International Journal of Innovations & Advancement in Computer Science
IJIACS
ISSN 2347 – 8616
Volume 6, Issue 6
June 2017

# ISSUES AND CHALLENGES OF TCP CONGESTION CONTROL IN MOBILE ADHOC NETWORKS

***S. Mohanarangan,*** *Assistant Professor, Dept. of Computer Science and Engineering, Arunai College of Engineering, Tiruvannamalai, Tamilnadu.*
***D. Sivakumar****, Professor, Dept. of Electronics and Communication Engineering, Easwari Engineering College, Chennai, Tamilnadu*

***Abstract:*** *Congestion control in wireless networks has been extensively investigated over the years and several schemes and techniques have been developed, all with the aim of improving performance in wireless net-work. With the rapid expansion and implementation of wireless technology it is essential that the congestion control problem be solved. This paper presents a congestion control schemes in various wireless network environments which are different in slow start threshold calculation, bandwidth estimation, and congestion window manipulation. A comprehensive comparison of these approaches is given in relation to assumptions, bandwidth estimation, congestion window size manipulation, performance evaluation, fairness and friendliness and improved throughput.*

## I. INTRODUCTION

As a result of the advancement of wireless technology and the proliferation of handheld wireless terminals, recent years have witnessed an ever-increasing popularity of wireless networks, ranging from wireless Local Area Networks (WLANs) and wireless wide-area networks (WWANs) to mobile ad hoc networks (MANETs). In WLANs (e.g., the Wi-Fi technology) or in WWANs (e.g., 2.5G/3G/4G cellular networks), mobile hosts communicate with an access point or a base station that is connected to the wired networks. Obviously, only one hop wireless link is needed for communications between a mobile host and a stationary host in wired networks. In contrast, there is no fixed infrastructure such as base stations or access points in a MANET. Each node in a MANET is capable of moving independently and functioning as a router that discovers and maintains routes and forwards packets to other nodes. Thus, MANETs are multi-hop wireless networks by nature. Note that MANETs may be connected at the edges to the wired Internet.

Basically the wired networks and wireless networks are significantly different in terms of bandwidth, propagation delay, and link reliability. The implication of the difference is that packet losses are no longer mainly due to network congestion; they may well be due to some wireless specific reasons. As a matter of fact, in wireless LANs or cellular networks, most packet losses are due to high bit error rate in wireless channels and handoffs between two cells, while in mobile ad hoc networks, most packet losses are due to medium contention and route breakages, as well as radio channel errors. Therefore, although TCP performs well in wired networks, it will suffer from serious performance degradation in wireless networks if it misinterprets such non-congestion-related losses as a sign of congestion and consequently invokes congestion control and avoidance procedures, as confirmed through analysis and extensive simulations carried out in [3, 4, 5, 9-12]. As TCP performance deteriorates more seriously in ad hoc networks compared to WLANs or cellular networks, we divide wireless networks into two large groups: one is called one-hop wireless networks that include WLANs and cellular networks and the other is called multi-hop wireless networks that include MANETs.

To understand TCP behavior and improve TCP performance over wireless networks, given these wireless specific challenges, considerable research has been carried out and many schemes have been proposed. As the research in this area is still active and many problems are still wide open, this chapter serves to pinpoint the primary causes for TCP performance degradation over wireless networks, and cover the state of the art in the solution spectrum, in hopes that readers can better understand the problems and hence propose better solutions based on the current ones.

## II. OVERVIEW OF TCP

The basic functions of TCP as a transport layer protocol include flow control, error recovery and congestion control, while the state-of-the-art techniques include fast retransmission and recovery, selective acknowledgment, etc., mainly focusing on how to promptly and effectively respond to network congestion.

### *A. Basic Functionality of TCP*

It is well known that TCP is a connection-oriented transport protocol that is aimed at guaranteeing end-to-end reliable ordered delivery of data packets over wired networks. For this purpose, basic functionalities such as flow control, error control, and congestion control are indispensable. While these functions have a clean-cut definition of their own, in practice they are closely coupled with one another in TCP implementation.

In TCP, a sliding window protocol is used to implement flow control, in which three windows are used, namely, *Congestion Window*, *advertised window*, and *Transmission Window*. Congestion window indicates the maximum number of segments (Without causing confusion, the term segment and packet are used interchangeably henceforth) that the sender can transmit without congesting the network. As shown next in details on congestion control, this number is determined by the sender based on the feedback from the network. *advertised window*, however, is specified by the receiver in the acknowledgements it. Advertised window indicates to the sender the amount of data the receiver is ready to receive in the future. Normally, it equals to the available buffer size at the receiver in order to prevent buffer overflow. Transmission window means the maximum number of segments that the sender can transmit at one time without receiving any ACKs from the receiver. Its lower edge indicates the highest numbered segment acknowledged by the receiver. Obviously, to avoid network congestion and receiver buffer overflow, the size of transmission window is determined as the minimum of the congestion window and the receiver's advertised window.

