Wireless Pers Commun (2018) 100:1791–1836
https://doi.org/10.1007/s11277-018-5675-8

TCP Variants for Mobile Adhoc Networks: Challenges and Solutions

Hardik K. Molia[1,2] · Amit D. Kothari[3]

Published online: 10 April 2018

Abstract Transmission Control Protocol (TCP) provides connection oriented and reliable transport layer services. Mobile Adhoc Networks (MANETs) are autonomous and infrastructure less wireless networks. A significant amount of performance degradation is found when TCP is used with the MANETs as compared to the wired networks. TCP suits well with the wired networks, where majority of the packet losses are due to network congestion. MANETs have various other issues like transmission errors, dynamic topologies, link layer contentions. Transmission errors or contention issues are responsible for channel losses. Dynamic topologies are responsible for route failure losses. This review focuses on discussion of traditional TCP variants and various losses in MANETs. TCP variants for MANETs are explained which are classified into cross-layer approaches and layered approaches. A review of a set of TCP variants based on loss handling approach is given according to loss differentiation, loss prediction and loss avoidance approaches. The main purpose of this review is to define existing issues and future directions for improvement of TCP for MANETs.

Keywords TCP · MANET · Congestion · Route failure · Channel loss · Contention

✉ Hardik K. Molia
hardik.molia@gmail.com

Amit D. Kothari
amitdkothari@gmail.com

[1] Gujarat Technological University, Ahmedabad, Gujarat 382424, India

[2] Government Engineering College, Rajkot, Rajkot, Gujarat 360005, India

[3] ITM Universe, Vadodara, Vadodara, Gujarat 391510, India

1 Introduction

Transmission Control Protocol (TCP) is one of the most widely deployed transport layer protocols of TCP/IP protocol suite. Data link layer cares of node to node delivery; network layer cares of source to destination delivery where the transport layer cares of end to end delivery. TCP ensures delivery between two processes running on two different communicating end devices. TCP has three way handshaking mechanism for connection management. As a part of connection oriented service, it supports stream communication and full duplex communication. Stream communication ensures that the delivery is exactly in the same order in which the data was sent. Full duplex communication facilitates bidirectional communication in a single TCP connection. TCP ensures reliable communication with congestion control, flow control and error control. Congestion control restricts the transmission rate as per the level of congestion in the network. Flow control restricts the transmission rate as per the capacity of the receiver. Error control uses acknowledgement based retransmissions to handle lost, corrupted or discarded packets [1, 2].

In recent years, wireless networks have became very popular because of freedom from physical connections. Mobile Adhoc Networks (MANETs) are wireless networks too. They support device mobility so that users can continue communication while changing their locations. They are adhoc networks as they have no permanent and fixed infrastructures in terms of internetworking devices like routes, access points etc. MANETs can be used to form networks on temporary basis when permanent setup is either not required or not feasible like at disaster recovery camps. MANETs can also be used to form temporary networks at small places such as cafeterias and conferences. A MANET can be formed by a set of devices connected with each other through their wireless transmission ranges only. Every device plays two roles; as a communicating device as well as a forwarding device. Mobility keeps the network topology dynamic; frequently changing which makes routing difficult [3, 4].

TCP was initially introduced for the wired networks where network congestion is the primary cause of packet losses. TCP works on the assumptions which are suitable for wired networks only. Wired networks have less chances of packet losses due to other issues like transmission errors and route failures as compared to the network congestion. TCP mainly focuses on congestion control and considers any packet loss as a cause of network congestion only. Subsequently, it reduces the transmission rate to avoid further network congestion. TCP's default interpretation of any packet loss as a cause of network congestion suits well with the wired networks but not with the wireless networks. Broadcast nature of wireless networks often causes packet losses due to transmission errors and contention issues. As MANETs support dynamic topology, device mobility and network partitions may loss packets due to route failures. TCP's inability of differentiating losses and to act accordingly degrades overall performance. Many approaches have been proposed for performance improvement of TCP when used with MANETs [5–7].

This review focuses on exploring the challenges while using TCP with MANETs. This review discusses various TCP variants which have been proposed so far. This review is organized as follows. Section 2 discusses basis of TCP and various traditional TCP variants. These variants are classified based on the approach of handling congestion. Reactive and proactive TCP variants are explained. Section 3 discusses various losses which affect TCP's performance. TCP variants for MANETs are classified into cross layered and layered approaches. Section 4 discusses TCP variants based on cross layered approaches according to the losses they handle. Section 5 discusses TCP variants based on layered approaches according to the losses they handle. Section 6 discusses recent TCP variants based on the way they handle losses. Loss differentiation, loss prediction and loss avoidance based TCP variants are explained. Section 7 is for simulation. Sections 8 and 9 are conclusions and future directions. TCP is a byte oriented protocol. In this paper Packet term is used to refer to a TCP segment for easier understanding.

2 Traditional TCP Variants

2.1 TCP

Transmission Control Protocol (TCP) acts as a logical vehicle to transfer data between two processes running on two different devices. TCP is a byte oriented protocol which assigns a 32-bit number called byte number to each of the bytes being sent. To ensure ordered delivery, TCP uses 32-bit number as a sequence number of every packet being sent which is the byte number of the 1st byte inside that packet. Receiver acknowledges successful reception of a packet by sending an ACK-Acknowledgement to the sender. An ACK contains the 32-bit number called acknowledgement number which is set to the sequence number of the next expected byte and so of the sequence number of the next expected packet [1].

TCP performs congestion control, flow control and error control for reliable communication. Congestion control finds suitable transmission rate as per the level of network congestion. Flow control finds suitable transmission rate as per the receiver's buffer capacity. The actual transmission rate is set to the minimum of these two values. Some packets may get lost or corrupted due to transmission issues like attenuation, distortion, noise etc. some packets may get discarded due to congestion. Error control identifies and retransmits such packets based on missing acknowledgements [1].

The transmission rate and receiver rate are decided with the help of byte oriented sliding window at sender side and receiver side respectively. Sender TCP decides the size of Congestion Window (Cwnd) with the help of congestion control. Receiver TCP decides the size of Receiver Window (Rwnd) for the flow control. Rwnd is the rate at which the receiver is able to receive without overflow of at its buffers. Cwnd cares about overall overload of the network. Rwnd cares about overload of the receiver only. Receiver sends value of Rwnd to the sender as a part of TCP Header in an ACK. TCP sender sets the actual Transmission windowWnd to minimum of Cwnd and Rwnd to ensure congestion control as well as flow control [1].

$$Wnd = Minimum[Cwnd, Rwnd] \tag{1}$$

In reality, $Cwnd \ll Rwnd$ most of the time.

TCP is ACK-Clocking protocol where the rate of ACKs is used to decide the rate of transmission. The more ACKs sender receives, the more packets it sends. The relationship between the transmission rate and ACK rate is decided by Additive Increase Multiplicative Decrease (AIMD) scheme of Standard TCP. Many variants have been proposed which modify AIMD scheme for performance improvements [1].

2.2 AIMD Scheme

TCP is based on Additive Increase Multiplicative Decrease (AIMD) scheme. AIMD scheme increases or decreases transmission rate as per the level of network congestion.

AIMD has three phases: Slow Start to increase transmission rate exponentially, Congestion Avoidance to increase transmission rate linearly. Congestion Detection and Recovery to reduce transmission rate with a predefined factor [1, 2].

2.2.1 Slow Start

TCP sender enters into the 1st phase of AIMD scheme called Slow Start after connection establishment. It sets initial value of Cwnd to 1 or 2 packets only. The main purpose of starting with a small value of Cwnd is to avoid sudden overwhelming of a network without analyzing present status of congestion. TCP increases Cwnd by 1 after receiving every ACK successfully. As Cwnd is increased with every ACK, on the successful completion of each round of transmission (after transmitting entire Wnd successfully), Cwnd becomes double. Thus Slow Start phase shows exponential growth of transmission rate as transmission rate depends on Cwnd. Slow Start Threshold (SSThresh) restricts the exponential growth. Once Cwnd reaches SSThresh, TCP sender enters into the next phase of AIMD scheme called Congestion Avoidance [1, 2, 8].

2.2.2 Congestion Avoidance

Just like Slow Start phase, Congestion Avoidance phase increases the transmission rate but less aggressively. TCP increases Cwnd linearly rather than exponentially. TCP increases Cwnd by 1 after completion of every round of transmission, after every Round Trip Time (RTT) rather than after every ACK. The main purpose of moving from Slow Start phase to Congestion Avoidance phase is to keep increasing transmission rate slowly [1, 2, 8].

2.2.3 Congestion Detection and Recovery

TCP predicates network congestion using Retransmission Time Out (RTO). After sending a few packets of total size equal to of Wnd, TCP waits for ACK, maximum for the duration of RTO. If TCP does not receive ACK within time, RTO time out occurs which is considered as an indication for loss of all or few of the transmitted packets due to network congestion. On RTO time out, TCP immediately comes out of Slow Start or Congestion Avoidance phase and Congestion Detection and Recovery phase is started. As RTO time out occurs due to unavailability of a required ACK, it is assumed that the ACK was not available either due to any of the following reasons [1, 2, 8].

- The packet was discarded by an intermediate device due to network congestion so it could not reach to the receiver.
- The packet was received successfully but was corrupted and so discarded by the receiver.
- The packet was received successfully. Receiver had sent ACK also. But sender could not receive ACK because of loss,corruption or discard of ACK during transmission.

In either of the above cases, the undelivered packets need to be retransmitted. TCP sets SSThresh to half of the current Cwnd, Cwnd to the initial size and enters into Slow Start phase. Based on the current Cwnd, packets are retransmitted [1, 2, 8]. AIMD scheme is shown in Fig. 1.

The AIMD phases discussed in this section are of standard TCP. Many TCP variants have been proposed by introducing modifications of phases or introducing new phases [1, 2, 8].

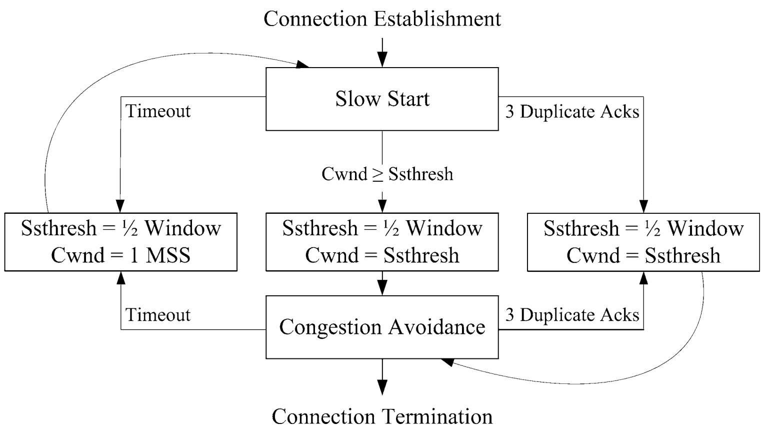

Fig. 1 TCP's AIMD scheme [1]

TCP variants can be classified based on the approach of handling network congestion. Reactive TCP variants are based on congestion detection and recovery approach. These variants use information of packet losses for decision making. RTT is less accurate as only one timer is maintained per TCP connection. Proactive TCP variants are less aggressive in terms of increasing transmission rate as their approach is Congestion Avoidance. These variants use information of packet delays for decision making. RTT estimation is more accurate as fine grained timers are used to track delay for every sent packet [1, 2, 8].

2.3 Reactive TCP Variants

2.3.1 TCP Tahoe

TCP Tahoe [9, 10] follows standard TCP's AIMD scheme. Tahoe introduces Fast Retransmission phase to detect congestion even before RTO timeout. A receiver may receive packets out of order. With every out of order received packet, receiver sends an ACK requesting for a packet which was expected as per the order. Sender counts how many such ACKs are received asking for the same packet already transmitted earlier. These ACKs are called duplicate ACKs as they are sent to request for a same packet more than once. When the count of duplicate ACKs becomes 3, Tahoe activates Fast Retransmission phase. Fast Retransmission retransmits the requested packet, sets SSThresh to half of Cwnd, sets Cwnd to initial size and switches to the Slow Start phase. Tahoe takes the same action when either 3 duplicate ACKs are received or RTO timeout occurs. Tahoe introduces Fast Retransmission phase to detect congestion earlier than RTO timeout event.

2.3.2 TCP Reno

TCP Reno [9, 10] modifies Tahoe by differentiating both the events: 3 duplicate ACKs and RTO time out. Reno considers arrival of 3 duplicate ACKs as a weaker possibility of congestion by assuming that the network is not completely congested as the receiver is still receiving packets for which duplicate ACKs are generated. At the same time, RTO time out event is considered as a stronger possibility of congestion. For both the events, it is necessary to reduce transmission rate but not equally. Tahoe reduces the transmission rate completely and moves to the Slow Start phase for both of these events. Reno reduces transmission rate completely when RTO time out occurs but halves transmission rate when receives 3 duplicate ACKs. Reno introduces an another phase called Fast Recovery. Reno moves to the Slow Start phase when RTO timeout occurs and to the Fast Recovery phase when it receives 3 duplicate ACKs.

On detection of 3 duplicate ACKs, Reno sets SSThresh to half of the current Cwnd. It sets Cwnd to SSThresh + 3. Here 3 is added as 3 packets left the network for which receiver has generated duplicate ACKs. Reno retransmits the oldest unacknowledged packet corresponding to the set of 3 duplicate ACKs and enters into the Fast Recovery phase. Reno exits Fast Recovery phase and moves to the Congestion Avoidance phase only once it receives an ACK, acknowledging a few or all of the sent packets. During Fast Recovery phase, for every duplicate ACK, Reno increments Cwnd by 1 and sends a new packet. On exit of Fast Recovery phase, Reno sets Cwnd to the value of SSTresh to start with the Congestion Avoidance phase. The main purpose of staying in Fast Recovery phase is to ensure that the packets are started being received properly.

2.3.3 TCP NewReno

TCP NewReno [11] improves TCP Reno. At any moment there will be a set of packets in transmission called outstanding packets. Any ACK which acknowledges only few of the outstanding packets is called a partial ACK. A Full ACK acknowledges all the outstanding packets. Reno comes out of the Fast Recovery phase either on receiving a partial ACK or a full ACK. There is a possibility that more than one packet from the same set of outstanding packets are not received. Reno performs well if there is one packet loss per set of outstanding packets. In other cases, it behaves almost similar to the Tahoe. Once Reno enters Fast Recovery phase, it waits for a fresh ACK (partial or full but no duplicate). In a set of outstanding packets, corresponding to the 3 duplicate ACKs of the 1st oldest undelivered packet, Reno enters into the Fast Recovery phase. For rest of the undelivered packets of the same set of outstanding packets, RTO timeous may occur where Reno behaves just like Tahoe by switching to the Slow Start phase. In some cases, Reno reduces the transmission rate more than once corresponding to the multiple undelivered packets belonging to the same set of outstanding packets.

