A Survey on Improving TCP Performance over Wireless Networks

Xiang Chen, Hongqiang Zhai, Jianfeng Wang, and Yuguang Fang
Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL 32611
E-mail: {xchen@ecel, zhai@ecel, jfwang@, fang@.ece}.ufl.edu

Contents

1 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.

Transmission control protocol (TCP) is a transport layer protocol which provides reliable end-to-end data delivery between end hosts in traditional wired network environment. In TCP, reliability is achieved by retransmitting lost packets. Thus, each TCP sender maintains a running average of the estimated round trip delay and the average deviation derived from it. Packets will be retransmitted if the sender receives no acknowledgment(ACK) within a certain timeout interval (e.g., the sum of smoothed round trip delay and four times the average deviation) or receives duplicate acknowledgments. Due to the inherent reliability of wired networks, there is an implicit assumption made by TCP that any packet loss is due to congestion. To reduce congestion, TCP will invoke its congestion control mechanisms whenever any packet loss is detected. Since TCP is well tuned, it has become the de facto transport protocol in the Internet that supports many applications such as web access, file transfer and email. Due to its wide use in the Internet, it is desirable that TCP remains in use to provide reliable data transfer services for communications within wireless networks and for those across wireless networks and the wired Internet. It is thus crucial that TCP performs well over all kinds of wireless networks in order for the wired Internet to extend to the wireless world.

Unfortunately, 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 [4, 5, 7, 18-21]. 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.

This chapter is organized as follows. We present in Section 2 a brief overview of TCP congestion control mechanisms and some current performance enhancement techniques. As the challenges TCP is facing differ in one-hop and multi-hop wireless networks and so do the solutions, it is suitable to separate them into two sections. Section 3 starts by identifying the challenges imposed on the standard TCP in one-hop wireless networks, followed by the classification of some existing solutions according to their design philosophy. Among the solutions, there are four large categories. The first class of schemes attempts to improve TCP performance by splitting a TCP connection into two at the base station or access point. Relying on an intelligent proxy located at the base station enforcing tasks such as local retransmission or ACK suppression/regulation, the second class eliminates the negative effects of wireless links on TCP. The approaches in the third class aim at hiding the characteristics of wireless links from TCP by providing a reliable link layer. The last category resolves the problems by slightly modifying TCP at the end systems, e.g., selective acknowledgment enabling or fast retransmission. In each class, the solutions are discussed in certain details. The structure of Section 4 is similar to that of Section 3, except that TCP performance over MANETs is the focus. Similarly, current solutions can also be grouped into three camps, according to their design philosophy. The first camp incorporates network feedback information into their designs to modify TCP's response to non-congestion-related packet losses while the second camp attempts to do so without explicit feedback. Unlike the previous two, the third one starts by tuning the lower layers in order for TCP to operate normally, while leaving TCP intact. With the understanding that current solutions fail to improve on some critical issues such as fairness, Section 5 gives some suggestions on future research issues. Finally, concluding remarks are given in Section 6.

2 Overview of TCP

Before we dive into the detailed discussion of questions such as *why TCP performs poorly in wireless networks*, *how TCP performance can be improved*, it is necessary to prepare the reader by presenting an overview of not only the basic functionality of TCP but also the state-of-the-art in 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.

2.1 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 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),

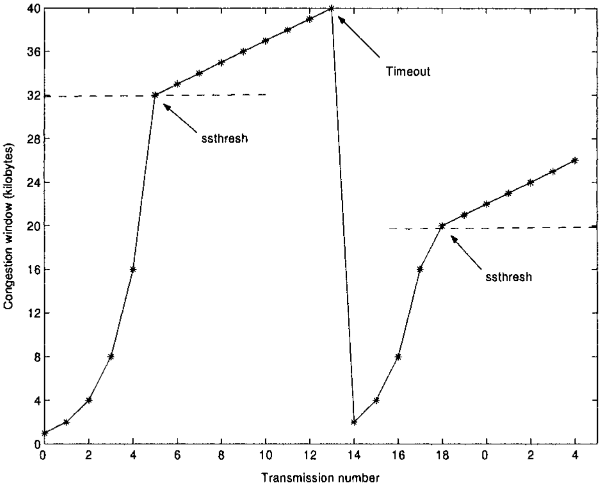

Figure 1: TCP congestion window dynamics ([44])

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, [25]). 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.

2.2 State-of-the-Art in Standard TCP

Most of the progress made in TCP is centered on error recovery and congestion control. Representative innovations include fast transmissions and fast recovery [42], selective acknowledgements [31], random early detection (RED, [17]) in routers, and explicit congestion notification (ECN, [39]). Notice that depending on what features are included, there are several TCP flavors, including TCP Tahoe, TCP Reno, TCP New Reno, etc. Among them, TCP Reno is by far most widely deployed. Next, we briefly describe these innovations in the following.

2.2.1 Fast Retransmission and Fast Recovery

As noted earlier, a packet can be assumed lost if three duplicate ACKs are received. In this case, TCP performs a fast retransmission of the packet. This mechanism allows TCP to avoid a lengthy timeout during which no data is transferred. At the same time, *ssthresh* is set to one half of the current congestion window, i.e., *cwnd*, and *cwnd* is set to *ssthresh* plus three segments. If the ACK is received approximately one round trip after the missing segment is retransmitted, fast recovery is entered. That is, instead of setting *cwnd* to one segment and starting with slow start, TCP sets *cwnd* to *ssthresh*, and then steps into congestion avoidance phase. However, only one packet loss can be recovered during fast retransmission and fast recovery. Additional packet losses in the same window may require that the RTO expire before retransmission.

2.2.2 Selective Acknowledgment

Owing to the fact that fast retransmission and fast recovery can only handle one packet loss from one window of data, TCP may experience poor performance when multiple packets are lost in one window. To overcome this limitation, recently the selective acknowledgement option (SACK) is suggested as an addition to the standard TCP implementation.