To notify the sender that data is correctly received, TCP employs a cumulative acknowledgement (ACK) mechanism. In other words, upon the receipt of an ACK, the sender knows that all previously transmitted data segments with a sequence number less than the one indicated in the ACK are correctly received at the receiver. In the case that an out-of-order segment (identified on the basis of sequence numbers) arrives at the receiver, a duplicate ACK is generated and sent back to the sender. It is important to note that in wired networks, an out-of-order delivery usually implies a packet loss. If three duplicate cumulative ACKs are received, the sender will assume the packet is lost. A packet loss is also assumed if the sender does not receive an ACK for the packet within a timeout interval called retransmission timeout (RTO), which is dynamically computed as the estimated round-trip time (RTT) plus four times the mean deviation. By retransmitting the lost packet, TCP achieves reliable data delivery.

It turns out that in wired networks, almost all the packet losses are due to network congestion rather than transmission errors. Thus, in addition to retransmission, TCP responds to packet losses by invoking its congestion control mechanism. TCP congestion control is also based on the sliding window mechanism described above and consists of two major phases: *slow start* and *congestion avoidance*. In the slow start phase, the initial congestion window size (*cwnd*) is set to one maximum segment size (MSS) and is incremented by one MSS on each new acknowledgement. After *cwnd* reaches a preset threshold (*ssthresh*), the congestion avoidance starts and it is increased linearly, i.e., it is increased by one segment for each RTT. Upon a timeout, *ssthresh* is set to the half of the current transmission window size (but at least two segments) and the congestion window is reduced to 1 MSS. Then slow start mechanism starts again. This procedure is also called the additive increase and multiplicative decrease algorithm (AIMD, [14]). The entire congestion control algorithm is illustrated in Fig. 1. Note that the sender reacts to three duplicate ACKs in a different way, which is described in *fast retransmission and fast recovery* in the next subsection.

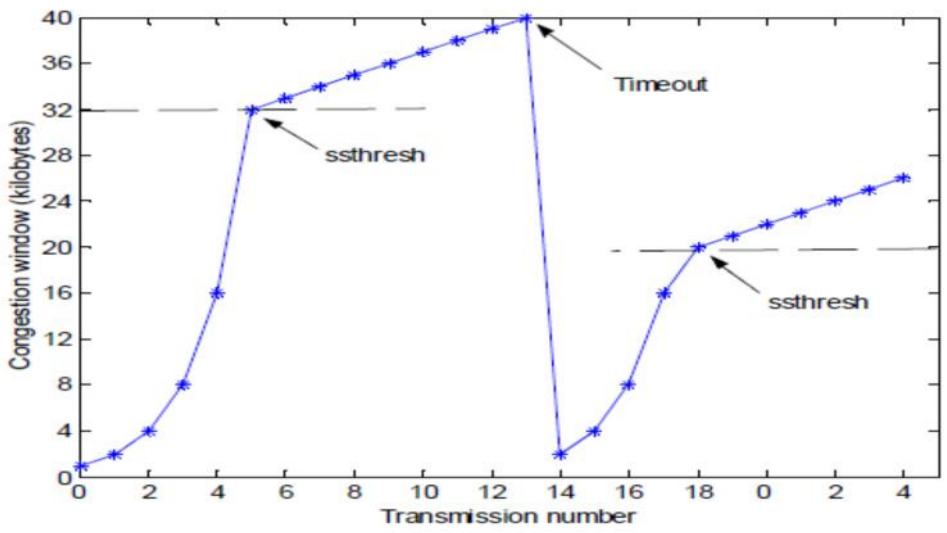

Figure 1: TCP congestion window dynamics

## III. TCP IN MOBILE AD HOC NETWORKS

TCP performance in mobile ad hoc networks is the focus of this section. It is expected that compared to one-hop wireless networks, TCP will encounter more serious difficulty in providing end-to-end communications in mobile ad hoc networks, as MANETs are, in essence, infrastructure less, self-organizing multi-hop networks, and lacking centralized network management. Next, we present the main problems in ad hoc networks, followed by recent solutions.

### *A. Challenges*

Some salient characteristics of mobile ad hoc networks, which seriously deteriorate TCP performance, include the unpredictable wireless channels due to fading and interference, the vulnerable shared media access due to random access collision, the hidden terminal problem and the exposed terminal problem, and the frequent route breakages due to node mobility. From the point of view of network layered architecture, these challenges can be broken down into five categories, i.e., a) the channel error, b) the medium contention, the hidden terminal problem, and the exposed terminal problem, c) the mobility, d) the multi-path routing, and e) congestion, whose adverse impacts on TCP is elaborated next.

#### *1) Channel Errors*

The effects of channel errors in ad hoc networks are similar to those in one-hop wireless networks except that they are more serious, since a TCP connection now may consist of multi-hop wireless links, unlike the situation in cellular networks or wireless LAN where only the last hop is wireless. Accordingly, the congestion window size at the sender may shrink more dramatically due to channel errors in several wireless hops, resulting in even lower throughput in ad hoc networks.

#### *2) Medium contention, Hidden terminal, and Exposed terminal problems*

Contention-based medium access control (MAC) schemes, such as IEEE 802.11 MAC protocol, have been widely studied and incorporated into many wireless test beds and simulation packages for wireless multi-hop ad hoc networks, where the neighboring nodes contend for the shared wireless channel before transmitting. There are three key problems, i.e., the hidden terminal problem, the exposed terminal problem, and unfairness. A hidden node is the one that is within the interfering range of the intended receiver but out of the sensing range of the transmitter. The receiver may not correctly receive the intended packet due to collision from the hidden node. An exposed node is the one that is within the sensing range of the transmitter but out of the interfering range of the receiver. Though its transmission does not interfere with the receiver, it could not start transmission because it senses a busy medium, which introduces spatial reuse deficiency. The binary exponential backoff scheme always favors the latest successful transmitter, and hence results in unfairness. These problems could be more harmful in multi-hop ad hoc networks than in Wireless LANs as ad hoc networks are characterized by multi-hop connectivity.

