EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

A survey of performance enhancement of transmission control protocol (TCP) in wireless *ad hoc* networks

Noor Mast[1,2*] and Thomas J Owens[1]

Abstract

Transmission control protocol (TCP), which provides reliable end-to-end data delivery, performs well in traditional wired network environments, while in wireless *ad hoc* networks, it does not perform well. Compared to wired networks, wireless *ad hoc* networks have some specific characteristics such as node mobility and a shared medium. Owing to these specific characteristics of wireless *ad hoc* networks, TCP faces particular problems with, for example, route failure, channel contention and high bit error rates. These factors are responsible for the performance degradation of TCP in wireless *ad hoc* networks. The research community has produced a wide range of proposals to improve the performance of TCP in wireless *ad hoc* networks. This article presents a survey of these proposals (approaches). A classification of TCP improvement proposals for wireless *ad hoc* networks is presented, which makes it easy to compare the proposals falling under the same category. Tables which summarize the approaches for quick overview are provided. Possible directions for further improvements in this area are suggested in the conclusions. The aim of the article is to enable the reader to quickly acquire an overview of the state of TCP in wireless *ad hoc* networks.

1. Introduction

Over the last decade, there has been a very rapid growth in wireless technology. One of the aims of wireless technology is to provide network availability to users everywhere, at all times and at low cost. Fundamentally, wireless networks can be divided into two types: infrastructure, and *ad hoc* networks (also called infrastructure less networks). Examples of infrastructure and wireless *ad hoc* networks are given in Figure 1a, b, respectively. In an infrastructure network, the wireless nodes have access to a wired network through a wired access point (AP) which works as a backbone. A wireless *ad hoc* network is a collection of nodes, and its characteristics can be summarized as follows [1,2]:

- Nodes communicate through wireless links using shared medium.
- A node can work as a router.
- There is no infrastructure and centralized control.
- Nodes can be static or free to move.
- Nodes can join or leave the network without any restriction.
- It can be setup anywhere.

Owing to their flexible structure, wireless ad hoc networks are well suited for scenarios where an infrastructure is unavailable. Thus, they can be used for crisis management services applications, such as in disaster relief operations where the quick restoration of communications infrastructures is essential. Other examples of their use include police exercises, urgent business meetings outside the office environment and in battlefield applications by the military including situation awareness systems, where IEEE 802.11 MAC protocol [3] provides support to establish *ad hoc* networks. It is obvious that wireless *ad hoc* networks have the potential to provide significant facilities to users. However, owing to these specific characteristics of wireless *ad hoc* networks, there are a lot of technical problems that need to be solved to achieve an efficient utilization of wireless technology. The research community is trying to solve these technical problems and formulate possible smooth deployment of wireless technology. Transmission control protocol (TCP) [4], which is a dominant transport

* Correspondence: nm.afridi@hotmail.com
[1]School of Engineering and Design, Brunel University, London, UK
Full list of author information is available at the end of the article

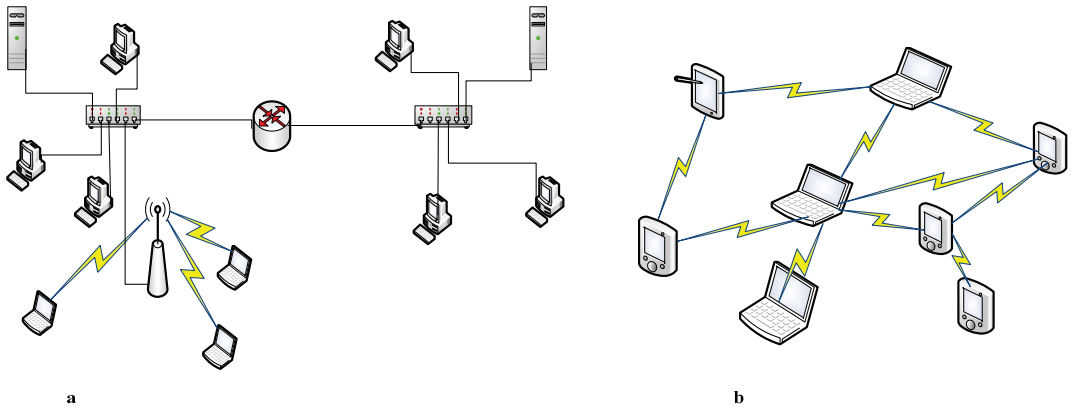

Figure 1 Examples of infrastructure and wireless *ad hoc* networks.

layer connection oriented and reliable end-to-end protocol, is facing challenges in wireless *ad hoc* networks. From the literature review reported in this article, it is clear that TCP faces the following challenges in wireless *ad hoc* networks:

- Route failure and network partition
- Shared medium
  - Hidden and exposed terminal problem
  - Channel contention
- High bit error rate and burst losses
- Inability to differentiate congestion losses from other losses

The objectives of this article are to provide a clear overview of different proposals suggested by the research community for performance improvement of TCP in wireless *ad hoc* networks and provide a guide as to what are the possible directions for further improvements in this area. In this way, it aims to provide an overview of the current state of TCP on wireless *ad hoc* networks. In the process, a classification of the proposals is provided to give the reader a new angle from which to view the work on TCP in wireless *ad hoc* networks. Discussion in the article will show that this classification makes it easy to compare approaches that fall under the same category. It is difficult to present a comprehensive comparison of all the approaches together because each one addresses specific problems.

This article is organized as follows. Section 2 presents a brief overview of the working mechanism of TCP, while Section 3 outlines the challenges TCP faces in wireless *ad hoc* networks. Section 4 presents a survey of the approaches available to improve the performance of TCP in wireless *ad hoc* networks and provides taxonomy of these approaches. Section 5 concludes the article and provides suggestions for possible directions for future study seeking to improve the performance of TCP in wireless *ad hoc* networks.

2. TCP working mechanisms

TCP is a window-based reliable transport layer protocol that achieves its reliability through sequence numbers and acknowledgements (ACKs). TCP uses the ACK as a clock to transmit data to the network and adjust its transmission rate according to the available network capacity; because of this mechanism, TCP is also known as a self-clocking algorithm.

During data transmission, TCP conceptually assigns a sequence number to each octet (byte) of data, and then packs these octets into segments[1]. The sequence number of the first octet of data in a segment is transmitted with that segment and is called a segment sequence number [4]. To ensure the reliable delivery of data segments, when a destination node receives a data segment, it replies to the sender to acknowledge that the data segment has been received correctly and to send the sequence number of next expected data octet. If an out-of-order data segment arrives at destination node, then it shows that a data segment is missing between the previously and currently arrived segments. Then, the receiver (destination node) sends an ACK to identify the missing data segment for retransmission. More than one ACK identifying the same segment of data to be retransmitted is called a duplicate ACK. After three duplicate ACKs, the sender assumes that the segment has been lost and retransmits it. Moreover, TCP also uses timeout to detect losses. After transmitting a segment, TCP starts a time down counter to monitor timeout occurrence. If timeout occurs before receiving the ACK, then the sender assumes that the segment has been lost. The lost segment is then retransmitted, and TCP initiates the slow start algorithm. The timeout interval is called retransmission timeout (RTO) and is computed

according to [5]. To identify segments damaged in transit, the TCP sender adds a checksum field to each segment, and the receiver must apply validate the checksum on receiving each segment, discarding the segment if the validation fails.

TCP also provides a mechanism for the receiver to control the amount of data a sender is sending to it. The receiver specifies in each ACK a window size (window size means the amount of data) named the advertised window or receiver window (rwnd) that the receiver is ready to accept. Similarly, a congestion window (cwnd) specifies the amount of data a sender can transmit to a receiver without receiving any ACK from the receiver for the data sent to it. The amount of data equal to the minimum of one of these windows will be transmitted over the network by the sender, i.e.

$$\text{data to be transmitted} = \min(\text{cwnd}, \text{rwnd}).$$

Slow start and congestion avoidance are the two main phases of the TCP congestion algorithm. In the slow start phase the cwnd is incremented exponentially until the slow start threshold is reached. Afterward, the congestion avoidance phase starts, and the cwnd is incremented by one maximum segment size (MSS) up to some predefined value. In each phase, the cwnd is incremented in response to a non-duplicate ACK. It should be clear that TCP will recover one lost packet per round trip time (RTT). Thus, when multiple losses occur in the same cwnd, TCP may experience very poor performance. To overcome this problem, a selective acknowledgment (SACK) [6] was introduced. TCP NewReno [7] provides an alternative way to tackle this problem; the working mechanisms of SACK and TCP NewReno are explained below.

To understand the working mechanism of SACK, let us suppose that a TCP sender is sending data segments with sequence numbers in the following order where each segment consists of 512 bytes:

$$N = 500,\ N + 512,\ N + 1024,\ N + 1536,\ N + 2048,\ N + 2560$$

Further suppose that the TCP receiver received two segments with sequence numbers of $N = 500$ and $N + 2560$, which means that the four segments having sequence numbers between $N = 500$ and $N + 2560$ are missing. After receiving the segment $N + 2560$, the receiver will ask for the retransmission of the segment $N + 512$ in ACK. Thus, the receiver confirms that it has received the segments with sequence numbers less than $N + 512$. Although the receiver has also received the segment of sequence number $N + 2560$, it does not provide any information to the sender about this segment. In contrast, the SACK has an option that allows the TCP receiver to acknowledge that it has received non contiguous data segments. Thus, in the case of a lost segment(s), the sender can infer from the SACK which segment(s) has(have) been lost and should be retransmitted. In the above example, the SACK can indicate that segments with sequence numbers $N = 500$ and $N + 2560$ have been received. As a result, the sender will be able to infer that the segments between these two (i.e. segments of sequence numbers $N + 512$, $N + 1024$, $N + 1536$ and $N + 2048$) have been dropped.

On the other hand, it is stated in [7] that, in the absence of SACK, some information is still available to a TCP sender to detect a single or multiple segments lost and make a retransmission decision. To detect that there is a loss of a single or multiple segments, the first information available to the TCP sender comes from receiving an ACK for the retransmitted segment. If a single segment has been dropped, then the ACK received for the retransmitted segment must confirm the reception of all those segments transmitted before the activation of the TCP retransmission algorithm. If the ACK confirms some but not all of the segments, then it is an indication of multiple segments lost and the ACK is known as a partial ACK. In [7], it is suggested that the TCP sender respond to the partial ACK by inferring that the segments not acknowledged have been lost, and retransmit the first unacknowledged segment. Thus, in response to each partial ACK, the TCP sender retransmits the next unacknowledged segment, until all the segments have been acknowledged. This modification to TCP Reno results in TCP NewReno.

3. Challenges for TCP in wireless *ad hoc* networks

TCP was designed for wired networks without considering the complexity of wireless networks. With the advent of wireless technology TCP was also employed in wireless environments. In wireless technology, changes were made in the lower layers of the communications stack without considering their effects on the upper layers. Consequently, in wireless networks, the communication environment is significantly different from that of wired networks, while TCP treats a wireless network like a wired one and, as a result, TCP faces challenges in wireless environments such as

I. Route failure and network partition

In wireless *ad hoc* networks nodes are free to remain static or move as well as can join or leave the network without any restriction. Owing to this freedom, wireless *ad hoc* networks have a dynamic topology. As a result, two types of problems arise. One is frequent route failure, which leads to packet loss because of which TCP takes a long time to recover from these losses and its performance decreases. There will be no transmission on the connections of the failed route until a new path

is computed, and the instantaneous throughput of TCP becomes zero. Therefore, frequent route failure means a lot of time is wasted in a network, just for searching new routes. During the path recovery process, there may be retransmission timeout (RTO) resulting in packets' retransmission followed by an activation of the slow start algorithm of congestion control. In slow start phase, the cwnd size is set to one segment which is the minimum amount of data require to transmit over the TCP connection, which causes the poor utilization of the network resources. The second type of problem is network partition where the sender and receiver end up being located in separate networks as a result of route failure. Network partition can be more serious than simple route failure if it remains unresolved for a long time, say longer than several RTO.

II. Shared medium

Owing to the shared medium, where the carrier sense multiple access with collision avoidance (CSMA/CA) method is used for medium access in the IEEE 802.11 MAC protocol [3], wireless networks have two main problems: (a) hidden and exposed terminals; and (b) channel contention.

