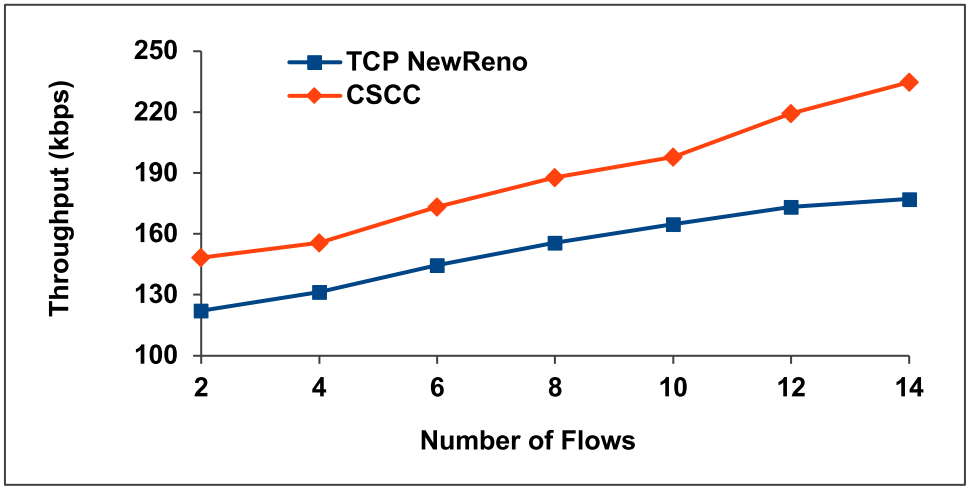

FIGURE 16. Throughput achieved on the 13 × 13 grid topology with DSR.

NewReno. A total of 169 nodes were simulated and placed in a 13 × 13 grid, depicted in Fig. 14. The distance between the adjacent nodes, as was the case in the previously considered scenarios, was 200 meters. Compared to the string topology

TABLE 7. Confidence intervals and fairness indexes for the 13 × 13 grid topology, where the routing protocol is AODV.

	TCP NewReno				CSCC			
Number of flows	Throughput	95% Confidence Interval		Fairness Index	Throughput	95% Confidence Interval		Fairness Index
		Lower Bound	Upper Bound			Lower Bound	Upper Bound	
2	120.75	116.49	125.01	0.990	157.18	152.52	161.84	0.994
4	134.14	129.64	138.63	0.697	174.28	170.81	177.74	0.995
6	158.07	153.38	162.75	0.748	200.83	197.19	204.47	0.990
8	177.76	173.94	181.58	0.670	222.84	218.60	227.08	0.990
10	194.62	190.54	198.70	0.658	241.97	238.46	245.48	0.970
12	209.16	205.92	212.40	0.693	266.55	260.99	272.10	0.897
14	224.47	220.41	228.53	0.661	283.96	277.76	290.15	0.890

TABLE 8. Confidence intervals and fairness indexes for the 13 × 13 grid topology, where the routing protocol is DSR.

	TCP NewReno				CSCC			
Number of flows	Throughput	95% Confidence Interval		Fairness Index	Throughput	95% Confidence Interval		Fairness Index
		Lower Bound	Upper Bound			Lower Bound	Upper Bound	
2	122.19	119.51	124.87	0.830	148.34	145.38	151.31	0.996
4	131.43	129.27	133.59	0.797	155.53	152.65	158.40	0.998
6	144.50	141.73	147.28	0.819	173.29	170.52	176.07	0.991
8	155.52	150.15	160.89	0.835	187.93	182.46	193.40	0.916
10	164.67	159.13	170.20	0.910	197.99	190.99	204.99	0.970
12	173.29	168.36	178.22	0.884	219.50	214.86	224.14	0.929
14	177.27	167.59	186.94	0.935	234.74	226.93	242.55	0.971