NewReno modifies the Fast Recovery phase by processing all the duplicate ACKs. For 1st set of 3 duplicate ACKs, it retransmits the requested packet. For every next duplicate ACK, it retransmits the requested packet immediately without waiting for two more duplicate ACKs. With every retransmission, retransmission timer is reset so earlier RTO time outs can be avoided. NewReno keeps incrementing Cwnd by 1 with every duplicate ACK to send fresh packets in addition of the retransmissions until all the outstanding packets are acknowledged. NewReno switches to the Congestion Avoidance phase once all the outstanding packets are acknowledged.

NewReno avoids unnecessary reduction of the transmission rate for the undelivered packets belonging to the same set of outstanding packets. A set of undelivered packets make holes in the ordered sequence of packets. Fast Recovery phase is used for the hole filling purpose. As NewReno avoids unnecessary RTO timeouts, it maintains high throughput when multiple holes with multiple packets within some holes need to be processed. NewReno outperforms Reno specially in the cases of high error rates.

2.4 Proactive TCP Variants

2.4.1 TCP Vegas

TCP Vegas [12] is a proactive TCP variant which uses packet delay information to predicate congestion. Vegas is less aggressive as compared to NewReno as far as increase in transmission rate is concerned. The main goal of Vegas is to identify the difference between the expected throughput and the actual throughput to increase or decrease current transmission rate linearly. Standard TCP increases transmission rate until queue overflows and discards a few packets causing network congestion. Subsequently standard TCP detects congestion and reduces the transmission rate. Vegas reduces the transmission rate before such event occurs to avoid congestion. Vegas introduces five novel techniques as explained below.

Accurate RTT Calculation
Vegas tries to avoid congestion by detecting it at an incipient stage by analyzing the increasing RTT values. Tahoe, Reno and NewReno estimate one value of RTT per transmission window. Vegas estimates RTT per packet. Thus RTT estimation is more accurate in Vegas. Vegas records system time for each of the sent packets. On receiving an ACK, RTT is calculated using the current time and recorded sent time. Accurate RTT calculation is used to calculate time out value more precisely. At the same time, the decision of retransmission can be done in more timely way.

Retransmission Requirement Detection
On receiving a duplicate ACK, Vegas checks whether the difference between the current time and the recorded sent time for the requested packet is more than time out value or not. If it is so then Vegas retransmits the requested packet immediately without waiting for additional duplicate ACKs. This helps to avoid timeout for the cases when either the window is too small or the losses are too high and so three duplicate ACKs could not be generated or received.

On receiving a non-duplicate ACK which acknowledges a few (not all) outstanding packets and is 1st or 2nd ACK after a retransmission, Vegas checks whether the difference between the current time and the recorded sent time for the next outstanding packet is more than time out value or not. If it is so then Vegas retransmits the next outstanding packet immediately without waiting for any duplicate ACKs. This helps to detect any further lost packet without waiting for the duplicate ACKs.

Window Size Policy
Vegas decreases the transmission rate by decreasing window size when the losses are happened at the current transmission rate. Any loss which is belonging to previous transmission rate is not considered for window size reduction. Vegas decreases transmission rate if the lost packet was sent after most recent window decrease.

Congestion Avoidance Mechanism
Vegas finds the difference between the expected throughput and the actual throughput. The number of bytes in transmission is called extra bytes. Vegas tries to maintain sufficient amount of extra bytes which neither cause network congestion nor under utilise available bandwidth. Expected throughput is calculated with reference of *BaseRTT* which is

minimum of all RTTs so far. Actual throughput is calculated with reference of most recent RTT. Window_Size is the size of *Cwnd*.

$$Exp\_Throughput = Window\_Size/BaseRTT \tag{2}$$

$$Act\_Throughput = RTT\_Len/RTT \tag{3}$$

$$Diff = Exp\_Throughput - Act\_Throughput \tag{4}$$

When *Diff* is negative, *BaseRTT* is set to most recent RTT. Two thresholds α and β set the lower and upper boundaries of extra bytes respectively. If $Diff < \alpha$, Cwnd is incremented by 1. If $Diff > \beta$, *Cwnd* is decremented by 1. No change is made if $\alpha \leq Diff \leq \beta$. In real life implementations, α and β define minimum and maximum number of required extra buffers to hold extra bytes.

Modified Slow Start Mechanism

Reno's Slow Start is very aggressive as it increases transmission rate exponentially every RTT. To avoid all of sudden congestion during Slow Start phase and to calculate the actual throughput efficiently, Vegas increases transmission rate exponentially every other RTT. When actual throughput falls below threshold γ, Vegas moves to the Congestion Avoidance mechanism.

3 TCP Variants for MANETs

3.1 TCP and Losses

3.1.1 TCP's Limitation

TCP ensures reliable delivery with its congestion control, flow control and error control mechanisms. These mechanisms work with the assumptions suitable for the wired networks only. Wired networks have fixed topologies with dedicated networking devices like routers to manage the traffic. When a router is not able to handle more packets, it starts discarding. Such situation is called network congestion. Wired networks get majority of the packet losses due to congestion as compared to the losses due to link failures or lost/corrupted packets due to transmission errors. TCP is able to decide the most suitable transmission rate, accordance with the present status of congestion. As TCP was initially introduced for wired network, it strongly follows this assumption by considering any loss as a cause of network congestion only. TCP reduces transmission rate and re-transmit lost packets on noticing congestion as discussed in section TCP variants. When TCP is deployed in wireless networks, a significant amount of performance degradation is found as it is not able to adopt itself with the issues related with wireless communications. Wireless networks are different than wired networks because of broadcast nature and inherent wireless transmission issues. Moreover wireless networks like MANETs support dynamic topologies with no dedicated networking devices. Wireless networks often suffer from packet losses or corrupted packets due to wireless transmission issues like attenuation, noise, fading, contention. Mobility of devices makes routing difficult and may partitions a network into two or more halves causing losses due to route failures. TCP's inability to differentiate various losses to act accordingly is one of the most critical issues of wireless networks specially of MANETs [5–7].

3.1.2 Route Failure Losses

Device mobility changes the network topology continuously, making it dynamic. Battery powered devices may shutdown all of sudden. Some devices may leave network all of sudden. Routing in dynamic topology is always challenging as it experiences frequent route failures. Routing protocols such as AODV, DSR are specially written to deal with dynamic routing in MANETs. Dynamic topology may partition a network into two or more halves where no devices from one half could communicate with any other device in any other halves due to limited transmission ranges. A route gets failed when a device playing a role of a forwarding device moves out of range or shutdown. Subsequently, a new route needs to be reestablished. Packets are often discarded by a device causing a route fail. As it takes time to reestablish a new route, TCP sender may experience a retransmission time out. Subsequently, TCP sender reduces the transmission rate completely and retransmits the oldest unacknowledged packet. In case of a route failure, TCP sender should neither reduce the transmission rate nor retransmit immediately. Once a new route is reestablished, it should retransmit the lost packets [5–7].

3.1.3 Congestion Losses

MANETs may experience network congestion due to limited bandwidth and buffering capabilities. A device starts discarding packets when it is running out of space while buffering packets. TCP performs congestion control through its AIMD scheme to reduce transmission rate to cope up with congestion. TCP variants for MANETs introduce different ways for detection of congestion [5–7].

3.1.4 Channel Losses and Contention Issues

Wireless transmission is more vulnerable to transmission impairments like attenuation, noise, distortion, interference which loss/corrupt packets. Lost packets never reach to the receiver while corrupted packets are discarded by the receiver which are not possible to recover. TCP works at the transport layer from where it can not predicate possibility of such losses accurately. Lost or corrupted packets generate holes in the receiver window. Subsequently, TCP receiver sends duplicate acknowledgements asking such packets. Duplicate acknowledgements are considered as weak possibility of congestion for which TCP sender starts retransmission after reducing transmission rate by half and switches to the Congestion Avoidance phase. When packets are not lost due to congestion, unnecessary reduction of transmission rate degrades overall performance by under utilizing available bandwidth [5–7].

Most of the wireless networks implement contention based protocols for allowing devices to communicate. Packets may get lost or corrupted because of various contention based issues like collision, hidden and exposed station problems. At the transport layer such issues could be classified as inter-flow interference and intra-flow interference. Inter-flow interference refers to the packet loss issue across multiple TCP flow. Intra-flow interference refers to the packet loss issue inside a single TCP flow—Issues between Data flow and ACK flow or in a bidirectional data flow [5–7].

3.2 Classification

TCP variants for MANETs can be classified according to the level of network awareness into cross layered approaches and layered approaches. A cross layered approach based TCP receives information for decision making from any of the lower layers like network layer or link layer. TCP uses this information without trying to collect it itself. These variants are complex to write as well as violate layer independency concept of TCP/IP protocol suite. These variants are based on Green box concept as the network communicates its state to the transport layer for decision making purpose. A layered approach based TCP collects information for decision making itself without being dependent on any of the lower layers. These variants are simple to write as implementation for layer interaction is not required. These variants are based on Black box concept as there is no information of network state is provided other than collected from congestion control statistics. Grey box concept based TCP variants estimate bandwidth related statistics from the transport layer itself. TCP Vegas is based on grey box concept. The following sections discuss TCP variants based on these approaches as well as based on what type of losses they try to handle [5–7].

4 Cross Layered Approaches

4.1 Introduction

TCP is a transport layer protocol ensuring connection oriented and reliable end to end delivery. TCP is not able to get present status of a network other than of congestion. Network layer gets information about route failures. Link layer gets information about transmission errors and contention issues. As discussed in Sect. 1, TCP's performance is degraded due to inability of differentiating various losses to act accordingly. TCP considers any loss as a cause of congestion because of the unavailability of information regarding other type of losses happened at lower layers. Cross Layered approach based TCP allows a lower layer to provide decision making information to the TCP working at transport layer. Route failure losses could be handled by interaction between Network layer and TCP. Channel losses and contention issues are handled by interaction between Link layer and TCP. This section discusses cross layered TCP variants based on type of issues they handle.

4.2 Route Failures Losses

4.2.1 TCP-F

TCP-F [13] (TCP-Feedback) informs a sender about a route failure with RFN-Route Failure Notification message. On receiving RFN, TCP enters into the snooze state to pause further transmission until a failed route is restored or a new route is established. It invalidates all timers to avoid unnecessary timeouts. It stores values of other variables like window size and retransmission timer to resume transmission later on. It starts RFT-Route Failure Timer to wait for restoration of a route. TCP sender remains in snooze state until it is notified for a new route with Route Reestablishment Notification (RRN). The main motive is to avoid time out, reduction of transmission rate and retransmission by invoking congestion control mechanisms unnecessarily. An intermediate node which detects a route failure sends RFN back to the actual sender. If any intermediate node which receives RFN has an alternate route to reach to the receiver, it may discard RFN and starts routing on a new route. In absence of availability of any alternate route, RFN reaches to the sender.

On receiving a RRN, TCP sender enters into the active state by flushing all the unacknowledged packets. The main reason is to avoid keep waiting for those packets which were most likely discarded due to a route failure. These packets are retransmitted as per the size of sender window. Value of RFT is set to the worst time required for a route reestablishment so far. RFT prevents sender from being in snooze state indefinitely. On expiration of RFT, TCP retransmits all the unacknowledged packets by moving from snooze state to the active state which may generate a large burst of traffic all of sudden. On receiving RRN, TCP resumes transmission with its old state which may not be suitable for the new route. An intermediate node which has earlier sent a RFN may send a RRN on availability of the same or alternate route. It discards all future RRN messages for the same connection.

4.2.2 TCP-ELFN

TCP-ELFN [14] (TCP with Explicit Link Failure Notification) informs sender about a route failure to avoid unnecessary activation of congestion control. ELFN could be implemented with “Host Unreachable” message of ICMP-Internet Control Message Protocol. If routing protocol supports route failure messages, the information of “Host Unreachable” message could be piggybacked with them. To identify a connection, information of sender and receiver (addresses and ports) and TCP sequence number are piggybacked. On receiving ELFN, TCP sender enters into the standby mode by pausing further transmission. TCP sender remains in standby mode until a failed route is restored or a new route is established. TCP sender periodically checks for availability of a route using a probe packet. On receiving an ACK for a probe packet, it comes out of standby mode and continues transmission.

TCP with ELFN is evaluated based on different values of the interval between probe packets, modification of RTO and Cwnd (once a new route is found) and different types of packets being used for probing purpose. If the interval between probe packets is very large then it may delay transmission even if a route is already available. If the interval between probe packets is very small then it may start sending probe packets before a route is available causing further issues like congestion. So the interval should be chosen carefully as a function of RTT. On route restoration, TCP can resume with its old state or modify value of Cwnd and/or RTO to their default values to start exploring a new route. It has been seen that modifying RTO to its default value improves throughput. A probe packet could be an oldest unacknowledged packet or a new packet. It has been seen that there is no significant effect on selecting either of these for the probing purpose.

4.2.3 TCP-BuS

TCP-BuS [15] (TCP-Buffering capability and Sequencing information) uses Associativity-Based Routing (ABR). When a node moves out of the radio ranges of the neighboring nodes, a route failure occurs. A node detecting a route failure has to initiate a route reconstruction process to discover a partial route to reach to the destination. A node sends a Localized Query (LQ) message for this purpose. Any downward node uses Route

Notification (RN) message to invalidate old partial path. The destination replies to LQ message with a REPLY message. It has four mechanisms discussed below.

Explicit Notifications
Pivot node (PN) detects a route failure being an intermediate node in a route from sender to the receiver. A route failure causes receiver unreachable from the PN. PN sends Explicit Route Disconnection Notification (ERDN) message to the sender to inform about a route failure. On receiving ERDN, TCP enters into standby mode (to pause transmission). PN sends Explicit Route Successful Notification (ERSN) message to the sender to inform about restoration of a route. On receiving ERSN, TCP enters into active mode (to resume transmission).

Extended Timers
There are two categories of affected packets due to a route failure. A few packets may get discarded by the node which caused a route failure. A few packets may get buffered at intermediate nodes between a sender and a PN. TCP sender experiences timeout for these packets due to unavailability of acknowledgements. To avoid unnecessary activation of timeout based congestion control, TCP BuS doubles RTO value on receiving ERDN message. By this time, TCP expects restoration of a new route.

Selective Retransmissions
There may be a few packets which are lost on a route from sender to the PN. These packets are not retransmitted until timeout occurs. Selective retransmission identifies such packets (lost on a route from sender to PN) and retransmit to PN for the buffering purpose. Once a route between PN and receiver is restored, PN starts transmission of buffered packets. The main purpose of this mechanism is to maintain ordered sequence of buffered packets at PN with retransmission of missing packets if any.

Avoid Unnecessary Fast Retransmissions
On a route failure, some packets may get lost on a route from PN to the receiver. Once a partial route between PN and receiver is restored, PN starts sending buffered packets. Buffered packets are probably next to the lost packets (lost on a route from PN to the receiver). Receiver may notice a hole of missing packets for which it sends duplicate acknowledgements. Sender may react to these duplicate acknowledgements by activation of fast retransmissions unnecessarily. TCP-BuS avoids fast retransmissions by sending information of lost packets to the sender. ERDN keeps information about the highest sequence number of a buffered packet at PN. ERSN keeps information about the highest sequence number of received packet at receiver. Sender can find the missing sequence of packets from this information and retransmit without waiting for duplicate acknowledgements.