(a) Hidden and exposed node problem

To explain the problem of hidden and exposed terminals, the following example is provided:

Suppose there are four nodes A, B, C and D and adjacent nodes are in transmission range of each other as shown in Figure 2. Both nodes B and C can communicate with two other nodes directly, while nodes A and D have only one node in direct communication range.

Further suppose that node A is transmitting data to node B while at the same time node C has started data transmission to node B. There will be data collision at node B because nodes A and C do not know about each other and are hidden from each other.

Now, suppose node B is sending data to node A and at the same time node C wants to transmit data to node D. When node C senses the medium, it finds that transmission is taking place. Therefore, node C defers transmission, but actually there is no problem with node C transmitting data to node D; this is called the exposed terminal problem.

The IEEE 802.11 standard provides two techniques to handle interference from other nodes: one is physical carrier sensing, while the second is the RTS/CTS (request to send/clear to send) technique, also known as virtual carrier sensing. The RTS/CTS is basically designed to tackle the hidden terminal problem. In the RTS/CTS mechanism, a sending node sends a RTS message to the receiving node before transmitting a data frame. Once the RTS message is sent, there are two possibilities: (1) If the RTS message is not answered with a CTS message, then the sender reschedules the RTS message. (2) If the RTS message is answered with a CTS message, then the sending node can transmit the data frame, and the other nodes defer their transmission on receipt of the CTS message. The interference range of a node is greater than its transmission range. Therefore, nodes out of the receiver's transmission range cannot receive the CTS message, which would defer their transmission, but can interfere with the transmissions of sending and receiving nodes which are within their interference range. This interference has been reported in [8] and causes performance degradation of the network. To further clarify these interference effects, consider the following example which is taken from [8].

Figure 2 Nodes of a wireless *ad hoc* network.

In this example, consider a chain topology of six nodes depicted in Figure 3. The distance between the nodes is 200 m, and the transmission and interference ranges of the nodes are 250 and 550 m, respectively.

When node 1 is transmitting to node 2, then nodes 2 and 3 cannot transmit at the same time, and thus, the channel transmission capacity is reduced to 1/3 of the total capacity of the channel. However, if the interference (sensing) range is considered, then the transmission capacity further reduces to 1/4 of the channel capacity, because node 4 also cannot transmit concurrently with node 1, since it will interrupt the reception at node 2. Thus, the IEEE 802.11 MAC protocol can schedule very well the transmissions of nodes 1, 2 and 3 with the help of RTS/CTS so that nodes 2 and 3 will defer sending data while node 1 is transmitting; however, it cannot schedule concurrent transmissions by nodes 1 and 4 because node 4 is not in the transmission range of nodes 1 and 2 and so does not receive the CTS message sent by node 2 in response to RTS message from node 1. Thus, the hidden and exposed node problems do not allow efficient use of the medium because of a spatial reuse problem, as only one transmission can take place at a time.

(b) Channel contention

In IEEE 802.11 networks, owing to the shared medium, there exists a race condition among nodes for medium access. As a result, the transmissions of data packets and their ACKs lead to channel contention [9] which causes collision and packet loss. Although the introduction of the RTS/CTS mechanism in IEEE 802.11 MAC protocol is a good solution for handling packets interference, still it is observed in a nine-node chain topology that, for a single flow, packets are dropped at a rate of

Figure 3 Effect of large Interference range.

0.83-3.63 packets/s due to channel contention [10]. A detailed analysis of RTS/CTS and its alternatives can be found in [11-13].

Furthermore, channel contention also leads to the problem of unfairness and can be classified into two types:

• Cases of unfairness that happen among flows passing through different paths in the neighbourhood or among flows passing through the same path. Furthermore, it is pointed out in [14] that, if there are two flows passing through the same path, then the flow starting later gains more bandwidth than the first one.

• The second type of unfairness is among the nodes. Therefore, it is necessary to ensure fair access to the medium for each node. If medium access is not fairly shared between the nodes then disadvantaged nodes will start dropping packets after a specified number of attempts. Meanwhile, it is also possible that the queue size will build up on a disadvantaged node, and the node starts dropping packets because of queue overflow.

In addition, the problem of wrong notification of route failure arises because of channel contention. In IEEE 802.11 MAC protocol, when the number of attempts for medium access exceeds a specified limit, the sender assumes that the receiver is out of range and stops its transmission attempts. The MAC protocol notifies the upper layer that the path is unavailable, and the upper layer starts a route recovery procedure [15]. At this stage, TCP stops transmission, and throughput becomes zero during the route recovery process. Moreover, if the route recovery process takes longer than the RTO, then there will be unnecessary activation of TCP congestion control.

III. High bit error (random losses) rate and burst losses

In a wireless network, the bit error rate is high compared to a wired network; in a wired network, the losses due to bit corruption or link errors can be negligible. For a wired network, the bit error rate typically varies from 10^{-6} to 10^{-8}, while for wireless networks, it varies from 10^{-1} to 10^{-3} [16]. This comparatively high bit error rate leads to non-optimal performance. TCP shows poor performance in the case of burst losses which mostly occur because of channel fading or a change in topology, but they can be due to interference. On receiving three duplicate ACKs for a segment, TCP assumes that the corresponding packet has been lost. After retransmitting the segment concerned, TCP determines the next lost packet on receiving three duplicate ACKs. Consequently, it takes some time to recover from the loss of multiple segments. Owing to this limitation of TCP, it performs very poorly in an environment prone to high losses [15].

IV. Unable to differentiate congestion losses from other losses

As mentioned above, packet loss is very high in wireless networks compared to wired networks. Packet loss may be due to interference, fading or channel contention, but TCP assumes that all packet losses are congestion losses, which leads to the activation of congestion control and a reduction in the cwnd.

In addition, in wireless *ad hoc* networks out-of-order packets can arrive because of the use of multipath routing protocols. When packets are forwarded through different paths to the same destination, the packet transmitted last could reach its destination before the packet transmitted first, but TCP always assumes packet loss in case of out-of-order packets, and this causes its poor performance. Thus, it also becomes difficult to implement multipath routing protocols in a system which is more sensitive to the reordering problem.

4. Available proposals for TCP improvement

4.1. Taxonomy of available proposals

Before introducing the novel taxonomy of proposals for improving the performance of TCP on wireless *ad hoc* networks, those readers who are interested in single-hop wireless networks are referred to [17]. The readers interested in the surveys of TCP enhancement in wireless networks can refer to [18] where six of the surveyed proposals are related to *ad hoc* networks, and five of these six had already been surveyed in [19]. In [19], the survey is focussed on the approaches related to TCP improvement in wireless *ad hoc* networks, in which a total of 15 proposals had been surveyed.

This article seeks to survey the most up-to-date and wide ranging of the TCP improvement proposals for wireless *ad hoc* networks. A total of 29 proposals are included in this article where 14 of these proposals were also studied in the survey articles mentioned above. To provide a different angle from which to view the existing proposals at the top level, as in [19], this article categorizes TCP improvement proposals into two groups, i.e. cross-layer approaches and layered approaches. The difference between the cross-layer and layered

approaches is explained later in this section. At second level, all proposals are grouped according to the problems that the proposal addresses. This makes it easier to compare the proposals falling under the same category where it is difficult to present a comprehensive comparison of all the proposals because each one addresses specific problems. This is illustrated by discussing each category and then comparing the proposals in that category. The resulting novel overall classification is shown in Figure 4. At the second level, the three categories of proposals are:

• *Route failure*: The proposals included in this category address the problem of route failure to tackle route failure in a proper way. Thus, the sender will be in the position to avoid misinterpreting losses that are not due to route failure, as being due to route failure.

• *Congestion and transmission losses*: The proposals in this category are focussed towards resolving the problems of congestion and transmission losses to avoid the injection of more data into the network than its available capacity can accommodate.

• *Shared medium*: As mentioned in Section 3, in wireless *ad hoc* networks, the medium is shared and, as a result, TCP faces problems such as channel contention and unfairness. Therefore, the approaches included in this category are those that address problems arising due to the shared medium.

Based on the cross-layer and layered categorisation, Tables 1 and 2 provided in Section 4 summarize the different proposals in more detail for quick overview. This tabular representation specifies the different characteristics of each proposal, such as which layer(s) is(are) involved in the proposal and clarifies whether the proposal is sender side, receiver side, or whether both sender and receiver are involved. The above tables also show whether or not a proposal relies on the involvement of intermediate nodes for feedback.

Now, let us explain that what is the difference between the cross-layer and layered approaches. The International Standards Organization (ISO) established a framework known as the open system interconnection (OSI) reference model aiming to standardize communication systems. The OSI model consists of seven layers each with specific functionalities. From bottom to top, these layers are the physical, data link, network, transport, session, presentation and application layers [20]. The objective of cross-layer design is to pass information from lower layers to upper layers to facilitate decision making in upper layers for better performance of the network. Passing information in such a way is a violation of the OSI reference model, because, according to OSI model, each layer must perform its task independently. This attempt to violate the principles of the OSI reference model is called the cross-layer design approach [21], while its opposite approach is called a layered approach. In [21], which is mainly focussed on the complexity of cross-layer design, those authors state that the traditional layered architecture is unable to

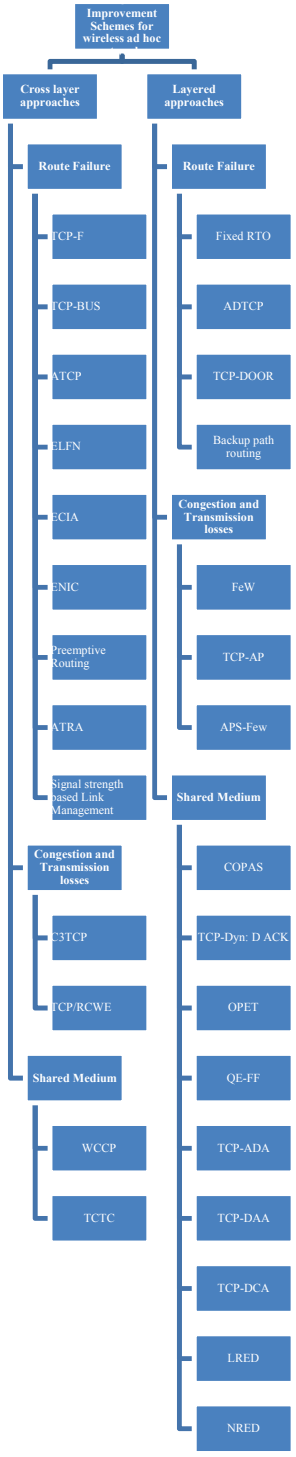

Figure 4 Classification of TCP improvement schemes for wireless *ad hoc* networks.

Table 1 Cross-layer approaches.

Proposed solution	Layers involved	Sender	Intermediate	Receiver	Route failure	Shared medium	Transmission losses	Unfairness	Congestion	Normal	SACK	DACK	Control messages	Routing protocol	Parameters	Proposed in	Simulation environment
TCP-feedback	T & N	Y	Y	-	Y	-	-	-	Y	Y	-	-	RFN RRN	-	-	1998	Mobile
TCP-BUS	T & N	Y	Y	-	Y	-	-	-	Y	Y	-	-	ERDN ERSN	ABR	-	2000	Mobile
ATCP	T & N	Y	Y	-	Y	-	Y	-	Y	Y	-	-	ICM, ECN	-	-	2001	Mobile
ENIC	T, N & MAC	Y	Y	Y	Y	-	-	-	-	-	Y	Y	EREN ERRN	AODV	-	2001	Mobile
Preemptive routing	N & P	Y	Y	-	Y	-	-	-	-	Y	-	-	Ping Pong	DSR AODV	Signal strength	2001	Mobile
ELFN	T & N	Y	Y	-	Y	-	-	-	Y	Y	-	-	ELFN	DSR	-	2002	Mobile
TCP/RCWE	T & N	Y	Y	-	Y	-	Y	-	Y	Y	-	-	ELFN	DSR	RTT	2002	Mobile
ATRA	N & MAC	Y	Y	-	Y	-	-	-	-	Y	-	-	ELFN	DSR	Signal Strength	2004	Mobile
ECIA	T & N	Y	Y	-	Y	-	-	-	Y	Y	-	-	EPLN BEAD	DSR	-	2004	Mobile
Signal strength based link manage	N, P	Y	Y	-	Y	-	-	-	-	Y	-	-	-	AODV	Signal Strength	2004	Mobile
C3TCP	T & MAC	Y	Y	Y	-	-	-	-	Y	-	Y	Y	-	-	Available Bandwidth & Delay	2006	Static
WCCP	T & MAC	Y	Y	Y	-	Y	-	Y	Y	Y	-	-	-	AODV	Channel Usage	2007	Static
TCTC	T, MAC	Y	Y	Y	-	Y	-	-	-	Y	-	-	-	DSR	Short-term Throughput & Transmission Delay	2008	Static

Note: The'-'mark is used if the approach does not involve a particular aspect, otherwise 'Y' is used except in the case of routing where the'-'mark means the approach does not clearly specify the routing protocol used.

