2003

TCP over Wireless Links: Mechanisms and Implications

Sonia Fahmy
Purdue University, fahmy@cs.purdue.edu

Venkatesh Prabhakar

Srinivas R. Avasarafa

Ossama M. Younis

Report Number:

03-004

TCP OVER WIRELESS LINKS: MECHANISMS AND IMPLICATIONS

Sonia Fahmy
Venkatesh Prabhakar
Srinivas R. Avasarala
Ossama M. Younis

CSD TR #03-004
February 2003

TCP over Wireless Links: Mechanisms and Implications

Sonia Fahmy, Venkatesh Prabhakar, Srinivas R. Avasarala, Ossama M. Younis
Department of Computer Sciences
1398 Computer Science Building
Purdue University
West Lafayette, IN 47907-1398
E-mail: fahmy@cs.purdue.edu

Abstract— **In wireless networks, the implicit assumption which TCP makes that losses indicate network congestion is no longer valid. Losses in wireless networks can result from bit errors, fading and handoffs. There are two different approaches to improve TCP performance in case of wireless losses: (1) hide non-congestion-related losses from the TCP sender, using either TCP-aware or TCP-unaware reliable link layer protocols, or using split connections with separate wireline and wireless TCP connections, or (2) adapt the TCP sender to realize that some losses are not due to congestion, e.g., using selective acknowledgments or explicit loss notification. In this paper, we classify popular mechanisms that have been recently proposed to solve the wireless transport problem, and contrast them according to their complexity, ease of deployment, and performance in different scenarios. We conclude that some TCP sender adaptations work well for certain environments only, while others are generally useful. Local recovery works best when the wireless link delay is small, and split connection schemes allow application/transport level optimizations, but exhibit overhead and lack robustness.**

Keywords— **TCP/IP, congestion control, wireless networks, satellite networks**

I. Introduction

Due to the low loss nature of wired links, TCP assumes that packet losses mostly occur due to congestion. TCP reacts to congestion by decreasing its congestion window [1] thus reducing network utilization. In wireless networks, however, losses may occur due to the high bit-error rate of the transmission medium or due to fading and mobility. TCP still reacts to losses according to its congestion control scheme, thus unnecessarily reducing the network utilization. This paper focuses on proposals that attempt to address this problem.

The typical wireless network system model is shown in figure 1. This model can represent both wireless local area networks (WLANs) such as 802.11 networks, and wireless wide area networks (WWANs) such as CDPD and GPRS networks. In ad hoc networks, on the other hand, there is no required infrastructure (such as base stations (BSs)), and nodes can organize themselves to establish communication routes. Satellite networks allow various configurations with potentially several wireless hops. In this paper, we mostly examine the configuration where the wireless link is the last hop, though we also classify approaches where this is not necessarily the case.

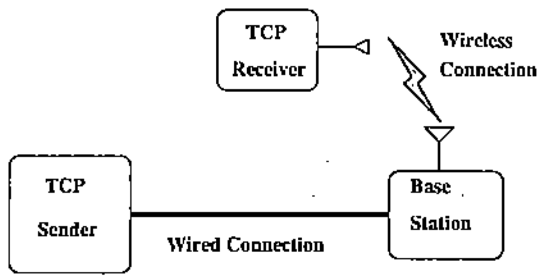

Fig. 1. Typical model of a TCP-wireless connection

Handling wireless losses may or may not be transparent to the sender. In other words, the sender implementation may be modified to be aware of losses due to wireless links, or local recovery can be used to handle such losses. The latter method shields the sender from knowing about the wireless link. A special case of this method terminates the TCP connection at the wireless interface, namely, the base station, which in turn uses some other reliable connection to connect to the destination. This solution is referred to as split-connection.

In this paper, we compare and contrast a number of wireless TCP mechanisms that have been recently proposed. We find that some TCP sender adaptations work well for certain environments only, while others are generally useful. Local recovery works best when the wireless link delay is small, and split connection schemes allow application/transport level optimizations, but exhibit high overhead and lack flexibility and robustness.

The remainder of this paper is organized as follows. Section II briefly reviews TCP congestion control. Section III discusses characteristics of wireless media that affect the performance of TCP. Section IV classifies the main mechanisms proposed to solve the wireless transport problem. Section V presents a comparative study of the various mechanisms and their implications. Section VI summarizes standardization efforts at the IETF. Finally, section VII concludes this survey.

II. Reno TCP Congestion Control

A TCP connection starts off in the **slow start** phase [2]. The slow start algorithm uses a variable called congestion window (*cwnd*). The sender can only send the minimum of *cwnd* and the receiver advertised window which we call *rwnd* (for receiver flow control). Slow start tries to reach equilibrium by opening up the window very quickly. The sender initially sets *cwnd* to 1 (RFC 2581 [1] suggests an initial window size value of 2 and RFC 2414 [3] suggests min(4×MSS,max(2×MSS,4380 bytes))), and sending one segment. (MSS is the maximum segment size.) For each ACK that it receives, the *cwnd* is increased by one segment. Increasing by one for every ACK results in exponential increase of *cwnd* over round trips, as shown in figure 2.

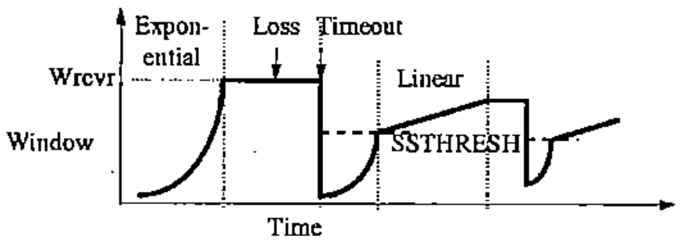

Fig. 2. TCP congestion control

TCP uses another variable *ssthresh*, the slow start threshold. Conceptually, *ssthresh* indicates the "right" window size depending on current network load. The slow start phase continues as long as *cwnd* is less than *ssthresh*. As soon as it crosses *ssthresh*, TCP goes into **congestion avoidance.** In congestion avoidance, for each ACK received, *cwnd* is increased by *1/cwnd* segments. This is *approximately* equivalent to increasing the *cwnd* by one segment in one round trip (an additive increase), if every segment is acknowledged, as shown in figure 2.