The SACK extension adopts two TCP options. One is an enabling option, which may be sent to indicate that the SACK option can be used upon connection establishment. The other is the SACK option itself, which may be sent by TCP receiver over an established connection if SACK option is enabled through sending the first option.

The SACK option contains up to four (or three, if SACK is used in conjunction with the Timestamp option used for RTTM [24]) SACK blocks, which specifies contiguous blocks of the received data. Each SACK block consists of two sequence numbers which delimit the range of data the receiver has received and queued. A receiver can add the SACK option to ACKs it sends back to a SACK-enabled sender. In the event of multiple losses within a window, the sender can infer which packets have been lost and should be retransmitted using the information provided in the SACK blocks. A SACK-enabled sender can retransmit multiple lost packets in one RTT instead of detecting only one lost packet in each RTT.

2.2.3 Random Early Detection

Random Early Detection (RED) is a router-based congestion control mechanism that seeks to detect incipient congestion and notify some TCP senders of congestion by controlling the average queue size at the router. To notify the TCP senders of congestion, the router may mark or drop packets, depending on whether the senders are cooperative. As a response, the senders should reduce their transmission rate. This is done in two algorithms. The first algorithm is to compute the average queue size by using exponential weighted moving average. If we denote by avg and q the average queue size and the current queue size, respectively, then $avg = (1 - wq) \times avg + wq \times q$, where wq is the queue weight. The other algorithm is to compute the packet-marking or packet-dropping probability p_a. If avg falls in between min_{th} and max_{th}, the packet marking probability $p_b = max_p(avg - min_{th})/(max_{th} - min_{th})$ and the final marking probability $p_a = p_b/(1 - count * p_b)$, where max_p and $count$ are design parameters, respectively, denoting the maximum value for p_b and the number of packets having arrived since last packet marking or dropping. If avg exceeds max_{th}, $p_a = 1$, which means that the router marks or drops each packet that arrives. Through control over the average queue size prior to queue overflow, RED succeeds in preventing heavy network congestion and global synchronization as well as improving fairness. Notice that numerous variants of RED have been proposed to improve various performance of the original RED [16, 29, 33, and 34].

2.2.4 Explicit Congestion Notification

Most of current Internet routers employ traditional "drop-tail" queue management. In other words, the routers drop packets only when the queue overflows, which could lead to the undesirable global synchronization problem as well as heavy network congestion. Recently, active queue management (AQM) mechanisms have been proposed since they can detect congestion before the queue overflows at the routers and inform TCP senders of the congestion, thereby avoiding some of these problems caused by the "drop-tail" policy. In the absence of Explicit Congestion Notification (ECN), however, the only choice that is available to AQM for indicating congestion to end systems is to drop packets at the routers. With ECN, AQM mechanisms have an alternative to allow routers to notify end systems of congestion in the network.

ECN requires some changes to the header of both IP and TCP. In the IP header, an ECN field with two bits is used. By setting this field to specific bits, the router can send an indication of congestion to end systems. For TCP, two new flags in the Reserve field of the TCP header are specified. By manipulating these two flags, the TCP sender and the TCP receiver can enable ECN via negotiation during connection setup; the receiver can inform the sender if it receives congestion indications from intermediate routers; and the sender can inform the receiver that it has invoked congestion control mechanisms [39].

3 TCP in One-Hop Wireless Networks

In this section, we focus on TCP performance in one-hop wireless networks, which typically include wireless LAN and wireless cellular networks. We first summarize some challenges adversely affecting TCP performance. Then, some representative schemes proposed to improve TCP performance are described. Notice that in this chapter we focus on how to improve TCP performance, so some schemes such as WTCP [41], which attempts to propose a totally different transport layer protocol, are not presented here since it is not an improvement scheme based on TCP.

3.1 Challenges

Compared with wired networks, one-hop wireless networks have some inherent adverse characteristics that will significantly deteriorate TCP performance if no action is taken. In essence, these characteristics include bursty channels errors, mobility and communication asymmetry.

3.1.1 Channel Errors

In wireless channels, relatively high bit error rate because of multipath fading and shadowing may corrupt packets in transmission, leading to the losses of TCP data segments or ACKs. If it cannot receive the ACK within the retransmission timeout, the TCP sender immediately reduces its congestion window to one segment, exponentially backs off its RTO and retransmits the lost packets. Intermittent channel errors may thus cause the congestion window size at the sender to remain small, thereby resulting in low TCP throughput.

3.1.2 Mobility

Cellular networks are characterized by handoffs due to user mobility. Normally, handoffs may cause temporary disconnections, resulting in packet losses and delay. TCP will suffer a lot if it treats such losses as congestion and invokes unnecessary congestion control mechanisms. The handoffs are expected to be more frequent in next generation cellular networks as the micro-cellular structure is adopted to accommodate an increasing number of users. Thing could be worse if TCP cannot handle handoffs gracefully. Similar problems may occur in wireless LAN, as mobile users will also encounter communication interruptions if they move to the edge of the transmission range of the access point.

3.1.3 Asymmetry

In one-hop wireless networks, the wireless link between a base station and a mobile terminal in nature is asymmetric. Compared with the base station, the mobile terminal has limited power, processing capability, and buffer space. Another asymmetry stems from the vastly different characteristics of wired links and wireless links. The former is reliable and has large bandwidth while the latter is error-prone and has limited and highly variable bandwidth. For example, the bandwidth of a typical Ethernet is 10Mbps (100Mbps or even higher for fast Ethernet) while the highest bandwidth for 3G networks is only about 2Mbps. Therefore, the wireless link is very likely to become the bottleneck of TCP connections.