MAC protocols have been shown to significantly affect TCP performance [11, 12, 21, 23, 25, and 27]. When TCP runs over 802.11 MAC, as [27] pointed out, the instability problem becomes very serious. It is shown that collisions and the exposed terminal problem are two major reasons to prevent one node from reaching the other when they are in each other's transmission range. If a node cannot reach its adjacent node for several times, it will trigger a route failure, which in turn will cause the source node to start route discovery. Before a new route is found, no data packet can be sent out. During this process, TCP sender has to wait and will invoke congestion control algorithms if it observes a timeout. When it comes down to TCP throughput, serious oscillation will be observed. Moreover, the random backoff scheme used in the MAC layer makes this worse [11]. Since large data packet sizes and back-to-back packet transmission both decrease the chance of the intermediate node to gain access of the channel, the node has to back off a random time and attempt again. After several failed attempts, a route failure is reported.

TCP may also encounter serious unfairness problems [11, 21, 23, and 27] for the reasons stated below:

- Topology causes unfairness because of unequal channel access opportunity for different nodes. As shown in Fig. 4, where the small circle denotes a node's valid transmission range and the large circle denotes a node's interference range, all nodes in a 7-node chain topology experience different amount of competitions. There are TCP flows, namely flow 1 from node 0 to 1 and flow 2 from node 6 to 2. The transmission from node 0 to node 1 experiences interference from three nodes, i.e., nodes 1, 2, and 3, while the transmission from node 3 to node 2 experience interference from five nodes, i.e., nodes 0, 1, 2, 4, and 5. Flow 1 will obtain much higher throughput than flow 2 due to the unequal channel access opportunity.

- Backoff mechanism in the MAC may lead to unfairness as it always favors the last successfully transmitting node.

TCP flow length influences unfairness. Longer flows implies longer round trip time and higher packet dropping probability, leading to lower and more fluctuating TCP end-to-end throughput. Through this chain reaction, unfairness is amplified, as the high throughput will become higher and the low throughput lower.

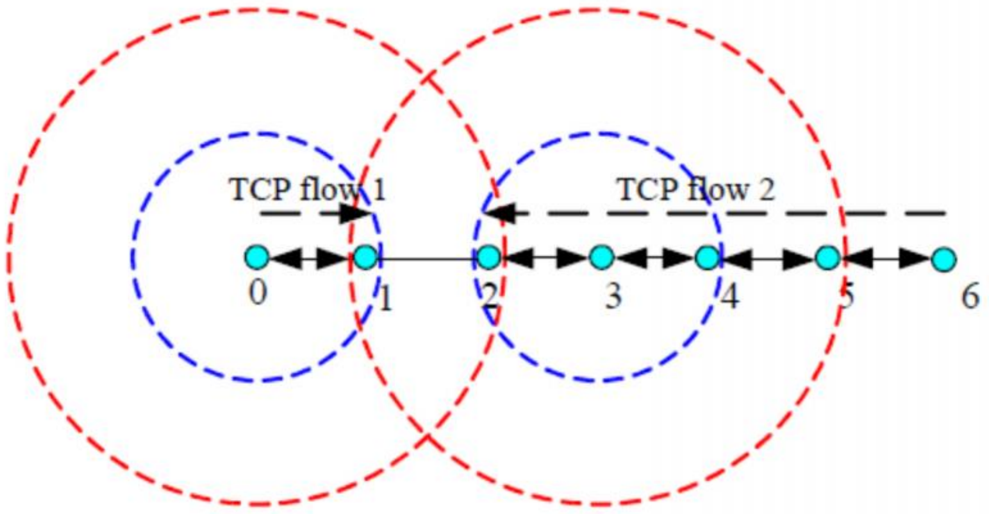

Figure 2: Node interference in chain topology

*3) Mobility*

Mobility may induce link breakage and route failure between two neighboring nodes, as one mobile node moves out of the other's transmission range. Link breakage in turn causes packet losses. As we said earlier, TCP cannot distinguish between packet losses due to route failures and packet losses due to congestion. Therefore, TCP congestion control mechanisms react adversely to such losses caused by route breakages [1, 8, and 16]. Meanwhile, discovering a new route may take significantly longer time than TCP sender's RTO. If route discovery time is longer than RTO, TCP sender will invoke congestion control after timeout. The already reduced throughput because of losses will further shrink. It could be even worse when the sender and the receiver of a TCP connection go into different network partitions. In such a case, multiple consecutive RTO timeouts lead to inactivity lasting for one or two minutes even if the sender and receiver finally get reconnected.

Fu et al. conducted simulations by considering mobility, channel error, and shared media-channel contention [10]. They indicated that mobility-induced network disconnections and reconnections have the most significant impact on TCP performance comparing to channel error and shared media-channel contention. TCP NewReno merely achieves about 10% of a reference TCP's throughput in such cases. As mobility increases, the relative throughput drops from almost 0% in a static scenario to 100% in a highly mobile scenario (when moving speed is 20m/sec). In contrast, congestion and mild channel error (say 1%) have less visible effect on TCP (with less than 10% performance drop compared with the reference TCP).