Table 2 Layered approaches

Proposed solution	Layer Involved	Sender	Intermediate	Receiver	Route Failure	Shared Medium	Transmission Losses	Unfairness	Congestion	Normal	SACK	DACK	Routing Protocol	Parameters	ACK After	Proposed in	Simulation environment
Fixed RTO	T	Y	-	-	Y	-	-	-	Y	Y	Y	Y	DSR, AODV, ADV	RTO	-	2001	Mobile
ADTCP	T	Y	-	Y	Y	-	Y	-	Y	Y	-	-	DSR	Inter-packet delay & short-term throughput	-	2002	Mobile
TCP-DOOR	T	Y	-	Y	Y	-	-	-	Y		Y	-	DSR	Out of order packet	-	2002	Mobile
Backup path routing	N	Y	-	-	Y		-	-	-	Y	-	-	SMR	-	-	2003	Mobile
COPAS	N	Y	Y	Y		Y	-	-	-	Y	-	-	DSR	Weighted. Ave Backoff	-	2003	Static
LRED	Link Layer	Y	Y	Y	-	Y	-	Y	-	Y	-	-	Manual Routing	Number of Try for medium access	-	2003	Static & Mobile
TCP-Dynamic Delay ACK	T	-	-	Y	-	Y	-	-	-	-	-	Y	AODV	-	One ACK/1-4 packets	2003	Static
OPET	MAC	Y	Y	Y	-	Y	-	-	Y	Y	-	-	Manual Routing	-	-	2004	Static
QE & FF	MAC	-	-	-	-	Y	-	-	-	Y	-	-	AODV, DSR	-	-	2004	Static & mobile
TCP-ADA	T	-	-	Y	-	Y	-	-	-	-	-	Y	DSR	-	One ACK/ Cwnd	2004	Static
TCP-DAA	T	Y	-	Y	-	Y	-	-	-	-	-	Y	-	-	2-4 Packets	2005	Static
FeW	T	Y	-	-	-	Y	-	-	Y	Y	-	-	DSR	-	-	2005	Static
NRED	Link Layer	Y	Y	Y	-	-	-	Y		Y	-	-	AODV	Queue length	-	2005	Static & mobile
TCP-AP	T	Y	Y	Y		Y				Y			AODV	-	AODV	2005	Static
TCP-DCA	T	-	-	Y	-	Y	-	-	-	-	-	Y	AODV, DSR and GPRS	-	Delay window based on hop	2008	Static
APS-FeW	T	Y	-	-	-	-	-	-	Y	Y	-	-	DSR	cwnd, packet size	-	2009	Static & mobile

Note: The'-'mark is used if the approach does not involve a particular aspect, otherwise 'Y' is used except in case of routing where the '-' mark means the approach does not clearly specify the underlying routing protocol used.

make efficient use of wireless network resources and, as a result, cross-layer design has been adopted to provide optimized operations in heterogeneous wireless environments. Cross-layered design has been adopted in various application areas. Those readers who are interested in understanding the various aspects of cross-layer design, such as its complexity and the communication overhead it introduces, are referred to [21]. In next two sections (i.e. 4.2 and 4.3), each proposal for improving the performance of TCP in a wireless *ad hoc* network is discussed in detail.

4.2. Cross-layer approaches

The cross-layered proposals for TCP improvement in wireless *ad hoc* networks are presented under their second-level subcategories.

4.2.1. Route failure

TCP-feedback (TCP-F) TCP-F [22] addresses the problem of TCP's inability to distinguish between the losses due to route failure and the losses due to congestion. If any node detects route failure, then it immediately informs the source about the route failure to avoid unnecessary activation of congestion control. When the network layer detects disruption in the route due to mobility, then it informs the source using a route failure notification (RFN) message. On receiving the RFN message, each intermediate node must prevent other packets from passing through the failed route. In addition, if at any intermediate node an alternate route to the destination is available, then the intermediate node must divert the traffic onto this path and discard the RFN message; otherwise, the intermediate node forwards the RFN message towards the source node.

On the other hand, when an RFN message arrives at the source node, then the source must enter snooze state. In the snooze state, the source must (a) stop all kinds of transmission, (b) freeze its state variables, (c) start a route failure timer, and (d) wait to receive the route re-establishment notification (RRN). There are two ways for the source to come out of the snooze state:

(i) On receiving the RRN message, the source breaks out of the snooze state, or

(ii) When the route failure timer expires, then the source breaks out of the snooze state.

Expiry of the route failure timer is the worst case as it causes retransmission of all the unacknowledged packets. If there are a large number of unacknowledged packets, then it can lead to a burst of traffic and a highly contended situation. If the source changes to its active state on receiving an RRN message, it restores the timer to the frozen value, and the cwnd also remains the same. However, continuing the transmission with the same cwnd may not be suitable for the new path. Similarly, while resuming transmission with the old values of timers, there is a chance that timeout occurs before receiving ACK for unacknowledged packets, which is a drawback

Explicit link failure notification (ELFN) The objective of ELFN [23] is to provide route failure information to the source to avoid unnecessary activation of congestion control. In [23], it is stated that one of the ways to inform a TCP sender about route failure is to use a 'host unreachable' ICMP (internet control message protocol) message for notification. However, in a case of route failure, the routing protocol will send a route failure message to the sender. The approach taken by ELFN is to piggy-back a route failure message for TCP on the routing protocol route failure message. The ELFN message contains the sender and receiver addresses and port numbers as well as the TCP segment's sequence number. To implement the ELFN scheme, the route failure message of dynamic source routing (DSR) [24] protocol was modified to piggy-back the route failure message for TCP.

When the TCP sender receives an ELFN message, it enters a 'standby' mode by disabling its retransmission timers. To gain information about the route re-establishment in the ELFN scheme, the sender sends a probe packet periodically in 'standby' mode. On arrival of ACK for the probe packets, the sender breaks out of the 'standby' mode restoring its timers and continues transmission with its cwnd unchanged. In addition, it is suggested to assign a fixed value to the time interval between two consecutive probe packets, and this value should be a function of the RTT.

TCP-buffering capability and sequence information (TCP-BuS) Considering the problem of TCP's inability to differentiate route failure losses from congestion losses, a scheme named TCP-BuS [25] was suggested to tackle the route failure losses. In TCP-BuS, the associativity-based routing protocol (ABR) [26] is the underlying routing protocol which is a source-initiated on-demand routing protocol. TCP-BuS is a feedback mechanism based on TCP-F [22] which includes reliable delivery of control messages and avoids the unnecessary retransmission of packets along the broken path. In this regard, TCP-BuS has the following five enhancement features compared to TCP-F:

(i) Explicit route notification: To inform the source about route failure, an explicit route disconnection notification (ERDN) message is generated at a pivoting node (PN)–a pivoting node is a node which detects a route failure. The explicit route successful notification (ERSN) is used to notify the source about route re-establishment and to resume transmission from the source.

(ii) Extending timeout values: During the route recovery process, the packets are buffered along the path from the source to the PN to avoid retransmission of

packets on route re-establishment. It is possible that timeout occurs for the buffered packets. Therefore, it is necessary to increase transmission timeout values to avoid timeout events. For ease of implementation, the proposed scheme just doubles the timeout values.

(iii) Selective retransmission requested by receiver for lost packets: When the retransmission timer value for the buffered packets at the source and along the path to the PN expires, it is adjusted to be double its current value. The lost packets are not retransmitted until the adjusted timer value expires. To handle the packet loss along the path from the source to the PN, an indication is made to the source so that it can retransmit the lost packets selectively before their timeout value expires.

(iv) Avoiding unnecessary requests for fast retransmission: On route restoration, the destination can notify the source about the lost packets. In response, the source simply retransmits the lost packets. The packets buffered along the path from the source to the PN may arrive at their destination earlier than the retransmitted packets, but the destination continues to send duplicate ACK until the expected packets arrive at the destination (via the fast retransmit method adopted by TCP-Reno). In TCP-BuS, these unnecessary request packets for fast retransmission are avoided.

(v) Reliable transmission of control messages: It is suggested, for a reliable transmission of the control messages, if a node 'A' sends an ERDN message to its upstream node 'B' then the ERDN message should be forwarded by node 'B' to its upstream node and must be overheard by node 'A'; otherwise, the ERDN message will be considered lost and node 'A' will retransmit the ERDN message. A similar technique is adopted for the reliable delivery of ERSN messages.

Ad hoc TCP (ATCP) The ATCP [27] is a sender-side solution that addresses the problems of route failure, high bit error rate and congestion. It inserts a layer between the TCP and IP layers to maintain compatibility with original TCP. ATCP monitors the network state information provided by explicit congestion notification (ECN) [28] and ICMP 'Destination unreachable' messages to make decisions. ATCP runs in one of four states: Normal, loss, congested and disconnected as shown in Figure 5. It starts in normal state and counts the number of duplicate ACKs. On receiving a third duplicate ACK, it stops forwarding the third duplicate ACK to TCP and puts TCP into 'persist' mode. Also, when RTO is about to occur, ATCP puts TCP into 'persist' mode and enters the loss state. In the loss state, it retransmits all the unacknowledged packets. When a new ACK arrives for any of these retransmitted unacknowledged packets, it is forwarded to TCP which comes out of its 'persist' mode and ATCP returns to its normal state.

Whenever ATCP observes that the ECN flag is on, it shifts to the congested state to activate TCP congestion control without any interruption. However, receipt of an ICMP 'destination unreachable' message means route failure, or network partition has occurred. In response, ATCP enters 'disconnected' state and puts TCP into 'persist' mode. In disconnected state, probe packets are used periodically to detect route re-establishment. On route re-establishment, ATCP returns to its normal state and takes TCP out of 'persist' mode into normal mode. ATCP sets the cwnd to one segment, as in the TCP slow start phase, along the new path.

Exploiting cross-layer information awareness (ECIA) The study carried out in [29] is based on TCP-ELFN and proposes two mechanisms for further improvements. In [29], it is stated that a number of data packets as well as ACK packets get lost before the sender goes to 'standby' mode. The loss of Data and ACK packets leads to retransmission timeout. Therefore, it is important for the network layer to be aware of these losses to help in reducing TCP timeout due to mobility-induced losses. In this regard, two mechanisms, namely, early packet loss notification (EPLN) and best-effort ACK delivery (BEAD) were suggested. In case of route failure,

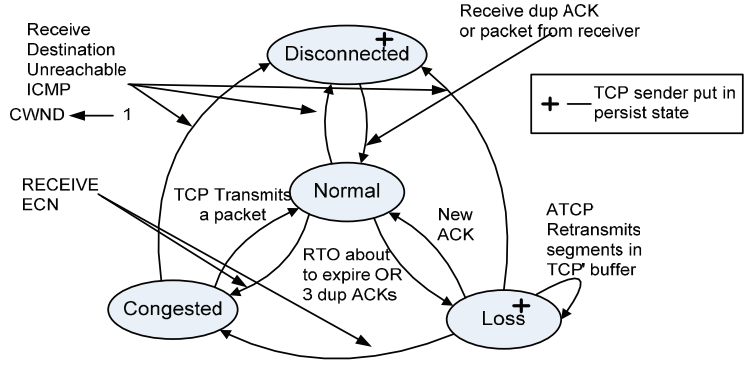

Figure 5 State transition diagram for ATCP at the sender [27].

if an intermediate node cannot salvage the data packet, then the task of EPLN is to send a notification which includes the sequence number of every dropped packet to the sender concerned. As a result, the TCP sender should disable its retransmission timer and retransmit the lost data packets with the lowest sequence number on route availability. In the same way, if ACKs are not salvaged by an intermediate node, then the task of BEAD is to notify the receiver that generated the ACK. In response, the receiver resends the ACK with the highest sequence number on route availability. The DSR routing protocol has been modified to implement the BEAD and EPLN mechanisms.