3.2 Current Solutions

The quest to overcome the deficiency of TCP over wireless links has been courting extensive efforts. Among the various solutions proposed to improve TCP performance, there are four major categories: *split-connection solutions*, *proxy-based solutions*, *link-layer solutions*, and *end-to-end solutions*. The split-connection solutions attempt to improve TCP performance by splitting a TCP connection into two at the base station so that the TCP connection between the base station and the mobile host can be specially

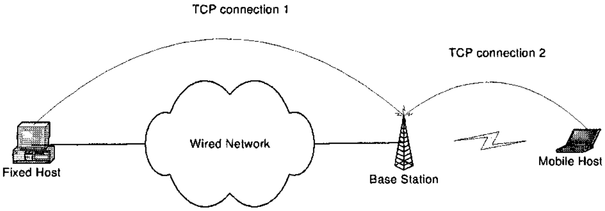

Figure 2: I-TCP, splitting a TCP connection into two connections

tuned for the wireless links. Realizing the base station is a critical point, approaches based on proxy put an implicit or explicit intelligent agent at the base station, detecting packet losses over wireless links and taking corresponding actions (such as duplicate ACK suppression and/or local retransmission) to ensure the TCP sender responds correctly. For the third category, a reliable link layer is built by adopting some link error recovery mechanisms, seeking to hide link errors from the TCP sender. Unlike the previous three classes, the end-to-end approaches enhance TCP by using SACK to quickly recover from multiple packet losses or by predicting incoming handoffs to avoid unnecessary congestion control invocation. Next, some representative schemes in each category are presented.

3.2.1 Split-Connection Solutions

Indirect TCP: Indirect-TCP (I-TCP) [7] protocol proposed by Bakre and Badrinath suggests that any TCP connection from a mobile host (MH) to a machine on the fixed network (FH) should be split into two separate connections: one between the MH and its base station (BS) over the wireless medium and the other between the BS and the FH over the fixed network, as shown in the Fig. 2. A packet sent to MH is first received by BS, it then sends an acknowledgment to FH and then the packet is forwarded to MH. If MH moves to a different cell while communicating with an FH, the whole connection information maintained at the current BS is transferred to the new BS and the new BS takes over thereafter. The FH is unaware of this indirection and is not affected when this switch occurs. Also, since the end-to-end connection is split, the TCP connection over the wireless link can use some wireless-link-aware TCP variation, which may be tailored to handle wireless channel errors and handoff disruption.

From the above description, we see that I-TCP separates the congestion control functionality on the wireless link from that on the fixed network, which enables the two kinds of links to identify different reasons for packet losses and then take corresponding actions. In addition, since the TCP connection is broken into two, it is possible for a mobile host to use some lightweight transport protocol instead of a full TCP/IP suite to communicate with the base station and access the fixed network through the base station. This feature is desirable since a mobile host, as pointed out earlier, has limited battery and processing power. The downside of this scheme, however, is the following. First, I-TCP violates the end-to-end semantics of TCP acknowledgments, as both the wired part and the wireless part of a connection have their own acknowledgments. Second, control overhead is considerable as the base station needs to maintain a significant amount of state for each TCP connection and all the state information needs to be transferred to the new base station in the event of a handoff, which could result in a long delay.

M-TCP: M-TCP [9] is another split-connection approach that breaks up a TCP connection between a FH and a MH into two parts: one between the FH and the BS, and the other between the BS and the MH. What makes it different from I-TCP is that it manages to preserve TCP end-to-end semantics.

M-TCP is assumed to operate upon the underlying three-level architecture shown in Fig. 3. A mobile host (MH) communicates with the BS in the cell. Several BSs are controlled by a supervisor host (SH), which, serving a gateway, is connected to the wired network. The authors opt for this architecture for two reasons. The first is that the functionalities at a BS can be transferred to SH, which may reduce the cost of the network as one SH is in charge of several BSs; the other is that the number of handoffs is greatly reduced since a MH roaming from one cell to another need not perform handoffs as long as the two cells are controlled by the same SH.

Another important assumption made by M-TCP is that a relatively reliable link layer is operating underneath M-TCP to recover losses such that the bit error rate over wireless links is low. The implication of this assumption is that TCP performance degradation is mainly due to frequent disconnections caused by handoffs.

M-TCP operates as follows. Assume that the MH has acknowledged bytes up to sequence number x, the SH sends an ACK for bytes up to $x - 1$ to the TCP sender. Note that this is different from I-TCP in that the SH only sends ACKs to the sender when it receives ACKs from the MH. If the SH does not receive ACKs beyond x for some time, the SH will assume this is due to temporary wireless link outage. Therefore, it sends an ACK for the last byte x with a zero window size. Upon receiving this ACK,

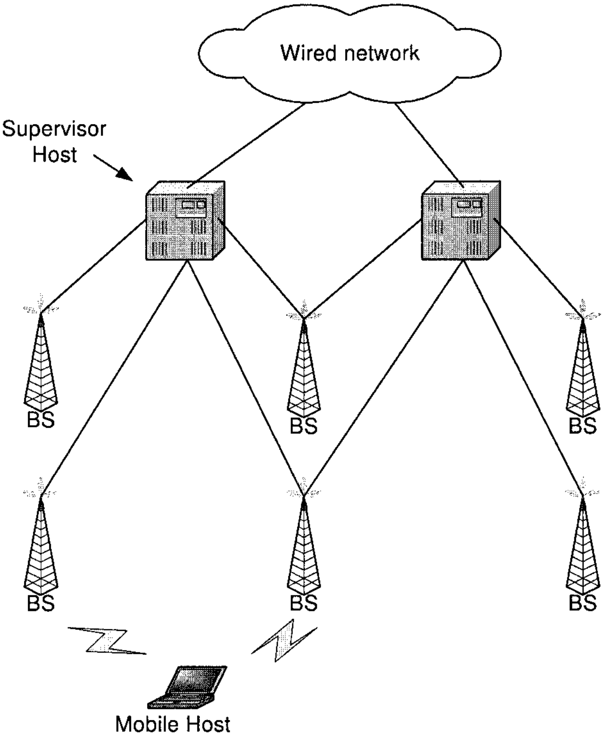

Figure 3: Three-level architecture underlving M-TCP

the sender will enter into *persist* mode, freezing all its transmission states such as RTO and congestion window. When the wireless link is regained, the MH will notify the SH by sending a greeting packet. The SH, in turn, informs the sender of this reconnection, allowing the sender to resume its transmission from the frozen state. Through this way, the adverse efforts of disconnections on TCP performance are gracefully eliminated since no congestion control is invoked.