Incoming_SEQ refers to the sequence number of incoming packet. ERDN has an argument ERDN_GEN_SEQ which is set to the packet which source will need to send once a route is restored (Next packet from the highest in-order received packet at PN). ERSN has an argument LAST_ACK which is set to the sequence number of the highest in-order received packet at the destination. For example, if source has sent packets up to 14 and it receives ERDN(10), packets from 10 to 14 are considered as buffered at PN after detection of a route failure. On receiving ERSN(6), Source assumes that the packets 7, 8 and 9 are lost due to route failure. Subsequently source retransmits these packets and continues with transmission of fresh packets from 15 onwards. Meanwhile, packets from

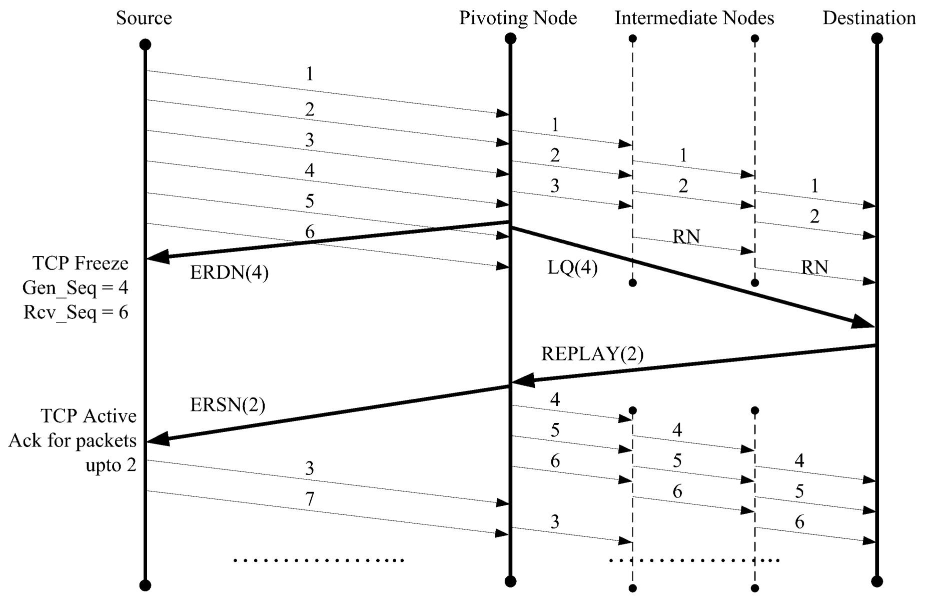

Fig. 2 TCP-BuS [15]

10 to 14 are transmitted by PN. So this way, TCP BuS avoids unnecessary activation of fast retransmission due to out of order delivery of packets. The process is shown in Fig. 2 [15].

4.2.4 Adhoc TCP

ATCP [16] (Adhoc TCP) is a thin layer between IP and TCP. ATCP detects route failures with ICMP messages and congestion with Explicit Congestion Notification (ECN). Duplicate acknowledgements and RTO timeouts are used to detect channel losses. ATCP is a sender side modification which introduces four states: Normal, Disconnected, Congested and Loss.

In Normal state, TCP follows AIMD scheme. ATCP monitors network state based on ICMP messages and ECN marked packets. Disconnected state is related with ICMP message. Congested state is related with ECN marked packet. Loss state is related with duplicate acknowledgements and RTO timeout.

In Disconnected state, ATCP puts TCP into persist mode to pause transmission until a route is restored. TCP continuously probe the network and ATCP switches to the normal state once a probe packet is acknowledged. On route restoration, TCP does not continue transmission with the old transmission rate but Cwnd is set to its initial size to explore new route to find most appropriate transmission rate. In Congested state, ATCP allows TCP to follow AIMD based congestion control. ATCP returns to the normal state on transmitting a packet.

When ATCP finds a third duplicate acknowledgement for a packet or when RTO timer is about to timeout, it puts TCP into persist mode and switches to Loss state. The reason is to avoid activation of congestion control. TCP retransmits all unacknowledged packets. On receiving a new acknowledgement, ATCP switches TCP to Normal state. The main 7 events of TCP Adhoc are listed below.

1. Receive Destination Unreachable ICMP Message
2. Receive ECN
3. TCP Transmits a packet
4. New ACK
5. Three Duplicate ACKs or RTO expiration
6. Receive Duplicate or New ACK
7. Retransmission

The process is shown in Fig. 3 [16].

4.2.5 TCP-ECIA

TCP-ECIA [17] (TCP-Exploiting Cross-layer Information Awareness) uses ELFN messages to handle route failures. On receiving an ELFN message, TCP sender enters into the freeze state. It is obvious that an intermediate node experiencing a route failure may discard a few data packets or ACK packets. The lost packets may cause timeouts at the TCP sender once it comes out of the freeze state after route restoration. TCP-ECIA tries to avoid these timeouts with its two mechanisms. Any intermediate node experiencing a route failure uses any of these two mechanisms based on type of packets it needs to discard. EPLN-Early Packet Loss Notification is used to inform TCP sender about discard of data packets. BEAD Best Effort ACK Delivery message is used to inform TCP receiver about discard of ACK packets. EPLN and BEAD messages contain sequence numbers of all dropped packets. On receiving an EPLN message, TCP sender disables retransmission time out timer and retransmits discarded packets with lowest sequence number on route restoration. On receiving a BEAD message, TCP receiver resends an ACK with highest sequence number on route restoration. DSR protocol is used to implement EPLN and BEAD mechanisms.

The process of Early Packet Loss Notification is shown in Fig. 4 [17]. A node A starts a TCP connection with node E using the route A–B–C–D–E. On failure of link C–D, C selects alternate route C–F–G–E. After a while, node F detects a link F–G is fail. As there is no alternate route can be found, F has to drop the packets. F sends a message with packet drop information to C, Msg1 (F, C). As this message is not from the TCP sender, Node C finds route to the TCP sender (A) from its cache to send Msg2 (F,C) (It can be considered as Msg2 (C,A) too) to A.

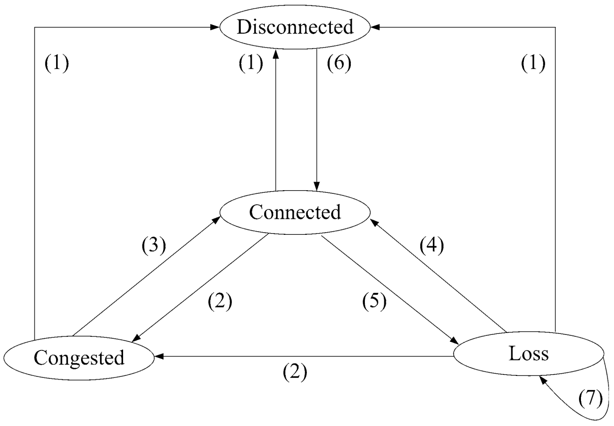

Fig. 3 Adhoc TCP [16]

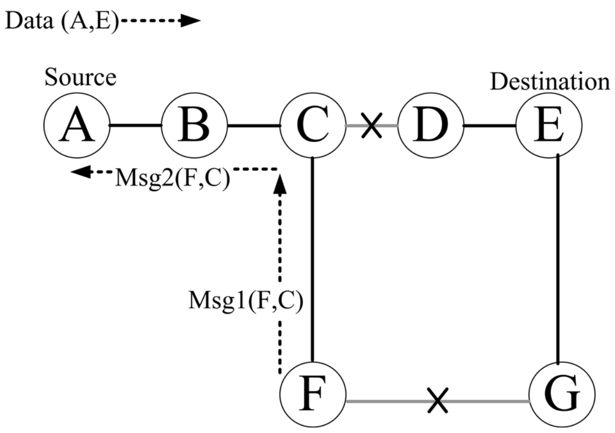

Fig. 4 EPLN mechanism of TCP-ECIA [17]

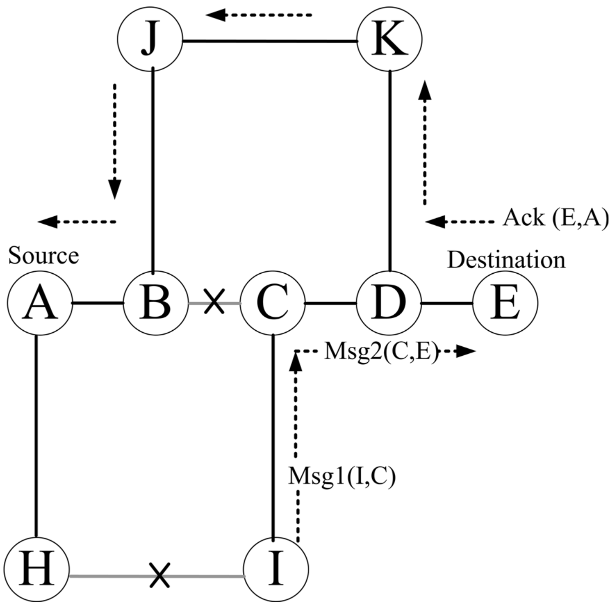

Fig. 5 BEAD mechanism of TCP-ECIA [17]

The process Best Effort ACK Delivery is shown in Fig. 5 [17]. A node E sends an ACK to node A using the route E–D–C–B–A. On failure of link C–B, C select alternate route CIHA. On failure of link I–H, Node I sends a message to node C, Msg1 (I, C). As the Notification_Message_1 is not from the TCP sender, node C sends a message to the TCP sender—E, Msg2(C,E). This notification is received by node D. If D finds an alternate route to node A, it uses it to forward the ACK, otherwise sends Msg2 (C, E) to Node E.

4.2.6 Comparison

Table 1 compares Cross Layered TCP variants to handle route failure losses.

Table 1 Cross layered TCP variants—route failure losses

Sr.	TCP variant	Features	Limitations
1	TCP-F [13] Feedback	Explicit Notifications. Non-probing method so it does not keep checking for a new route. An intermediate node may select alternate route	Expiration of route failure timer causes retransmission of all unacknowledged packets which may generate issues like contention, congestion, sudden traffic burst
2	TCP-ELFN [14] Explicit Link Failure Notification	Explicit Notifications. Probing based method so it may find a new route earlier than of TCP-F. There is no Route reestablishment process, so Intermediate nodes do not select alternate route	There is no Route reestablishment process. Sender has to initiate route reestablishment by periodic probing the network
3	TCP-BuS [15] Buffering capability and Sequencing information	Explicit Notifications. As a part of Route reestablishment process, Intermediate node may select alternate route. Selective transmissions and extended timeout mechanisms avoid unnecessary congestion control during a route failure. Avoids fast retransmissions by keeping records of lost packets during a route failure	Not suitable for a network with frequent route failures due to complexity
4	Adhoc TCP [16]	Combination of ICMP and ECN. ICMP based notification is used to identify route failure. ECN is used to identify congestion. 3 Duplicate ACKs and RTO events identify channel loss. A new route is explored to find suitable transmission rate	There is no Route reestablishment process. Sender has to initiate route reestablishment by periodic probing the network
5	TCP-ECIA [17] Exploiting Cross-layer Information Awareness	ELFN informs only sender about a route failure. While ECIA informs sender and receiver about a route failure as well as about discarded packets	Information of packets to be retransmitted is provided to sender and receiver but intermediate nodes do not buffer packets

4.3 Congestion Losses

4.3.1 TCP-ECN

TCP with ECN [18, 19] (Explicit Congestion Notification) uses feedback from intermediate nodes having Random Early Detection (RED) based routing capability. RED enabled nodes analyze average queue length to predict future congestion. RED enabled nodes drop packets with a predefined probability based on the current value of its queue. Two thresholds Min_{acl} and Max_{acl} are set for a queue. Packets are dropped with a predefined probability if average of queue length value is between Min_{acl} and Max_{acl}. Packets are not dropped if average of queue length value is less than Min_{acl}. All packets are dropped if average of queue length value is greater than Max_{acl}.

ECN is an extension of RED where an intermediate node marks packets instead of dropping. Such marking is used as an explicit congestion signaling for the TCP receiver and TCP sender. Packets are marked with a probability P if average of queue length value is between Min_{acl} and Max_{acl}. Packets are not marked if average of queue length value is less than Min_{acl}. All packets are marked if average of queue length value is greater than Max_{acl}.

$$P = ((Avg - Min_{acl})/(Max_{acl} - Min_{acl})) * P_{Max} \tag{5}$$

An intermediate node sets ECN bits of IP header to inform TCP receiver about future possibility of network congestion. On receiving a packet with IP header's ECN bits are set, TCP receiver sets ECE Explicit Congestion notification Echo bit of TCP header of next packet to inform TCP sender about congestion. On receiving an ECE marked packet, TCP sender reduces the transmission rate by activation of congestion control. TCP sender informs TCP receiver about reduction of transmission rate by setting CWR-Congestion Window Reduce flag in next packet's header. The rate of ECN marked packets can be used as a parameter of congestion control mechanism. In [20], average number of ECN marked packets is used to set receiver's advertised window.

4.3.2 TCP-RCWE

TCP-RCWE [21] (TCP-Restricted Congestion Window Enlargement) differentiates congestion loss and channel loss. RCWE uses ELFN messages to detect route failures. It estimates network state based on variation in consecutive RTO Retransmission Timeout values.

$$NetworkState = True, ifRTO_{New} \leq RTO_{Old} \tag{6}$$

$$NetworkState = False, ifRTO_{New} > RTO_{Old} \tag{7}$$

If current RTO is greater than previous RTO, it is considered as network congestion where TCP continues transmission without increasing Cwnd. If current RTO is less than previous RTO, TCP Cwnd increases Cwnd as per AIMD scheme. RCWE is a simple scheme which tries to avoid congestion.

4.3.3 C^3-TCP

C^3-TCP [22] (Cross layer Congestion Control-TCP) interacts with link layer to decide suit able transmission rate for Congestion Avoidance purpose. Every intermediate node partici pates in the process of measuring available bandwidth and total delay from the link layer.

Bandwidth Measurement
In a wireless network, available bandwidth is shared among all the nodes which are present in the range of sender and receiver. A node can measure available bandwidth (B=D/T) of a link by finding time (T) to send data of size (D). Every node uses medium access mechanism such as Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) with control signals like Request to Send (RTS), Clear to Send (CTS). C^3 TCP uses queuing delay, time required to acquire medium and time required for the data transmission for the calculation of total time T.

$$T = T_{out} - T_{in} + T_{tr} \tag{8}$$

T_{in} is the time a packet arrives at the link layer for transmission. T_{out} is the time a packet is transmitted. $T_{out} - T_{in}$ is the time representing queuing delay and delay for acquiring medium. T_{tr} is the time to transmit a packet and to receive corresponding ACK.