Enhanced inter-layer communication and control (ENIC) The scheme named explicit notification with ENIC [30] was proposed to solve the problem of TCP performance degradation due to route failure. ENIC uses an explicit route state notification (ERSN) mechanism for inter-process communication (IPC). The ERSN has two types of control messages: explicit route error notification (EREN), and explicit route recover notification (ERRN). For external process communication, ENIC uses routing protocol messages to feedback the route status. Route request (RREQ), route reply (RREP) and route error (RERR) are the three types of external routing messages amongst different nodes.

On detecting a route failure, a node generates a RERR broadcast control message for all the related source and destination nodes, while intermediate nodes receiving this message will drop all the packets related to this failed route.

On receiving a RERR message, the source initiates a route recovery process by broadcasting a RREQ control message and stops the transmission of data packets (new and retransmission); In addition, it puts TCP into suspension state by freezing values of state variables and initializing the route recovery timer. If the source does not receive a RREP before the expiry of the route recovery timer, then the route recovery process is repeated until the pre-specified maximum number of attempts allowed for route recovery is reached. On making the maximum number of unsuccessful attempts allowed, the source closes the connection. On receiving the RREP message, the source breaks the suspension state and transmits all the unacknowledged packets, while returning all variables to their original states except the retransmission timer. This approach uses the DACK (delay acknowledgement) and SACK mechanisms.

Preemptive routing A scheme, named preemptive routing [31], which addresses the problem of route failure, tries to detect when route failure is near to occurring to potentially avoid the disconnection altogether. The signal strength is used for determining closeness to route failure. When the signal strength goes below a specified threshold (called the preemptive threshold), then it means that link failure is near to occurring. In such a situation, the node concerned must inform the source; as a result, the source initiates discovery of a new route.

Ping-pong messages were proposed to measure the signal strength and avoid initiating a false route failure warning. Ping-pong messages are actually small size packets used for probing a link state. A node sends a ping message and receives the pong message in response from the other node, to measure whether signal strength is either below or above the particular threshold. DSR and *ad hoc* on-demand distance vector (AODV) [32] routing protocol were modified to implement this technique, and the modified versions are called preemptive DSR and preemptive AODV routing, respectively.

ATRA In [33], on the basis of analysis, those authors proposed a framework called ATRA. The goal of ATRA is (a) to minimize the probability of route failure, (b) to predict a route failure in advance and compute an alternative path before the failure of an existing one, and (c) to minimize the latency in conveying route failure information to the source. ATRA consists of three mechanisms to achieve its aims. Symmetric route pinning (SRP) is one of these mechanisms: its task is to forward the Data and ACK packets through the same path. Route failure prediction (RFP) is the second mechanism which is based on using the signal strength to predict a route failure in advance. The RFP mechanism maintains the history of the signal strength, from which the speed at which the two nodes in the network are moving away from each other is determined, together with how long before the nodes will be outside of communication range of each other, to inform a source that path failure is about to occur, so that the source can compute a new path before failure of the existing one. If this mechanism does not detect the route failure in advance, then a third mechanism proactive route errors (PREs) will inform the source about route failure. The PRE mechanism tries to minimize the latency involved in passing the route failure information to the source. In the case of route failure, the PRE mechanism informs all the sources that have used this failed link in the past T seconds (the authors [33] used T = 1 s during their simulation). Moreover, ATRA uses the mechanism of [23] to pass the link failure notification to the source. On receipt of this message, the source enters a freeze state, freezing its current state in terms of window size and timers. The source restores its active state when an ACK is received for the probe packets. However, the mechanism proposed in [23] only notifies the source of the connection from which the packed is dropped, while ATRA notifies all the sources connections of which have passed through the affected route in the last T seconds. The ATRA framework uses DSR as a routing protocol;

however, the authors of ATRA claim that ATRA will work with all on-demand routing protocols.

Signal strength-based link management The objective of the scheme proposed in [34] is to overcome packet loss due to mobility and packet loss due to false route failure information. When, in IEEE 802.11 MAC protocol, a sending node fails to establish RTS/CTS handshake with a receiving node after a specified number of attempts due to channel contention, then the sending node notifies the network layer that the path is not available and drops the packet. This type of route failure notification is a wrong (false) notification, because the path is available, but the sending node is unable to establish RTS/CTS handshake because of channel contention. In the proposed approach, it is suggested to double the number of attempts for medium access if there is a high probability that the neighbouring nodes are still within the transmission range of each other. Each node maintains a record of the signal strength of its neighbouring nodes to observe how the signal strength changes over time and computes the probability of a node being present within its transmission range.

Two other mechanisms were also proposed in [34] known as proactive and reactive link managements (LM). The aim of proactive LM is to inform the routing protocol that a link is near to breaking, and the routing protocol then informs the packet source. As a result, the source stops sending packets, and route discovery is initiated. Reactive LM increases the transmission range of a node to restore a broken link for a short time to give a chance to the packets in transit to traverse the high power link. For successful transmission, it is necessary that both the nodes (the nodes between which the route failure has occurred/is near to occurring) shift to a high transmission power to have an RTS/CTS handshake. However, a node must shift quickly to the lower transmission range to communicate with other nodes. The nodes are not allowed to broadcast a RREQ message with high transmission power because the new route must contain a default power link. AODV is used as the routing protocol. In [34] changes were made at the MAC and routing layers to implement the proposed mechanisms.

Comparison Nine cross-layer proposals have been studied: TCP-F, TCP-BuS, ELFN, ATCP, ECIA, ENIC, preemptive routing, ATRA, and signal strength-based LM. These proposals address the problem of route failure and seek to ensure that route failure and congestion losses are not misinterpreted.

The TCP-F mechanism used the route failure and route re-establishment notification to inform the sender about route status. The TCP-F mechanism also has the route recovery timer. When this timer expires before receiving route re-establishment notification, then the sender retransmits all the unacknowledged packets. This retransmission of unacknowledged packets can lead to the injection of a burst of packets into the network and can create network congestion. In contrast, restoring variables to their original values on receiving RRN can lead to two types of problems: (1) it is possible that the new path will be long and timeout occurs; (2) the bandwidth will be different on the new path, which may be unsuitable for continued transmission with same cwnd. TCP-BuS is based on TCP-F using the network layer notification about route failure and re-establishment. TCP-BuS provides a reliable mechanism for control messages over TCP-F and provides selective ACK for the retransmission of lost packets.

A group of four approaches ELFN, ATCP, ECIA and ENIC does not use the route re-establishment mechanism like TCP-F and TCP-BuS. The ELFN mechanism is the first approach in this group. It modified the route failure message of DSR to piggy-back the route failure notification for TCP, while using the probe packet to detect route re-establishment. On route availability, ELFN comes out of standby mode into normal mode, restoring the retransmission timer and cwnd to their original states. It is possible that the original retransmission timer and cwnd may not be suitable for the new path as is the case with TCP-F. The authors state that for efficient performance of a network, it is better to restore the cwnd to its original state (the state before the route failure) rather than restore the cwnd to its initial size as in the slow start phase. Thus, both the TCP-F and the ELFN mechanisms fail to find out the actual size of the cwnd required on the new path. In ELFN, there is no mechanism to divert the traffic on intermediate nodes as in TCP-F.

ATCP also uses the probe packet to detect route re-establishment and shrinks the cwnd on route re-establishment, which causes a slow down in the traffic and can lead to poor utilization of network resources, especially on long paths. ATCP and ELFN are contradictory in adapting the size of cwnd on the new path, and further analysis is needed to determine which option is the best. When ATCP moves to disconnected or loss state, it puts the sender in 'persist' mode. There will be no transmission in 'persist' mode. Because ATCP stops the forwarding of ACK packets to TCP in 'persist' mode, it causes the clocking mechanism of TCP to be lost.

Another approach known as ECIA is also based on ELFN, and as a result incorporates the characteristics of ELFN. However, ELFN informs just the sender about route failure, whereas ECIA informs both the sender and the receiver of the route failure, and if intermediate nodes cannot salvage data or ACK packets, then the

sender/receiver transmits those data/ACK packets to avoid transmission timeout.

The approach named ENIC is also based on ELFN to notify both the sender and the receiver about the route failure. However, it uses broadcast messages for all the related source/destination nodes. In ENIC, the intermediate nodes drop the packets on receiving the route failure message, which is the opposite strategy to that adopted in TCP-BuS which buffers the packets, while TCP-F diverts the traffic if another route is available. Therefore, there are three opinions on what to do with the packets on intermediate nodes in the event of route failure, which necessitates further evaluation of these approaches.

The three approaches named preemptive routing, signal strength-based LM and ATRA are based on signal strength to tackle route failure. The significance of these three approaches is that they inform the sender of route failure before it occurs.

In preemptive routing, when the signal strength between two nodes decreases below a specified threshold, then the source is informed before the path failure occurs to minimize the delay in establishing a new route. One of the unfavourable aspects of this approach is that it uses ping-pong control messages which create an extra overhead on the channel. Furthermore, the performance of this approach is dependent on the selection of the preemptive threshold. If the value is too low, then there will not be enough time to find an alternative path. If the value is too high, then it causes unnecessary route discovery to be initiated and unnecessary usage of the medium.

In the signal strength-based LM scheme, route discovery is also initiated before the failure of the current route. At route failure, an increase in the transmission power of the two nodes allows the packets to traverse the path. However, in the signal strength-based LM scheme, the second mechanism introduced, which is to double the number of reattempts for medium access, is suspect. This is because of wrong estimation in the second mechanism, which just causes unnecessary attempts for data transmission.

The third approach based on signal strength is ATRA. The route failure and re-establishment mechanisms of ATRA and ELFN are the same, but ATRA notifies all the sources that used the failed path in last T seconds of the route failure, whereas ELFN notifies only the source from which connection the packet was dropped. It is difficult to decide at this point without any further evaluation which technique is the best.

4.2.2. Congestion and transmission losses

Restricted congestion window enlargement (TCP/RCWE) TCP/RCWE [35] employs the ELFN [23] mechanism to tackle route failure. Its innovation is to tackle the congestion and packet loss due to a high bit error rate, and, for this reason, it is discussed under the category of 'Congestion and transmission losses.' In the TCP/RCWE mechanism, a network state estimator (NSTATE) has been introduced to detect whether a network is congested or not, which is based on the RTO and can be represented as

$$\text{NSTATE} = \text{true} \quad \text{if } \text{RTO}_{\text{new}} \leq \text{RTO}_{\text{old}}$$

$$\text{NSTATE} = \text{false} \quad \text{if} \quad \text{RTO}_{\text{new}} > \text{RTO}_{\text{old}}$$

According to the NSTATE, if the current RTO (RTO_{new}) is greater than the previous one (RTO_{old}), it means that the network is congested. In this case, the sender should continue its transmission without any increase in the size of the cwnd. If the current RTO is less than or equal to the previous one, then it means that there is no congestion and the sender can increase the size of the cwnd. During the running phase, the TCP/RCWE scheme will be in one of the three states, i.e. Normal, Congested, and Link break, see Figure 6. In the congested state, the sender decision is based on the NSTATE as mentioned above. On the other hand, the aim of the Link break state is to tackle route failure, but TCP/RCWE is totally dependent on the mechanism of ELFN in this case; there is no integration of any new idea for handling route failure in TCP/RCWE.