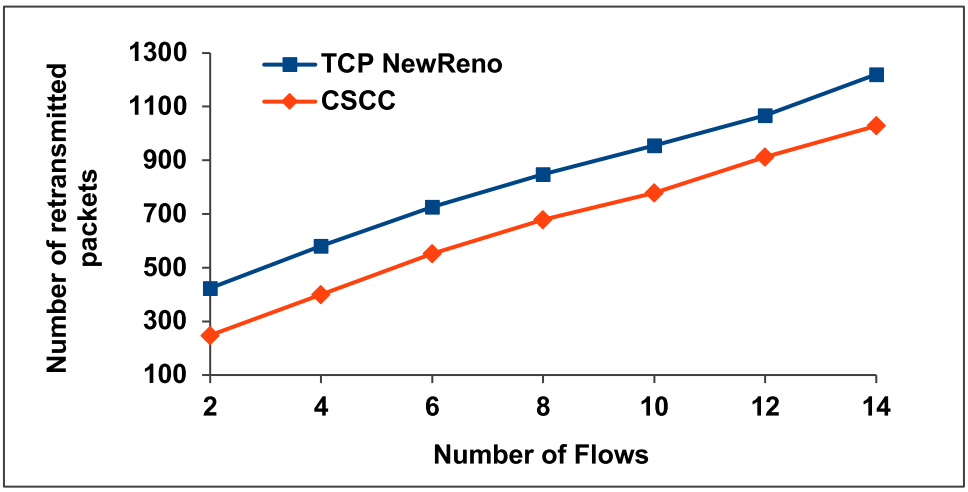

FIGURE 17. The number of retransmitted packets on the 13 × 13 grid topology with AODV.

considered in previous subsections, the grid topology has more nodes, and more data flows are considered to create a highly contended environment. At the start, two flows (F1 and F2) were considered so that the flows cross each other

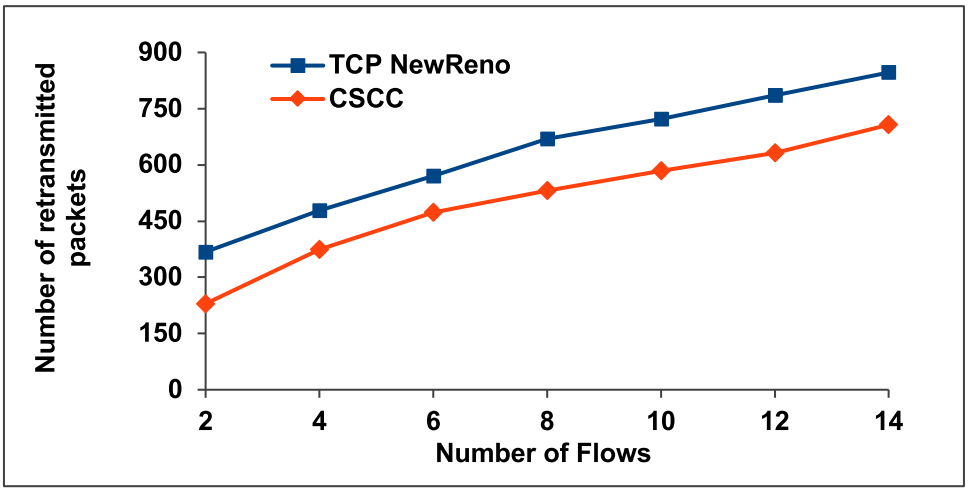

FIGURE 18. The number of retransmitted packets on the 13 × 13 grid topology with DSR.