Delay Measurement TCP provides end-to-end delivery without having any information about the network structure like number of hops, link status etc. Every TCP connection works as a single data pipe between sender and receiver. TCP uses RTT to measure forward delay (sender to receiver) and backward delay (delay in receiving an ACK) within a pipe. In C^3 TCP, every node measures single hop forward delay which includes time required to acquire medium and time required for the data transmission. Figure 6 [22] shows IEEE 802.11 medium access mechanism.

C^3 TCP is implemented with modification of IEEE 802.11 MAC header's options field to propagate estimated bandwidth and delay information from sender to the receiver. $B_{end-to-end}$ is the minimum of all bandwidth values estimated by intermediate nodes as it indicates the bandwidth of the bottleneck link. $D_{end-to-end}$ is the total of all delay values estimated by intermediate nodes. $B_{end-to-end}$ and $D_{end-to-end}$ are sent to the TCP sender as a part of MAC header of a TCP ACK. C^3 TCP adjusts transmission rate as per the received bandwidth delay product. Figure 7 [22] shows how C^3 TCP estimates bandwidth and delay. Bab is bandwidth between A and B. Dab is delay between A and B.

4.3.4 *TCP-R*

TCP-R [23] is implemented with ADV-CC routing. TCP-R compares throughput and RTT with current value of Cwnd to decide cause of a RTO time out. TCP-R is used with Adhoc Distance Vector with Congestion Control (ADV-CC). ADV-CC calculates queue size to determine network congestion state. Every node shares such state with its neighbors and with the source. This information helps sender in controlling the transmission rate. The detail arithmetic is given in [23].

4.3.5 *Comparison*

Table 2 compares Cross Layered TCP variants to handle congestion losses.

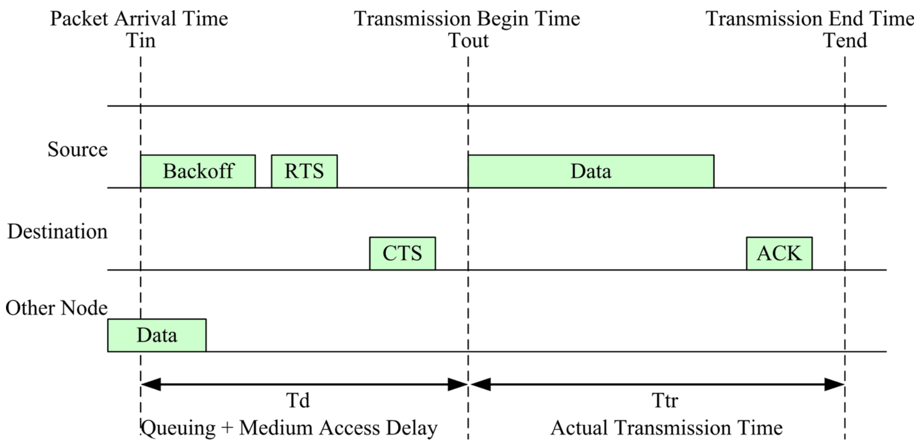

Fig. 6 IEEE 802.11 medium access mechanism [22]

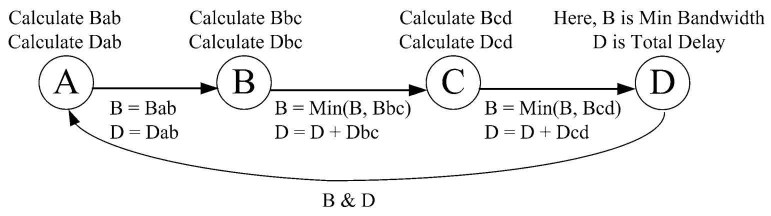

Fig. 7 C^3 TCP [22]

Table 2 Cross layered TCP variants—congestion losses

Sr.	TCP variant	Features	Limitations
1	TCP-ECN [18, 19] Explicit Congestion Notification	Intermediate nodes help in avoiding congestion. The rate of ECN marked packets can be used as a parameter of congestion control	Not suitable for high mobility based networks where intermediate nodes are changed frequently
2	TCP-RCWE [21] Restricted Congestion Window Enlargement	Increase and Decrease of RTO value is used to decide transmission rate. Implementation is easy and less complex	ELFN messages are used to handle route failures. There is no improvement over TCP-ELFN as far as only route failures are concerned
3	C^3 TCP [22] Cross layer Congestion Control	It estimates end-to-end bandwidth delay product from the data link layer. This variant estimates transmission rate more accurately	C^3 TCP is complex to implement for wireless adhoc networks. It also needs to be implemented at every intermediate node for bandwidth delay calculation
4	TCP-R [23] RTT based TCP	TCP is used with AODV's variant Adhoc Distance Vector with Congestion Control (ADV-CC)	Difficult to implement as routing protocol needs to be modified at every node

4.4 Channel Losses and Contention Issues

4.4.1 TCTC

TCTC [24] (TCP based ConTention Control) addresses the issue of intra-flow contention. Intra-flow contention refers to the self interference within a TCP connection. Self interference refers to the collisions across the packets belonging to the same TCP connection. Self interference could be any of the following three categories.

- Self interference between TCP Packets (Data and ACK).
- Self interference between IEEE 802.11 Control Packets (RTS and CTS).
- Self interference between TCP Packets and IEEE 802.11 Control Packets.

Nodes participating in a single TCP connection may compete for the channel access at a time. In such situations, a few packets might collide and lost. TCP's intra-flow stability refers to the transmission of limited number of packets which could be properly handled by the network. Large number of outstanding packets may cause intra-flow contention issues. Less number of outstanding packets may underutilize the available bandwidth. Intra-flow contention increases link layer retransmissions which increases retransmission by TCP due to timeouts. TCTC finds the most suitable number of outstanding packets based on available bandwidth and level of contention. The problem is shown in Fig. 8 [24].

Contention Window

TCTC is a receiver side modification to probe the network periodically for finding level of contention in the form of contention delay. TCP receiver monitors the actual throughput and level of contention during a probe period. TCP receiver maintains Contention Window (Ctwnd) to represent level of contention in terms of number of packets. TCP receiver sends minimum of Ctwnd (Preferred transmission rate based on level of contention) and Rwnd (Preferred transmission rate based on receiver's buffering capacity) to the TCP sender as a part of TCP header's WindowSize field. TCP Sender sets the transmission rate to the minimum of WindowSize and Cwnd (preferred transmission rate based on network's congestion). TCTC's probe process has four stages.

Fast Probe

Fast Probe is similar to the Slow Start phase of TCP. TCTC starts with this phase where Ctwnd is increased exponentially (doubles every probe interval). TCTC switches to this phase after recovering from severe contention phase.

Slow Probe

TCTC enters into the Slow Probe phase on detection of decreasing throughput and decreasing contention delay as compared to the previous probe interval. This phase increases Ctwnd linearly (1 packet every probe interval).

Light Contention

TCTC enters into the Light Contention phase on detection of increasing throughput as well as increasing contention delay compared to the previous probe interval. This phase decreases Ctwnd linearly (1 packet every probe interval).

Severe Contention

TCTC enters into the Severe Contention phase on detection of decreasing throughput but increasing contention delay as compared to the previous probe interval. This phase sets Ctwnd to its initial size of (2 packets). The selection of probe interval is a critical task. If it is too large, TCTC will not able to adopt network changes quickly. If it is too small, TCTC will keep changing Ctwnd frequently. Probe interval of one RTT is suitable for Ctwnd's stable measurement.

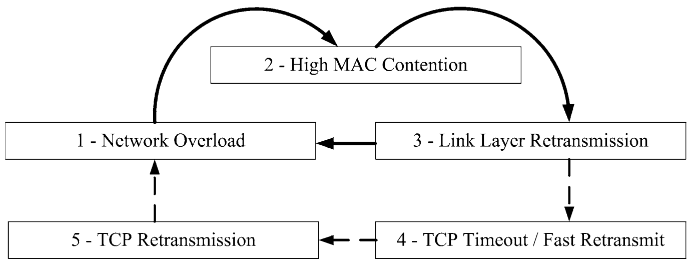

Fig. 8 TCP Issue [24]

4.4.2 COPAS

TCP with COPAS [25] (TCP with Contention based PAth Selection) uses two mechanisms to deal with intra-flow contention. Two disjoint routes (having no common intermediate nodes) are used for forward (sender to receiver—Data traffic) and reverse (receiver to sender-ACK traffic) traffic to reduce contentions inside a single TCP connection. COPAS's dynamic contention balancing mechanism keeps changing forward and reverse routes based on corresponding level of contention.

Route Establishment

COPAS can be implemented with any of the reactive routing protocols such as AODV, DSR. During route establishment process, every node has to broadcast Route Request packet (RREQ). Every node keeps record of average number of times it has to backoff while trying to broadcast RREQ. This information helps a node to measure level of contention in its neighborhood. Average number of backoff attempts information is propagated along with the RREQ packet to the receiver (TCP receiver to which a route needs to be established). Receiver has a RREQ Collection timer in which it accepts all RREQ to find multiple routes along with their contention levels. On expiration of this timer, Receiver selects two routes based on two criteria: Disjointness and Routes with less contention. Receiver replies with two RREP Route Reply packets with direction flag set to inform sender about forward and reverse routes. Disjoint routes are also useful when one of the routes is failed. Packets could be rerouted on another route until a failed route is restored.

Dynamic Contention Balancing

Dynamic topology may change the level of contention in any route any time. A route with less contention may experience severe contention later on. COPAS periodically measures contention and initiate Route Construction if corresponding route is experiencing high contention. Sender and receiver monitor the contention status of reverse and forward route respectively. If either of these route experiences contention beyond a threshold, it is replaced by a newly constructed and overall less contended route.

The problem is shown in Fig. 9 [25].

4.4.3 Comparison

Table 3 compares Cross Layered TCP variants to handle channel losses and contention issues.

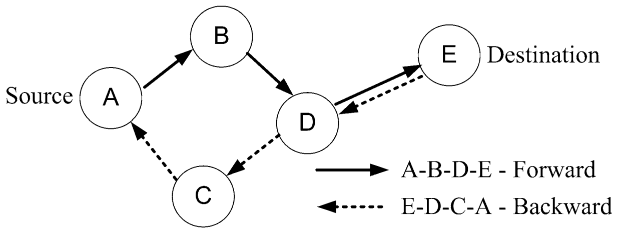

Fig. 9 TCP Issue [25]

Table 3 Cross layered TCP variants—channel losses and contention issues

Sr.	TCP variant	Features	Limitations
1	TCTC [24] TCP based ConTention Control	Transmission rate is decided as per the throughput and contention delay experienced by receiver	A network may get under utilized or over utilized if the time interval is not appropriate
2	COPAS [25] Contention based PAth Selection	It uses two distinct routes for forward and backward traffic to avoid contention. It reduces Intra-flow contention efficiently	Applicable only with reactive routing protocols. As this method is completely independent of TCP, there is no improvement at TCP in absence of contention issues

5 Layered Approaches

5.1 Introduction

Layered approach based TCP does not interact with lower layers to get decision making information. There is no explicit feedback from any of the lower layers. TCP itself tries to find out various issues which may arise at one of the lower layers. TCP being a transport layer protocol, uses its own statistics to predict route failure, channel loss or contention related issues. Layered approaches are simple to implement as well as do not violate layer independence property of a layer based network.

5.2 Route Failures Losses

5.2.1 Fixed RTO

TCP with Fixed RTO [26] is a simple heuristic, implemented with TCP Reno to identify possibility of a route failure. Traditional TCP increases value of RTO timer exponentially (doubles on every timeout event). This process is called TCP's exponential back off mechanism. Fixed RTO assumes a route failure when there is no ACK is received between two consecutive timeout events. In such situation, RTO is not increased and TCP continues with its AIMD scheme. In other cases, Fixed RTO follows TCP's exponential back off mechanism to increase RTO value.

5.2.2 TCP-DOOR

TCP-DOOR [27] (TCP-Detection of Out-of-Order and Response) uses the order of delivered packets to predict a route failure. On a route failure, a few packets are not delivered in the same order in which they were sent. Frequent route failures cause retransmissions of such packets which TCP receiver receives later on causing out-of-order delivery. TCP sender and TCP receiver detects out-of-order deliverers of ACKs and data packets respectively. DOOR considers out-of-delivery at TCP. TCP always provide ordered delivery to the higher level application.

Detection of Out-of-Order ACKs

TCP ACK's sequence number specifies how much data has been received in order. Every ACK has a sequence number which is larger than of any other previous ACKs. TCP's inherent monotonically increasing property of ACK numbering makes it easy to detect out-of-order delivery. If a TCP sender receives an ACK with a sequence number smaller than of any of the previously received ACKs, it is considered as an Out-of-Order delivery. As ACKs are never retransmitted there is no possibility of out-of-order delivery due to retransmissions. DUPACKs-Duplicate acknowledgements carry same sequence number so it is not possible to find out-of-order delivery among a set of DUPACKs having same sequence number. DOOR uses TCP header's option field to send ADSN-ACK Duplicate Sequence Number to set order among a same set of DUPACKs. ADSN of 1st DUPACK is set to 0 and incremented by 1 for every additional DUPACK.

Detection of Out-of-Order Data

Ordered delivery of Data packets is violated in case of retransmissions because sequence numbers of retransmitted data packets remain same as of earlier transmissions. As data packets do not carry any inherent ordered property across the retransmissions, an explicit ordering is required. DOOR uses TCP header's option field to send TPSN-TCP Packet Sequence Number. TPSN of 1st data packet is set to 0 and incremented by 1 for every data packet sent including retransmissions. TCP receiver can easily find out out-of-order delivery by checking TPSN values. ADSN and TPSN could be set to the timestamps also.

DOOR Response

Out-of-Order delivery is considered as a cause of a route failure. This information is mainly used by TCP sender to avoid activation of congestion control unnecessarily. TCP sender detects out-of-order ACKs directly. TCP receiver informs TCP sender about out-of-order data packets by setting OOO bit in TCP ACK header. TCP sender disables congestion control temporarily (for a period T1) by not changing values of variables like congestion window, RTO timer etc. If TCP sender has recently (during past period of T2) switched to Congestion Avoidance phase, it immediately restores its state to the state prior to the Congestion Avoidance phase. The reason is the assumption that a recent route failure had probably caused activation of congestion control unnecessarily. T1 and T2 values are set dynamically with reference of RTT values.

5.2.3 ADTCP

ADTCP [28] handles four states: Congestion, Channel Error, Route Change and Disconnection. ADTCP introduces four end-to-end measurements: Inter Delay Difference (IDD), Short Term Throughput (STT), Packet Out-of-order delivery Ratio (POR) and Packet Loss Ratio (PLR). These measurements are used to detect various states. Multiple measurements are used to detect a state for accuracy purpose.

Measurements

$$IDD = A_{i+1} - A_i - (S_{i+1} - S_i) \tag{9}$$

$$STT = N_p(T)/T \tag{10}$$

$$POR = N_{po}(T)/N_p(T) \tag{11}$$

$$PLR = N_l(T)/N_p(T) \tag{12}$$

A_i is arrival time of *ith* packet.
S_i is sending time of *ith* packet.
$N_p(T)$ is number of received packets during interval T.
$N_{po}(T)$ is number of out-of-order packets during T.
$N_l(T)$ is number of lost packets during interval T.