The TCP sender assumes congestion in the network when it times out waiting for an ACK. *ssthresh* is set to max(2, min(*cwnd*/2, rwnd)) segments, *cwnd* is set to one, and the system goes to slow start [1]. If a TCP receiver receives an out of order segment, it immediately sends back a duplicate ACK (dupack) to the sender. The **fast retransmit** algorithm uses these dupacks to make retransmission decisions. If the sender receives n dupacks ($n = 3$ was chosen to prevent spurious retransmissions due to out-of-order delivery), it assumes loss and retransmits the lost segment without waiting for the retransmit timer to go off. It also updates *ssthresh*. **Fast recovery** keeps track of the number of dupacks received and tries to estimate the amount of outstanding data in the network. It inflates *cwnd* (by one segment) for each dupack received, thus maintaining the flow of traffic. The sender comes out of fast recovery when the segment whose loss resulted in the duplicate ACKs is acknowledged. TCP then deflates the window by returning it to *ssthresh*, and enters the congestion avoidance phase. Variations on TCP congestion control include NewReno, SACK, FACK and Vegas, presented in [4], [5], [6], [7], [8].

III. Characteristics of Wireless Media

A number of inherent characteristics of wireless media affect TCP performance including:

- **Channel Losses:** Signals carried by wireless media are subject to significant interference from other signals, and subsequently, losses due to modification of bits while frames are being transmitted. These losses are difficult to recover from at the link layer despite the presence of error correction techniques and typically require retransmission. Retransmission can be performed at the link layer or at the transport layer (TCP). TCP performance is affected by the frequent losses occurring at the link layer, because TCP inherently assumes all losses occur due to congestion and invokes the congestion control algorithms upon detecting any loss.
- **Low Bandwidth:** Bandwidth of wireless links may be low, which can sometimes result in excessive buffering at the base station. This could lead to packets being dropped at the base station, or transmitted back-to-back on the wireless link, which in turn results in high observed round trip times [9].
- **Signal Fading:** Fading typically occurs when a wireless host is mobile. Interference from physical factors like weather, obstacles, unavailability of channels, overlapping areas of different cells, could result in signal fading and blackouts. Such blackouts can exist for prolonged periods of time.
- **Movement across Cells:** Mobility of a wireless host involves addressing connection handoff. In addition to the link layer state that has to be handed off, the base station may maintain connection state about the transport layer which might need to be handed off. Other problems like signal fading as mentioned above occur with a high probability when a host moves across cells.
- **Channel Asymmetry:** Resolving channel contention is usually asymmetric. The sending entity gets more transmission time than the receiving entity. This could lead to TCP acknowledgments being queued for transmission at

the link layer of the receiving entity and sent back to back when channel access is permitted. This can lead to larger round trip times measured by the TCP sender and to bursty traffic, which subsequently reduces the throughput of the TCP connection.

• **Link Latency:** Wireless links may exhibit high latencies and when such delays are a significant fraction of the total round trip times observed by TCP, the retransmission timeouts of TCP are set to high values, which subsequently affects TCP performance. Such conditions occur in wireless WANs and satellite networks. In addition, high variance in the measured round trip times have been observed, which impact the TCP round-trip-time (RTT) estimation algorithm [9].

IV. Wireless TCP Mechanisms

Several mechanisms have been proposed to enhance the performance of TCP over wireless links. The proposed approaches can be classified as end-to-end, local recovery or split connection. We briefly discuss the various mechanisms below and compare them in later sections. Figure 3 illustrates the three categories and sample schemes in each category.

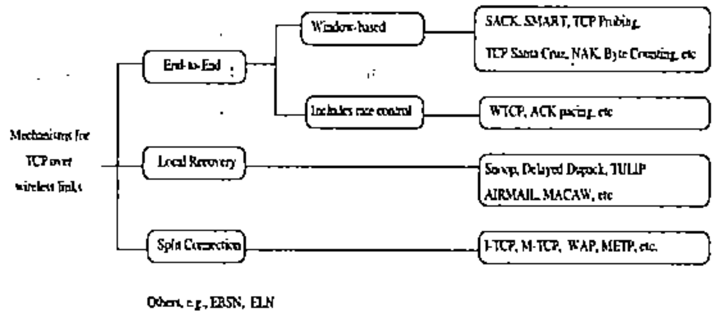
Mechanisms for TCP over wireless links
End-to-End
Window-based
SACK, SMART, TCP Probing, TCP Santa Cruz, NAK, Byte Counting, etc
Includes rate control
WTCP, ACK pacing, etc
Local Recovery
Snoop, Delayed Dupack, TULIP AIRMAIL, MACAW, etc
Split Connection
I-TCP, M-TCP, WAP, METP, etc.
Others, e.g., EBSN, ELN

Fig. 3. Categorization of mechanisms for TCP over wireless links

A. *End-to-End Mechanisms*

End-to-end mechanisms solve the wireless loss problems at the transport layers of the sender and receiver. We examine the main proposals in this section. Other alternatives being standardized are discussed in section VI.

A.1 Selective Acknowledgments

TCP Selective Acknowledgments (SACK) [10], [6], [7] uses the options field of TCP to precisely inform the sender which segments were not received. This enables the sender to retransmit only those segments that are lost, thereby not unduly wasting network bandwidth. TCP SACKs were proposed as a solution to recover from multiple losses in the same window without degrading the throughput of the connection. The same idea can be applied to networks with wireless hops to aid the sender in better recovering from non-congestion related losses. TCP still performs congestion control as in [6] or as in [7] upon detecting packet loss even if it is on the wireless link as there is no way to deduce where the loss occurred.

A.2 SMART Retransmissions

A Simple Method to Aid ReTransmissions (SMART) [11] decouples flow and error control. Each ACK packet from the receiver carries both the standard cumulative ACK and the sequence number of the packet that initiated the ACK. This informs the sender of the packets that are lost, so that the sender can selectively retransmit. The scheme uses a different heuristic to detect lost retransmissions. It also avoids the dependence on timers for the most part except for the worst case when all packets and all ACKs are lost. The scheme uses two different windows, one called the error-control window at the receiver for buffering the out-of-sequence packets, and another one called the flow-control window at the sender for buffering unacknowledged packets. Thus, error control and flow control are decoupled.

A.3 TCP Probing

With TCP probing [12], when a data segment is delayed or lost, the sender, instead of retransmitting and reducing the congestion window size, enters a probe cycle. A probe cycle entails exchanging probe segments between the sender and receiver to monitor the network. The probes are TCP segments with header options and no payload. This helps alleviate congestion because the probe segments are small compared to the retransmitted segments. The cycle terminates when the sender can make two successive RTT measurements with the aid of receiver probes. In cases of persistent errors, TCP decreases its congestion window and threshold. But for transient random errors, the sender resumes transmission at the same window size it used before entering the probe cycle.