*4) Multi-path Routing*

Routes in MANETs are short-lived due to frequent link breakages. To reduce delay due to route re-computation, some routing protocols such as TORA [19] maintain multiple routes between a sender-receiver pair and use multi-path routing to transmit packets. In such a case, packets coming from different paths may not arrive at the receiver in order. Being unaware of multi-path routing, TCP receiver would misinterpret such out-of-order packet arrivals as a sign of congestion. The receiver will thus generate duplicate ACKs that cause the sender to invoke congestion control algorithms like fast retransmission (upon reception of three duplicate ACKs).

*5) Congestion*

It is known that TCP is an aggressive transport layer protocol. Its attempt to fully utilize the network bandwidth makes ad hoc networks easily go into congestion. In addition, due to many factors such as route change and unpredictable variable MAC delay, the relationship between congestion window size and the tolerable data rate for a route is no longer maintained in ad hoc networks. The congestion window size computed for the old route may be too large for the newly found route, resulting in network congestion as the sender still transmits at the full rate allowed by the old congestion window size.

Congestion/overload may give rise to buffer overflow and increased link contention, which adversely affects TCP performance. As a matter of fact, [17] showed the capacity of wireless ad hoc networks decreases as traffic and/or competing nodes arise.

*B. Current Solutions*

As is shown in the previous section, there is a magnitude of research work on improving TCP performance over one-hop wireless networks. However, many of these mechanisms are designed for infrastructure-based networks and depend on the base stations in distinguishing the error losses from congestion losses. Since mobile ad-hoc networks do not have such an infrastructure, they are hard to be applied in mobile ad-hoc networks directly.

More recently, several schemes have been proposed to improve TCP performance over mobile ad hoc networks. We classify the schemes into three groups, based on their fundamental philosophy: TCP with feedback schemes, TCP without feedback schemes, and TCP with lower layer enhancement schemes. Through the use of feedback information to signal non-congestion-related causes of packet losses, the feedback approaches help TCP distinguish between true network congestion and other problems such as channel errors, link contention, and route failures. On the other end of the solution spectrum, TCP without feedback schemes makes TCP adapt to route changes without relying on feedback from the network, in light of the concern that feedback mechanisms may bring about additional complexity and cost in ad hoc networks. The third group, lower layer enhancement schemes, starts with the idea that TCP sender should be hidden from any problems specific in ad hoc networks while lower layers such as routing layer and MAC layer need to be tailored with TCP's congestion control algorithms in mind. As expected, this idea guarantees that TCP end-to-end semantics is maintained for ad hoc networks to seamlessly internetwork with the wired Internet. In the following, we present some representative schemes according to the aforementioned taxonomy.

*1) TCP with Feedback Solutions TCP-F*

In the mobile ad hoc networks, topology may change rapidly due to the movement of mobile hosts. The frequent topology changes result in sudden packet losses and delays. TCP misinterprets such losses as congestion and invokes congestion, leading to unnecessary retransmission and loss of throughput. To overcome this problem, TCP-F (TCP-Feedback) [6] was proposed so that the sender can distinguish between route failure and network congestion. Similar to Freeze-TCP and M-TCP discussed above, the sender is forced to stop transmission without reducing window size upon route failure. As soon as the connection is reestablished, fast retransmission is enabled.

TCP-F relies on the network layer at an intermediate node to detect the route failure due to the mobility of its downstream neighbor along the route. A sender can be in an active state or a snooze state. In the active state, transport layer is controlled by the normal TCP. As soon as an intermediate node detects a broken route, it explicitly sends a route failure notification (RFN) packet to the sender and records this event. Upon reception of the RFN, the sender goes into the snooze state, in which the sender completely stops sending further packets, and freezes all of its timers and the values of state variables such as RTO and congestion window size. Meanwhile, all upstream intermediate nodes that receive the RFN invalidate the particular route in order to avoid further packet losses. The sender remains in the snooze state until it is notified of the restoration of the route through a route reestablishment notification (RRN) packet from an intermediate node. Then it resumes the transmission from the frozen state. The state machine of TCP-F is shown in Fig. 5.

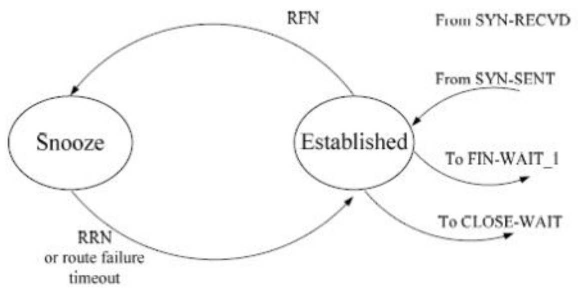

Figure 3: TCP-F F State Machine

*TCP-ELFN*

Holland and Vaidya proposed another feedback-based technique, the Explicit Link Failure Notification (ELFN) [13]. The goal is to inform the TCP sender of link and route failures so that it can avoid responding to the failures as if congestion occurs. ELFN is based on DSR [15] routing protocol. To implement ELFN message, the route failure message of DSR is modified to carry a payload similar to the “host unreachable” ICMP message. Upon receiving an ELFN, the TCP sender disables its congestion control mechanisms and enters a “stand-by” mode, which is similar to the snooze state of TCP-F mentioned above. Unlike TCP-F using an explicit notice to signal that a new route has been found, the sender, while on stand-by, periodically sends a small packet to probe the network to see if a route has been established. If there is a new route, the sender leaves the stand-by mode, restores its RTO and continues as normal. Recognizing most of popular routing protocols in ad hoc networks are on demand and route discovery/rediscovery is event driven, periodically sending a small packet at the sender is appropriate to restore route with mild overhead and without modification to the routing layer.