State Identification

A congestion state is identified when IDD is High and STT is Low. In other cases, non congestion state is identified. A non congestion state needs to be detected as either a channel error or a route change or a disconnected state. A route change state is identified when POR is High. A channel error state is identified when PLR is High. A disconnected state is identified when retransmission time out occurs along with any of the non congestion states. Relative sample density based classification is used to define values of High and Low thresholds. Congestion state activates congestion control. Channel error state retransmits lost packets. Route change state retransmits lost packets and sets Cwnd. Disconnect state probes network for a new route.

5.2.4 Comparison

Table 4 compares Layered TCP variants to handle route failure losses.

Table 4 Layered TCP variants—route failure losses

Sr.	TCP variant	Features	Limitations
1	TCP-Fixed RTO [26]	The interval between two RTO timeout events is analyzed to check for a route failure. It is easy to implement with any of the AIMD based TCP variants	It is a heuristic which does not ensure performance improvement in all situations
2	TCP-DOOR [27]Detection of Out-of-Order and Response	Unordered delivery is considered as a cause of a route failure. Other than TCP's inherent sequencing mechanism, explicit sequencing is used. It detects route failures efficiently	It is not able to adopt with the new route immediately. Packet sequencing needs additional bytes of header too
3	ADTCP [28]	New measurements are used to define four states. Congestion, Channel Error, Route Change, Disconnected. Multi measurement based decision making increases accuracy	Difficult to define the threshold values in terms of Low and High for various measurements

5.3 Congestion Losses

5.3.1 TCP-WELCOME

TCP-WELCOME [29] (TCP-Wireless Environment, Link losses and COngestion packet loss ModEls) differentiates congestion loss, route failure loss and channel loss. It analyzes the amount of changes in RTT values. TCP WELCOME has two parts: Loss Differentiation Algorithm and Loss Recovery Algorithm.

LDA-Loss Differentiation Algorithm
TCP WELCOME sender activates LDA on either 3 DUPACK event or RTO timeout event. TCP WELCOME keeps analyzing RTT values. LDA uses this information to differentiate a loss. If RTT values are continuously increasing with time, a loss is considered as a cause of network congestion irrespective of type of event (3 DUPACK or RTO timeout). If RTT values are not fluctuating and remain almost stead around average, a loss is considered as a cause of a non congestion event. TCP WELCOME further differentiates non congestion loss into two categories: Route failure loss or channel loss. If TCP sender experiences 3 DUPACK event then the cause is assumed to be a channel loss. If TCP sender experience RTO timeout then the cause is assumed to be a route failure loss.

LRA-Loss Recovery Algorithm
TCP WELCOME retransmits the lost packet immediately in all the cases. Later on it takes appropriate actions for RTO timer value and transmission rate. RTO timer value and transmission rate are changed as per the congestion control scheme in case of a congestion loss. RTO timer value and transmission rate are not changed in case of a channel loss. RTO timer value and transmission rate are changed as per the statistics of a new route in case of a route failure loss. RTT_{New} is as per the new route and RTT_{Old} is as per the previous route.

$$RTO_{New} = (RTT_{New}/RTT_{Old}) * RTO_{Old} \tag{13}$$

$$CWND_{New} = (RTT_{Old}/RTT_{New}) * CWND_{Old} \tag{14}$$

5.3.2 TCP-Westwood

TCP-Westwood [30] is a sender side modification to handle congestion with consideration of possibility of channel losses. TCP Reno always halves values of Congestion Window (Cwnd) and Slow Start Threshold (SSThresh) on three duplicate ACKs event. TCPW estimates end-to-end bandwidth to set values of Cwnd and SSThresh according to the level of congestion. TCPW analyses the flow of ACKs for bandwidth estimation. Faster recovery phase is introduced to set values of Cwnd and SSThresh.

Bandwidth Estimation
TCPW sender monitors average rate of ACKs for bandwidth estimation. Bandwidth is estimated by averaging amount of delivered data over time. TCPW uses ACKs rate and information carried by every ACK (amount of data delivered) to calculate bandwidth samples. For example, if TCPW receives an ACK for a packet of d_k bytes at time t_k and t_{k-1} is the time at which previous ACK was received, Sample bandwidth b_k is $d_k/(t_k - t_{k-1})$. Low pass filter based averaging is used over sample bandwidths to deal with

delayed acknowledgements. On 3 duplicate ACKs or timeout event values of Slow Start threshold is changed as below. Bandwidth Estimated (*BWE*). *SegSize* is Size of payload in TCP Segment.

$$SSThresh = (BWE * RTTmin)/SegSize \tag{15}$$

On three duplicate acknowledgement event, Cwnd is set to maximum of current value of Cwnd and new value of SSThresh. On timeout vent, Cwnd is set to its default initial size.

5.3.3 TCP-Jersey

TCP-Jersey [19] is based on the similar concept of TCP-Westwood to estimate available bandwidth to set transmission rate accordingly. The process of Available Bandwidth Estimation (ABE) is based on analyzing the flow of ACKs at TCP sender. This information is used to find SSThresh and Cwnd values. One key difference between TCP-Jersey and TCP-Westwood is that the former uses ECN type mechanisms to notify TCP sender about congestion explicitly. TCP-Jersey introduces simple way of explicit congestion notification with CW-Congestion Warning scheme. CW uses the same fields of IP header and TCP header used by ECN but the parameters and the process of finding average queue length is simple.

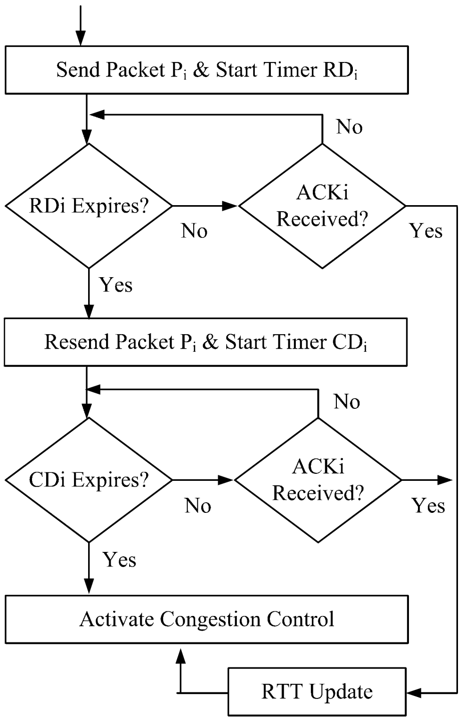
Send Packet Pi & Start Timer RDi
No
RDi Expires?
No
ACKi Received?
Yes
Yes
Resend Packet Pi & Start Timer CDi
No
CDi Expires?
No
ACKi Received?
Yes
Yes
Activate Congestion Control
RTT Update

Fig. 10 TCP-NCL flow chart [31]

5.3.4 TCP-NCL

TCP-NCL [31] (TCP-Non Congestion Losses) modifies congestion control mechanism to handle packet reordering and channel loss issues. TCP-NCL separates packet retransmission mechanism from the congestion control mechanism. Retransmission Decision (RD) Timer and Congestion Response Decision (CD) Timer are used to trigger retransmission and congestion control mechanisms respectively. TCP-NCL sends a packet P_i and starts RD_i timer. RD_i is cancelled if ACK is received for P_i. On expiration of RD_i, P_i is retransmitted and CD_i timer is started. CD_i is cancelled if ACK is received for P_i. On expiration of CD_i, Congestion control is activated. The values of RD and CD timers are very crucial. TCP-NCL uses statistical distribution of RTT values to set these values dynamically. The problem is shown in Fig. 10 [31].

5.3.5 TCP-FIT

TCP-FIT [32] uses virtual TCP sessions inside a single TCP connection for efficient calculation of Cwnd value. The algorithm is based on using packet loss and packet delay information. Packet loss information is used to calculate Cwnd value in each of the virtual TCP sessions. Packet delay information is used to set number of virtual TCP sessions dynamically. A TCP-FIT connection is logically a set of N TCP Reno virtual sessions. Cwnd is calculated as below. The detail arithmetic is given in [32].

$$\begin{aligned} EachRTT &: Cwnd = Cwnd + \gamma N \\ EachLoss &: Cwnd = Cwnd - (Cwnd/2N) \end{aligned} \tag{16}$$

$$\gamma = RTT/Base_{RTT} \tag{17}$$

5.3.6 Comparison

Table 5 compares Layered TCP variants to handle congestion losses.

5.4 Channel Losses and Contention Issues

5.4.1 TCP-VENO

TCP-VENO [33] is a combination of TCP Vegas and TCP Reno. TCP Vegas monitors network congestion to set Cwnd for Congestion Avoidance purpose. TCP Vegas tries to prevent further congestive packet losses without differentiating current loss. TCP Veno monitors network congestion in the same way as TCP Vegas, not to set Cwnd but to differentiate congestion loss and channel loss. Any loss which is found during congestion state is considered as congestion loss. Any other loss is considered as a channel loss. TCP Veno follows TCP Reno as a base variant. TCP Reno's Multiplicative Decrease phase is modified to set value of SSThresh as per the level of congestion rather than with a fixed decrease of half in all cases. TCP Reno's Congestion Avoidance phase is modified to select increment Cwnd by 1 every ACK or increment Cwnd by 1 every other ACK according to the level of congestion.

Table 5 Layered TCP variants—congestion losses

Sr.	TCP variant	Features	Limitations
1	TCP-WELCOME [29]Wireless Environment, Link losses and COngestion packet loss ModEls	Values of RTTs are used to differentiate causes of 3 Duplicate ACKs and RTO Timeout events. Easy to implement with any of the AIMD based TCP variants	Loss recovery algorithm is less efficient as it uses only one new RTT value to set RTO and transmission rate. Only RTT based decision making is inaccurate
2	TCP-Westwood [30]	Handles congestion with consideration of possibility of channel losses. ACK flow is used to estimate available bandwidth to set values of Cwnd and SSThresh	It does not differentiate non congestion losses from congestion losses
3	TCP-Jersey [19]	Simple methods to estimate bandwidth as compared to TCP Westwood. Explicit congestion notifications with Congestion Warning messages	Explicit congestion notifications are less costly than TCP-ECN still it requires to be done by an intermediate node. Not suitable for high mobility based networks where intermediate nodes are changed frequently
4	TCP-NCL [31]Non Congestion Losses	It is a single solution to avoid unnecessary activation of congestion control in case of unordered delivery or losses due to channel issues	It is difficult to set appropriate values of Retransmission Decision (RD) Timer and Congestion Response Decision (CD) Timer
5	TCP-FIT [32]	Virtual TCP sessions inside a single TCP connection helps in finding Cwnd value accurately	Complex process as it requires more computation at TCP Sender

5.4.2 ACK Thinning Techniques

ACK Thinning Techniques [34] target intra-flow contention issues. The available bandwidth is shared for the data flow as well as for the ACK flow of a TCP connection. ACK Thinning is a process of reducing the rate of ACKs to spare more bandwidth for the data packets. ACK Thinning reduces contention issues as well as helps in increasing throughput. Most of the TCP variants are ACK clocking where the transmission rate depends on the ACK rate. In such situation, a large reduction of ACK rate restricts increase in transmission rate. It may generate unnecessary timeouts too. ACK Thinning Techniques provide systematic way to decide up to what amount reduction of ACK rate is suitable without underutilizing a network and without activating congestion control unnecessarily.

TCP's concept of cumulative acknowledgement is one of the simplest ACK Thinning scheme called *TCP-DA* [34] (TCP with Delayed ACK). TCP-DA sends a combine ACK for every 2 packets. In case of an out-of-order packet, it sends an ACK immediately.

TCP-ADA [35] (Adaptive Delayed ACK) tries to reduce number of ACKs to 1 per Cwnd. TCP-ADA calculates Average Packet Interval (API) from the inter arrival times of packets. API is used to define Acknowledgement Expiration Timer (AET). The maximum postponement time is defined by MDT Timer-Maximum Defer Time (500 ms).

API is calculated using previous API and Last Packet Arrival (LPA) time as below. α is set to 0.8

$$API = \alpha * API + (1 - \alpha) * (Now - LPA) \tag{18}$$

AET Timer is calculated using API as below. β is set to 0.8

$$AET = \beta * API \tag{19}$$

An ACK is postponed until expiration of either AET timer or of MDT timer.

TCP-DDA [36] (Dynamic Delayed ACK) avoids temporary pause in transmission when the size of Cwnd is very small and ACK delay is very high. In such case, TCP sender waits for an ACK to increase its transmission rate. At the same time, TCP receiver waits for arrival of a few packets to send ACK. Subsequently timeout may occur at TCP sender. TCP-DDA sets number of ACKs to delay (number of packets per cumulative ACK) (1, 2, 3 or 4) dynamically. Initially delay parameter is set to 1. Delay parameter is increased or decreased as per the number of in-order received packets.

5.4.3 Comparison

Table 6 compares Layered TCP variants to handle channel losses and contention issues.

6 Recent TCP Variants

6.1 Introduction

This section discusses some of the recent TCP variants based on the way they handle losses. These TCP variants are classified into loss differentiation, loss avoidance and loss prediction based approaches.

Table 6 Layered TCP variants—channel losses and contention issues

Sr.	TCP variant	Features	Limitations
1	TCP-VENO [33]	The estimated bandwidth is used to differentiate losses	Not suitable for asymmetric network. Even if forward path has no congestion, VENO will not allow TCP sender to increase Cwnd in case of congestion in backward path
2	ACK Thinning Techniques [34] TCP-DA [34] TCP-ADA [35] TCP-DDA [36]	Various techniques try to reduce flow of ACKs to provide more bandwidth for data flow. These techniques help TCP to reduce intra-flow contention easily	It is difficult to associate an ACK Thinning technique with a TCP variant. Selection of one ACK Thinning technique in all scenarios may not be efficient

6.2 Loss Differentiation

6.2.1 TCP-NRT

TCP-NRT [37] (TCP Non congestion Retransmission Timeout) detects non congestion events such as spurious delay and random packet loss. TCP-NRT differentiates RTO timeout events into congestion RTO and non congestion RTO (spurious RTO and RTO due to random packet loss). Sudden delay in communication may cause timeout which is considered as a spurious RTO. Channel loss may cause timeout which is considered as a RTO due to random packet loss. TCP-NRT has three phases: Detection, Differentiation and Reaction.