A.4 TCP Santa Cruz

TCP Santa Cruz [13], like TCP probing, makes use of the options field in the TCP header. Its congestion control algorithm is based on relative delays that packets experience with respect to one another in the forward direction, an idea that was first introduced in TCP Vegas [8]. Therefore, ACK losses and reverse path congestion do not affect the throughput of a connection. The scheme also improves the error recovery mechanism by making better RTT estimates than other TCP implementations. Retransmissions are included in making RTT estimates to take into consideration the RTT during congestion periods.

TCP Santa Cruz uses selective acknowledgments (SACK) as discussed earlier.

A.5 Negative Acknowledgments

Negative ACKs (NAKs) can be included in the options field of TCP header to explicitly indicate which packet has been received in error so that retransmission of that packet can be initiated quickly. It works only under the assumption that a corrupted packet can still reach the destination and that the source address of the packet is still known. The sender, on receiving a NAK, can retransmit the packet without modifying the congestion window size. RTT measurement from the retransmitted packets is ignored to avoid inflating the RTT estimate.

A.6 Wireless TCP (WTCP99)

Wireless TCP (WTCP99) [9] uses rate-based transmission control and not a self-clocking window-based scheme like TCP. (We call this WTCP99 to distinguish it from a number of earlier proposals which also used the name WTCP.) The transmission rate is determined by the receiver using a ratio of the average inter-packet delay at the receiver to the average inter-packet delay at the sender. The sender transmits its current inter-packet delay with each data packet. The receiver updates the transmission rates at regular intervals based on the information in packets, and conveys this information to the sender in ACKs. WTCP99 computes an appropriate initial transmission rate for a connection based on a packet-pair approach rather than using slow-start. This is useful for short lived connections in wireless wide area networks (WWANs) where round trips are large. WTCP99 achieves reliability by using selective ACKs [10]. No retransmission timeouts are triggered as it is difficult to maintain good round trip time estimates. Instead, the sender goes into "blackout mode" when ACKs do not arrive at sender-specified intervals. In this mode, the sender uses probes to elicit ACKs from the receiver, similar to probing TCP [12].

A.7 ACK Pacing

ACK pacing [14] is a rate based approach to ACK generation at the receiver. ACK pacing results in rate-controlled sender packets, and hence avoids bursty traffic that can result in packet losses, delays and lower throughput. Pacing, however, does not distinguish between congestion losses and wireless losses.

A.8 Explicit Bad State Notification (EBSN)

Losses on the wireless link of a connection may cause timeouts at the sender, as well as unnecessary retransmissions over the entire wired/wireless network. The Explicit Bad State Notification (EBSN) scheme [15] proposes sending EBSN messages from the base station to the sender whenever the base station is unsuccessful in transmitting a packet over the wireless network. This scheme is *not* entirely end-to-end, but we discuss it here because it requires sender support like most end-to-end schemes.

EBSN receipt at the sender re-starts the TCP timer and prevents the sender from decreasing its window when there is no congestion. Although this scheme requires modifications to the TCP implementation at the sender, the required changes are minimal, no state maintenance is required, and the clock granularity (timeout interval) has little impact on performance.

A.9 Explicit Loss Notification (ELN) Strategies

Like EBSN, Explicit Loss Notification (ELN) is not purely end-to-end. ELN has been proposed to recover from errors that occur when the wireless link is the first hop [16]. A base station is used to monitor TCP packets in either direction. When the receiver sends dupacks, the base station checks if it has received the packet. If not, then the base station sets an ELN bit in the header of the ACK to inform the sender that the packet has been lost on the wireless link. The sender can then decouple retransmission from congestion control, i.e., it does reduce the congestion window. In contract to snoop [17] (discussed below), the base station need not cache any TCP segments in this case, because it does not perform any retransmissions.

B. Split Connection Mechanisms

In this class of mechanisms, the TCP connection is split at the base station. TCP is still used from the sender to the base station, whereas either TCP or some other reliable connection-oriented transport protocol is used between the base station and the receiver. The TCP sender is only affected by the congestion in the wired network and hence the sender is shielded from the wireless losses. This has the advantage that the transport protocol between the base station and mobile node can make use of its knowledge of the wireless link characteristics, or even of the application requirements. The problem with these proposals, however, is their efficiency, robustness, and handoff requirements.

B.1 Indirect TCP (I-TCP)

Indirect TCP (I-TCP), developed in 1994–1995, was one of the earliest wireless TCP proposals. With I-TCP, a transport layer connection between a mobile host and a fixed host is established as two separate connections: one over the wireless link and the other over the wired link with a "mobile support router" serving as the center point

[18]. Packets from the sender are buffered at the mobile support router until transmitted across the wireless connection. A handoff mechanism is proposed to handle the situation when the wireless host moves across different cells. A consequence of using I-TCP is that the TCP ACKs are not end-to-end thereby violating the end-to-end semantics of TCP.

B.2 Mobile TCP (M-TCP)

Mobile TCP (M-TCP) also uses a split connection based approach but tries to preserve end-to-end semantics [19]. M-TCP adopts a three level hierarchy. At the lowest level, mobile hosts communicate with mobile support stations in each cell, which are in turn controlled by a "supervisor host." The supervisor host is connected to the wired network and serves as the point where the connection is split. A TCP client exists at the supervisor host. The TCP client receives the segment from the TCP sender and passes it to an M-TCP client to send it to the wireless device. Thus, between the sender and the supervisor host, standard TCP is used, while M-TCP is used between the supervisor host and the wireless device. M-TCP is designed to recover quickly from wireless losses due to disconnections and to eliminate serial timeouts. TCP on the supervisor host does not ACK packets it receives until the wireless device has acknowledged them. This preserves end-to-end semantics, preserves the sender timeout estimate based on the whole round trip time, and handles mobility of the host with minimal state transformation.

B.3 Mobile End Transport Protocol (METP)

The mobile end transport protocol proposes the elimination of the TCP and IP layers from the wireless hosts and replacing them with a Mobile End Transport Protocol designed specifically to directly run over the link layer [20]. This approach shifts the IP datagram reception and reassembly for all the wireless hosts to the base station. The base station also removes the transport headers from the IP datagram. The base station acts as a proxy for TCP connections. It also buffers all the datagrams to be sent to the wireless host. These datagrams are sent using METP and a reduced header, thus providing minimal information about source and destination addresses and ports, and connection parameters. In addition, it uses retransmissions at the link layer to provide reliable delivery at the receiver. It does not require any change in the application programs running on the wireless host as the socket API is maintained as in a normal TCP/IP stack. However, when handoff occurs, all the state information needs to be transferred to the new base station.