Some comments are in order. First and foremost, while maintaining end-to-end TCP semantics, M-TCP works well under the wireless environment where frequent disconnections between the MH and the BS are common. Second, during handoffs from one domain of SH to the domain of another SH, little overhead is incurred since compared to I-TCP, a small amount of state is transferred from the old SH to the new SH. However, in order to achieve the expected performance improvement, it relies largely on its underlying link layer to hide the effects of high bit error rate.

3.2.2 Proxy-Based Solutions

SNOOP: Balakrishnan et al. [6] sought to improve TCP performance by modifying the network-layer software at a BS while preserving end-to-end TCP semantics. *Snoop* protocol gets its name because it adds a snooping module to network layer, which monitors every packet that passes a BS in either direction. In the following, we describe how snoop module deals with packet losses in both directions.

If TCP packets are sent from a FH to a MH, the snoop module caches each packet that has not yet been acknowledged by the MH. Meanwhile, the snoop module also keeps track of all the acknowledgments sent from the mobile host. The snoop module determines that a packet loss occurs by detecting if either it receives a duplicate acknowledgment or its local timer times out. In this case, the lost packet is retransmitted if it has been cached. The duplicate acknowledgments, if any, are suppressed. Through this way, unnecessary congestion control mechanism invocations are avoided since packet losses due to wireless channel errors are hidden from the FH.

In the case that packets are transmitted from an MH to an FH, since the MH cannot tell whether a packet loss is due to errors on the wireless link or due to congestion elsewhere in the network, TCP SACK option is used. At the BS, when the snoop module notices a gap in the inbound sequence numbers of the packets sent from the MH, selective acknowledgements are sent to the MH. Upon receiving such SACKs, SACK-enabled MH will retransmit the lost packets for local loss recovery.

SNOOP is also designed to handle handoffs. Several BSs near a MH will form a multicast group and buffer some latest packets sent from the FH. Prior to a handoff, the MH will send control messages to determine that a BS with strongest signal should be the primary one, i.e., the one forwarding packets to the MH, and that all other BSs just buffer packets. Therefore, both handoff latency and packet losses are reduced.

The major merit of this approach is that, it improves TCP performance through performing local retransmissions across the wireless link without affecting end-to-end TCP semantics. On the other hand, although TCP performance during handoffs may be improved, considerable overhead is incurred for maintaining the multicast BS group and state transfer from one BS to another. Finally, it is worth noting that special care needs to be taken to handle the interaction of the snoop module retransmission and TCP end-to-end retransmission because SNOOP is similar to link-level retransmission approaches over the wireless links.

Ack Regulator: Since link layer enhancement schemes are shown to successfully improve TCP performance over wireless link, they have been adopted in 3G wireless networks. For example, reliable link-layer protocols such as RLP [49] and RLC [47] are respectively used in 3G1X [48] and UMTS [46]. However, as pointed out in [11], these link layer protocols also introduce increased delay and rate variability, which may cause bursty ACK arrivals (called ACK compression [56]) and consequently degrade TCP throughput. This effect becomes more pronounced as some channel state based scheduling schemes [8] is used in 3G wireless networks as well.

To reduce such negative effects, Chan and Ramjee ([11]) proposed a network-based solution called *Ack Regulator*, which is implemented at the radio network controller (RNC) to regulate the flow of ACKs sent from the mobile host to the TCP sender. Notice that since most applications like web browsing mainly use the downlink, this solution is designed for TCP connections toward the mobile hosts. The key idea is that, the RNC should control the number of ACKs sent back to the sender each time a data packet is transmitted to the mobile host or an ACK arrives from the mobile host, such that there is at most one packet loss due to buffer overflow in one window of transmitted packets. In this way, the TCP sender operates mainly in the congestion avoidance phase. In this scheme, the RNC maintains a data queue for each TCP flow from the sender to the mobile host and an ACK queue from the mobile host to the sender. By monitoring the current available buffer space and estimating the number of future incoming data packets, the RNC decides how many ACKs it should send to the sender each time. Significant performance improvement has been reported, which, nevertheless, is achieved at the expense of increased complexity and buffer space at the RNC.

Advertised Window Control: While a great deal of effort has been made on the IEEE 802.11 MAC, little research work has been focused on the interaction of TCP with WLANs. However, to fully understand TCP behaviors over WLAN is very important, as TCP is the de facto transport layer protocol for most applications over WLANs.

In [38], the work by Pilosof et al. has shed some light in better understanding TCP fairness over WLANs. Through analysis and simulation, it is discovered that the upstream (from the mobile host to the base station) and downstream (from the base station to the mobile host) TCP flows do not fairly share the wireless medium, with a throughput ratio between them as high as ten times, in favor of upstream flows. They discovered that this ratio is sensitive to the buffer size at the base station. In particular, TCP unfairness may fall into four different regions as the buffer size is varying. Part of the reason is that given the TCP receiver window size, the downstream TCP window size fails to reach the receiver window size if some data packets are lost due to insufficient buffer, while upstream TCP window size can reach the receive window size because it can tolerate some ACK losses.

Thus, they proposed to modify the receiver's advertised window field in the ACKs when they pass through the base station. More precisely, given that there are n TCP flows in the WLANs and the buffer size at the base station is B, the advertised window size will be set to the minimum of the original advertised window size and $\lfloor B/n \rfloor$. Simulations and experiments show that the throughput ratio between upstream and downstream TCP flows is almost 1 after adopting this change.