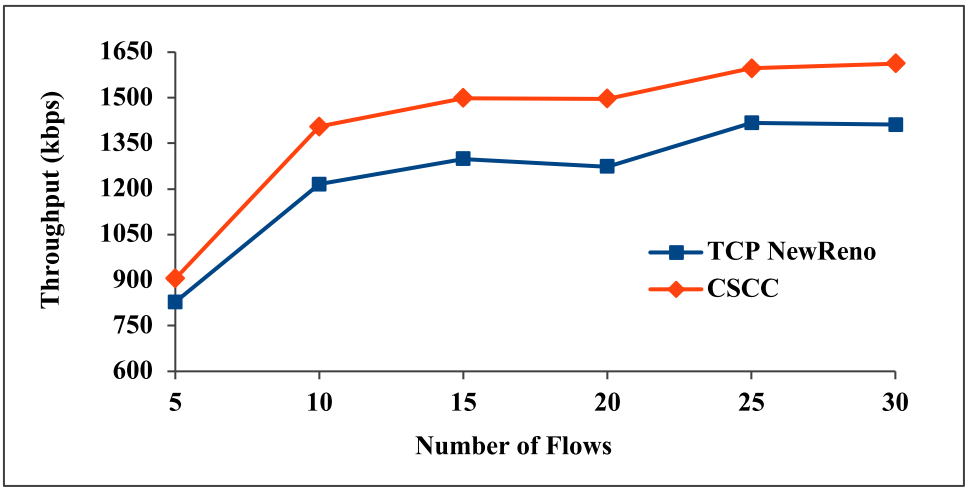

FIGURE 19. Throughput achieved on the random topology with AODV.

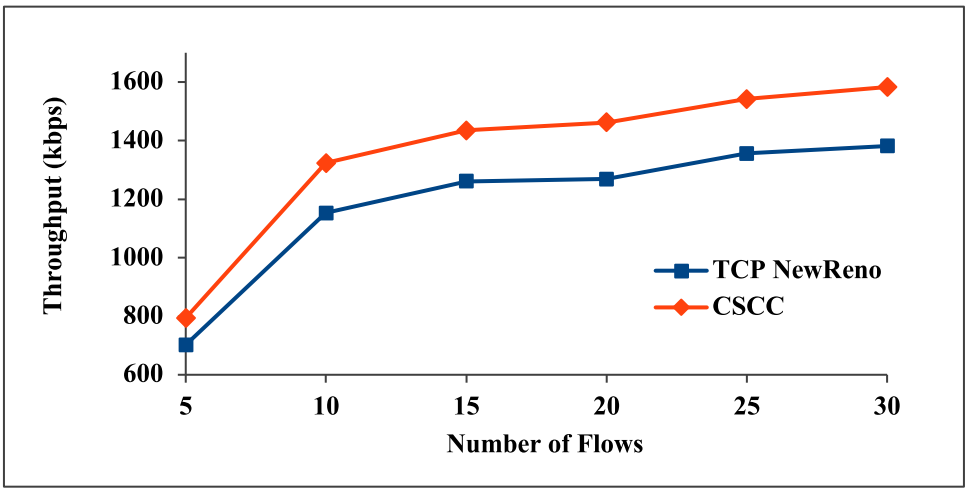

FIGURE 20. Throughput achieved on the random topology with DSR.

and move from one end to the other end of the grid, as shown in Fig. 14. Then two further flows (F3 and F4) were added, one starting on each side and flowing opposite to the first one. The number of flows was increased to 14 by adding two

TABLE 9. Confidence intervals and fairness indexes for the random topology, where the routing protocol is AODV.

	TCP NewReno				CSCC			
Number of flows	Throughput	95% Confidence Interval		Fairness Index	Throughput	95% Confidence Interval		Fairness Index
		Lower Bound	Upper Bound			Lower Bound	Upper Bound	
5	828.08	809.67	846.50	0.370	838.15	810.80	865.50	0.450
10	1215.84	1169.04	1262.63	0.290	1405.30	1379.00	1431.60	0.391
15	1299.04	1252.20	1345.89	0.273	1498.57	1453.49	1543.65	0.389
20	1273.79	1219.14	1328.44	0.223	1497.07	1458.40	1535.73	0.360
25	1417.22	1358.29	1476.14	0.192	1596.25	1548.48	1644.02	0.344
30	1411.76	1344.62	1478.91	0.184	1612.11	1569.67	1654.55	0.341

TABLE 10. Confidence intervals and fairness indexes for the random topology, where the routing protocol is DSR.