Through explicit route failure notification, TCP-EFLN and TCP-F allow the sender to instantly enter snooze state and avoid unnecessary retransmissions and congestion control which wastes precious MH battery power and scarce bandwidth. With explicit route reestablishment notification from intermediate nodes or active route probing initiated at the sender, these two schemes enable the sender to resume fast transmission as soon as possible. But neither of these two considers the effects of congestion, out-of-order packets, or bit errors, which are quite common in wireless ad hoc networks. In addition, both TCP-ELFN and TCP-F use the same parameter sets including congestion window size and RTO after reestablishment of routes as those before the route failure, which may cause problem because congestion window size and RTO are route specific. Using the same parameter sets helps little to approximate the available bandwidth of new route if the route changes significantly.

*ATCP*

ATCP (Ad hoc TCP) [18] also utilizes the network layer feedback. The idea of this approach is to insert a thin layer called ATCP between IP and TCP, which ensures correct behavior in the event of route failures as well as high bit error rate. The TCP sender can be put into a *persist* state, *congestion control* state or *retransmit* state, respectively, corresponding to the packet losses due to route breakage, true network congestion and high bit error rate. Note that unlike the previous two feedback-based approaches, packet corruption caused by channel errors has also been tackled. The sender can choose an appropriate state by learning the network state information through explicit congestion notification (ECN) messages and ICMP “Destination Unreachable” messages.

The state transition diagram for ATCP at the sender is shown in Fig. 6. Upon receiving a “Destination Unreachable” message, the sender enters the persist state. The TCP at the sender is frozen and no packets are sent until a new route is found, so the sender does not invoke congestion control. Upon receipt of an ECN, congestion control is invoked without waiting for a timeout event. If a packet loss happens and the ECN flag is not set, ATCP assumes the loss is due to bit errors and simply retransmits the lost packet. In case of Multi-path routing, upon receipt of duplicate ACKs, TCP sender does not invoke congestion control, because multi-path routing shuffles the order in which segments are received. So ATCP works well when the multi-path routing is applied.

ATCP is considered to be a more comprehensive approach in comparison with TCP-F and TCP-ELFN in that it accounts for more possible sources of deficiency including bit errors and out of order delivery due to multipath routing. Through re-computation of congestion window size each time after route reestablishment, ATCP may adapt to change of routes. Another benefit of ATCP is that it is transparent to TCP, and hence nodes with and without ATCP can interoperate.

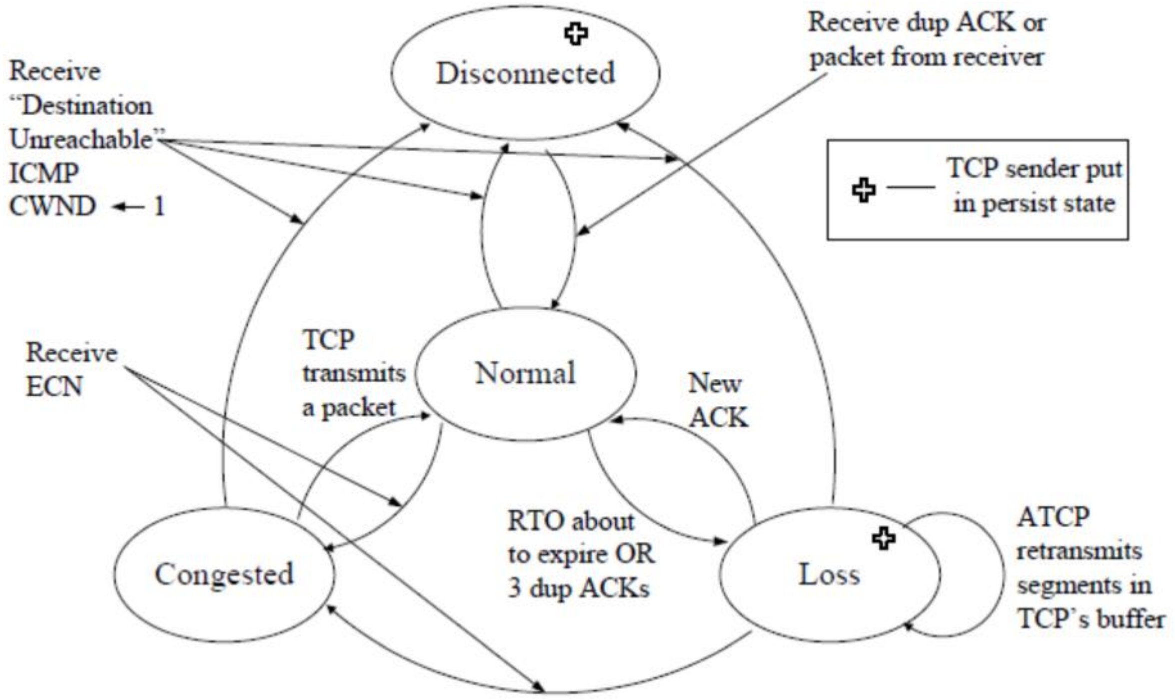
Disconnected
Receive "Destination Unreachable" ICMP CWND ← 1
Receive dup ACK or packet from receiver
✚ — TCP sender put in persist state
TCP transmits a packet
Normal
Receive ECN
New ACK
RTO about to expire OR 3 dup ACKs
Congested
Loss
ATCP retransmits segments in TCP's buffer