Cross-layer congestion control for TCP (C^3TCP) Kliazovich et al. [36] proposed a C^3TCP scheme which is a feedback system to mitigate the problem of network congestion. C^3TCP is focussed on limiting the amount of data emitted to the network in order to avoid congestion. In this regard, C^3TCP measures the available bandwidth at each node, and the delay encountered by each Link, and the gathered information, is inserted into the option field of the MAC header. After measuring bandwidth for the first node and its link delay, the second node must compare the bandwidth of both nodes and pick the minimum bandwidth, while the delay on the second link is summed with the delay on the first link, and this operation is repeated at each node. When a TCP data packet reaches its destination node, its MAC header will contain information on the minimum available bandwidth and the link delay for the whole path. When the destination node replies with TCP ACK, the gathered information is also transmitted to the sender. Based on the bandwidth and the delay information, the sender should adjust its outgoing traffic. It is noted that the Link Delay is obtained as the sum of the forward and backward delays.

Comparison The innovation of the TCP/RCWE mechanism is based on using the values of RTO to

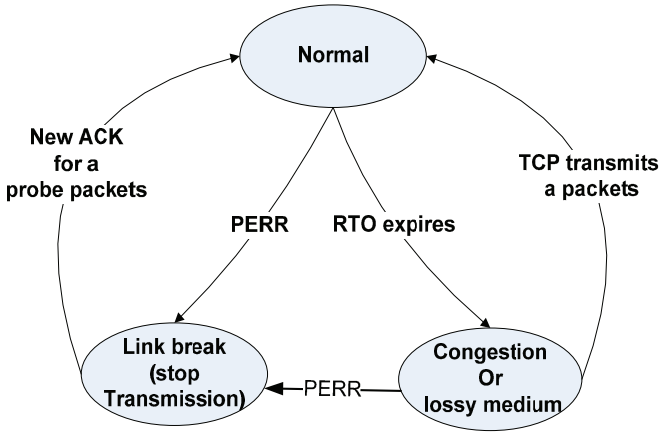

Figure 6 State diagram of TCP/RCWE [35].

increase or decrease the injection of packets into the network, while incorporating the ELFN mechanism to tackle the route failure. During simulation, TCP/RCWE was compared with TCP to establish that its performance is better. However, for route failure, TCP/RCWE is totally dependent on the ELFN mechanism, and the simulations failed to show any improvement of TCP/RCWE over ELFN. C^3TCP is measuring the end-to-end delay during each packet transmission and finding out the minimum available bandwidth along the path, and using the information to inject the data into the network accordingly. To compute the transmission rate, this mechanism uses the bandwidth-delay product which is the product of a data link's capacity and its end-to-end delay. The idea behind this proposal seems quite good, but the demand of this proposal is to provide optional field support to the existing IEEE 802.11 MAC protocol.

4.2.3. Shared medium

Wireless congestion control protocol (WCCP) Zhai et al. [10] proposed a scheme named WCCP to handle the contention problem. The authors of [10] state that contention is the main reason for TCP unfairness and throughput instability. WCCP uses the channel's busyness ratio to characterize the network utilization and congestion status. WCCP measures the available bandwidth in the shared channel and, based on this, internode and intra-node resource allocation schemes are proposed to determine the available resources for each node and each flow. WCCP uses two modules, one at the transport layer to replace the TCP windows algorithm with a rate based algorithm, while the other is between the network and the MAC layers to monitor and possibly modify the feedback field in each TCP data packet. This field is sent back to the sender in an ACK and, on the basis of received feedback, the sender regulates its transmission rate.

TCP contention control (TCTC) In the TCTC [37] scheme, the problem of TCP intra-flow instability, which lies in overloading the network by sending more data than the capacity of the channel, has been addressed. Intra-flow instability refers to a situation where the successive transmissions of packets in a single flow interfere with each other and, as a result, a large number of packets are dropped. To dynamically adjust the data transmission rate and provide intra-flow stability, the TCTC technique monitors two things at the receiver end for a fixed probe interval, one being the achieved throughput and the other the level of contention delay experienced by packets. Based on these observations, the receiver estimates the optimum amount of traffic and passes this information to the sender. As a result, the sender will regulate its transmission accordingly. Moreover, the contention delay-measuring time starts when a node places the first fragment of a packet at the beginning of a buffer and continues until the packet leaves the buffer for actual transmission. If a packet is dropped because of unsuccessful RTS/CTS handshake, then the delay is added to the contention delay of next packet. To estimate the optimum amount of traffic that should be sent by the sender, TCTC defines a new variable called the TCP contention window (*ctwnd*), the value of which is computed according to the following four stages of the TCTC scheme:

(i.) Fast probe: After connection establishment, TCTC enters the fast probe phase. In this phase, the *ctwnd* increases exponentially as it does in the TCP slow start phase. Whenever TCTC observes severe contention, then it enters a fast probe phase.

(ii.) Slow probe: If the throughput and contention delay are decreased compared to the previous probe interval, then it is an indication that the network resources are underutilized and TCTC enters a slow

probe phase. In this phase, the aim of the receiver is to increase the amount of network traffic by adding one MSS to the *ctwnd* after each probe interval.

(iii.) Light contention: When the receiver observes that there is an increase in the achieved throughput and contention delay, it means that the network is overloaded and, as a result, the TCTC scheme adopts the light contention phase. In this phase, the receiver reduces the amount of network traffic by subtracting one MSS from the *ctwnd*, while ignoring the decrease in the throughput.

(iv.) Severe contention: If the contention delay increases and the throughput decreases, then it is an indication of severe contention. At this stage, TCTC sets its *ctwnd* to 2*MSS to force the sender to slow down the data transmission rate, and TCTC enters a fast probe phase.

Comparison In this group, there are two cross-layer solutions, named WCCP and TCTC, to tackle the contention problem that arises because of the shared medium. WCCP uses the busyness ratio of the medium to measure the network utilization and congestion status, and then uses the measured value to adjust the transmission rate. The TCTC scheme measures the throughput achieved and the contention delay experienced by the packets on receiver side for a fixed time interval and then computes the optimum transmission rate for the sender. Thus, TCTC is totally dependent on the value of the time interval, if measurements are taken on expiry of a properly selected time interval well then; otherwise, the network will be underutilized or more congested.

4.3. Layered approaches

The layered proposals for TCP improvement are now presented under their second-level subcategories.

4.3.1. Route failure

Fixed RTO Dyer and Boppana [38] analysed the performance of TCP over three routing protocols which included two on-demand routing protocols named DSR [24] and AODV [32], and an adaptive distance vector (ADV) [39] routing protocol that combines an on-demand approach with proactive distance vector routing. It was observed that ADV performed well under variety of conditions compared to the on-demand routing protocols, DSR and AODV.

Moreover, a scheme is proposed in [38] to distinguish between route-failure losses and congestion losses which is a sender-side scheme named fixed-RTO (retransmission timeout). However, ADV does not get any benefit from this scheme because its route repair does not occur fast enough. In traditional TCP, RTO increases exponentially, but in fixed-RTO, on the expiry of two RTOs consecutively without receiving the ACK, it is assumed that route failure has occurred, and then the unacknowledged packets are retransmitted without increasing the RTO for a second time. According to [38], Fixed-RTO is only applicable in wireless environments and can be implemented easily in UNIX, by modifying the *getsockopt* and *setsockopt* functions.

TCP-friendly transport protocol The scheme proposed in [40] is an end-to-end approach named ADTCP which tries to detect congestion, route disconnection/failure and channel errors, and trigger an appropriate action. Moreover, ADTCP uses two metrics, the inter-delay difference (IDD), and the short-term throughput (STT) to detect network congestion. The IDD is the interval between two consecutive packets arriving at the receiver end. An increase in this interval is a sign of network congestion. STT is the throughput measured for a specific interval in the near past at the receiver end. A decrease in STT is also a sign of network congestion. These two metrics are used in combination to determine the network state which is either congested or not, and how to regulate the data transmission. On the other hand, the out-of-order packets and loss ratio are used for detecting route changes and channel losses. The feedback is provided to the sender in ACK which then reacts accordingly.

TCP detection of out-of-order and response (*TCP-DOOR*): Wang et al. [41] proposed a transport-layer scheme named TCP-DOOR, which is a pure end-to-end approach, to tackle the problem of route failure. In conventional TCP, when a sender sends packets, they should arrive in sequence at the receiver end. Otherwise, the receiver assumes that packets have been lost. However, on receipt of out-of-order packets, TCP-DOOR assumes that a route change has occurred.

In a TCP session, the delivery of out-of-order packets can happen in either direction, i.e. it can happen with data packets or ACK packets. The sender detects out-of-order delivery of ACK packets from the ACK sequence number, since every new ACK sequence number must be greater than the previous one if the ACKs are received in correct order. However, two duplicate ACKs have the same contents. Therefore, additional information is required to detect out-of-order delivery. In this regard, it is proposed to add a one byte TCP option to the TCP-ACK header called an ACK duplication sequence number (ADSN). The initial value of the ADSN will be zero with each ACK, while sending a duplicate ACK for the same sequence number will trigger an increment in the ADSN number.

At the receiver end, to detect the delivery of the out-of-order packets, two mechanisms were suggested. The first one is the addition of a two-bytes TCP option to the TCP header called a TCP packet sequence number (TPSN). If the initial value of the TPSN is zero, then it will be incremented with each data packet sent,

including retransmitted data packets. Thus, in a case of retransmission, the TPSN will be different from the previous one. The second method is to add a time stamp to each data packet instead of a TPSN.

On detecting out-of-order events, the receiver must inform the sender by setting the out-of-order bit in the TCP-ACK. After the sender knows about the out-of-order event, it can take the following two actions: the sender disables the congestion control for a pre-defined time and the values of state variables remain unchanged, and as a second action (known as instant recovery), TCP sender adopts a somewhat older state, which was invoked in the past T seconds. It should be clear that TCP-DOOR is a derivative of TCP-SACK.

Backup path routing Lim et al. [42] proposed a backup path routing protocol to improve path availability in mobile environments. In [42], the performance of TCP was analysed over an on-demand multipath *ad hoc* routing protocol named split multipath routing (SMR) [43] which is an extension of DSR protocol. It was observed that the multipath routing is detrimental to the performance of TCP. As a result, the backup path routing protocol strategy was suggested.

SMR uses two approaches to discover a new path. In the first approach, a new route discovery process is initiated as soon as a route becomes invalid. In the second approach, a route discovery process is initiated, when all routes become invalid. Through analysis, it is found that the second approach is better than the first one. Therefore, it is adopted in the proposed backup path routing protocol.

In addition, it was reported in [42] that TCP performed better when a single path was used for a TCP connection compared with using multiple paths for a single TCP connection. This is because forwarding data through different paths maximizes the chance of out-of-order packets, while TCP generates duplicate ACK in case of out-of-order packets and, as a result, there will be data retransmission. Moreover, each route has different RTT, and so there will be inaccurate average RTT calculation. Therefore, in the proposed backup path routing protocol, it is suggested to discover two paths between the source and the destination for each connection. However, only a single path is used for data transmission and the second one maintained as a backup path to minimise the chance of route unavailability. To select a primary path and a backup path, two selection options were suggested. The first option is to select the shortest-hop path as the primary path and the shortest-delay path as the backup path. In the second option, the shortest-delay path is selected as the primary path and the maximally disjoint path as the backup path. In both cases, it is a good practice to select paths where both the primary and the backup paths have minimum overlapping nodes.

Comparison Four layered proposals addressing the problem of route failure have been studied in this section. Three of them named Fixed RTO, TCP-DOOR and AD-TCP are transport-layer solutions, while backup path routing is a network-layer solution. In Fixed RTO when two consecutive RTOs expire, it is assumed that the path is not available, and the packet is retransmitted without increasing the RTO. Thus, it has an effect like an ELFN-probing packet, but Fixed RTO is easier to implement than ELFN. The second transport-layer approach named TCP DOOR has been shown to deliver a 50% improvement in throughput, but it fails to set the transmission rate properly on the new path.

The AD-TCP mechanism uses the IDD and short-term throughput to detect network congestion, while out-of-order packets and loss ratio are used for detecting route change and channel losses. However, from an implementation point of view, 300 lines of code are needed for AD-TCP which adds an extra computational load on the system. In contrast, the backup path-routing mechanism has been shown to deliver a 30% improvement in TCP performance, while discovering two paths for transmission. One of these paths is used for transmission and the other one as a backup path. Thus, the data and ACK packets of a connection are forwarded through the same path for better performance, but the TCP-COPAS (TCP-contention-based path selection) mechanism, which will be discussed in Section 4.3.3, contradicts this strategy.