	TCP NewReno				CSCC			
Number of flows	Throughput	95% Confidence Interval		Fairness Index	Throughput	95% Confidence Interval		Fairness Index
		Lower Bound	Upper Bound			Lower Bound	Upper Bound	
5	702.23	650.58	753.87	0.37	793.98	736.42	851.54	0.45
10	1152.24	1119.92	1184.56	0.29	1323.29	1288.20	1358.38	0.39
15	1261.29	1232.84	1289.73	0.27	1435.18	1409.65	1460.71	0.39
20	1268.77	1250.31	1287.23	0.22	1461.83	1442.30	1481.37	0.36
25	1356.24	1321.76	1390.72	0.19	1541.81	1515.60	1568.02	0.34
30	1381.37	1325.19	1437.56	0.18	1582.95	1561.30	1604.60	0.34

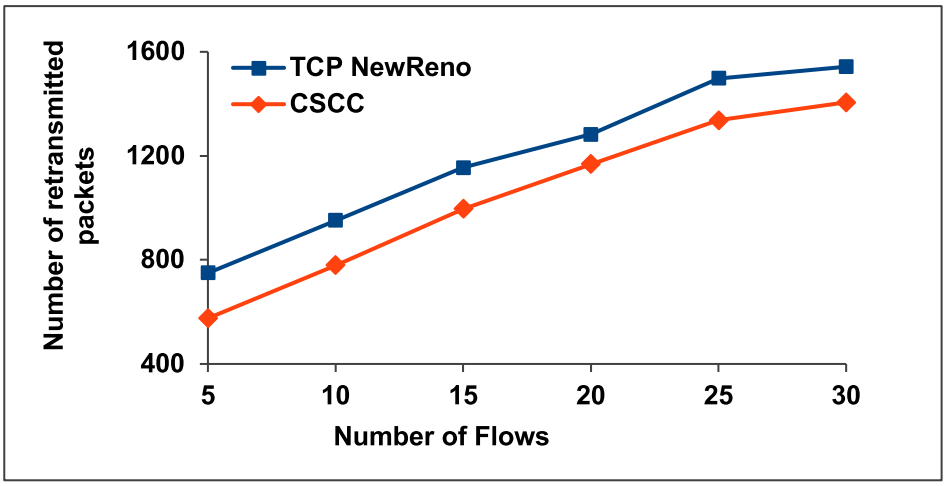

FIGURE 21. Number of retransmitted packets on the random topology with AODV.

successive flows at a time. The throughput recorded using the AODV and DSR routing protocols, respectively, is depicted in Fig. 15 and Fig. 16 for each scenario. The improvement achieved by the CSCC mechanism against TCP NewReno ranges from 24.33 to 30.17% with AODV and from 18.33 to 32.42% with DSR. At the same time, the 95% confidence intervals and fairness indexes computed for the achieved throughput for each scenario are listed in Tables 7 and 8.