3.2.3 Link-layer Solutions

AIRMAIL: Since TCP performance degradation is partly due to the high bit error rate of the wireless link, it is intuitive to shield TCP from such errors. With a reliable link-layer protocol in place, unnecessary TCP congestion control invocation due to channel errors can be avoided, and hence TCP performance is improved. Based on this idea, a reliable link-layer protocol named AIRMAIL (AsymmetrIc Reliable Mobile Access In Link-layer) was proposed in [4]. In AIRMAIL, two well-known link error recovery techniques, i.e., forward error correction (FEC) and automatic repeat request (ARQ) are employed. Moreover, in order to accommodate the asymmetry lying between the two ends of a wireless link, i.e., the BS and the MH, AIRMAIL purposely devises some asymmetric ARQ error control and window-based flow control techniques as shown in the following:

- Timers are always at the BS regardless of whether it is transmitting or receiving. Thus all timer-related operations are conducted in the BS.
- The base station receiver sends its status to a mobile transmitter periodically. However, this is not the case for the mobile receiver to send its status to the base station transmitter. Rather, the mobile receiver sends status messages based on an event-driven mechanism. The difference in the mechanisms of sending status messages is justified by the power constraint at the mobile host.

In addition to ARQ, three levels of FEC, namely bit-level FEC, byte-level FEC, and packet-level FEC, have been employed to provide increased error correction capability under different mobile environment.

Several comments on AIRMAIL are in order. First, since AIRMAIL only involves changes at the link layer, no modifications need to be made to TCP. Obviously, it fits in well with the layered structure of network protocols. Second, when designing ARQ techniques, AIRMAIL takes into account the asymmetry between the BS and the MH, a desirable feature which may relieve the requirement of computing power on the mobile host and prolong the battery life of the mobile host as well. Third, the drawback of AIRMAIL is that it cannot account for temporary disconnections due to handoff. Thus, even though it succeeds in reducing bit error rate over wireless links, it can do little to prevent TCP from timing out when an acknowledgment is not received on time because of long disconnections. In fact, this observation might apply to various link layer approaches. Finally, the interaction between link-layer retransmissions and end-to-end retransmissions can be complicated, as shown in [14]. It showed that link-layer retransmission protocols only improve TCP performance when the packet error rate exceeds a certain threshold. Further study on the interaction is needed in order to improve TCP performance with the aid of the link-layer enhancement.

TULIP: TULIP (Transport Unaware Link Improvement Protocol) is a TCP-unaware link layer protocol that works upon the MAC layer [36]. Because TULIP is targeted for half-duplex wireless links, to avoid collision between two opposite data streams, it only passes one packet at a time to the MAC layer. The procedure is described as follows. After receiving a TCP packet from the upper layer, TULIP passes it to the MAC layer. When starting to transmit the packet, the MAC layer notifies TULIP by sending a signal *TRANS*. Upon reception of *TRANS*, TULIP starts a timer Δt_1, which is estimated as the time duration between the beginning of data packet transmission and the end of the reception of a link-layer ACK (or a link-layer ACK piggybacked with a data packet). In the case that Δt_1 is underestimated because of packet length variations, the MAC layer will inform TULIP by sending another signal *WAIT*, specifying the additional time Δt_2. Readers are referred to [36] for details on how to set Δt_2 as it involves the specific MAC layer mechanism. To locally recover lost or corrupted packets due to channel errors, a link-layer selective acknowledgment mechanism is used to retransmit packets, which is assigned high priority compared to normal data packets in order for fast recovery. Moreover, to save bandwidth over the wireless link, a mechanism called MAC acceleration is introduced to piggyback a TCP ACK with the TULIP ACK. Through simulation it is shown that TULIP achieves a bit better performance compared to SNOOP in the environment where errors are exponentially distributed over the wireless channel.

3.2.4 End-to-end Solutions

Fast Retransmission: Fast retransmission is perhaps the simplest end-to-end scheme to improve TCP performance. Based on the observation that TCP encounters unacceptably long pauses in communication during handoffs which cause increased delays and packet losses, fast retransmission was proposed by Caceres and Iftode to overcome this problem [10]. The MH will send duplicate ACKs to the TCP sender immediately after the handoff process is completed. In this way, the TCP sender can begin retransmission without waiting for the timeout, hence preventing serious throughput drop.

Selective Acknowledgement: As described earlier, the TCP selective acknowledgment mechanism can allow a SACK-enabled sender to retransmit in one RTT multiple lost packets in one transmission window and hence avoid continuous timeouts. However, this mechanism does not distinguish the reasons for packet losses and still assumes all losses are caused by congestion. Consequently, TCP congestion control procedures are inappropriately called for, which throttles the sender's transmission rate. As shown in [5], SACK is useful over the error-prone wireless link where losses occur in bursts.

Freeze-TCP: It is observed that most current TCP schemes require base stations to monitor the TCP traffic and actively participate in flow control in order to enhance performance. However, such schemes might be undesirable or even useless for several reasons. First, to be compatible with currently existing infrastructure, it is ideal that no modification should be made to intermediate nodes, since such nodes may belong to other organizations and hence are unavailable for modification. Second, as network security is becoming increasingly important, end-to-end traffic is likely to be encrypted and hence inaccessible to intermediate nodes. As a result, some schemes such as SNOOP, I-TCP or M-TCP can no longer work in such scenarios since they all require the base station to access the traffic before taking actions. Finally, overly relying on mediation at the intermediate nodes may cause a significant amount of control overhead, creating network bottlenecks under heavy traffic load.

To overcome these deficiencies, the author in [22] proposed Freeze-TCP, a true end-to-end TCP enhancement scheme. The key idea of this scheme is to exonerate the base station from intervening in the end-to-end TCP connections. By constantly observing its received signal strength, a mobile host can predict a temporary disconnection due to handoffs or fading. Once such an event is predicted, the mobile host sends an ACK to the TCP sender with a zero advertised window size. Upon reception of such an ACK, the sender enters a persist mode. That is, the sender freezes all retransmission timers and sends *zero window probes* (ZWP) until the mobile host advertises a non-zero receiving window size. Since ZWPs are sent out with exponentially backoff, it is possible that the sender remains idle even the mobile host has recovered from the disconnection. To tackle this problem, the same technique as in [10] is employed. Namely, as soon as the mobile host knows that it has reconnected, it will send three duplicate ACKs to the sender, forcing the sender to start fast retransmission. The main advantage of this scheme is that it improves TCP performance without any modification to the intermediate nodes. However, it could be easily seen that the actual performance depends largely on the accuracy with which the mobile host predicts an impending disconnection.