4.3.2. Congestion and transmission losses

Fractional window increment (FeW) The scheme is proposed by Nahm et al. [44]. First of all, it was analysed how the network congestion and MAC contention effect the interaction between TCP and on-demand *ad hoc* routing protocol. It was observed that one of the reasons for TCP's low throughput lies in its window mechanism. By nature, TCP is aggressive in incrementing the cwnd, which causes network congestion and channel contention.

In addition, it was also observed that in wireless networks, packets are dropped at the link layer because of channel contention. These losses greatly affect the performance of routing protocols, because on-demand routing protocols perceive such losses as a route failure and initiate a new route discovery process. As a result, there will be increase in the network congestion, and more packets are dropped because of channel contention. This situation may cause TCP timeout, and then, the TCP slow start algorithm will be initiated. Therefore, the FeW scheme restricted the growth in the cwnd to reduce the aggressiveness of TCP and reduce the loss

ratio, so that TCP can achieve a high throughput. The mathematical equation of [45] was also used for the evaluation of FeW, and it was found that it could increase the throughput compared to standard TCP.

TCP with adaptive pacing (TCP-AP) *TCP-AP* [46] is a novel congestion control algorithm to handle the injection of data into the network and avoid large bursts of packets. TCP-AP is a pure end-to-end approach that seeks to keep a proper pacing between the packets before injecting them into the network. To introduce adaptive pacing between successive packets transmission, TCP-AP uses an estimation of the four-hop propagation delay (FHD) and the coefficient of variation of the recently measured round trip times (RTTs). The FHD is the time a packet takes, after its transmission, to arrive at a node four hops away from the source node. It is claimed that TCP-AP achieved up to 84% increase in the goodput.

Adaptive packet size on top of FeW (APS-FeW) Wang et al. [47] studied the window adjustment mechanism of *FeW* [44] and found that it cannot make full use of its predicted window. The fractional part of the cwnd has no effect on transmission and, consequently, a certain amount of predicted network capacity is wasted. Thus, the FeW scheme is too strict in limiting the amount of data transmission. Therefore, an adaptive packet size (APS) [47] scheme was proposed to fully utilize the size of the predicted window. The packet size in FeW is fixed, whereas APS-FeW can adapt the packet size to the current predicted window. APS uses equation (A) below to compute the current packet size.

$$packetsize\_ = ((cwnd\_ \times initPacketsize\_)/cwnd\_) \qquad (A)$$

where *initPacketsize_* is the initial packet size which has a fixed value, *cwnd_* is the cwnd size and *packetsize_* is the current packet size. The TCP sender uses equation (A) to compute the new packet size when there is any change in the cwnd. Moreover, when retransmission timeout occurs, the TCP sender goes into the slow-start phase resetting the cwnd to 1. The TCP source needs to repack the data in its buffer with an initial packet size. When the TCP source enters the fast retransmit phase due to three duplicate ACKs, then there is no need to repack the lost data packet, but just to transmit the packets in the buffer. The current packet's size never goes beyond double that of the initial packet size

Comparison In this group, three approaches have been studied, namely TCP-AP, FeW and APS-Few. The objective of these approaches is to limit the injection of data into the network to avoid network congestion, and to reduce the packet losses. The TCP-AP approach uses an estimate of the FHD between the successive packets and achieves an up to 84% improvement in goodput over TCP NewReno. On the other hand, instead of introducing some pacing mechanism to slow down the traffic and avoid network congestion, FeW and APs-FeW both limit the increase in the cwnd to reduce the aggressiveness of TCP. However, FeW is too restrictive in limiting the cwnd, to the point of wasting some predicted network capacity, and so the aim of APS-FeW is to fix this problem with the FeW mechanism to achieve a higher throughput.

4.3.3. Shared medium

COPAS Cordeiro et al. [48] proposed a scheme named contention-based path selection (COPAS) to handle the problem of channel contention by forwarding the data and ACK packets through different least-contended paths. For route selection, two criteria were adopted: first of all to find out all the possible routes between the source and the destination, then select two disjoint routes for forwarding data and ACK packets. To select the least-contended routes, COPAS monitors the paths continuously for channel contention and diverts the traffic towards the least-contended path. To determine the contention in its neighbourhood, each node counts how many times it has been backed off during the last T_{BACKOFF} seconds, and then computes the weighted average (μBACKOFF). Each node appends the weighted average to the non-duplicate RREQ packet, and a decision is made on the basis of this weighted average as to which path is the least contended. The COPAS mechanism is applicable to any source initiated on-demand routing protocol such as AODV and DSR.

Dynamic delay acknowledgement (DD ACK) Altman et al. [49] proposed a scheme named DD ACK which is a receiver-side solution for dealing with the contention problem by limiting the number of ACK. This technique is based on RFC 1122 [50] which defines a standard for generating an ACK after **d** (**d** = 2) packets or after some specific time if **d** packets are not received in this time. In DD ACK, **d** varies from 1 to 4. To begin with, DD ACK generates an ACK for one packet (**d** = 1), and then gradually moves towards generating an ACK after every four packets (**d** = 4). Moreover, [49] defines three thresholds ***l1, l2 and l3*** to control the increase in the value of **d** such that **d** = 1 if the sequence number ***N*** is less than ***l1***; if the sequence number ***N*** is such that ***l1*** < ***N*** < ***l2***, then **d** = 2; **d** = 3 if ***l2*** <***N*** < ***l3;*** and **d** = 4 when ***l3*** < N. When **d** reaches 4, then there is no mechanism to bring it back down in this scheme. Suppose at some point the TCP connection enters a slow start phase, where **d** = 4, then more time is required to increase the cwnd to achieve good utilization of the available bandwidth.

Optimum packet scheduling for each traffic flow (OPET) Zhai et al. [51] proposed a scheme which

consists of two mechanisms. The first mechanism provides high-priority medium access to the current receiver to avoid intra-flow contention at each node. The second mechanism is backward-pressure scheduling which restricts the forwarding node from sending more packets to its downstream node. The downstream node will be ready to receive more packets for a particular flow when it forwards the previous packets of this flow. To restrict the forwarding node, each downstream node receives packets from the forwarding node up to a particular limit named the backward-pressure threshold. After reaching the backward-pressure threshold, a node sends a negative-clear-to-send (NCTS) message as a response to a RTS message instructing the upstream node to stop forwarding further packets. To resume the transmission after a NCTS message, the receiver initiates the three-way handshake mechanism CTS/DATA/ACK instead of the RTS/CTS/DATA/ACK handshake mechanism. In the CTS/DATA/ACK three-way handshake mechanism, the receiver should clearly identify the source address and flow ID.

Quick exchange (QE) and fast-forward (FF) (QE & FF) Two MAC layer mechanisms named QE and FF were proposed by Berger et al. [52] to tackle the problem of self-contention. QE aims to handle the problem of inter-flow self-contention (contention between packets of the same connection moving in opposite directions). FF aims to overcome intra-flow self-contention (contention among packets of the same connection moving in the same direction).

The quick exchange mechanism allows the two packets to be exchanged through a single RTS/CTS control message as shown in Figure 7, which adds an extra data packet (DATA2) to the normal exchange procedure of RTS/CTS/DATA/ACK, where DATA2 is a packet moving in the opposite direction to that of packet DATA1. In addition, the time field of the CTS message is modified by adding an extra time interval to reserve the channel for transmission of DATA2. All the neighbouring nodes should update their status on receiving the CTS message. After receiving the data packet DATA1, the receiver replies with ACK and must piggy-back the DATA2 packet with the ACK. Afterward, the original sender must acknowledge the DATA2 packet to complete the transmission process. If there is no DATA2 packet to transmit in the opposite direction, then the nodes continue transmission using the original RTS/CTS/DATA/ACK mechanism.

The FF mechanism is shown in Figure 8, where the RTS/CTS taking place is normal at the beginning, but during the ACK for the first set of DATA1 packets, the receiver sends the RTS messages with a piggybacked ACK if it has data packets to send in the opposite direction.

TCP-ADA (TCP with adaptive delayed ACK) The solution proposed by Singh et al. [53] is known as TCP-ADA, which is a receiver-side solution. The authors claim that TCP-ADA is the best choice for generating an ACK for one cwnd to handle the problem of channel contention and collision among Data and ACK packets.

TCP dynamic adaptive ACK (TCP-DAA) strategy TCP-DAA [54] is another solution that belongs to the category of generating a delayed ACK, which generates an ACK according to the channel conditions. If channel conditions are good, then the ACK is delayed for up to four packets; otherwise an ACK is generated after two packets have arrived. If out-of-order packets are received or packets are filling the gaps in the sequence space of packets in the receiver buffers, then an ACK is generated immediately. Fast retransmission of packets takes place after receiving three duplicate ACKs in conventional TCP. In TCP-DAA, this number is decreased from three to two packets to minimize unnecessary retransmission. Figure 9 shows how the receiver dynamic window (*dwin*) changes. Under normal conditions, it is maintained at a maximum size of four

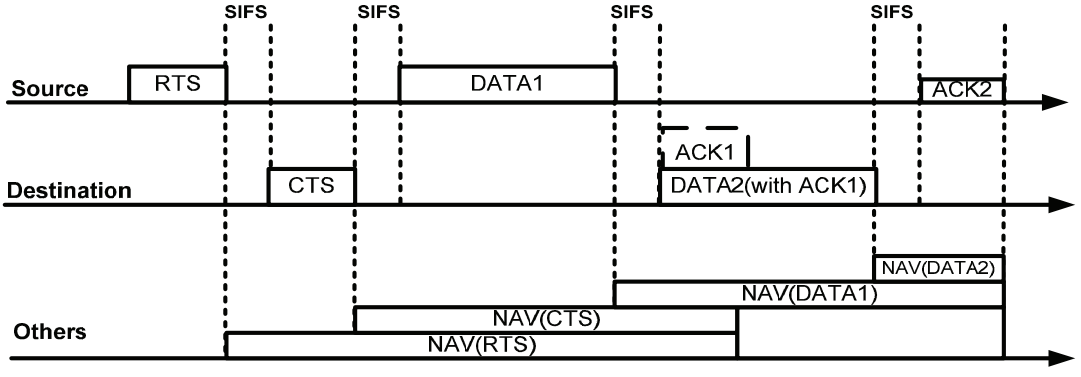

Figure 7 802.11 with Quick Exchange [52].

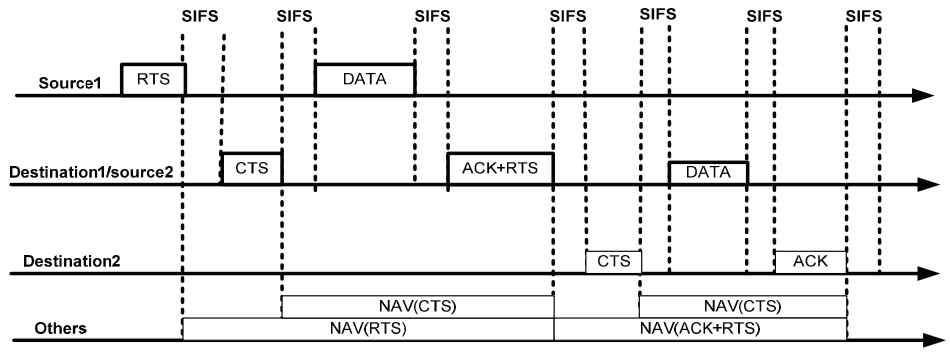

Figure 8 802.11 with Fast Exchange [52].

packets. When there is any packet drop, then TCP-DAA reduces the *dwin* size to two, i.e. it generates an ACK after two packets. Afterward the size of *dwin* is incremented to delay ACK up to three packets and then four.

TCP-delayed cumulative ACK (TCP-DCA) Chen et al. [55] proposed a scheme known as TCP-DCA which also belongs to the category of generating a delayed ACK. TCP-DCA is inspired by the idea underpinning TCP-DAA. It tries to determine the delay window based on the hop count. For a short path ($h \leq 3$, h represents the number of hops) TCP-DCA will delay an ACK for the whole cwnd to achieve the best performance. For paths where $3 < h \leq 9$, TCP-DCA will send an ACK after five packets; if $h \geq 10$ then an ACK is sent after three packets. The delay windows proposed by TCP-DCA based on hop count are listed in Table 3.