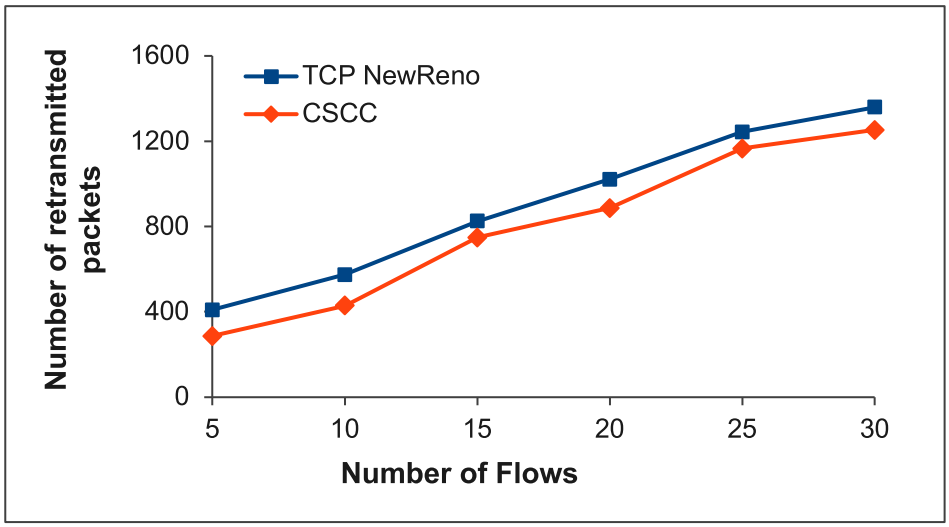

FIGURE 22. Number of retransmitted packets on the random topology with DSR.

Jain's fairness index was calculated according to equation (5).

$$f(x) = \frac{\left[\sum_{i=n}^{n} x_i\right]^2}{n \times \sum_{i=n}^{n} x_i^2} \tag{5}$$

In equation (5), n counts for the total number of flows, and xi for the ith flow's throughput. Equation (5) will provide a result between 0 and 1. The fairness increases as the calculated result approach one and decreases as it approaches zero (0).

Looking at Fig. 17 and Fig. 18, it is clear that the number of retransmitted packets in the case of the CSCC mechanism is less than in the case of TCP NewReno, in the presence of both the AODV and DSR routing protocols. Thus, the CSCC mechanism handles CC more efficiently in a dense network.

C. RANDOM TOPOLOGY

The suggested CSSS mechanism's ability to handle growing traffic flows—from five to thirty connections—is evaluated using simulation experiments. A random network topology is employed in this simulation, with 100 nodes distributed at random throughout an area measuring 1000 by 1000 meters. Like all previous scenarios, the outcomes are averaged over 15 runs. For conducting traffic flow experiments, the throughput achieved by TCP NewReno and the proposed CSCC mechanism with AODV and DSR is illustrated in Fig. 19 and Fig. 20, respectively. Moreover, Fig. 21 and Fig. 22 show each case's retransmitted packets. Analyzing these graphs, the CSCC mechanism has achieved high throughput than TCP NewReno, and less retransmission is observed in the case of the proposed CSCC mechanism. Furthermore, the 95% confidence interval and Jain's fairness index computed in random topology, in the presence of AODV and DSR, are listed in Tables 9 and 10, respectively. The improvement in throughput achieved by the proposed CSCC mechanism against TCP NewReno ranges from 9.27 to 17.53% and from 13.07 to 15.22% in the presence of AODV and DSR, respectively.

VI. CONCLUSION

Improving the performance of TCP in WANETs is the main objective of the proposed CSCC mechanism. In the proposed mechanism, each node calculates the WMA of the number of tries attempted for a frame transmission at the MAC layer to reflect CC. When the WMA at any node hits a pre-defined threshold, the node begins marking packets to alert the sender about contention. Consequently, the sending node must adjust the injection of packets into the network based on the cwnd size of the data flow.

The performance of the proposed CSCC mechanism has been evaluated against TCP NewReno and observed that the proposed mechanism outperformed TCP NewReno in terms of throughput. The number of retransmitted packets is fewer with the proposed mechanism than TCP NewReno, which is a sign of contention control. Moreover, fewer retransmission means the packet drop rate is low.

For the string topology, the CSCC mechanism achieved 10.99% to 56.43% and 12.58% to 54.71% improvement in throughput against TCP NewReno with the AODV and DSR routing protocols, respectively. When the grid topology was considered, the CSCC mechanism achieved 24.33% to 30.17% and 18.33% to 32.42% improvement in throughput against TCP NewReno with the AODV and DSR routing protocols, respectively. A random topology was also considered to evaluate the ability of the CSCC mechanism to handle an increasing number of flows; the CSCC mechanism achieved 9.27% to 17.53% and 13.07% to 15.22% improvement in throughput against TCP NewReno with the AODV and DSR routing protocols, respectively.