B.4 Wireless Application Protocol (WAP)

In 1997, the Wireless Application Protocol (WAP) forum [21] was founded by Ericsson, Nokia, Motorola, and Phone.Com (formerly Unwired Planet). Like METP, WAP includes a full protocol stack at the receiver, as well as gateway support, as shown in figure 4. The WAP protocol layers are shown in figure 5.

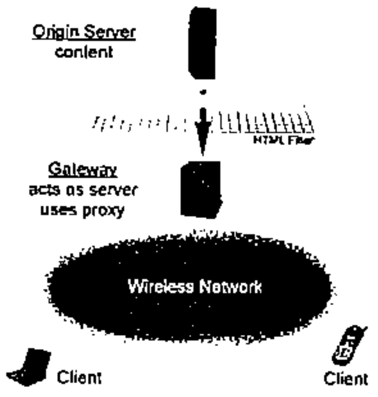

Fig. 4. WAP gateway performs translation

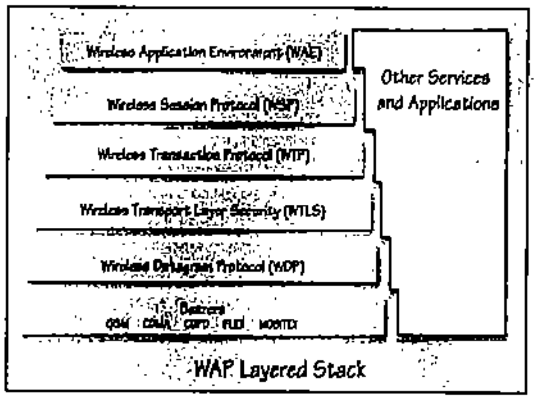
Wireless Application Environment (WAE)
Wireless Session Protocol (WSP)
Wireless Transaction Protocol (WTP)
Wireless Transport Layer Security (WTLS)
Wireless Datagram Protocol (WDP)
Bearers
GSM
CDMA
CDPD
FLEX
MOBITEX
Other Services and Applications
WAP Layered Stack

Fig. 5. Wireless application protocol layers

The top WAP layer, the Wireless Application Environment (WAE) establishes an environment to allow users to build applications that can be used over a wide variety of wireless systems. WAE is composed of user agents such as browsers, text editors, date book or phone book. WAE also includes scripting, higher-lever programming languages and image formats. WAE uses languages such as WMLScript (similar to JavaScript) and WML (similar to HTML).

The Wireless Session Protocol (WSP) handles communication between the client and proxy or server. WSP opens a session of communication between client and server, exchanges encoded data, exchanges requests and replies, and supports several asynchronous transmission modes of data.

The Wireless Transaction Protocol (WTP) handles transactions, re-transmission of data, and separation and concatenation of data. This particular protocol has a separate interface that manages and referees the WTP layer and the settings of the handheld device. This management application is known as the WTP Management Entity. For WTP to work, the following factors are important: (1) The handheld device is within the coverage area of base agent; (2) The handheld device is turned on and is reliable; (3) Resources are adequate, e.g., CPU and memory; (4) WTP settings are correctly input.

The Wireless Transport Layer Security (WTLS) is the layer that handles security of data and validity of data between two communicating entities. To transport data, WTLS needs the source address and port number to identify the message creator, and from where the message is being sent, the destination address and port number to which data is being sent, and of course, the data itself. WTLS has a connection interface which provides a connection protocol between client and server.

The Wireless Data Protocol (WDP) acts as the communication layer between the upper level protocols (WTLS, WTP, and WSP), and the bearer services. The function of WDP is to provide a stable environment so that any of the underlying bearers can operate using WAP. WDP can be adapted to different bearers with different services. However, the services offered by WDP remain constant thus providing a continuous interface to the upper layers of the WAP stack.

C. *Local Recovery Mechanisms*

The motivation behind this approach is that wireless losses are local and hence recovery from them should be local. The mechanisms discussed below entail detecting wireless losses at the base station or at the receiver and using local retransmissions and sometimes forward error correction for recovery. This class of mechanisms can be considered as a compromise between the above two categories (end-to-end and split connections) [22].

C.1 The Snoop Protocol

The Snoop protocol caches TCP packets at the base station [17]. State is maintained for all the TCP connections passing through the base station to the wireless device. The link layer is TCP-aware, thereby making it possible to monitor TCP segments and infer packet losses on the wireless link from duplicate ACKs (dupacks). Packet losses are also inferred through local timeouts. Lost TCP segments are retransmitted on the wireless link. This is possible because all TCP segments are cached until ACKs for them arrive from the wireless device. Additionally, the dupacks are suppressed at the base station in order to prevent the sender from unnecessarily invoking congestion control algorithms. This method works well [22] especially in wireless local area networks (WLANs).

C.2 Delayed Duplicate ACKs (DDA)

Delayed duplicate ACKs (DDA) recovers from wireless losses using link level retransmissions, while *delaying* the dupacks at the receiver to prevent the sender from retransmitting [23]. If the retransmitted packets are received after a time interval d, the dupacks are discarded; otherwise they are allowed to return to the sender. DDA, unlike snoop, is TCP-unaware in that it relies on link level (not TCP-level like snoop) segments and ACKs. This has the advantage that it can work even when the IP payload is encrypted, such as with IPSEC. However, due to the delaying of all receiver dupacks by a time d, the sender may not invoke congestion control algorithms like fast retransmit and recovery (see section II) and time-out with both congestion and wireless losses. Thus DDA only works well when congestion losses are minimal.

C.3 Transport Unaware Link Improvement Protocol (TULIP)

Like DDA, TULIP recovers from wireless losses by link level retransmissions [24]. The protocol is TCP-unaware (though it is aware of reliability requirements). Dupack generation at the receiver is avoided by only delivering in-order frames to the receiver TCP/IP. Lost packets are detected through a bit vector that is returned as a part of the link layer ACK, which helps immediate loss recovery. TULIP is designed for efficient operation over half-duplex radio channels. It introduces a media access control (MAC) acceleration feature to accelerate the return of link layer ACKs in order to improve throughput.

C.4 TCP interaction with MAC layer (MACAW)

MACAW [25] investigates the interaction between TCP and MAC layer backoff timers, which may cause unfairness and capture conditions. The authors show these effects are very pronounced in CSMA and FAMA MAC protocols. They propose adding link layer ACKs and a less aggressive backoff policy to improve performance of TCP over these MAC protocols. The new MAC layer protocol is called MACAW.