Figure 4: State Transition Diagram for ATCP at the sender

In summary, as shown by the simulations, these feedback-based approaches improve TCP performance significantly while maintaining TCP's congestion control behavior and end-to-end TCP semantics. However, all these schemes require that the intermediate nodes have the capability of detecting and reporting network states such as link breakages and congestion. Enhancement at the transport layer, network layer, and link layer are all required. It deserves further research on the ways to detect and distinguish network states in the intermediate nodes.

*2) TCP without Feedback Solutions*

*Adaptive Congestion Window Limit Setting*

Based on the observation that TCP's congestion control algorithm often over-shoots, leading to network overload and heavy contention at the MAC layer, Chen et al. [7] proposed an adaptive congestion window limit (CWL, measured in the number of packets) setting strategy to dynamically adjust TCP's CWL according to the current round-trip hop-count (RTHC) of the path, which can be obtained from routing protocols such as DSR. More precisely, the CWL should never exceed the RTHC of the path.

The rationale behind this scheme is very simple, as shown in the following. It is known that to fully utilize the capacity of a network, a TCP flow should set its CWL to the bandwidth-delay product (BDP) of the current path, where a path's BDP is defined as the product of the bottleneck bandwidth of the forward path and the packet transmission delay in a round trip. On the other hand, the CWL should never exceed the path's BDP in order to avoid network congestion. In ad hoc networks, if we assume the size of a data packet is S and the bottleneck bandwidth along the forward and return paths is the same and equal to $b_{min}$, it can be easily seen that the delay at any hop along the path is less than the delay at the bottleneck link, i.e., $S/b_{min}$. Since the size of a TCP acknowledgement is normally smaller than that of the data packet, according to the definition of the BDP, we know BDP <= RTHC×S. Therefore, the CWL, which is bounded by the path's BDP, should never exceed the RTHC of the path.

This upper bound can be further tightened when the IEEE 802.11 MAC layer protocol is adopted. In fact, it is shown that, in a chain topology, a tighter upper bound exists, which is approximately 1/5 of the RTHC of the path. According to this tighter upper bound, the maximum RTO is set to a relatively small value of 2 seconds, which enables TCP to probe the route quickly should it break (due to false link failure). Simulation results showed that this simple but useful strategy is able to improve TCP-Reno performance by 8% to 16% in a dynamic MANET.

*TCP-DOOR*

TCP-DOOR [24] attempts to improve TCP performance by detecting and responding to out-of-order (OOO) packet delivery events and thus avoiding invoking unnecessary congestion control. By definition, OOO occurs when a packet sent earlier arrives later than a subsequent packet. In ad hoc networks, OOO may happen multiple times in one TCP session because of route changes.

In order to detect OOO, ordering information is added to TCP ACKs and TCP data packets. OOO detection is carried out at both ends: the sender detects the Out-of-Order ACK packets and the receiver detects the Out-of-Order data packets. If the receiver detects OOO, it should notify the sender, considering the fact that it is the sender who takes congestion control actions. Once the TCP sender knows of an OOO condition, it may take one of the two responsive actions: temporarily disabling congestion control and instant recovery during congestion avoidance. The first action means that, whenever an OOO condition is detected, TCP sender will keep its state variables such as RTO and the congestion window size constant for a time period $T_1$. The second action means that, if during the past time period $T_2$, the TCP sender has already entered the state of congestion avoidance, and it should recover immediately to the state prior to such congestion avoidance. The main reason is the detection of OOO condition implies that a route change event has just occurred.

However, OOO can be detected only after a route has recovered from failures. As a result, TCP-DOOR is less accurate and responsive than a feedback-based approach that is able to determine whether congestion or route errors occur, and hence report to the sender at the very beginning. Furthermore, it may not work well with multi -path routing since multi-path routing may cause OOO as well. Therefore, it is concluded that TCP-DOOR may work as an alternative to the feedback-based approach to improve TCP performance over ad hoc network, if the latter is not available.

*Fixed RTO*

In TCP congestion control, TCP doubles the RTO and retransmits the oldest unacknowledged packet when the retransmission timer expires. Although this exponential backoff mechanism of the RTO could handle network congestion gracefully, it is no longer suitable in MANETs when the loss of packets or ACKs is caused by temporary route breakages, as discussed earlier. In such a case, the RTO should be recalculated, if possible, according to the new route instead of being doubled. Furthermore, when the new route is established, TCP sender should start the transmission immediately instead of waiting for the expiration of retransmit timer.