ACKNOWLEDGMENT

The authors wish to thank Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R384), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

REFERENCES

[1] J. Postel, *Transmission Control Protocol*, document RFC 793, 1981.
[2] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, Aug. 1988.
[3] M. Allman, V. Paxson, and W. Stevens, *TCP Congestion Control*, document RFC 2581, 1999.
[4] M. Al Shinwan, L. Abualigah, N. D. Le, C. Kim, and A. M. Khasawneh, "An intelligent long-lived TCP based on real-time traffic regulation," *Multimedia Tools Appl.*, vol. 80, no. 11, pp. 16763–16780, May 2021, doi: 10.1007/s11042-020-08856-z.
[5] M. J. A. Jude, V. C. Diniesh, M. Shivaranjani, S. Madhumitha, V. K. Balaji, and M. Myvizhi, "Improving fairness and convergence efficiency of TCP traffic in multi-hop wireless networks," *Wireless Pers. Commun.*, vol. 121, no. 1, pp. 459–485, Nov. 2021, doi: 10.1007/s11277-021-08645-3.
[6] R. Rukaiya, M. U. Farooq, S. A. Khan, F. Hussain, and A. Akhunzada, "CFFD-MAC: A hybrid MAC for collision free full-duplex communication in wireless ad-hoc networks," *IEEE Access*, vol. 9, pp. 35584–35598, 2021, doi: 10.1109/ACCESS.2021.3061943.
[7] *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks-Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Redline*, IEEE Standard 802.11-2020 (Revision of IEEE Standard 802.11-2016), Feb. 2021, pp. 1–7524.
[8] L. D. Hieu, B. T. Tung, P. T. Giang, T. Q. Vinh, and L. Nam, "Improving fairness in IEEE 802.11 EDCA Ad Hoc networks based on fuzzy logic," *J. Adv. Comput. Intell. Intell. Inform.*, vol. 24, no. 5, pp. 615–620, 2020, doi: 10.20965/jaciii.2020.p0615.
[9] A. Deshpande and D. A. K. Shrivastava, "A review of various approaches to improve usage of TCP in mobile ad-hoc networks," *J. Inf. Comput. Sci.*, vol. 9, no. 9, pp. 93–100, Sep. 2019.
[10] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Boston, MA, USA: Springer, 1996, pp. 153–181.
[11] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," presented at the 2nd IEEE Workshop Mobile Comput. Syst. Appl., 1999.
[12] F. Zhang and G. Yang, "A stable backup routing protocol for wireless ad hoc networks," *Sensors*, vol. 20, no. 23, p. 6743, Nov. 2020, doi: 10.3390/s20236743.
[13] V. K. Sharma, L. P. Verma, M. Kumar, R. K. Naha, and A. Mahanti, "A-CAFDSP: An adaptive-congestion aware Fibonacci sequence based data scheduling policy," *Comput. Commun.*, vol. 158, pp. 141–165, May 2020, doi: 10.1016/j.comcom.2020.04.047.
[14] A. S. Sharma and D. S. Kim, "Energy efficient multipath ant colony based routing algorithm for mobile ad hoc networks," *Ad Hoc Netw.*, vol. 113, Mar. 2021, Art. no. 102396, doi: 10.1016/j.adhoc.2020.102396.
[15] N. Mast, M. A. Khan, M. I. Uddin, S. A. A. Shah, and A. Khan, "Channel contention-based routing protocol for wireless ad hoc networks," *Complexity*, vol. 2021, Jan. 2021, Art. no. 2051796, doi: 10.1155/2021/2051796.
[16] A. Al Hanbali, E. Altman, and P. Nain, "A survey of TCP over ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 7, nos. 1–4, pp. 22–36, 2005.
[17] A. M. Al-Jubari, M. Othman, B. Mohd Ali, and N. A. W. Abdul Hamid, "TCP performance in multi-hop wireless ad hoc networks: Challenges and solution," *EURASIP J. Wireless Commun. Netw.*, vol. 2011, no. 1, p. 198, Dec. 2011, doi: 10.1186/1687-1499-2011-198.
[18] H. K. Molia and A. D. Kothari, "TCP variants for mobile adhoc networks: Challenges and solutions," *Wireless Pers. Commun.*, vol. 100, no. 4, pp. 1791–1836, Jun. 2018, doi: 10.1007/s11277-018-5675-8.
[19] H. Zhai, X. Chen, and Y. Fang, "Improving transport layer performance in multihop ad hoc networks by exploiting MAC layer information," *IEEE Trans. Wireless Commun.*, vol. 6, no. 5, pp. 1692–1701, May 2007, doi: 10.1109/TWC.2007.360371.