4 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, infrastructureless, self-organizing multi-hop networks, and lacking centralized network management. Next, we present the main problems in ad hoc networks, followed by recent solutions.

4.1 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 and collision, c) the mobility, d) the multi-path routing, and e) congestion, whose adverse impacts on TCP is elaborated next.

4.1.1 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.

4.1.2 Medium Contention and Collision

Contention-based medium access control (MAC) schemes, such as the IEEE 802.11 MAC protocol [9], have been widely studied and incorporated into many wireless testbeds 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 inefficiency. The binary exponential backoff scheme always favors the latest successful transmitter and results in unfairness. These problems could be more harmful in multi-hop ad hoc networks than in Wireless LAN as ad hoc networks are characterized by multi-hop connectivity.

MAC protocols have been shown to significantly affect TCP performance [20, 21, 40, 45, 51, and 54]. When TCP runs over 802.11 MAC, as [54] pointed out, the instability problem becomes very serious. It is shown that collisions and the exposed terminal problem are two major reasons for preventing one node from reaching the other when the two nodes 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. Serious oscillation in TCP throughput will thus be observed. Moreover, the random backoff scheme used in the MAC layer exacerbates this [20]. Since large data packet sizes and back-to-back packet transmissions both decrease the chance of the intermediate node to obtain the channel, the node has to back off a random period of time and try again. After several failed tries, a route failure is reported.

TCP may also encounter serious unfairness problems [20, 40, 45, and 54] for the reasons stated below:

- Topology causes unfairness because of unequal channel access opportunity for different nodes. As shown in Fig. 1, 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 degree of competitions. There are two 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 experiences 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.

- The 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 high throughput will become higher and low one lower.

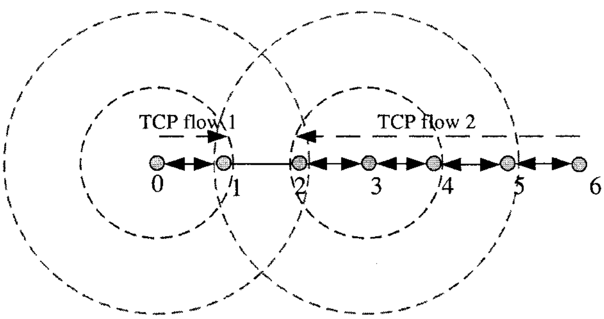

Figure 4: Node interference in a chain topology

4.1.3 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, 15, and 27]. 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 due to losses will further shrink. It could be even worse when the sender and the receiver of a TCP connection fall 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 considering mobility, channel error, and shared media-channel contention [19]. 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 drop ranges from almost 0% in a static case to 100% in a highly mobile case (when moving speed is $20m/s$). 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).

4.1.4 Multi-path Routing

Routes are short-lived due to frequent link breakages. To reduce delay due to route re-computation, some routing protocols such as TORA [35] 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 congestion. The receiver will thus generate duplicate ACKs that cause the sender to invoke congestion control algorithms like fast retransmission (upon reception of 3 duplicate ACKs).

4.1.5 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 if 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 degrades TCP performance. As a matter of fact, [28] showed the capacity of wireless ad hoc networks decreases as traffic and/or competing nodes arise.

4.2 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.

4.2.1 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) [12] 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.

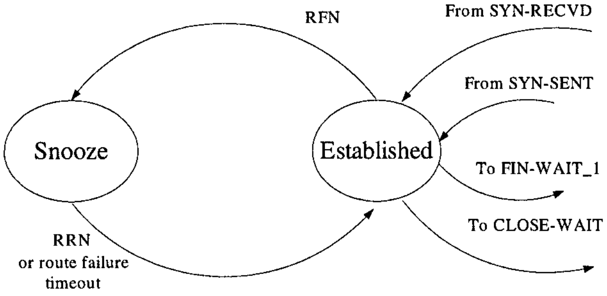

Figure 5: The TCP-F state machine [12]

Then it resumes the transmission from the frozen state. The state machine of TCP-F is shown in Fig. 5.

TCP-ELFN: Holland and Vaidya proposed another feedback-based technique, the *Explicit Link Failure Notification* (ELFN) [23, 32]. 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 [26] 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 into 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 routes 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 problems 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) [30] 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 or 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 into 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, realizing 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.

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 de-

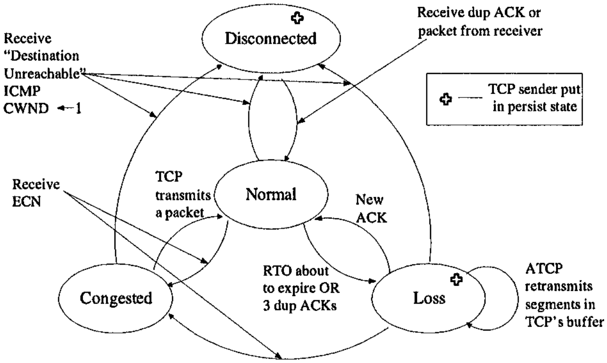
Disconnected
Receive dup ACK or packet from receiver
Receive "Destination Unreachable" ICMP CWND ←1
TCP sender put in persist state
TCP transmits a packet
Normal
New ACK
Receive ECN
RTO about to expire OR 3 dup ACKs
Congested
Loss
ATCP retransmits segments in TCP's buffer

Figure 6: State transition diagram for ATCP at the sender [30]

tecting 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.

4.2.2 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. [13] 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 bw_{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/bw_{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 $BDP <= RTHC \times S$. 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 [50] 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 [15], 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 [37] 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.