The simulations of [55] used each of AODV, DSR and greedy perimeter stateless routing (GPRS) [56] as the routing protocol.

Link randomly early detection (LRED) Fu et al. [57] observed that there exists an optimal cwnd size for which TCP can achieve the best performance, but that, instead of trying to find the optimal cwnd size, TCP increases its cwnd aggressively, which leads to packets being dropped at the link layer. It is also observed in [57] that the drop of packets due to buffer overflow is rare in wireless networks if the buffer size at a node is greater than 10 packets. Actually, in wireless *ad hoc* networks, medium contention is the major cause of dropped packets, which is also an indication that the network has been overloaded. As a solution, two mechanisms were proposed to address contention and unfairness named LRED and adaptive pacing.

The LRED algorithm maintains the average number of attempts for medium access. When the average number reaches a particular threshold, then the packets' dropping probability is computed according to the random early detection (RED) [58] algorithm. The RED algorithm is a mechanism for wired networks to drop packets from the router queue, when the queue size becomes greater than a particular threshold. In notifying the sender, in the RED algorithm, the dropped packets are taken to be a sign of congestion, whereas in the LRED algorithm, they are taken to be a sign of contention.

The second mechanism, adaptive pacing, is used for regulating the data flow in a more balanced and fair way. Adaptive pacing adds an extra interval to the backoff time, which is equal to the transmission time of the previous data packet. Addition of this extra interval to the backoff time causes a slowdown in the data flow rate and, as a result, poor utilization of the network resources.

Neighbourhood randomly early detection (NRED) Contention among the nodes due to the shared medium also leads to the problem of unfairness; in this regard,

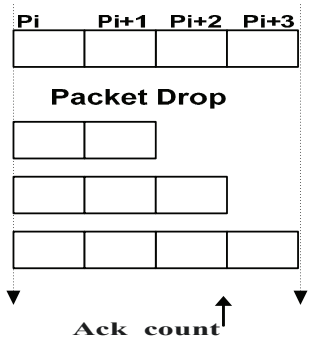

Figure 9 Receiver dynamic window for delaying packets in TCP-DAA [54].

Table 3 Delay window at the TCP-DCA receiver

Path length (h)	Delay window limit
$h \leq 3$	cwnd
$3 < h \leq 9$	5
$h \geq 10$	3

Xu et al. [59] proposed a NRED scheme that is based on the RED [58] algorithm to reduce the impact of unfairness among nodes. The NRED scheme introduced a distributed neighbourhood queue approach which is an aggregation of neighbourhood queues so that each neighbour node holds a portion of the distributed queue. Monitoring the channel usage, a node estimates the size of the neighbourhood-distributed queue and computes its dropping/marking probability to ensure a fair share of dropped/marked packets. The proposed technique is based on the link layer and consists of three sub-schemes called neighbourhood-congestion detection (NCD), neighbourhood-congestion notification (NCN), and distributed neighbourhood packet drop (DNPD). The task of NCD is to compute the average queue size of the distributed queue. Analysing the channel utilization for different time slots, when and how a node informs its neighbouring nodes about the congestion is the task of NCN where DNPD is responsible for computing the local drop probability.

Comparison Nine approaches were studied in this section, in which COPAS is the network-layer mechanism, and OPET, LRED, QE & FF, and NRED are MAC- and link-layer approaches. Whilst the third group, which includes the TCP-Delay ACK, TCP-ADA, TCP-DAA and TCP-DCA approaches, is based on the transport layer, the COPAS mechanism is only applicable in source-initiated routing and has shown a 90% improvement in throughput forwarding data and ACK packets through different paths. However, the simulations were carried out assuming a static environment. On the other hand, in the ATRA framework, the symmetric route pinning (SRP) mechanism forwards the DATA and ACK packets through the same path. Furthermore, the authors of the ATRA framework state that the use of asymmetric routes in a static environment is not an issue, but in a mobile environment, using asymmetric paths increases the probability of route failure.

The OPET mechanism, to give high priority to the current receiver for the medium access, sets the value of the backoff window to 8. This technique produces an effect like TCP-AP, which provides a pacing between the successive packets based on the FHD and the coefficient of variation of recent round trip times. However, the backward-pressure mechanism seems restrictive and is dependent on the permission of other nodes. The LRED mechanism achieved between 5 and 30% improvement in the throughput. Its adaptive facing mechanism adds an extra interval to the backoff time which is equal to the transmission time of the previous packet. Like OPET, in LRED, the transmitting node provides a chance to the next hop to transmit first by increasing its own backoff time, and this causes a slowdown in the traffic.

The NRED algorithm provides fairness based on the estimated size of a distributed queue. Its performance is mainly dependent on two parameters; the time interval T_{interval}, and the weight of the queue W_{q}. Proper selection of these two parameters is key to the success of NRED. On the whole, the idea of QE & FF is quiet promising, and simulation results show that the FF mechanism can improve the throughput of UDP, but also that there is high variance in the round trip time (RTT) of TCP with the FF mechanism. Thus, TCP fails to perform better due to this high variance in RTT.

A group of approaches, which includes TCP-Delay ACK, TCP-ADA, TCP-DAA and TCP-DCA, reduces the number of ACK to decrease the channel contention. The TCP-Delay ACK approach decreases the number of ACK on the basis of a sequence number, but the sequence number has no relation with congestion and contention. To explain its other unfavourable aspect, suppose TCP-Delay is at the stage of generating an ACK after four packets. Let TCP enter the slow start phase; at this stage, it takes a long time before TCP's cwnd is increased because of the unavailability of the mechanism to generate ACK quickly in such a situation without waiting for the arrival of four packets.

TCP-ADA states that the best option is to generate ACK after one cwnd. The problem with TCP-ADA is that, if there is ACK loss, then generating another ACK after receiving another cwnd will lead to retransmission timeout. TCP-DAA also generates ACK after four packets if the channel condition is good, but in the case of a packet loss it has a mechanism to generate ACK after two packets to avoid unnecessary delay. TCP-DCA is based on TCP-DAA, but its novelty is to delay ACK on the basis of hop counts.

5. Conclusions and possible future directions

In this article, a survey has been reported in the area of wireless *ad hoc* networks focussing on the approaches suggested for performance improvement of transmission control protocol (TCP) in wireless *ad hoc* networks. This article clearly mentions the problems that TCP faces in wireless *ad hoc* networks, namely, frequent route failure and channel contention, which are serious problems and provide the grounds for other problems to arise.

The enhanced classification of TCP performance improvement approaches that is presented in this article makes it easy to compare the approaches that fall under the same category (address the same problem(s)). This was illustrated by a category-by-category comparison of proposals. It is difficult to present a comprehensive comparison of all the approaches together because each one addresses a specific problem(s). Tables have been provided, which summarize each proposed approach

considered in more detail to further assist readers in finding out as to which specific layer(s) is(are) involved in the implementation of each approach. The tables also identify whether the approach is sender side, receiver side or a combination of sender-and-receiver side, and whether the approach uses intermediate nodes for feedback.

Now considering the different problems associated with the use of TCP in wireless *ad hoc* networks and the proposed approaches to solve them, it is clear that the transport layer must be aware of the network state to react properly and achieve maximum performance. To make itself aware of the network state, TCP must receive feedback from the lower layers. Approaches are available, which provide feedback from the lower layers and achieve promising results, indicating that cross-layer solutions are among the best approaches to tackle the problems of TCP on wireless *ad hoc* networks.

Looking at the problem of route failure, a number of cross-layer approaches as well as layered approaches are available. Most of the cross-layer approaches concern the involvement of the network klayer. When a route is re-established after failure, some of the approaches suggest continuing transmission at the same rate, i.e. there is no need to reduce the cwnd size, while other approaches suggest reducing the cwnd size to its minimum value to keep the transmission rate to a minimum. There is a need to evaluate which of these contradictory approaches is the best one. In the case of route failure, some of the approaches just send probe packets to determine whether the path has been re-established or not, while in some cases the sender is waiting to receive a route re-establishment message from the intermediate nodes. At the same time, it would be useful to find out the transmission rate a sender should adopt on a new route, because in reality, it is possible that a different bandwidth will be available on the new route.

Channel contention is one of the main reasons for TCPs performance degradation on wireless *ad hoc* networks. Most of the approaches dealing with contention try to limit the amount of data being injected into the network. The delay ACK approaches are focussed on reducing the amount of data flow by reducing the number of ACK. On the other hand, TCP, which is a self-clocking protocol, is dependent on ACK to increase the cwnd size. Therefore, in case of Delay ACK, the cwnd size increases very slowly which may result in poor utilization of available bandwidth. There is no guarantee that the delayed ACKs are delivered successfully; if not, then the TCP source must wait for the next delayed ACK before it can take an action. As a result, transmission timeout may occur. The overall mechanism must be such that it achieves efficient utilization of the available bandwidth while minimizing the chance of transmission timeout. Furthermore, the TCP self-clocking mechanism should not be disturbed.

End notes

In general discussion about transmitting information from one node to another, the term 'packet' is used loosely to refer to a piece of data. However, the specific packet of data formed by TCP in the transport layer is called a 'segment' [60].

Abbreviations

AP: access point; *ADSN:* ACK duplication sequence number; *ACKs*: acknowledgements; *AODV: ad hoc* on-demand distance vector; *ATCP: ad hoc* TCP; *ADV:* adaptive distance vector; *APS:* adaptive packet size; *APS-FeW:* adaptive packet size on top of FeW; *ABR:* associativity-based routing protocol; *BEAD:* best effort ACK delivery; *CSMA/CA:* carrier sense multiple access with collision avoidance; *cwnd*: congestion window; *ctwnd*: contention window; *COPAS:* contention-based path selection; C^3TCP: cross-layer congestion control for TCP; *DACK:* delay acknowledgement; *DNPD:* distributed neighbourhood packet drop; *DD: ACK* dynamic delay acknowledgement; *DSR:* dynamic source routing; *dwin*: dynamic window; *EPLN:* early packet loss notification; *ENIC:* enhanced inter-layer communication and control; *ECN:* explicit congestion notification; *ELFN:* explicit link failure notification; *ERDN:* explicit route disconnection notification; *EREN:* explicit route error notification; *ERRN:* explicit route recover notification; *ERSN:* explicit route state notification; *ERSN:* explicit route successful notification; *ECIA:* exploiting cross-layer information awareness; *FF:* fast-forward; *FeW:* fractional window increment; *GPRS:* greedy perimeter stateless routing; *IDD:* inter-delay difference; *IPC:* inter-process communication; *ISO:* International Standards Organization; *ICMP:* internet control message protocol; *LM:* link management; *LRED:* link randomly early detection; *MSS:* maximum segment size; *NCTS:* negative clear to send; *NCD:* neighbourhood congestion detection; *NCN:* neighbourhood congestion notification; *NRED:* neighbourhood randomly early detection; *NSTATE:* network state estimator; *OSI:* open system interconnection; *OPET:* optimum packet scheduling for each traffic flow; *PN:* pivoting node; *PRE:* proactive route errors; *FHD:* four-hop propagation delay; *QE:* quick exchange; *RED:* random early detection; *rwnd*: receiver window; *RTS/CTS:* request to send/ clear to send; *RCWE:* restricted congestion window enlargement; *RTO:* retransmission timeout; *RTT:* round trip time; *RERR:* route error; *RFN:* route failure notification; *RFP:* route failure prediction; *RRN:* route re-establishment notification; *RREP:* route reply; *RREQ:* route request; *SACK:* selective acknowledgment; *STT:* short-term throughput; *SMR:* split multipath routing; *SRP:* symmetric route pinning; *TCTC:* TCP contention control; *TCP-DOOR:* TCP detection of out-of-order and response; *TCP-DAA:* TCP dynamic adaptive ACK; *TCP-DAA:* TCP dynamic adaptive acknowledgement; *TPSN:* TCP packet sequence number; *TCP-ADA:* TCP with adaptive delayed acknowledgement; *TCP-AP:* TCP with adaptive pacing; *TCP-DCA:* TCP with delayed cumulative ACK; *TCP-BuS:* TCP-buffering capability and sequence information; *TCP-COPAS:* TCP-contention-based path selection; *TCP-DCA:* TCP-delayed cumulative ACK; *TCP-F:* TCP-feedback; *WCCP:* wireless congestion control protocol.