[20] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP performance," *IEEE Trans. Mobile Comput.*, vol. 4, no. 2, pp. 209–221, Mar. 2005, doi: 10.1109/TMC.2005.30.
[21] 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, doi: 10.1016/j.comnet.2020.107692.
[22] R. M. Bhavadharini, S. Karthik, and R. Sabitha, "An energy-efficient priority-based packet scheduling mechanism for enhancing quality of service in mobile ad hoc network," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 6, p. e6784, 2022, doi: 10.1002/cpe.6784.
[23] S.-T. Chou, Z.-B. Chen, J. Yao, and S.-S. Chou, "A priority contention window mechanism for ad hoc network," in *Proc. IEEE 3rd Eurasia Conf. Biomed. Eng., Healthcare Sustainability (ECBIOS)*, May 2021, pp. 84–87, doi: 10.1109/ECBIOS51820.2021.9511029.
[24] S. Wu, K. Liu, W. Zhang, Z. Xu, F. Liu, and X. Luo, "A distributed cooperative MAC protocol with relay collision avoidance for wireless ad hoc networks," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2018, pp. 325–330.
[25] M. Z. Oo, M. Othman, and T. O'Farrell, "A proxy acknowledgement mechanism for TCP variants in mobile ad hoc networks," *J. Commun. Netw.*, vol. 18, no. 2, pp. 238–245, Apr. 2016, doi: 10.1109/JCN.2016.000033.
[26] A. M. Al-Jubari, M. Othman, B. Mohd Ali, and N. A. W. Abdul Hamid, "An adaptive delayed acknowledgment strategy to improve TCP performance in multi-hop wireless networks," *Wireless Pers. Commun.*, vol. 69, no. 1, pp. 307–333, Mar. 2013, doi: 10.1007/s11277-012-0575-9.
[27] E. Altman and T. Jiménez, "Novel delayed ACK techniques for improving TCP performance in multihop wireless networks," presented at the IFIP Int. Conf. Pers. wireless Commun., 2003.
[28] R. Braden, *Requirements for Internet Hosts-Communication Layers*, document RFC1122, 1989.
[29] R. de Oliveira and T. Braun, "A dynamic adaptive acknowledgment strategy for TCP over multihop wireless networks," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Societies.*, 2005, vol. 3, pp. 1863–1874, doi: 10.1109/INFCOM.2005.1498465.
[30] A. K. Singh and K. Kankipati, "TCP-ADA: TCP with adaptive delayed acknowledgement for mobile ad hoc networks," presented at the IEEE Wireless Commun. Netw. Conf., 2004.
[31] C. D. A. Cordeiro, S. R. Das, and D. P. Agrawal, "COPAS: Dynamic contention-balancing to enhance the performance of TCP over multi-hop wireless networks," in *Proc. 11th Int. Conf. Comput. Commun. Netw.*, Oct. 2002, pp. 382–387.
[32] K. Xu, M. Gerla, L. Qi, and Y. Shu, "TCP unfairness in ad hoc wireless networks and a neighborhood RED solution," *Wireless Netw.*, vol. 11, no. 4, pp. 383–399, Jul. 2005.
[33] 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, doi: 10.1109/90.251892.
[34] D. Kliazovich and F. Granelli, "Cross-layer congestion control in ad hoc wireless networks," *Ad Hoc Netw.*, vol. 4, no. 6, pp. 687–708, Nov. 2006, doi: 10.1016/j.adhoc.2005.08.001.
[35] V. Rakocevic and E. Hamadani, "A cross layer solution to address TCP intra-flow performance degradation in multihop ad hoc networks," *J. Internet Eng.*, vol. 2, no. 1, pp. 146–156, 2008.
[36] R.-S. Cheng and H.-T. Lin, "A cross-layer design for TCP end-to-end performance improvement in multi-hop wireless networks," *Comput. Commun.*, vol. 31, no. 14, pp. 3145–3152, Sep. 2008, doi: 10.1016.comcom.2008.04.017.
[37] M. Wisniewski, *Quantitative Methods for Decision Makers*. London, U.K.: Pearson, 2009.
[38] K. Ramakrishnan, S. Floyd, and D. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, document RFC 3168, 2001.
[39] *The Network Simulator–NS-2*. Accessed: Aug. 25, 2022. [Online]. Available: https://www.isi.edusnam/ns/