4.2.3 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 [3]. 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. [18] 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 [18] 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. [53] 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 comprises 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.

5 Future Research Directions

At this point, after we discussed the challenges and visited some representative solutions, it is well recognized that in order for TCP to deliver a comparable performance in wireless networks to that in wired networks, quite a few critical issues need to be addressed. Note that compared with its one-hop counterparts, ad hoc networks require more efforts to handle as things are much more complicated. In this section we discuss some of these open issues for which searching for a better solution demands special efforts. It is worthy noting that we do not mean to list all. Rather, we concentrate on those that we believe are most important.

5.1 TCP Fairness

TCP unfairness becomes pronounced in wireless LANs [38]. In mobile ad hoc networks, the unfairness problem is more severe. It is shown that in a mobile ad hoc network with multiple flows, the throughput can be significantly different among competing flows. This phenomenon is particularly evident when comparing flows of short paths to those of long paths [20]. Compared with the considerable effort paid to improve TCP end-to-end throughput, fairness is a critical issue that deserves more attention. In fact, this insufficiency can be seen from the number of proposed scheme targeted for fairness: among all the schemes we present in this chapter, only *Advertised Window Control* and *Neighborhood RED* address this issue, although a few schemes have touched upon fairness. Since bandwidth over wireless links is very limited bandwidth compared with that over wired links, it is crucial for every flow to fairly share the bandwidth in wireless networks. Therefore, more mature approaches are highly expected.

5.2 Interactions among Different Layers

Layered network architecture brings a myriad of advantages. At the same time, it requires a close look at the interactions among different layers when designing a good scheme. Currently, many solutions are focused on one specific layer, attempting to isolate the problem and solve it. It is true that TCP might perform better with a highly effective and efficient link layer or routing layer, e.g., an MAC protocol which can quickly resolve medium contentions, or a mobility-aware routing protocol which can gracefully handling route changes. However, this approach may be problematic or even counterproductive, as suggested in [14]. Furthermore, as many factors such as bursty channel errors, medium access contention, and route breakage are all contributing to TCP throughput deterioration in mobile ad hoc networks, a unified solution is justified which takes into account the interaction among different layer. We thus argue that a cross-layer approach seems more desirable and promising.

5.3 Compatibility with the Wired Internet

For the purpose of internetworking with the wired Internet as required in future pervasive mobile computing, whatever TCP is designed for ad hoc networks should be fully compatible with the Internet. This quest for compatibility translates into two requirements for future research. First, TCP's end-to-end semantics must be maintained. Second, TCP performance should be considered when TCP connections span both the wired networks and mobile ad hoc networks.

6 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.

This chapter presents the state-of-the-art in recent efforts to improve TCP performance. Given the reasons, almost all the proposed schemes attempt to achieve better TCP performance with either of the two ideas: TCP should be capable of distinguishing non-congestion-related packet losses from congestion caused packet losses such that corresponding actions can be taken to deal with the losses; or non-congestion-related losses should be reduced such that TCP can work normally without any modifications. Interestingly enough, there seems little study attempting to combine these two ideas.

Again, we choose to present the proposed schemes after separating those for one-hop wireless networks from those for multi-hop wireless networks for the purpose of clarity. In the realm of one-hop wireless networks, there are four groups of schemes, i.e., split-connection approaches, proxy-based approaches, link-layer enhancement approaches, and end-to-end approaches. According to [5], a TCP-aware reliable link-layer protocol such as SNOOP performs best. However, TULIP is claimed to deliver better performance than SNOOP under some circumstances. In case of frequent and long disconnections, M-TCP appears to perform well. In the realm of multi-hop wireless networks, there are also three groups of schemes, namely, TCP with feedback approaches, TCP without feedback approaches, and lower layer enhancement approaches. In conclusion, feedback-based schemes seem to be able to react more quickly to non-congestion-related packet losses, thus to be more effective in enhancing TCP performance [50]. However, the price to be paid is that they are more difficult to implement, for they require end nodes and intermediate nodes to cooperate with each other. On the other hand, approaches without feedback information are relatively simple to implement, although the performance gain may not be high enough. Meanwhile, some solutions by enhancing the link layer and routing layer shed insights into how to reduce non-congestion-related losses in order to improve TCP performance.

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.

[2] I. Ali, R. Gupta, S. Bansal, A. Misra, A. Razdan and R. Shorey, Energy efficiency and throughput for TCP traffic in multi-hop wireless networks, *IEEE INFOCOM'02* (New York 2002).

[3] 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).

[4] 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).

[5] 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).

[6] H. Balakrishnan, S. Seshan and R. H. Katz, Improving reliable transport and handoff performance in cellular wireless networks, *ACM Wireless Networks* (Dec. 1995).

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

[8] P. Bhagwat, P. Bhattacharya, A. Krishna and S. K. Tripathi, Enhancing throughput over wireless LANs using channel state dependent packet scheduling, *IEEE INFOCOM'96* (San Francisco, Mar. 1996).

[9] K. Brown and S. Singh, M-TCP: TCP for mobile cellular networks, *ACM computer communication review* Vol.27 No.5 (Oct. 1997).

[10] R. Caceres and L. Iftode, Improving the performance of reliable transport protocols in mobile computing environments, *IEEE JSAC* Vol.19 No.7 (Jul. 2001).

[11] M. C. Chan and R. Ramjee, TCP/IP performance over 3G wireless links with rate and delay variation, *MobiCom'02* (Sep. 2002).

[12] 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* Vol.8 No.1 (Feb. 2001) pp. 34-39.

[13] K. Chen, Y. Xue and K. Nahrstedt, On setting TCP's congestion window limit in mobile ad hoc networks, *IEEE ICC'03* (Anchorage, Alaska, May 2003).

[14] A. DeSimone, M. C. Chuah and O. C. Yue, Throughput performance of transport-layer protocols over wireless LANs, *Proc. Globecom '93* (Dec. 1993).