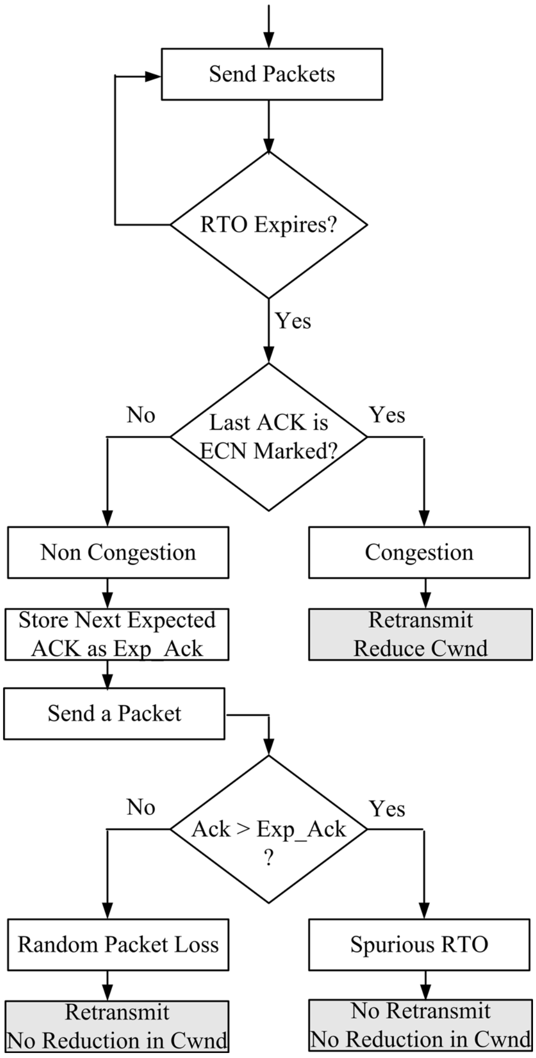

Fig. 11 TCP-NRT [37]

Detection TCP-NRT detects non congestion RTO from congestion RTO using modified Explicit Congestion Notification (ECN). In Modified ECN, an intermediate node marks a packet only when the Active Queue Length (AQL) is greater than Maximum Threshold. Modified ECN improves bandwidth utilization as packets are not marked in other conditions (discussed in Sect. 4.3.1). As TCP sender gets comparatively less number of ECN marked packets, Cwnd is not reduced frequently. On occurrence of RTO event, ECN status of last (most recent) ACK is checked. If it is ECN marked, congestion RTO is considered otherwise non congestion RTO is considered.

Differentiation A non congestion RTO is further differentiated into either a spurious RTO or a RTO due to random packet loss. This phase sets *Seq_Exp_Ack* to the sequence number of the next expected ACK. As a part of differentiation process, a fresh packet is sent. On receiving an ACK, its sequence number (*Seq_New_Ack*) is checked. If $Seq\_Exp\_Ack < Seq\_New\_Ack$ then spurious RTO is considered otherwise RTO due to random packet loss is considered. In case of a sudden delay, no packet gets lost and so the expected ACK can be achieved in the form of a cumulative ACK of a new packet.

Reaction On congestion RTO, congestion control mechanism of TCP NewReno is activated. On spurious RTO, new packets are sent based on the current value of Cwnd. On RTO due to random packet loss, lost packet is retransmitted and further transmission is continued without changing Cwnd. The complete process is shown in Fig. 11 [37].

6.2.2 TCP-NCE

TCP-NCE [38] (TCP Non Congestion Events) detects non congestion events such as random packet loss and unordered delivery issue on three duplicate ACK event. These issues generate holes (missing packets) in the TCP receiver window. Subsequently TCP receiver generates duplicate ACKs for these missing packets. These duplicate ACKs activate congestion control at TCP sender to reduce transmission rate. Wireless networks often experience random packet loss due to transmission issues and unordered delivery due to various reasons like multi-path routing, parallelism, retransmissions etc. TCP-NCE tries to avoid unnecessary reduction of transmission rate on three duplicate ACKs event, even before RTO timeout occurs. TCP-NCE has three phases: Detection, Differentiation and Reaction.

Detection TCP-NCE detects non congestion event from congestion event by measuring queue length of bottleneck link. Timestamp based RTT calculation is used for accuracy purpose. TCP sender sends current timestamp (TS_i) as a part of options field of TCP header of *ith* packet. TCP receiver sends TS_i back to the TCP sender as a part of an ACK of *ith* packet. TCP sender calculates RTT for *ith* packet and queue length as below.

$$RTT = Current_{Time} - TS_i \tag{20}$$

$$QL_{Now} = (RTT_{Now} - RTT_{Min}) * B \tag{21}$$

B is available Bandwidth.
RTT_{Now} is the current (most recent) RTT.
RTT_{Min} is the minimum RTT so far.

A threshold value is set with reference of buffer space at the queue of intermediate nodes. Detail of setting this threshold value is given in [38]. On receiving three duplicate ACKs, if current value of QL_{Now} is greater than threshold value then congestion event is considered. Otherwise non congestion event is considered.

Differentiation A non congestion event is further differentiated into either a random packet loss or an unordered delivery. A Delay_Thresh-Dynamic Delay Threshold value is calculated based on the number of outstanding packets on three duplicate ACK event. Delay_Thresh is calculated as below.

$$No\_PPkts = No\_SPkts - No\_APkts \tag{22}$$

No_PPkts is Total number of pending packets.
No_SPkts is Total number of sent packets.
No_APkts is Total number of acknowledged packets out of *No_SPkts*.

As three duplicate ACKs are corresponding to reception of three packets, the remaining number of packets to be delivered is calculated as below.

$$Delay\_Thresh = No\_PPkts - No\_DACKs \tag{23}$$

Delay_Thresh is used for retransmission delay. TCP sender neither retransmits oldest unacknowledged packet nor reduces Cwnd immediately. TCP sender sends a fresh packet (Cwnd is incremented) and waits for ADack (Additional Duplicate Acknowledgement). For each ADack, a fresh packet is sent (Cwnd is incremented). When TCP sender receives n ADacks (including initial three duplicate ACKs) equal to or greater than Delay_Thresh, random packet loss is considered. The reason is that TCP receiver generates n number of ADacks for two types of packets: Those packets which were in transmission at the time of calculation of Delay_Thresh and those fresh packets which were sent as a part of differentiation process. A random packet loss is considered on receiving a same duplicate ACK even after reception of all outstanding and fresh packets. In other cases when TCP sender receives higher ACKs, unordered delivery issue is considered. Unordered packets are expected to be delivered earlier than of fresh packets.

Reaction On congestion event, congestion control mechanism of TCP NewReno is activated. On random packet loss, lost packet is retransmitted. On unordered delivery, TCP continues transmission with its current state and transmission rate.

6.2.3 Enhanced TCP-NCE

Enhanced TCP-NCE [39] improves performance of TCP-NCE [38]. TCP-NCE needs accurate detail of bottleneck bandwidth to differentiate congestion event from non congestion events. As it is difficult to estimate bottleneck bandwidth accurately in MANETs, Enhanced TC-NCE replaced TCP NCE's loss differentiation process with a loss discrimination process.

Discrimination Queuing delay based two values are calculated as below.

$$QD = RTT_{Current} - RTT_{New} \tag{24}$$

$$QD_{Max} = RTT_{Max} - RTT_{Min} \tag{25}$$

The ratio of QD and QD_{Max} is compared with a threshold $\Lambda\ (0.5)$. If the ratio is less than Λ, congestion loss is considered otherwise channel loss is considered. It also tries to handle packet reordering issue more precisely using SACK option. A receiver sends information about missing packets using SACK field of a Duplicate ACK. This information is used by

TCP sender to update list of delivered packets. Initially, TCP sender considers a loss (Receiving three Duplicate ACKs) as a channel loss. At this stage, it continues transmission similar to of TCP - NCE by processing a series of Duplicate ACKs. Once number of Duplicate ACKs becomes 80% of the current size of Cwnd, packet loss is reconsidered as a packet reordering issue.

6.2.4 TCP-CERL

TCP-CERL [40] (Congestion control Enhancement for Random Loss) is conceptually similar to TCP-VENO [33]. TCP-CERL also monitors level of congestion to differentiate random packet loss and congestion loss. Every RTT is considered to be of three values: Queuing delay, Propagation delay and Service delay. On every new calculation of RTT, QL-Queue length at bottleneck link is estimated as below.

$$QL = (RTT_{Now} - RTT_{Min}) * B \tag{26}$$

RTT_{Now} is current RTT.
RTT_{Min} is minimum of all RTT values so far.
B is estimated bandwidth.

TCP-CERL sets dynamic queue length threshold as below.

$$QL_{Thresh} = A * QL_{Max} \tag{27}$$

QL_{Max} is largest value of QL so far and A = 0.55.

TCP-CERL updates values of RTT_{Min}, QL, QL_{Max} and QL_{Thresh} on every new RTT (RTT_{Now}). On three duplicate ACKs event, current values of QL and QL_{Thresh} are used to differentiate random packet loss from congestion loss. If $QL < QL_{Thresh}$ then random packet loss is considered, otherwise congestion loss is considered. In case of a random packet loss, retransmission is performed without changing TCP state. In case of a congestion loss, TCP-CERL ensures that the Cwnd is reduced only once per window even if multiple packets are lost within a window. TCP-CERL performs better than TCP-VENO due to dynamic threshold and maximum one time reduction of transmission rate per window.

For example, a TCP sender sends packets 1–10. Packets 2 and 7 are lost. On receiving three Duplicate ACKs for packet 2, TCP sender retransmits packet 2. It stores the highest sequence number of sent packet (the time congestion control is activated) to record most recent Cwnd reduction point. Later on, TCP receiver sends three Duplicate ACKs for packet 7 too. At this moment, TCP sender finds that the sequence number of requested packet (7) is smaller than the most recent Cwnd reduction point (10). This condition is considered as multiple packet loss in a single transmission. So packet 7 is retransmitted without reduction of Cwnd.

6.2.5 Packet Reordering Solution

A cross layer solution has been proposed to deal with packet reordering [41] issue raised due to mobility or link layer retransmissions. A trust based routing protocol has been proposed to find optimal route. This approach enables TCP sender to identify which packets are dropped due to congestion and which packets are reordered. Each intermediate node maintains a table to maintain a list of records to keep information about discarded packets per TCP flow. With every TCP flow, two numbers are stored: Min_{Drop} and

Max_{Drop}, Minimum and Maximum sequence numbers of dropped packets respectively. With every discarded packet, corresponding record is updated or inserted in the table. With every non discarded packet, corresponding record is sent along with it to the receiver. TCP receiver uses this information to maintain a reorder list a list of yet to be received packets which will need to be reordered and dropped list a list of already dropped packets which will need to be retransmitted. TCP receiver informs TCP sender about the number of packets in current reorder list K using a field named as Reorder bits in TCP's header of ACK. TCP sender waits for 3 + K number of Duplicate ACKs to smooth out the packet reordering effect. TCP receiver receives a packet with various possible drop details and other values as below.

Min_Drop is Value of Minimum Drop field of current packet.
Max_Drop is Value of Maximum Drop field of current packet.
Seq_H is Highest sequence number of the received packets.
Seq_C is Sequence number of current packet.

Proposed solution has many cases. Two of them are discussed here. Other cases are listed in [41].

Case 1: If current packet has no values for *Min_Drop* and *Max_Drop*

If $Seq\_C > Seq\_H$, all packets between *Seq_H* and *Seq_C* are considered as dropped. The rest of the packets are added in reorder list.

If $Seq\_C < Seq\_H$, current packet is removed from the reorder list.

Case 2: If current packet has values for *Min_Drop* and *Max_Drop*, it is removed from the reorder list.

Table 7 TCP variants—loss differentiation

Sr.	TCP variant	Features	Limitations
1	TCP-NRT [37] Non congestion Retransmission Timeout	A systematic way to handle losses. Modified ECN is more efficient as compared to ECN	Activation of loss handling is late(RTO timeout only)
2	TCP-NCE [38] Non Congestion Events	A systematic way to handle losses. Activation of loss handling is earlier (On three duplicate ACKs)	Difficult to find a bottleneck link in case of high mobility
3	Enhanced TCP-NCE [39] Enhanced TCP with Non Congestion Events	Bandwidth estimation of bottleneck link is removed. SACK information is used to identify missing packets	Thresholds need to be set dynamically
4	TCP-CERL [40] Congestion control Enhancement for Random Loss	It performs better than TCP-VENO due to dynamic thresholds. Only one time transmission rate reduction for multiple losses within a single window	Difficult to find a bottleneck link in case of high mobility. The decision making mainly depends on queue length a bottleneck link which is inefficient
5	Packet Reordering Solution [41]	TCP Sender is informed about out of order delivery so it can process duplicate ACKs accordingly	It needs to be implemented at every intermediate node too. Not suitable for high mobility environment

If $Seq\_C > Seq\_H$ and $Min\_Drop > Seq\_H$ and $Max\_Drop < Seq\_C$ then packets between Seq_H and Min_Drop, Max_Drop and Seq_C are added in reorder list. Packets between Min_Drop and Max_Drop are removed from the reorder list.

6.2.6 Comparison

Table 7 compares recent and advanced loss differentiation based TCP variants.

6.3 Loss Prediction

6.3.1 TCP-FR

TCP-FR [42] (TCP with Fastest Retransmission) introduces another phase called Fastest Retransmission over AIMD phases. TCP-FR avoids pauses in transmission due to large number of random packet losses or packet drops due to congestion. TCP receiver is unable to send duplicate ACKs if it does not receive any packets after a loss. In such situation, TCP sender does not receive sufficient number of duplicate ACKs (mostly 3) to activate Fast Retransmission phase. Subsequently, TCP sender experiences RTO timeout which slows down transmission rate unnecessarily. During this duration, there is a pause in transmission too. TCP-FR predicts possibility of such pauses and provides earlier retransmission with its Fastest Retransmission phase. If TCP sender does not receive any ACK within a Fastest Retransmission Time T1, an oldest unacknowledged packet is retransmitted to keep transmission going on. The process is shown in Fig. 12 [42].

Value of timer T1 is dynamically set based on the values of RTT and RTO as shown below.