NOOR MAST has been actively involved with academia and research. He is a Faculty Member with the Institute of Computing, Kohat University of Science and Technology, Kohat, Pakistan. His research interests include the design of routing protocols, congestion, and channel contention control in wireless networks.

SHAFIULLAH KHAN has been actively involved with academia and research. He is a Professor and the Director with the Institute of Computing, Kohat University of Science and Technology, Kohat, Pakistan. His research interests include wireless networks, ad hoc networks, and game theory.

M. IRFAN UDDIN received academic qualifications in the computer science domain and has explored various topics of computer science as research. He was a scientific researcher in multiple European Union funded projects. He is involved in teaching and research activities related to different diverse topics of computer science and has more than 17 years of teaching and research experience. He is currently a Faculty Member with the Institute of Computing, Kohat University of Science and Technology, Kohat, Pakistan. He has been actively involved in organizing national and international seminars, workshops, and conferences. He has published various research papers in reputed international journals and conferences. His research interests include machine learning, data science, artificial neural networks, deep learning, convolutional neural networks, recurrent neural networks, attention models, reinforcement learning, generative adversarial networks, computer vision, image processing, machine translation, natural language processing, speech recognition, big data analytics, parallel programming, multi-core, many-core, and GPUs. He is a member of ACM and HiPEAC.

YAZEED YASIN GHADI received the Ph.D. degree in electrical and computer engineering from Queensland University. He was a Postdoctoral Researcher with Queensland University. He is currently an Assistant Professor of software engineering with Al Ain University. He has published over 80 peer-reviewed journals and conference papers and holds three pending patents. His current research interests include developing novel electro-acoustic-optic neural interfaces for large-scale high-resolution electrophysiology and distributed optogenetic stimulation. His dissertation on developing novel hybrid plasmonic photonic on-chip biochemical sensors received the Sigma Xi Best Ph.D. Thesis Award. He was a recipient of several awards.

HEND KHALID ALKAHTANI received the B.Sc. degree in computer science from the School of Engineering and Applied Science, The George Washington University, in 1992, the M.Sc. degree in information management from the Department of Engineering Management, The George Washington University, in 1993, and the Ph.D. degree in information security from the Department of Computer Science, Loughborough University, in 2018. She has 23 years of work experience as a lecturer, the computer center president, and the statistic center president with faculty colleges. She is an Assistant Professor with the Department of Information Systems, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University. She received an award from SIDF Academy: Leading Creative Transformation in Critical Time Program, Stanford University, and Center for Professional Development. She has two conference publications in which she received a certificate as the best publication presented.

SAMIH M. MOSTAFA received the bachelor's and M.Sc. degrees in computer science from the Computer Science–Mathematics Department, Faculty of Science, South Valley University, in 2004 and 2010, respectively, and the Ph.D. degree in computer science from the Advanced Information Technology Department, Graduate School of Information Technology, Kyushu University, Japan, in 2017. He is currently a fellow with the Academy of Scientific Research and Technology (ASRT), Egypt. His research interests include machine learning and CPU scheduling.