[15] T. D. Dyer and R. V. Boppana, A comparison of TCP performance over three routing protocols for mobile ad hoc networks, *ACM Mobihoc* (Oct. 2001).

[16] S. Floyd and K. Fall, Router mechanisms to support end-to-end congestion control, *LBL Technical report* (Feb. 1997).

[17] S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, *IEEE/ACM Transaction on Networking* Vol.1 No.4 (Aug. 1993).

[18] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang and M. Gerla, The impact of multihop wireless channel on TCP throughput and loss, *IEEE INFOCOM'03* (San Francisco, Mar. 2003).

[19] Z. Fu, X. Meng and S. Lu, How bad TCP can perform in mobile ad-hoc networks, *IEEE Symposium on Computers and Communications* (Italy, Jul. 2002).

[20] 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).

[21] M. Gerla, K. Tang and R. Bagrodia, TCP performance in wireless multihop networks, *Proceedings of IEEE WMCSA'99* (New Orleans, LA, Feb. 1999).

[22] T. Goff, J. Moronski, D. S. Phatak and V. Gupta, Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environment, *IEEE INFOCOM'00* (Tel-Aviv, Mar. 2000).

[23] G. Holland and N. H. Vaidya, Analysis of TCP performance over mobile ad hoc networks, *MOBICOM'99* (Seattle, Aug. 1999).

[24] V. Jacobson, R. Braden and D. Borman, TCP extensions for high performance, *RFC 1323*, (May 1992).

[25] V. Jacobson and M. Karels, Congestion avoidance and control, *Proceedings of ACM SIGCOMM'88* (Aug. 1988).

[26] D. B. Johnson, D A. Maltz and 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).

[27] D-K. Kim, C.-K. Toh and Y. Choi, TCP-BuS: improving TCP performance over wireless ad hoc networks, *IEEE Comsoc Journal On Communications And Networks (JCN)* Vol.3 No.2 (2001).

[28] J. Li, C. Blake, D. S. J. De Couto, H. Lee and R. Morris, Capacity of ad hoc wireless networks, *MobiCom'01* (Rome, Italy, Jul. 2001).

[29] D. Lin and R. Morris, Dynamics of random early detection, *ACM Computer Communication Review* Vol.27 No.4 (Oct. 1997).

[30] J. Liu and S. Singh, ATCP: TCP for mobile ad hoc networks, *IEEE JSAC* Vol.19 No.7 (Jul. 2001).

[31] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, TCP selective acknowledgement options, *RFC 2018* (Oct. 1996).

[32] J. P. Monks, P. Sinha and V. Bharghavan, Limitations of TCP-ELFN for ad hoc networks, *MOMUC* (2000).

[33] T. J. Ott, T. V. Lakshman and L. H. Wong, SRED: stabilized RED, *Proceedings IEEE INFOCOM '99* (New York, Mar. 1999).

[34] R. Pan, B. Prabhakar and K. Psounis, CHOKe: a stateless active queue management scheme for approximating fair bandwidth allocation, *Proceeding of INFOCOM'00* (Tel-Aviv, Mar. 2000).

[35] V. D. Park and M. S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, *Proceedings of IEEE INFOCOM'97* (Kobe, Japan, Apr. 1997).

[36] C. Parsa and J. J. Garcia-Luna-Aceves, Improving TCP performance over wireless networks at the link layer, *ACM Mobile Networks and Applications* Vol. 5 (2000).

[37] C. E. Perkins, E. M. Belding-Royer and S. Das, Ad hoc on demand distance vector (AODV) routing, *IETF RFC 3561*

[38] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt and P. Sinha, Understanding TCP fairness over wireless LAN, *IEEE INFOCOM'03* (San Francisco, Mar. 2003).

[39] K. Ramakrishnan, S. Floyd and D. Black, The addition of explicit congestion notification (ECN) to IP, *RFC 3168*, Sep. 2001.

[40] E. Royer, S. J. Lee and C. Perkins, The effects of MAC protocols on ad hoc network communication, *IEEE WCNC* (Chicago, IL, Sep. 2000).

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

[42] W. Stevens, TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, *RFC 2001* (Jan. 1997).

[43] W. Stevens, *TCP/IP Illustrated* Vol.1 (Addison-Wesley, 1996).

[44] A. S. Tanenbaum, *Computer Networks*, (4th Edition, Prentice-Hall International, Inc. 2002).

[45] 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).

[46] *Third Generation Partnership Project*, (Release 1999).

[47] *Third Generation Partnership Project*, RLC protocol specification (3G TS 25.322:), (1999).

[48] *TIA/EIA/cdma2000*, Mobile station - base station compatibility standard for dual-mode wideband spread spectrum cellular systems, Washington: Telecommunication Industry Association, (1999).

[49] *TIA/EIA/IS-707-A-2.10*, Data service options for spread spectrum systems: radio link protocol type 3, (Jan. 2000).

[50] F. Wang and Y. Zhang, Improving TCP performance over mobile adhoc networks with out-of-order detection and response, *ACM MobiHoc'02* (Lausanne, Switzerland, Jun. 2002).

[51] H. Wu, Y. Peng, K. Long, S. Cheng and J. Ma, Performance of reliable transport protocol over IEEE 802.11 wireless LAN: analysis and enhancement, *IEEE INFOCOM'02* (New York, Jun. 2002).

[52] K. Xu, S. Bae, S. Lee and M. Gerla, TCP behavior across multihop wireless networks and the wired internet, *ACM WoWmoM'02* (Sep. 2002).

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

[54] S. Xu and T. Saadawi, Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? *IEEE Communications Magazine* (Jun. 2001).

[55] S. Xu and T. Saadawi, Revealing TCP unfairness behavior in 802.11 based wireless multi-hop networks, *IEEE PIMRC'01* (Oct. 2001).

[56] L. Zhang, S. Shenker and D. Clark, Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic, *Proceedings of ACM SIGCOMM'91* (Sep. 1991).