In the fixed RTO approach [8], no feedback from lower layers is needed. Rather, a heuristic is employed to distinguish route failures and congestion. When timeouts occur consecutively, i.e., an ACK is not received before the second RTO expires, the sender assumes a route failure rather than network congestion takes place. Therefore, the unacknowledged packet is retransmitted again without doubling the RTO. The RTO remains fixed until the route is re-established and the retransmitted packet is acknowledged. By adopting this strategy, the TCP sender avoids waiting for a long period of time before attempting to retransmit. This fast retransmission would force routing protocol especially like AODV [20] and DSR to repair routes fast, which in turn leads to a large congestion window on average and high TCP throughput. Actually, this technique complements TCP-DOOR.

*3) Lower Layer Enhancement Solutions*

*Routing Layer Enhancement*

A framework termed Atra, due to Anantharaman et al., aims to improve TCP performance over ad hoc networks by enhancing routing layers [2]. Three mechanisms, called *Symmetric Route Pinning (SRP), Route Failure Prediction (RFP),* and *Proactive Route Errors (PRE),* are introduced to minimize the probability of route failures, to predict route failures in advance, and to minimize the latency in conveying route failure information to source, respectively. Since asymmetric path would increase the probability of route failure for a connection, in the first mechanism, the ACK path of a TCP connection is always kept the same as the data path. Based on the progression of signal strengths of packet receptions from the concerned neighbor, the second mechanism enables the node to predict the occurrence of link failure more accurately. Finally, with PRE, when a link failure is detected, all sources that have used the link in the past certain period are informed of the link failure. This mechanism reduces the latency involved in the route failure information delivery and consequently reduces the number of packet losses and also triggers early alternate route computations.

*Link Layer Enhancement*

Fu et al. [9] have discussed the interaction between TCP and 802.11 MAC. Their studies reveal two interesting results. First, given a specific network topology and flow pattern, there exists a TCP window size, say W*, at which TCP throughput is maximized since the best spatial reuse can be achieved; further increasing the window size will reduce throughput. However, the standard TCP protocol does not operate around W*, typically with an average window much larger than W*. As a result, TCP experiences throughput reduction due to reduced spatial reuse and increased packet loss. In the simulated scenarios, 4%

to 21% throughput reduction from maximum throughput is observed. Second, most packet drops experienced by TCP are not due to buffer overflow, but due to link-layer contention that are incurred by hidden terminals. They showed that contention drops exhibit a load-sensitive loss feature: as the injected TCP packets exceed W* and further increase, the link dropping probability becomes non-negligible and increases accordingly; after the injected TCP packets exceed another threshold W, the link dropping probability saturates and flattens out. It turns out that the link-layer dropping probability is not significant enough to make the average TCP window oscillate around W*, which subsequently leads to suboptimal TCP throughput.

Therefore, two link layer techniques were proposed in [9] to improve TCP efficiency: a *Link-RED* (*Random Early Detection*) algorithm to tune the wireless link's packet dropping probability and an adaptive link-layer pacing scheme to reduce the medium contention. The Link-RED algorithm attempts to maintain the optimum congestion window size at the TCP sender. At the link layer each node measures the average number of the retries for recent packet transmissions. Normally, when the TCP sender increases the congestion window size and injects more packets into the network, this average number will increase, as more packets will aggravate medium contention. The head-of-line packet is dropped from the buffer or marked as congested with a probability calculated based on this average number. Once it detects packet losses or the congestion flag in the ACKs, the TCP sender invokes the congestion control algorithm that could help maintain the congestion window size around the optimum value and hence improve TCP's throughput.

The goal of adaptive link-layer pacing is to alleviate the medium contention especially when the congestion window size exceeds the optimum value. It is enabled from within the Link-RED algorithm. When a node (which just sends a packet) notices its average number of retries is less than a predefined threshold, it calculates its backoff time as usual. Otherwise, it increases the backoff period by an interval equal to the transmission time of the previous data packet, and backs off accordingly.

*Neighborhood RED*

As described in the previous subsection on challenges, TCP exhibits serious unfairness in ad hoc networks as a result of the combination of MAC-inherent problems such as medium contention, the hidden terminal problem, and the exposed terminal problem. As these problems are likely to exist in nodes which are located in a neighborhood, Xu et al. [26] proposed a scheme named *neighborhood RED* (NRED) that seeks to improve TCP fairness from the point of view of a neighborhood. By definition, a node's neighborhood consists of the node itself and the nodes which can interfere with this node's signal. To make things simpler, a node's neighborhood considered in the scheme is comprised of the node itself and its one-hop and two-hop neighbors.

The key idea of NRED is that each node forms a distributed queue of a neighborhood based on the individual queues maintained at every node located in the node's neighborhood, and the RED scheme can be applied to the distributed queue to address the fairness issue, as it has proven to be effective, in wired networks, in improving fairness among TCP flows by controlling average queue size at routers.

The NRED scheme boils down to three algorithms, namely, *Neighborhood Congestion Detection (NCD), Neighborhood Congestion Notification (NCN), and Distributed Neighborhood Packet Drop (DNCP).* Instead of counting on each node actively advertising its own queue size information and then measuring the neighborhood queue size, which may cause a large amount of overhead or even aggravate congestion, NCD intelligently gets around the difficult task by monitoring channel utilization. Normally, channel utilization can serve as an indicator of the queue size, based on the observation that channel utilization around a node is likely to increase when the queues at its neighboring nodes build up. An early congestion is assumed to take place as the channel utilization exceeds a certain threshold. If congestion is detected, the node will calculate the packet dropping probability and send it in a NCN packet to its neighbors, provided certain conditions are met in order to avoid "overreaction". The neighbors, upon the reception of such notification, will drop some packets according to DNCP.