C.5 AIRMAIL

AIRMAIL [26] is a popular link layer protocol designed for indoor and outdoor wireless networks. It provides a reliable link layer by using local retransmissions and forward error correction (FEC) at the physical layer. The mobile terminal combines several ACKs into a single event-driven

ACK, and the base station sends periodic status messages. However, this entails that no error correction can be done until the ACKs arrive which can cause TCP to time out if the error rate is high.

V. Comparison of Wireless TCP Mechanisms

In this section, we compare and contrast the various wireless TCP mechanisms based on interaction among protocol stack layers, division of responsibilities among entities, complexity, and performance in various conditions.

A. Interaction between TCP and the Link Layer

End-to-end mechanisms typically do not need any support from the link layer. On the other hand, approaches based on recovering from a wireless loss locally need substantial support at the link layer. The snoop protocol [27] monitors and caches TCP segments and performs local retransmissions of the TCP segments which necessitates the link layer to be TCP aware. TULIP [24] relies on link layer retransmission and MAC acceleration to improve the throughput of the wireless link. Both TULIP and DDA do not require base station TCP-level support. AIRMAIL [26] relies on event-driven acknowledgments and reliable retransmissions at the link layer. Mobile End Transport Protocol (METP) [20], which uses a split connection, also relies on a reliable retransmission and efficient demultiplexing of the reduced header at the link layer. MACAW [25] uses link layer ACKs and a less aggressive backoff policy to improve the performance of TCP over the link layer.

B. Interaction between TCP and the Network Layer

Typically, mechanisms to improve TCP in wireless networks do not need support from the IP layer. With METP [20], however, the IP and TCP layers are transferred from the wireless device to the base station. METP is directly implemented over the link layer and uses link layer reliability mechanisms and reduced headers to improve the throughput in the wireless link. Hence, this requires modifying the network layer at the base station and the wireless device. WAP [21] also modifies the protocol stack on the mobile device, as well as the gateway.

C. Retransmission Approaches and ACKs

End-to-End mechanisms perform retransmissions at the TCP sender. TCP SACKs [10] are used by the TCP sender to selectively retransmit lost segments. SMART retransmissions [11] infer the packets lost from the additional (receiver-transmitted) sequence number of the packet that caused the cumulative ACK. This eliminates the need for a bit vector in the ACK to specify lost packets. TCP probing [12] enters a probe cycle upon detecting a lost packet instead of retransmitting. This helps in alleviating congestion while continuing to obtain round trip measurements. TCP Santa Cruz [13] uses an ACK window from the receiver, similar to SACK. In addition, it uses retransmitted packets to obtain the more accurate round trip time measurements.

Local recovery mechanisms perform retransmissions only on the wireless link. The snoop protocol [27] performs retransmissions of the whole TCP segments on the wireless link upon receipt of dupacks at the base station. TULIP [24], delayed dupacks [23], and AIRMAIL [26] depend on *link layer* retransmissions as the base station used need not be TCP-aware. TULIP uses a bit vector in link layer ACKs, similar to SACK at the transport layer.

Split connection mechanisms like I-TCP [18] and M-TCP [19] use a separate connection between the base station and the wireless device. So the retransmissions are done by the transport protocol between the wireless device and the base station as appropriate. Mobile End Transport Protocol [20] uses link layer retransmissions to recover from losses.

D. Flow Control

Among the approaches we have examined, only WTCP99 [9] and ACK pacing (to some extent) [14] use rate based flow control. In WTCP99, the transmission rate is controlled by the receiver using inter packet delay. The motivation behind this idea is that the round trip time calculation for a wireless WAN is highly susceptible to fluctuations. All the other schemes use window based flow control. With ACK pacing, the sender transmission of packets is controlled by the rate at which ACKs arrive. This avoids bursty traffic and reduced throughput.

E. Loss Detection and Dependence on Timers

SMART [11] tries to avoid timeouts. The sender discovers lost packets by the additional cumulative ACK information. SMART uses RTT estimates to detect loss of retransmissions. Timers are used only to deal with the worst case where all packets and all ACKs are lost. In schemes like EBSN [15], NAK and ELN [16], where the wireless channel state is explicitly conveyed to the sender, the timeout for the transmitted packets can be reset on the arrival of loss notification. TCP Santa Cruz [13] measures RTTs for retransmissions to provide better RTT estimates. This eliminates the need for timer-backoff strategies where the timeout is doubled after every timeout and retransmission.

In the Snoop approach [17], the base station detects wireless losses from dupacks on the reverse path. In situations when wireless channel losses are very high and loss detection is hampered, it uses a timer, set to a fraction of the wireless RTT to perform local retransmissions. Using coarse TCP timers and fine link layer timers avoids retransmission contention between the two layers. However, unnecessary link layer transmissions may result in out-of-order packets at the receiver which generates dupacks, resulting in retransmissions.

F. Effect of Reverse Path Asymmetry

Cumulative ACK schemes used in most TCP implementations offer robustness from ACK losses occurring on the reverse path from the receiver to the sender. In cases where ACKs follow slower or more congested links, RTT estimates at the sender may be inflated and do not reflect the forward path state. TCP Santa Cruz [13] avoids this problem by making use of only forward delay measurements for congestion control. WTCP99 [9] makes use of the ratio of inter packet delay at the receiver and the sender for congestion control and, therefore, avoids the RTT inflation problem.

G. Effect of Wireless Link Latency on RTT

Local recovery schemes like [17] that use either TCP or link layer retransmissions work under the assumption that the wireless link latency is small compared to latency of the entire connection. But in cases where the wireless link latency is considerably high, there will be contention between TCP retransmissions from the sender and local retransmissions resulting in reduced throughput.

H. State Maintained

End-to-end mechanisms do not need any state at any intermediate node. Some local recovery mechanisms need to maintain state while others do not. The Snoop protocol [27] places the complexity at the base station, so TCP segments need to be cached until acknowledged by the receiver. In Explicit Loss Notification strategies for recovering from wireless losses when the wireless link is the first hop [16], the TCP segments need not be maintained by the base station as no retransmission is performed. However, information like sequence numbers of the packets that pass through the base station need to be maintained to detect the losses. Other protocols like TULIP [24], Delayed Dupacks [23], and AIRMAIL [26] rely on link level retransmissions and state, as the protocols are TCP-unaware. Split connection mechanisms require the TCP packets be stored at the base station to be transmitted on a separate connection across the wireless link.