$T1 = 0.5 * RTT$

Where $0.2 * RTO < 0.5 * RTT$

$T1 = 0.2 * RTO$

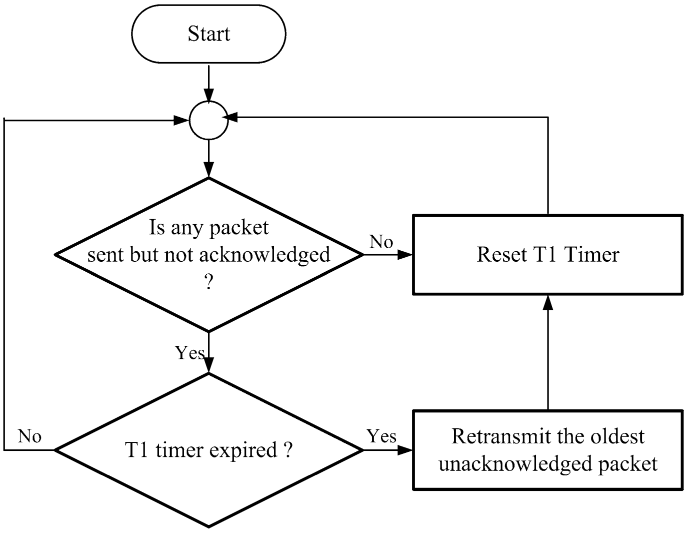
Start
Is any packet sent but not acknowledged ?
No
Reset T1 Timer
Yes
No
T1 timer expired ?
Yes
Retransmit the oldest unacknowledged packet

Fig. 12 TCP with Fastest Retransmission [42]

Where $0.5 * RTT \leq 0.2 * RTO < 1.5 * RTT$
$T1 = 1.5 * RTT$
Where $0.2 * RTO > 1.5 * RTT$

6.3.2 TCP-LRA

TCP-LRA [43] (TCP Loss Recovery Architecture) separates loss recovery from congestion control. TCP sender's send window is maintained as per the sequence number of packets being sent. TCP-LRA maintains two lists of packets sorted by their transmission order. This helps in improving loss prediction and recovery phases. A WaitList stores all outstanding packets (Sent but not yet acknowledged) as per their transmission order. A RetList stores all packets to be retransmitted. After retransmission of a packet, it is removed from RetList and inserted at the end of WaitList. A fresh packet is inserted at the end of WaitList after being sent. A packet is removed from WaitList after receiving a corresponding ACK or is inserted at the end of RetList if considered as lost. All fresh packets are inserted at the end of WaitList to maintain transmission order along with the retransmitted packets. TCP-LRA requires no interaction with TCP receiver to maintain transmission order.

TCP-LRA extends the concept of fast retransmission phase at packet level by maintaining one DAckC—Duplicate ACK Counter for every packet. Each entry in the list records Sequence Number, Duplicate ACK Counter and Timestamp. On receiving an ACK for a packet, DAckC of all packets which were sent earlier but not yet acknowledged are incremented. Packets with DAckC equal to 3 are removed from WaitList and inserted into RetList for retransmission purpose. In case of multiple transmissions of a packet, Timestamp is used to check which instance of packet is acknowledged. Congestion control selects packets from RetList for retransmissions. If RetList is empty, new packets are selected for transmission.

6.3.3 Hybrid-TCP

TCP-HYBRID [44] is a cross layer solution to predict losses to act accordingly. This variant is named as Hybrid as it combines Signal Strength method and RTT method. Signal strength based analysis helps a routing algorithm to estimate location and mobility pattern of a node. Such information helps in activating alternate route discovery process before disconnection of a current route. Hybrid TCP estimates RTT value, based on the signal strength value. Estimated value is compared with actual RTT to decide cause of a loss and to set transmission rate accordingly. Hybrid TCP also uses noise value to adjust various MAC layer transmission parameters like selection of channel. RTT estimation done by TCP-Hybrid is as below.

$$RTT_{i+1} = (\alpha * RTT_i + (1 - \alpha) * M) * \beta \tag{28}$$

α is a constant between 0 and 1. M is Time taken to receive ACK. $\beta = f(S)$ where S represents lowest signal strength among all nodes through which packet travelled

6.3.4 Comparison

Table 8 compares recent and advanced loss prediction based TCP variants.

Table 8 TCP variants—loss prediction

Sr.	TCP variant	Features	Limitations
1	TCP-FR [42] Fastest Retransmission	Simple way to handle pauses in transmission by earlier retransmission	Fastest Retransmission Timer needs more precise calculation. Always one packet is retransmitted irrespective of number of outstanding packets
2	TCP-LRA [43] TCP Loss Recovery Architecture	Separation of loss recovery from congestion control is more accurate. Fast retransmission is implemented at packet level. No interaction with TCP receiver to maintain transmission order	Complex to implement because of requirements of extra data structures
3	Hybrid-TCP [44]	Signal strength based analysis helps to estimate node location accurately. RTT values calculated with two completely different methods are compared. Transmission parameters can be set based on signal strength and related information	Signal strength based RTT estimation method needs to be analyzed for its accuracy. It is extremely complex to implement and resource consuming

6.4 Loss Avoidance

6.4.1 TCP-DATR

TCP-DATR [45] (TCP Dynamic Adjustment of Transmission Rate) estimates transmission rate according the available bandwidth and present loss rate. Generally losses affect the transmission rate as per the AIMD scheme. Most of the TCP variants maintain loss information per window. TCP-DATR calculates loss rate as one of the parameters while setting value of Cwnd. The occurrences FR-Fast Retransmission and RTO timeout events are counted to calculate PLR-Packet Loss Rate. On occurance of *ith* FR/RTO event, *ith* PLR is calculated as below.

$$PLR_i = (1/Sent\_Data) * 100 \tag{29}$$

$Sent\_Data$ is number of packets sent between $i - 1th$ FR/RTO and ith FR/RTO.

PLR is smoothed out using exponential moving average with reference of the interval between two FR/RTO events. The smoothed PLR is calculated as below.

$$SPLR_i = PLR_{i-1} * (1 - A_i) + PLR_{i-1} * (A_i) \tag{30}$$

$0 \leq A_i \leq 1$. A large value of A_i sets current SPLR more towards the current estimation while a small value of A_i sets it towards the previous SPLR.

$$A_i = Int_{Cur}/Int_{Max} \tag{31}$$

Int_{Cur} is the Interval between $i - 1th$ FR/RTO and ith FR/RTO.

Int_{Max} is the Maximum interval so far.

TCP-DATR defines the relationship between Cwnd and SPLR with two thresholds α and β. TCP-DATR sets RCwnd-Reduced Cwnd in the range of β (Minimum) to Max_Cwnd(Maximum). As SPLR increases, RCwnd decreases from Max_Cwnd to β. if

SPLR is greater than α, RCwnd is set to β. if SPLR is greater than 0 but less than α, RCwnd is set as below.

$$RCwnd = \frac{\beta - Max\_Cwnd}{\alpha} * SPLR + Max\_Cwnd \tag{32}$$

The actual Cwnd is set to minimum of present value of Cwnd and RCWnd to avoid extreme reduction of transmission rate.

6.4.2 TCP-PR

TCP-PR [46] (TCP Packet Recycling) is a cross layer approach based on best effort delivery at MAC layer. TCP-PR increases the retransmission limit at MAC layer as per the RTO timeout limit at TCP. An intermediate node drops a packet after a specific number of unsuccessful transmission attempts from MAC layer. Subsequently, TCP sender experiences RTO timeout which activates congestion control (unnecessarily in some cases). One another disadvantage of dropping a packet at an intermediate node is the requirement of its retransmission from the TCP sender node. TCP-PR allows an intermediate node to keep trying for transmission of a packet as long as there is no RTO timeout at TCP sender. TCP-PR sender sends Remaining Time (Based on RTO value) along with a packet. Every intermediate node has to update this field as below.

$$T_{New\_Rem} = T_{Rem} - (T_{Dep} - T_{Arr}) \tag{33}$$

T_{New_Rem} is New remaining time.
T_{Rem} is current remaining time.
T_{Dep} is departure time.
T_{Arr} is arrival time.

An intermediate node keeps trying for transmission of a packet till Reamining Time allows. This scheme may increase number of retransmissions at MAC layer but ensures comparatively less number of retransmissions at TCP.

6.4.3 TCP-RCS

TCP-RCS [42] (TCP Rate Control Scheme) tries to handle issues related with channel contention and unordered delivery. TCP in wireless networks often experiences sudden burst of traffic due to disturbance in its self-clocking mechanism. As the transmission rate is controlled by the rate of ACKs, disturbance in the reception of ACKs may prevent smooth increasing or decreasing change in transmission rate. Too late arrival of an ACK (because of contention issues or unordered delivery) may increase Cwnd and subsequently a burst of traffic is injected to the network. A sudden traffic burst may cause network unstable with congestion or contention. It also affects flow of other TCP connections. As a proactive approach, TCP-RCS restricts transmission rate by a constant factor of β. β's too large value increases possibility of traffic burst while β's too small value increases possibility of network underutilization. β is set to 1/5 based on experimental analysis.

6.4.4 TCP-CC

TCP-CC [47] (TCP Contention Control) tries to avoid issues related with sudden burst of traffic by controlling the flow of injecting packets to the network. A sudden burst may invite high contention in the network. Most of the research work is done towards the improvement of congestion control queue management which still not solved the performance degradation due to contention issues. TCP-CC is a cross layer approach which introduces delay based transmissions to distribute packets throughout the RTT. A similar mechanism exists in IEEE 802.11 called Contention Window Mechanism. Most of the TCP variants uses flow of ACKs to measure RTT and packet losses which does not provide exact information to the TCP about the lower layers. TCP-CC introduces cross layer based approach to define the relationship between contention window and TCP throughput. TCP-CC introduces a delay between two packets (Rate based pacing) and a delay between two ACKs (ACK pacing). TCP-CC has made two assumptions: (1) Every TCP sender always has something to transmit and every queue is always non empty. (2) RTT is independent of Cwnd. TCP-CC has implemented for one hope and multi hop networks. The detail calculation of delay values are discussed in [47].

6.4.5 Dynamic TCP-Vegas

Dynamic TCP-Vegas [48] tries to overcome rerouting based issues of TCP Vegas. It also tries to improve Slow Start phase for better congestion control. TCP Vegas uses RTT value for adjustment of Cwnd. MANETs often experience rerouting where an increase in RTT may misinterpreted as a cause of congestion. Inaccurate estimation of Base RTT may invite unfairness issue among TCP connections too. Under estimation (greater than actual) of Base_RTT degrades performance while over estimation (less than actual) of Base_RTT increases lose.

Base RTT Estimation Dynamic TCP-Vegas tries to estimate Base RTT accurately using cuckoo search optimization algorithm. The optimal function to estimate Base RTT is,

$$Max(Base_{RTT}) = \frac{AAR(k)}{AAR(k-1)} * \frac{S(k-1) - D(k)}{S(k)} \tag{34}$$

k is iteration number.
$AAR(k)$ is Actual Average Rate at iteration k.
$S(k)$ is Total Sent packets at iteration k.
$D(k)$ is Total Dropped packets at iteration k.

Dynamic Slow Start TCP Vegas increases or decreases Cwnd by a constant in Slow Start phase. Dynamic TCP Vegas decides change in Cwnd and SSThresh values dynamically according to the transmission rate. Bandwidth (δ) is estimated to set values of SSThresh and Cwnd as below.

$$\delta = (Rate_{Expected} - Rate_{Actual}) * Base_{RTT} \tag{35}$$

$$SSThresh = \delta * RTT \tag{36}$$

$$Cwnd = Cwnd + \left(\frac{\alpha - \delta}{\alpha}\right), \quad if \quad \delta < \alpha \tag{37}$$

$$Cwnd = Cwnd - \left(\frac{\delta - \beta}{\beta}\right), \quad if \quad \delta > \beta \tag{38}$$

6.4.6 CCPRCLA

CCPRCLA [49] Congestion Control and Packet Recovery for Cross Layer Approach addresses overshooting window and poor throughput problems due to contention and congestion traffic. It uses Congestion RTT as well as Contention RTT to decide transmission rate. An another part of the approach is packet recovery. Packets may lost due to contention or congestion. This approach introduces caching of packets. In a group of nodes, some nodes act as recovery assistant nodes which help in forwarding cached packets. The detail arithmetic is given in [49].

6.4.7 Comparison

Table 9 compares recent and advanced loss avoidance based TCP variants.

Table 9 TCP variants—loss avoidance

Sr.	TCP variant	Features	Limitations
1	TCP-DATR [45] TCP Dynamic Adjustment of Transmission Rate	Packet loss rate is explicitly calculated used to control transmission rate. This scheme also avoids extreme reduction of transmission rate	Threshold needs to set more accurately
2	TCP-PR [46] Packet Recycling	Packet Recycling increases link layer retransmissions to reduce retransmissions at TCP	Complex to implement. Active participation of intermediate nodes is required. Less suitable for high mobility
3	TCP-RCS [42] Rate Control Scheme	It handles all of sudden traffic burst related issues with a simple heuristic to limit transmission rate	Threshold needs to be dynamic
4	TCP-CC [47] Contention Control	It handles all of sudden traffic burst related issues by controlling flow of packets	Complex to implement. Active participation of receiver is required. Less suitable for high mobility
5	Dynamic TCP Vegas [48]	Dynamic Slow Start and Base RTT calculation improves performance as compared to TCP Vegas	Base RTT calculation is time consuming
6	CCPRCLA [49]	Congestion and Contention based transmission rate calculation. Packet caching for recovery	Difficult to implement as both the sub approaches require cross layer implementation

Table 10 MANET scenarios

ID	Nodes	Mobility speed (s)	Mobility redefine interval (s)	Error rate
M1	25	20	10	0.005
M2	50	30	15	0.005
M3	75	40	20	0.005
M4	100	30	15	0.005
M5	125	20	10	0.005

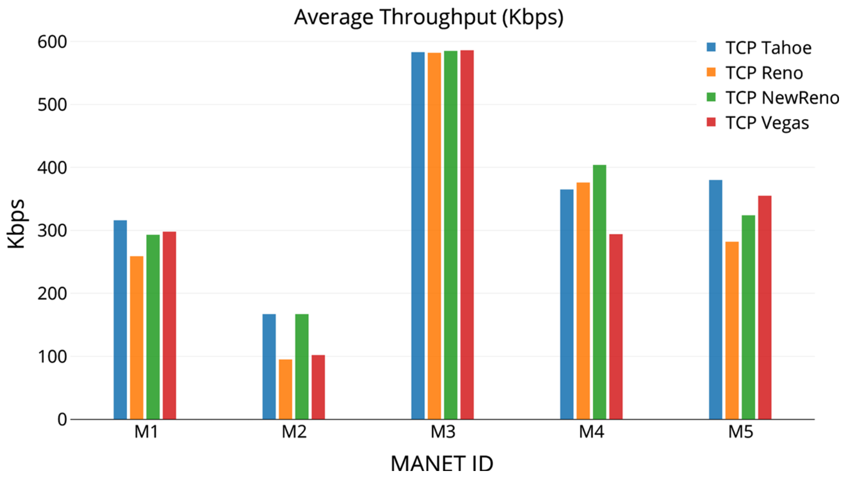

Fig. 13 Average throughput

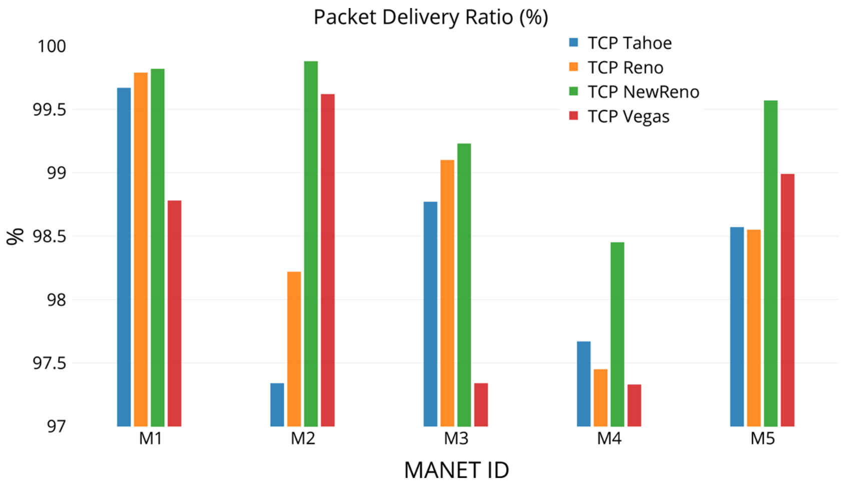

Fig. 14 Packet delivery ratio

7 Simulation

Simulation has been done in Network Simulator 2.35 [50]. Performances of few of the most widely used TCP variants are discussed in this paper as performance analysis of all TCP variants is out of the scope of this paper. Various scenarios have been generated to implement realistic MANETs. A scenario is mainly defined by Number of Nodes, Mobility Speed and Mobility Redefine Interval and Error Rate. Mobility Speed is how fast a node tries to reach to the destination. Mobility Redefine Interval is how frequently a mobile node changes its destination. Error Rate is how frequently packets are dropped. A set of five sample scenarios and their results are discussed in this section.