Simulation studies show that the NRED can improve TCP fairness to some extent in ad hoc networks. However, the price paid is that the aggregate throughput in the network is actually reduced, which shows there is still room for further improvement.

## IV. CONCLUSIONS

As the assumption made by TCP that any packet loss is due to network congestion is no longer valid in wireless networks, TCP performs poorly in such networks. In this chapter, we point out the major reasons for this performance degradation. In particular, factors such as error-prone wireless channels and handoffs result in the poor TCP performance over one-hop wireless networks, while, aside from these factors, other factors such as medium access contention, frequent route changes, and breakages are considered to lead to the poor TCP performance over multi-hop wireless networks. Compared with one-hop wireless networks, we can see it is more difficult to make TCP perform well in multi-hop wireless networks.

Finally, although some encouraging improvements have been reported by employing the proposed schemes, none of them can work well in all scenarios and meet all the challenges mentioned. Therefore, there is still much work to be done in the near future. To serve as guidance for future research, some critical issues regarding improving TCP performance and fairness are identified.

## REFERENCES

[1] A. Ahuja, S. Agarwal, J. P. Singh and R. Shorey, "Performance of TCP over different Routing Protocols in Mobile Ad-hoc Networks," IEEE Vehicular Technology Conference 2000, vol. 3, pp. 2315 - 2319, Tokyo.

[2] V. Anantharaman, S.-J. Park, K. Sundaresan, and R. Sivakumar, "TCP Performance over Mobile Ad-hoc Networks: A Quantitative Study," 'To appear in Wireless Communications and Mobile Computing Journal (WCMC), Special Issue on Performance Evaluation of Wireless Networks, 2003.

[3] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani and R. D. Gitlin, "AIRMAIL: a link-layer protocol for wireless networks," ACM Wireless Networks, Feb. 1995.

[4] H. Balakrishnan, V. Padmanabhan, S. Seshan and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," Proceedings of ACM SIGCOMM'96, Aug. 1996.

[5] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," Proc. 15th International Conf. On Distributed Computing systems (ICDCS), May 1995.

[6] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback-based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks," IEEE Personal communications, 8 (1):34-39, February 2001.

[7] K. Chen, Y. Xue, K. Nahrstedt, "On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks," IEEE ICC'03, Anchorage, Alaska, May, 2003.

[8] T. D. Dyer and R. V. Boppana, "A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks," ACM Mobihoc, October 2001.

[9] Z. Fu, P.Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss", IEEE INFOCOM'03, San Francisco , March 2003.

[10] Z. Fu, X. Meng, and S. Lu, "How Bad TCP can Perform in Mobile Ad-Hoc Networks," IEEE Symposium on Computers and Communications, Italy, July 2002

[11] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang, "TCP over Wireless Multihop Protocols: Simulation and Experiments," Proceedings of IEEE ICC'99, Vancouver, Canada, Jun. 1999.

[12] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multihop Networks," Proceedings of IEEE WMCSA'99, New Orleans, LA, Feb. 1999.

[13] G. Holland and N. H. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," MOBICOM'99, Seattle, August 1999.\

[14] V. Jacobson and M. Karels, "Congestion avoidance and control," Proceedings of ACM SIGCOMM'88, Aug. 1988.

[15] D. B. Johnson, D A. Maltz, Y. Hu, "The dynamic souce routing protocol for mobile ad hoc networks," IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-08.txt, 2003.

[16] D-K. Kim, C.-K. Toh, and Yanghee Choi, "TCP-BuS: Improving TCP Performance over Wireless Ad Hoc Networks," IEEE Comsoc Journal On Communications And Networks (JCN), Vol. 3, No. 2, 2001.

[17] J. Li, C. Blake, D. S. J. De Couto, H. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," MobiCom'01, Rome, Italy, July 2001.

[18] J. Liu, S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks," IEEE JSAC. Vol.19 No.7, July 2001.

[19] V. D. Park and M. S. Corson. "A Highly Adaptive Distributed Routing Algorithm for MobileWireless Networks". *Proceedings of IEEE INFOCOM*, Kobe, Japan, April 1997.

[20] C.E Perkins et al., "Ad hoc on demand distance vector routing." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt, 2003.

[21] E. Royer, S. J. Lee and C. Perkins, "The Effects of MAC Protocols on Ad Hoc Network Communication", IEEE WCNC, Chicago, IL, September 2000

[22] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan, "WTCP: a reliable transport protocol for wireless wide-area networks," MobiCom'99, Aug. 1999.

[23] K. Tang and M. Gerla, "Fair Sharing of MAC under TCP in Wireless Ad hoc Networks," Proceedings of IEEE MMT'99, Venice, Italy, Oct. 1999.

[24] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response," MobiHoc'02, pp. 217-225, Lausanne, Switzerland, Jun 2002.

[25] H. Wu, et al., "Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement," INFOCOM 2002.

[26] K. Xu, M. Gerla, L. Qi and Y. Shu, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED," MobiCom'03, Sep. 2003.

[27] S. Xu and T. Saadawi, “Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad hoc Networks?” IEEE Communications Magazine, June 2001.