I. Robustness and Overhead

Copying is required in split connection based approaches. This is because the connection is terminated at the base station and hence the packets destined for the wireless host need to be buffered at the base station until they are transmitted on the wireless link. Overhead is incurred by the kernel when copying the packets across buffers for different connections [22]. The difference between this and TCP state in snoop, for example, is that in the latter, the state is soft. A failure of the base station in snoop only temporarily affects the throughput of the connection. However, if the base station fails in split connection based approaches, the receiving TCP entity might not receive the packets that have been transmitted by the sending entity, and subsequently ACKed by the base station. M-TCP avoids this through delaying ACKs to the sender until packets are received at the mobile node. However, split connection approaches generally suffer from robustness, efficiency and flexibility problems.

J. Preserving End-to-End Semantics

End-to-end mechanisms clearly preserve end-to-end semantics. Local recovery mechanisms also preserve end-to-end semantics except that dupacks are suppressed as in the Snoop protocol [27] or delayed as in DDA [23]. Split connection approaches typically terminate the connection at the base station [28], [20], thereby violating the end-to-end semantics of TCP in the strict sense. The M-TCP approach [19], however, attempts to maintain end-to-end semantics while splitting the connection at the base station by acknowledging packets back on the wired network only when the ACKs arrive on the wireless link.

Another important point is that TCP-aware local recovery, such as snoop [17] reads TCP headers and thus needs to be adapted when the IP payload is encrypted such as with IPSEC. DDA and TULIP, as well as approaches like MACAW and AIRMAIL, are not TCP aware, and can work with encrypted IP payloads.

K. Suitability to different Network Environments

Some protocols work very well for certain types of network environments. TULIP [24] is tailored for the half-duplex radio links and provides a MAC acceleration feature to improve the throughput in wireless links. The Snoop protocol [27] and delayed duplicate ACKs work best in local area network environments where wireless round trip times are small and fading losses do not occur frequently. In fact, snoop [27] has little benefit if the link layer already provides reliable in-order delivery, or if the wired link delay is small. WTCP99 [9] is tuned to perform well in wireless WANs with low bandwidths, large round trip times, asymmetric channels, and occasional blackouts and signal fading.

L. Adaptation to Long Fading Periods

WTCP99 [9] adapts to long fading periods effectively. This is because the motivation for its design stems from wireless WANs which exhibit fading. WTCP99 uses rate based control, inter-packet delay as a congestion metric, and blackout detection to adapt to long fading periods. Snoop [27] and similar protocols adapt to fading by making use of local timers in absence of dupacks. TCP probing [12] also adapts to long fading periods because it enters a probe cycle upon detection of a packet loss. The cycle terminates only after the sender makes two successive RTT measurements.

VI. The Internet Engineering Task Force (IETF) Efforts

Recently, the performance implications of link characteristics (PILC) working group at the IETF has recommended certain changes to help TCP adapt to (1) asymmetry, (2) high error rate, and (3) low speed links.

Asymmetry in network bandwidth can result in variability in the ACK feedback returning to the sender. Several techniques can mitigate this effect, including using *header compression* [29]; reducing ACK frequency by taking advantage of cumulative ACKs; using TCP congestion control for ACKs on the reverse path; giving scheduling priority to ACKs over reverse channel data in routers; and applying backpressure with scheduling.

The TCP sender must also handle infrequent ACKs. This can be done by bounding the number of back-to-back segment transmissions. Taking into account cumulative ACKs and not number of ACKs at the sender can also improve performance. This scheme is called *byte (versus ACK) counting* [30] because the sender increases its congestion window based on the number of bytes covered (ACKed) by each ACK.

In addition, reconstructing the ACK stream at the sender; router ACK filtering (removing redundant ACKs from the router queue); or ACK compaction/expansion (conveying information about discarded ACKs from the compacter to the expander) can be used.

For *high error rate* links, experiments show that approaches such as explicit congestion notification (ECN) [31], fast retransmit and recovery and SACK are especially beneficial. Explicit loss notification, delayed duplicate acknowledgments (section IV-C.2), persistent TCP connections, and byte counting are a few of the open research issues in this area. TCP-aware performance enhancing proxies (PEPs), such as split connection mechanisms and snoop can also be used.

For *low speed* links, in addition to *compressing the TCP header and payload* [29], [32], several changes to the congestion avoidance algorithm are recommended. First, hosts that are directly connected to low-speed links should advertise small receiver window sizes to prevent unproductive probing for non-existent bandwidth.

Second, maximum transmission units (MTUs) should be carefully selected to not monopolize network interfaces for human-perceptible amounts of time (e.g., 100-200 ms) and to allow delayed acknowledgments. Large MTUs which monopolize the network interfaces for long periods are likely to cause the receiver to generate an ACK for every segment rather than delayed ACKs. Using a smaller MTU size will decrease the queuing delay of a TCP flow compared to using larger MTU size with the same number of packets in the queue.

Third, the receiver advertised window size, *rwnd*, should be carefully selected. Dynamic allocation of TCP buffers (or *buffer auto-tuning*) [33] based on the current effective window can be used.

Finally, binary encoding of web pages, such as with WAP (section IV-B.4) can be used to make web transmissions more compact. Many of the suggested solutions mentioned in this section are still in the research phase.

RFC 2757 [34] provides a good summary of various TCP over wireless WAN proposals.

A. TCP over Satellites

In addition to bandwidth asymmetry, restricted available bandwidth, intermittent connectivity, and high error rate due to noise, Geostationary Earth Orbit (GEO) satellite links are characterized by very high latency. This is because such satellites are usually placed at an altitude of around 36,000 km, resulting in a one-way link delay of around 279 ms, or a round trip delay of approximately 558 ms. This results in a long feedback loop and a large delay bandwidth product. For Low Earth Orbit (LEO) and Medium Earth Orbit (MEO) satellites, the delays are shorter, but inter-satellite links are more common and round trip delays are variable.

Allman et al recommend in RFC 2488 [35] several techniques to mitigate the effect of these problems. These techniques include using path MTU discovery; forward error correction (FEC); TCP slow start, congestion avoidance, fast retransmit, fast recovery, and selective acknowledgments (SACK). The TCP window scaling option must also be used to increase the receiver window (*rwnd*) to place a larger upper bound on the TCP window size. The algorithms companion to window scaling, including protection against wrapped sequence space (PAWS) and round trip time measurements (RTTM) are also recommended.

A number of additional mitigations are still being researched, and are summarized in RFC 2760 [36]. These include using larger initial window sizes (to eliminate the timeout observed with delayed ACKs at startup and reduce transmission time for short flows); transaction TCP (eliminates the TCP 3-way handshake with every connection); using multiple TCP connections for a transmission; pacing TCP segment transmissions; persistent TCP connections; byte counting at the sender; ACK filtering; ACK congestion control; explicit loss notification; using delayed ACKs only after slow start; setting the initial slow start threshold (ssthresh) to the delay bandwidth product as in [5]; header compression; SACK, FACK, random early detection (RED) at routers, ECN and TCP-friendly control.