Acknowledgements

Noor Mast thanks the Kohat University of Science & Technology (KUST), Pakistan, and the Higher Education Commission, Pakistan, for funding his PhD studies.

Author details

[1]School of Engineering and Design, Brunel University, London, UK [2]Kohat University of Science and Technology, Kohat, Pakistan

Competing interests

The authors declare that they have no competing interests.

Received: 23 January 2011 Accepted: 13 September 2011
Published: 13 September 2011

References

1. ZJ Haas, M Gerla, DB Johnson, CE Perkins, MB Pursley, M Steenstrup, *et al*, Guest editorial wireless *ad hoc* networks. IEEE J Sel Areas Commun. **17**, 1329–1332 (1999). doi:10.1109/JSAC.1999.779916
2. DB Johnson, DA Maltz, Protocols for adaptive wireless and mobile networking. IEEE Pers Commun. **3**, 34–42 (1996)
3. IEEE Standard 802.11-2007, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification* (June 2007)
4. J Postel, *Transmission Control Protocol RFC 793* (September 1981)
5. V Paxson, M Allman, *RFC 2988: Computing TCP's Retransmission Timer* (Internet RFC, 2000)
6. M Matins, S Floyd, A Romanow, *TCP Selective Acknowledgment Options* (RFC.2018, 1996).
7. S Floyd, T Henderson, A Gurtov, *The NewReno Modification to TCP's Fast Recovery Algorithm* (RFC 2582, Internet Request for Comments, 1999)
8. K Xu, M Gerla, S Bae, Effectiveness of, RTS/CTS handshake in IEEE 802.11 based *ad hoc* networks. *Ad hoc Netw*. **1**, 107–123 (2003). doi:10.1016/S1570-8705(03)00015-5
9. A Gupta, I Wormsbecker, C Williamson, Experimental evaluation of TCP performance in multi-hop wireless *Ad hoc* networks, in *Proceedings–IEEE Computer Society's Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS*, 3–11 (2004)
10. H Zhai, X Chen, Y Fang, Improving transport layer performance in multihop *ad hoc* networks by exploiting MAC layer, information. IEEE Trans Wirel Commun. **6**, 1692–1701 (2007)
11. R Garces, JJ Garcia-Luna-Aceves, Floor acquisition multiple access with collision resolution, in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, 187–197 (1996)
12. CL Fullmer, JJ Garcia-Luna-Aceves, Solutions to hidden terminal problems in wireless networks. Comput Commun Rev. 39–49 (1997)
13. JJ Garcia-Luna-Aceves, CL Fullmer, Floor acquisition multiple access, (FAMA) in single-channel wireless networks. Mob Netw Appl. **4**, 157–174 (1999). doi:10.1023/A:1019146831447
14. S Xu, T Saadawi, Does the IEEE, 802.11 MAC protocol work well in multihop wireless *ad hoc* networks? IEEE Commun Mag. **39**, 130–137 (2001)
15. R Cheng, H Lin, A cross-layer design for TCP end-to-end performance improvement in multi-hop wireless networks. Comput Commun. **31**, 3145–3152 (2008). doi:10.1016/j.comcom.2008.04.017
16. WCY Lee, R Roy, *Mobile Communications Design Fundamentals*, 2nd edn. (Wiley, New York, 2006)
17. X Chen, H Zhai, J Wang, Y Fang, A survey on improving TCP, performance over wireless, networks. Resour Manag Wirel Netw. **16**, 657–695 (2005). doi:10.1007/0-387-23808-5_23
18. K-C Leung, V Li, Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges. IEEE Commun Surv Tutor. **8**, 64–79 (2006)
19. Al Hanbali, E Altman, P Nain, A survey of TCP, over *ad hoc* networks. IEEE Commun Surv Tutor. **7**, 22–36 (2005). doi:10.1109/COMST.2005.1610548
20. G Held, *Ethernet Networks: Design Implementation, Operation, Management*. (John Wiley and Sons Inc, New York, 2003)
21. F Foukalas, V Gazis, N Alonistioti, Cross-layer design proposals for wireless mobile networks: a survey and taxonomy. IEEE Commun Surv Tutor. **10**, 70–85 (2008)
22. K Chandran, S Raghunathan, S Venkatesan, R Prakash, A feedback based scheme for improving TCP performance in ad-hoc wireless networks, in *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS)*, 472–479 (1998)
23. G Holland, N Vaidya, Analysis of TCP performance over mobile *ad hoc* networks, in *Proceedings of the 5th Annual ACM/IEEE international Conference on Mobile Computing and Networking* (Seattle, Washington, United States, August 15-19, 1999) pp. 219–230
24. DB Johnson, DA Maltz, Dynamic source routing in *ad hoc* wireless networks. Mobile Comput, 153–181 (1996)
25. D Kim, Y Choi, TCP-BuS: improving TCP performance in wireless *ad hoc* networks, in *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*. **3**, 1707–1713 (2003)
26. CK Toh, Associativity-based routing for *ad hoc* mobile, networks. Wirel Pers Commun. **4**, 103–139 (1997). doi:10.1023/A:1008812928561
27. J Liu, S Singh, ATCP: TCP for mobile *ad hoc* networks. IEEE J Sel Areas Commun. **19**, 1300–1315 (2001). doi:10.1109/49.932698
28. S Floyd, TCP and explicit congestion, notification. ACM SIGCOMM Comput Commun Rev. **24**, 8–23 (1994)
29. X Yu, Improving TCP performance over mobile *ad hoc* networks by exploiting cross-layer information awareness, in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, 231–244 (2004)
30. D Sun, H Man, ENIC–An improved reliable transport scheme for mobile *ad hoc*, networks. in *Conference Record/IEEE Global Telecommunications Conference*. **5**, 2852–2856 (2001)
31. T Goff, N Abu-Ghazaleh, D Phatak, R Kahvecioglu, Preemptive routing in *ad hoc*, networks, J Parallel Distrib Comput. **63**, 123–140 (2003). doi:10.1016/S0743-7315(02)00059-X
32. CE Perkins, EM Royer, Ad-hoc on-demand distance vector routing, in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on* 90–100, (1999)
33. V Anantharaman, S Park, K Sundaresan, R Sivakumar, TCP Performance over mobile *ad hoc*, networks: a quantitative study. Wirel Commun Mob Comput. **4**, 203–222 (2004). doi:10.1002/wcm.172
34. F Klemm, Z Ye, SV Krishnamurthy, SK Tripathi, Improving TCP performance in *ad hoc* networks using signal strength based link, management. Ad hoc Netw. **3**, 175–191 (2005). doi:10.1016/j.adhoc.2004.07.005
35. M Gunes, D Vlahovic, The performance of the TCP/RCWE enhancement for ad-hoc networks. in *Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on* 43–48, (2002)
36. D Kliazovich, F Granelli, Cross-layer congestion control in *ad hoc* wireless, networks. *Ad hoc* Netw. **4**, 687–708 (2006). doi:10.1016/j.adhoc.2005.08.001
37. E Hamadani, V Rakocevic, A cross layer solution to address TCP, intra-flow performance degradation in multihop *ad hoc*, networks. J Internet Eng. **2**, 146–156 (2008)
38. TD Dyer, RV Boppana, A comparison of TCP performance over three routing protocols for mobile *ad hoc* networks, in *Proceedings of the 2001 ACM International Symposium on Mobile Ad hoc Networking and Computing: MobiHoc 2001* 56–66 (2001)
39. RV Boppana, SP Konduru, An adaptive distance vector routing algorithm for mobile, *ad hoc* networks, in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. **3**, 1753–1762 (2001)
40. Z Fu, B Greenstein, X Meng, S Lu, Design and implementation of a TCP-friendly transport protocol for *ad hoc* wireless networks, in *Proc of the 10th IEEE International Conference on Network Protocols (ICNP'02)*, 216–225 (2002)
41. F Wang, Y Zhang, Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response. in *Proceedings of the International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc)*, 217–225 (2002)
42. H Lim, K Xu, M Gerla, TCP performance over multipath routing in mobile *ad hoc* networks. IEEE Int Conf Commun. **2**, 1064–1068 (2003)
43. MS Lee, Gerla, Split multipath routing with maximally disjoint paths in *Ad hoc* networks. IEEE Int Conf Commun. **10**, 3201–3205 (2001)
44. K Nahm, A Helmy, C-J Kuo, TCP over multihop 802,11 networks: Issues and performance enhancement. in *Proceedings of the International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc)*, 277–287 (2005)
45. J Padhye, V Firoiu, D Towsley, J Kurose, Modeling TCP throughput: a simple model and its empirical validation. Comput Commun Rev. **28**, 303–314 (1998). doi:10.1145/285243.285291
46. SM ElRakabawy, A Klemm, C *Lindemann TCP with Adaptive Pacing For Multihop Wireless Networks*, 288–299 (2005)
47. X Wang, Y Han, Y Xu, APS-FeW: improving TCP throughput over multihop adhoc networks. Comput Commun. **32**, 19–24 (2009). doi:10.1016/j.comcom.2008.08.024
48. CDA Cordeiro, SR Das, DP Agrawal, COPAS: dynamic contention-balancing to enhance the performance of TCP over multi-hop wireless networks, in *Computer Communications and Networks, 2002. Proceedings Eleventh International Conference on*, 382–387 (2002)
49. E Altman, T Jiménez, Novel delayed ACK techniques for improving TCP performance in multihop wireless networks, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **2775**, 237–250 (2003)
50. R Braden, *RFC1122: Requirements for Internet Hosts-Communication Layers*, (RFC Editor United States, October 1989)

51. H Zhai, X Chen, Y Fang, Alleviating intra-flow and inter-flow contentions for reliable service in mobile *ad hoc* networks. *Proceedings–IEEE Military Communications Conference MILCOM*, 1640–1646 (2004)
52. D Berger, Ye Zhenqiang, P Sinha, S Krishnamurthy, M Faloutsos, SK Tripathi, TCP-friendly medium access control for ad-hoc wireless networks: alleviating self-contention, in *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, 214–223 (2004)
53. AK Singh, K Kankipati, TCP-ADA: TCP with adaptive delayed acknowledgement for mobile *ad hoc* networks, in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*. **3**, 1685–1690 (2004)
54. R de Oliveira, T Braun, A dynamic adaptive acknowledgment strategy for TCP over multihop wireless networks, in *INFOCOM, 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. **3**, 1863–1874 (2005)
55. J Chen, M Gerla, YZ Lee, MY Sanadidi, TCP with delayed ack for wireless, networks. *Ad hoc* Netw. **6**, 1098–1116 (2008). doi:10.1016/j.adhoc.2007.10.004
56. B Karp, HT Kung, GPSR: Greedy Perimeter Stateless Routing for wireless networks, in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, 243–254 (2000)
57. Z Fu, P Zerfos, H Luo, S Lu, L Zhang, M Gerla, The impact of multihop wireless channel on TCP throughput and loss, INFOCOM 2003, in *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, **3**, 1744–1753 (2003)
58. S Floyd, V Jacobson, Random early detection gateways for congestion avoidance. Netw IEEE/ACM Trans. **1**, 397–413 (1993). doi:10.1109/90.251892
59. K Xu, M Gerla, L Qi, Y Shu, TCP unfairness in *ad hoc* wireless networks, a neighborhood RED, solution. Wirel Netw. **11**, 383–399 (2005). doi:10.1007/s11276-005-1764-1
60. T Sheldon, *McGraw-Hill's Encyclopedia of Networking and Telecommunications* (McGraw-Hill Professional, New York, 2001)

doi:10.1186/1687-1499-2011-96
Cite this article as: Mast and Owens: **A survey of performance enhancement of transmission control protocol (TCP) in wireless *ad hoc* networks.** *EURASIP Journal on Wireless Communications and Networking* 2011 **2011**:96.