Several MANET scenarios based on different Number of Nodes, Mobility Speed and Mobility Redefine Interval, Traffic Pattern, Error Rate are designed. A set of five sample scenarios are listed in Table 10.

Average Throughput and Packet Delivery Ratio for various TCP variants for M1 to M5 MANET Scenarios are shown in Figs. 13 and 14 respectively.

8 Conclusion

This paper discussed various TCP variants based on their approaches to handle various losses. Other than congestion, TCP's performance in MANETs is influenced by underlying layer issues like route failure losses and channel issues (random channel losses, contention issues). Traditional TCP variants are not suitable for MANETs because of their default consideration of any loss as a cause of congestion only. TCP variants for MANETs can be classified into Cross-Layered approaches and Layered approaches. A Cross Layered TCP variant uses decision making information provided by lower layer protocol(s) for loss handling while a Layered TCP variant uses its own statistics to handle losses experienced at lower layers. Cross Layered TCP variants are more accurate but complex and difficult to implement. They also violate layer independence concept of a layered network which is based on designing every protocol algorithmically independent from protocols of other layers. MANETs are often formed for temporarily usage with battery powered devices most of the time. In such scenarios, deploying a complex protocol is time consuming and resource consuming.

Features and limitations of Cross Layered based TCP variants for route failure losses (Table 1), congestion losses (Table 2), channel losses and contention issues (Table 3) are listed. As compared to Cross Layered TCP variants, Layered TCP variants are less accurate but simple and easy to implement. Layered approaches are more suitable for MANETs from the implementation point of view. Features and limitations of Layered based TCP variants for route failure losses (Table 4), congestion losses (Table 5), channel losses and contention issues (Table 6) are listed.

In recent years, more advanced TCP variants have been proposed primarily focused on efficient loss handling. Features and limitations of loss differentiation (Table 7), loss prediction (Table 8) and loss avoidance (Table 9) based TCP variants are listed.

Even though a numerous TCP variants have been proposed for MANETs, no TCP variant seems to be providing complete solution to all the problems in all scenarios. Every TCP variants either targets a specific set of problems or performs better in some specific scenarios only. This is the reason why there is no ultimate TCP solution has been found.

The need of an ultimate and complete solution invites further research in the direction of efficient loss handling by TCP.

9 Future Directions

As no TCP Variant has been found as an ultimate and complete solution for all the issues and scenarios which may arise in MANETs, one or more proposals could be combined to design a more generic TCP Variant. Any Future TCP Variant needs to be designed with some more concrete measurements rather than implementing RTT based decision making. The loss handling problem can be tried to solve using techniques of Artificial Intelligence and Data Mining too. A few decision making questions are explained as below.

What should be done after identifying a route failure?

A route failure may cause multiple retransmission timeouts. Whether TCP should continue increasing retransmission timeout timer values exponentially between subsequent timeouts or not. After each timeout, should TCP retransmits the oldest unacknowledged packet which acts as a probing packet or not. How to find whether a new route is available or not. Should TCP starts transmission which will subsequently cause network layer to initiate a route discovery or should TCP waits for network to take necessary action

What should be done after a new route is found?

TCP should not reduce transmission rate completely as route failures are not as same as network congestion. Whether to explore new route to find most suitable transmission rate or to continue with the transmission rate at which TCP was working with old route.

What should be done after identifying a channel loss?

The lost packet must be retransmitted. If packet was lost due to wireless transmission issues then there is no need to reduce transmission rate. If packet was lost due to contention issues, transmission rate could be reduced for a while to reduce channel contention. Avoiding channel losses due to wireless transmission issues are more towards using frequencies properly, using devices like repeaters etc. channel losses due to contention issues may be avoided by reducing unnecessary retransmission, reducing flow of acknowledgements etc.

References

1. Forouzan, B. (2009). *TCP/IP protocol suite*. New York City: McGraw-Hill.
2. Stevens, W. R. (2000). TCP/IP illustrated, volume 1: The protocols. Addison-Wesley professional computing series.
3. Sarkar, S. K., Basavaraju, T. G., & Puttamadappa, C. (2013). *Ad hoc mobile wireless networks: Principles, protocols, and applications*. Boca Raton: CRC Press.
4. Mohapatra, P., & Krishnamurthy, S. V. (2005). *Ad hoc networks technologies and protocols*. Berlin: Springer.

5. Mast, N., & Owens, T. J. (2011). A survey of performance enhancement of transmission control protocol (TCP) in wireless adhoc networks. *EURASIP Journal on Wireless Communications and Networking, 96*, 1–23.
6. Al-Jubari, A. M., Othman, M., Ali, B. M., & Hamid, N. A. W. A. (2011). TCP performance in multi-hop wireless ad hoc networks challenges and solution. *EURASIP Journal on Wireless Communications and Networking, 198*, 1–25.
7. Tsaoussidis, V., & Matta, I. (2002). Open issues on TCP for mobile computing. *Journal on Wireless Communications and Mobile Computing*, *2*, 3–20.
8. Abed, G. A., Ismail, M., & Jumari, K. (2012). Exploration and evaluation of traditional TCP congestion control techniques. *Journal of King Saud University - Computer and Information Sciences*, *24*(2), 145–155.
9. Qureshi, B., Othman, M., & Hamid, N. A. W. (2009). Progress in various TCP variants. In *2nd IEEE international conference on computer, control and Communication* (pp. 1–6).
10. Stevens, W. (1997). TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *RFC 2001.*
11. Allman, M., Paxson, V., & Stevens, W. (1999). TCP congestion control. *RFC 2581.*
12. Brakmo, L. S., OMalley, S. W., & Peterson, L. L. (1994). TCP Vegas: New techniques for congestion detection and avoidance. In *SIGCOMM'94 the conference on communications architectures, protocols and applications* (pp. 24–35).
13. Chandran, K., Raghunathan, S., Venkatesan, S., & Prakash, R. (1998). A feedback based scheme for improving TCP performance in ad-hoc wireless networks. In *18th IEEE international conference on distributed computing systems* (pp. 34–39).
14. Holland, G., & Vaidya, N. (2002). Analysis of TCP performance over mobile ad hoc networks. *Wireless Networks*, *8*(2), 275–288.
15. Kim, D., Toh, C.-K., & Choi, Y. (2001). TCP-BuS: improving TCP performance in wireless ad hoc networks. *IEEE Journal of Communications and Networks*, *3*, 1–12.
16. Liu, J., & Singh, S. (2001). ATCP: TCP for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, *19*(7), 1300–1315.
17. Yu, X. (2004). Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness. In *Proceedings of the 10th annual international conference on mobile computing and networking* (pp. 231–244).
18. Kadangode, R., Sally, F., & Black, D. (2001). The addition of explicit congestion notification (ECN) to IP. *RFC 3168.*
19. Kai, X., Tian, Y., & Ansari, N. (2004). TCP-Jersey for wireless IP communications. *IEEE Journal on Selected Areas in Communications*, *22*(4), 747–756.
20. Kim, K.-W., Lorenz, P., & Lee, M. M.-O. (2005). A new tuning maximum congestion window for improving TCP performance in MANET. In *IEEE proceedings of systems communications* (pp. 73–78).
21. Gunes, M., & Vlahovic, D. (2002) The performance of the TCP/RCWE enhancement for ad-hoc networks. In *7th IEEE international symposium on computers and communications* (pp. 43–48).
22. Kliazovich, D., & Granelli, F. (2006). Cross-layer congestion control in ad hoc wireless networks. *Ad Hoc Networks*, *4*(6), 687–708.
23. Sirajuddin, M. D., Rupa, C., & Prasad, A. (2016). Advanced congestion control techniques for MANET. *Information Systems Design and Intelligent Applications. Advances in Intelligent Systems and Computing*, *433*, 271–279.
24. Rakocevic, V., & Hamadani, E. (2008). A cross layer solution to address TCP intra-flow performance degradation in multihop ad hoc networks. *Journal of Internet Engineering*, *2*(1), 146–156.
25. Cordeiro, C. D. A., Das, S. R., & Agrawal, D. P. (2002). COPAS: dynamic contention-balancing to enhance the performance of TCP over multi-hop wireless networks. In *Eleventh IEEE international conference on computer communications and networks* (pp. 382–387).
26. Dyer, T. D., & Boppana, R. V. (2001). A comparison of TCP performance over three routing protocols for mobile ad hoc networks. In *2nd ACM international symposium on mobile ad hoc networking & computing* (pp. 56–66).
27. Wang, F., & Zhang, Y. (2002). Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response. In *3rd ACM international symposium on mobile ad hoc networking & computing* (pp. 217–225).
28. Fu, Z., Greenstein, B., Meng, X. & Lu, S. (2002). Design and implementation of a TCP-friendly transport protocol for ad hoc wireless networks. In *IEEE-10th international conference on network protocols* (pp. 216–225).

29. Seddik-Ghaleb, A., Ghamri-Doudane, Y., & Senouci, S.-M. (2009). TCP welcome TCP variant for wireless environment, link losses, and congestion packet loss models. In *1st IEEE international conference on communication systems and networks and workshops* (pp. 1–8).
30. Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M. Y., & Wang., R. (2002). Tcpwestwood: End-to-end congestion control for wired wireless networks. *Wireless Networks*, *8*(5), 467–479.
31. Lai, C., Leung, K.-C., & Li, V. O. K. (2009). TCP-NCL: A unified solution for TCP packet reordering and random loss. In *IEEE 20th international symposium on personal, indoor and mobile radio communications* (pp. 1093–1097).
32. Wang, J., Wen, J., Zhang, J., & Han, Y. (2010). TCP-fit—a novel TCP congestion control algorithm for wireless networks. In *IEEE GLOBECOM 2010 workshop on advances in communications and networks* (pp. 2065–2069).
33. Fu, C. P., & Liew, S. C. (2003). TCP enhancement for transmission over wireless access networks. *IEEE Journal on Selected Areas in Communications*, *21*, 216–228.
34. de Oliveira, R., & Braun, T. (2005). A dynamic adaptive acknowledgment strategy for TCP over multihop wireless networks. In *24th annual joint conference of the IEEE computer and communications societies* (Vol. 3, pp. 1863–1874).
35. Singh, A. K., & Kankipati, K. (2004). TCP-ADA: TCP with adaptive delayed acknowledgement for mobile ad hoc networks. *IEEE Wireless Communications and Networking Conference*, *3*, 1685–1690.
36. Altman, E., & Jimenez, T. (2003). Novel delayed ACK techniques for improving TCP performance in multihop wireless networks. In *IFIP international conference on personal wireless communications* (pp. 237–250). Springer.
37. Sreekumari, P., & Lee, M. (2013). TCP NRT: A new TCP algorithm for differentiating non-congestion retransmission timeouts over multihop wireless networks. *EURASIP Journal on Wireless Communications and Networking, 172,* 1–20.
38. Sreekumari, P., & Chung, S.-H. (2011). TCP NCE: A unified solution for non-congestion events to improve the performance of TCP over wireless networks. *EURASIP Journal on Wireless Communications and Networking, 23*, 1–20.
39. Govindarajan, J., Vibhurani, N., & Kousalya, G. (2018). Enhanced TCP NCE: A modified non-congestion events detection, differentiation and reaction to improve the end-to-end performance over manet. *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications. Advances in Intelligent Systems and Computing*, *519*, 443–454.
40. El-Ocla, H. (2010). TCP CERL: Congestion control enhancement over wireless networks. *Wireless Networks*, *16*, 183–198.
41. Sunitha, D., Nagaraju, A., & Narsimha, G. (2017). Cross-layer based routing protocol and solution to packet reordering for TCP in MANET. *Cluster Computing, 1,* 1–8.
42. Jung, S., Lee, J., Lee, G., Pyun, S.-Y., & Cho, D.-H. (2014). Novel fastest retransmission and rate control schemes for improving TCP performance in wireless ad hoc networks. *Wireless Personal Communications*, *75*, 557–567.
43. Kang, M., Park, H., & Mo, J. (2012). Implementation and evaluation of a new TCP loss recovery architecture. *EURASIP Journal on Wireless Communications and Networking, 149,* 1–9.
44. Douga, Y., & Bourenane, M. (2013). A cross layer solution to improve TCP performances in ad hoc wireless networks. In *IEEE international conference on smart communications in network technologies* (pp. 1–5).
45. Park, M.-Y., Chung, S.-H., & Ahn, C.-W. (2012). TCP's dynamic adjustment of transmission rate to packet losses in wirelessnetworks. *EURASIP Wireless Networks, 149,* 1–9.
46. Al-Zubi, R. T., Krunz, M., Al-Sukkar, G., Hawa, M., & Darabkh, K. A. (2014). Packet recycling and delayed ACK for improving the performance of TCP over manets. *Wireless Personal Communications*, *75*, 943–963.
47. Xie, H., & Boukerche, A. (2015). TCP-CC: Cross-layer TCP pacing protocol by contention control on wireless networks. *EURASIP Wireless Networks, 21,* 1061–1078.
48. Xie, H., & Boukerche, A. (2017). Dynamic TCP-Vegas based on cuckoo search for efficient congestion control for MANET. *International Journal of Signal and Imaging Systems Engineering, 10,* 1–8.
49. Gowtham, M. S., & Subramaniam, K. (2018). Congestion control and packet recovery for cross layer approach in manet. *Cluster Computing, 1*, 1–8.
50. Issariyakul, T., & Hossain, E. (2012). *Introduction to network simulator NS2*. Berlin: Springer.

Hardik K. Molia is pursuing his Doctoral Studies (Ph.D. in Computer Engineering) at Gujarat Technological University, Gujarat under supervision of Dr. Amit D. Kothari. He is working as an Assistant Professor in Computer Engineering at Government Engineering College-Rajkot, 360005 Gujarat, India.

Dr. Amit D. Kothari is working as an Associate Professor and HOD-MCA at Institute of Technology and Management Universe (ITM Universe) Vadodara, 391510 Gujarat, India. Hardik Molia is pursuing his Doctoral Studies under his supervision.