Table I summarizes the current research issues with link characteristics. A blank entry means the technique was not mentioned in the relevant RFC/draft. Note that slow start, congestion avoidance, fast retransmit and fast recovery are required and SACK is recommended for all cases. Table II summarizes the RFCs describing TCP congestion control and their contents.

VII. Summary and Open Issues

Table III gives a brief summary of the mechanisms discussed in this paper and their requirements and implications.

Providing reliable transport for wireless networks has been an active topic of research since 1994. Many new protocols and TCP modifications have been proposed to improve performance over wireless links. We have presented a taxonomy of these approaches and compared them in terms of complexity, ease of deployment, and performance in different conditions.

Local recovery mechanisms work well when the wireless link latency is small compared to the RTT of the connection. For local recovery mechanisms that use link layer retransmissions, it is important that the granularity of the link layer timers be very fine compared to the TCP timers. Otherwise, contention between the two timers can reduce throughput. Another important observation is that out-of-order delivery of link layer retransmissions at the receiver [22] may trigger duplicate ACKs and TCP retransmissions by the sender. TULIP [24] avoids this problem by preventing out-of-order delivery. In addition, hiding wireless losses from the sender by suppressing dupacks may cause inaccurate RTT measurements at the sender. This is especially important in cases where the wireless link latency is comparable to the RTT.

Split connection schemes shield the sender from wireless losses, and allow application and transport level optimizations at the base station which is aware of the wireless link characteristics. However, split connection schemes do not preserve end-to-end semantics, and hence they are not robust to base station crashes. Moreover, the schemes are inefficient and involve significant copying overhead and handoff management.

A few of the TCP sender adaptations work well for certain environments only, while others, such as selective ACKs, are generally useful. Modifications proposed to TCP for lossy, asymmetric, low speed or high delay links are being standardized by IETF. It is important to study the interactions between these mechanisms to ensure no adverse effects occur. For example, some straightforward mechanisms, such as byte counting, result in bursty traffic, and should be used with caution even though they are very useful for satellite environments. Many of the proposed approaches have been designed for specific environments and must be carefully examined if they are to be universally deployed. For example, WTCP99 [9] is well suited for wireless wide area networks. TULIP [24] is tailored for half-duplex radio links, while MACAW [25] is designed for CSMA and FAMA links.

References

[1] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, April 1999, Also see http://tcpsat.lerc.nasa.gov/tcpsat/papers.html.

[2] V. Jacobson, "Congestion avoidance and control," in *Proceedings of the ACM SIGCOMM*, August 1988, vol. 18, pp. 314–329, ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[3] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," RFC 2414, September 1998.

[4] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," RFC 2582, April 1999, Also see http://www.aciri.org/floyd/tcp_small.html.

[5] J. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," in *Proceedings of the ACM SIGCOMM*, August 1996, pp. 270–280. http://www.acm.org/sigcomm/ccr/archive/1996/conf/hoe.ps.

[6] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP," in *ACM Computer Communication Review*, July 1996, vol. 26, pp. 5–21, ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z.

[7] M. Mathis and J. Mahdavi, "Forward acknowledgment: Refining TCP congestion control," in *Proceedings of the ACM SIGCOMM*, August 1996, Also see http://www.psc.edu/networking/papers/papers.html.

[8] L. Brakmo, S. O'Malley, and L. Peterson, "TCP vegas: New techniques for congestion detection and avoidance," in *Proceedings of the ACM SIGCOMM*, August 1994, pp. 24–35, http://netweb.usc.edu/yaxu/Vegas/Reference/vegas93.ps.

[9] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," in *ACM Mobicom '99*, Seattle, Washington, Aug. 1999.

TABLE I
MITIGATIONS FOR SPECIAL LINK CHARACTERISTICS

Technique	Satellites	Asymmetry	High Error Rate	Low Speed
Path MTU discovery	recommended			recommended small
Forward error correction	recommended			
Transaction TCP	recommended			
Larger initial window	recommended			
Delayed ACK after Slow Start	recommended			
Estimating ssthresh	recommended			
Advertised receiver window	large (window scaling)			small/auto-tuned
Byte counting	recommended	recommended	recommended	
Explicit loss notification	recommended		recommended	
Explicit congestion notification	recommended		recommended	
Multiple connections	recommended			
Pacing segments	recommended			
Header and payload compression	recommended	recommended		recommended
Persistent connections	recommended		recommended	
ACK congestion control	recommended	recommended		
ACK filtering	recommended	recommended		
ACK compaction		recommended		
ACK scheduling priority and backpressure		recommended		

[10] M. Mathis, J. Madhavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," RFC 2018, October 1996.

[11] S. Keshav and S.P. Morgan, "SMART retransmission: Performance with random losses and overload," in *Proc. IEEE INFOCOM '97*, Kobe, Japan, Apr. 1997.

[12] V. Tsaoussidis and H. Badr, "TCP-Probing: Towards an error control schema with energy and throughput performance gains," Technical Report TR4-20-2000, Dept. of Computer Science, SUNY Stony Brook, Apr. 1999, Also appears in Proceedings of ICNP 2000.

[13] C. Parsa and J.J.Garcia-Luna-Aceves, "Improving TCP congestion control over internets with heterogeneous transmission media," in *Proc. of the Seventh Annual International Conference on Network Protocols*, Toronto, Canada, Nov. 1999.

[14] Amit Aggarwal, Stefan Savage, and Tom Anderson, "Understanding the performance of TCP pacing," in *Proceedings of the 2000 IEEE Infocom Conference*, Tel-Aviv, Israel, Mar. 2000.

[15] B.S. Bakshi, P. Krishna, N.H. Vaidya, and D.K. Pradhan, "Improving performance of TCP over wireless networks," in *Proc. of the 17th International Conference on Distributed Computing Systems '97*, Baltimore, Maryland, May 1997.

[16] Hari Balakrishnan and Randy H. Katz, "Explicit Loss Notification and wireless web performance," in *Proc. IEEE Globecom Internet Mini-Conference*, Sydney, Australia, Nov. 1998.

[17] H. Balakrishnan, S. Seshan, E. Amir, and R.H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. of 1st ACM Conference on Mobile Computing and Networking*, Berkeley, California, Nov. 1995, http://daedalus.cs.berkeley.edu/publications/mcn.ps.

[18] Ajay Bakre and B. R. Badrinath, "Handoff and system support for indirect TCP/IP," in *Second USENIX Symposium on Mobile and location-independent computing proceedings*, Ann Arbor, Michigan, Apr. 1995.

[19] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Computer Communication Review*, vol. 27, no. 5, pp. 19–43, Oct. 1997.

[20] K. Y. Wang and S. K. Tripathi, "Mobile-end transport protocol: An alternative to TCP/IP over wireless links," in *INFOCOM*, San Francisco, California, March/April 1998, p. 1046.

[21] "The WAP forum," http://www.wapforum.org, 2001.

[22] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, Dec. 1997, http://www.ccrc.wustl.edu/~ton/dec97.html.

[23] Miten N. Mehta and Nitin H. Vaidya, "Delayed Duplicate Acknowledgements: A proposal to improve performance of TCP on wireless links," Interim report, Texas A&M University, College Station, Texas, Dec. 1997.

[24] C. Parsa and J.J. Garcia-Luna-Aceves, "TULIP: A link-level pro-

TABLE II

IMPORTANT TCP CONGESTION CONTROL-RELATED RFCS

RFC	Describes
2018	selective acknowledgments (SACK)
2309	random early detection (RED)
2414–6	increasing initial window size
2481	explicit congestion notification
2488	TCP over satellite enhancements
2525	known TCP implementation problems
2581	slow start, congestion avoidance, fast retransmit, fast recovery, idle periods, ACK generation
2582	NewReno
2757	long thin networks, e.g., wireless WANs
2760	ongoing TCP satellite research
2861	congestion window validation (decay during idle periods)
2883	SACK extensions (use of SACK for acknowledging duplicate packets)
2884	performance of explicit congestion notification
2923	MTU discovery
2988	retransmission timer (RTO) computation
3042	limited transmit option (recovery with a small window or multiple losses)

TABLE III

SUMMARY OF MECHANISMS FOR TCP OVER WIRELESS LINKS

Scheme	*Type*	*Sender Support*	*Receiver Support*	*BS Support*	*State at BS*	*Retrans–mit at*	*LL Support*	*Flow Control*	*BS Crash Impact*	*Preserves E2E*
SACK	E2E	Yes	Yes	No	No	Sender	No	WB	No	Yes
SMART	E2E	Yes	Yes	No	No	Sender	No	WB	No	Yes
TCP-P	E2E	Yes	Yes	No	No	Sender	No	WB	No	Yes
TCP-SC	E2E	Yes	Yes	No	No	Sender	No	WB	No	Yes
NAK	E2E	Yes	Yes	No	No	Sender	No	WB	No	Yes
Byte-C	E2E	Yes	No	No	No	Sender	No	WB	No	Yes
WTCP99	E2E	Yes	Yes	No	No	Sender	No	RB	No	Yes
ACK-P	E2E	No	Yes	No	No	Sender	No	WB/RB	No	Yes
EBSN	E2E+	Yes	No	LL	No	Sender	No	WB	No	Yes
ELN	E2E+	Yes	No	TL	Minimal	Sender	Yes	WB	No	Yes
I-TCP	SC	No	Mostly	TL/AL	Yes	BS	No	WB	Yes	No
M-TCP	SC	No	Yes	TL/AL	Yes	BS	No	WB	Yes	Yes
METP	SC	No	Yes	TL/AL	Yes	BS	Yes	WB	Yes	No
WAP	SC	No	Yes	TL/AL	Yes	BS	No	WB	Yes	No
Snoop	LR	No	No	TL/LL	TL/LL	BS	Yes	WB	Small	Yes
DDA	LR	No	LL	LL	LL	LL-BS	Yes	WB	Minimal	Yes
TULIP	LR	No	LL	LL	LL	LL-BS	Yes	WB	Minimal	Yes
AIRMAIL	LR	No	LL	LL	LL	LL-BS	Yes	WB	Minimal	Yes
MACAW	LR	No	LL	LL	LL	LL-BS	Yes	WB	Minimal	Yes

TCP-P = TCP Probing, TCP-SC = TCP Santa Cruz, Byte-C = Byte Counting, ACK-P = ACK Pacing, WB = Window Based, RB = Rate Based, AL = application level, TL = transport level, and LL = link level

tocol for improving TCP over wireless links," in *Proc. IEEE Wireless Communications and Networking Conference 1999*, New Orleans, Louisiana, Sept. 1999, Also appears in the MONET journal in 2000.

[25] Mario Gerla, Ken Tang, and Rajiv Bagrodia, "TCP performance in wireless multihop networks," in *Proceedings of IEEE WMCSA*, New Orleans, LA, Feb. 1999.

[26] 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*, vol. 1, no. 1, pp. 47–60, 1995.

[27] H. Balakrishnan, S. Seshan, and R.H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks*, vol. 1, no. 4, Dec. 1995.

[28] A. Bakre and B.R. Badrinath, "I-TCP: Indirect tcp for mobile hosts," in *Proc. of the 15th International Conference on Distributed Computing Systems*, Vancouver, BC, May 1995.

[29] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links," RFC 1144, February 1990.

[30] Mark Allman, "TCP Byte Counting Refinements," *ACM Computer Communication Review*, vol. 29, no. 3, July 1999.

[31] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP," RFC 2481, January 1999.

[32] A. Shacham, R. Monsour, R. Pereira, and M. Thomas, "IP Payload Compression Protocol," RFC 2393, December 1998.

[33] Jeffrey Semke, Jamshid Mahdavi, and Matthew Mathis, "Automatic tcp buffer tuning," *ACM Computer Communication Review*, vol. 28, no. 4, Oct. 1998.

[34] G. Montenegro, S. Dawkins, M. Kojo, V. Magret, and N. Vaidya, "Long thin networks," RFC 2757, January 2000.

[35] M. Allman, D. Glover, and L. Sanchez, "Enhancing TCP over satellite channels using standard mechanisms," RFC 2488, January 1999.

[36] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, , J. Heidemann, S. Ostermann, K. Scott, J. Semke, and J. Touch, "Ongoing TCP research related to satellites," RFC 2760, February 2000.

[37] Alhussein Abouzeid, Sumit Roy, and Murat Azizoglu, "Stochastic modeling of TCP over lossy links," in *INFOCOM*, Tel Aviv, Israel, Mar. 2000.

[38] Anurag Kumar, "Comparative performance analysis of versions of TCP in local network with a lossy link," *IEEE/ACM Transactions on networking*, vol. 6, no. 4, pp. 485–498, August 1998, http://www.ccrc.wustl.edu/~ton/aug98.html, http://ece.iisc.ernet.in/~anurag.