Improving TCP Performance over Mobile Networks

HALA ELAARAG

Stetson University

Transmission Control Protocol (TCP) is the most commonly used transport protocol on the Internet. All indications assure that mobile computers and their wireless communication links will be an integral part of the future internetworks. In this paper, we present how regular TCP is well tuned to react to packet loss in wired networks. We then define mobility and the problems associated with it. We discuss why regular TCP is not suitable for mobile hosts and their wireless links by providing simulation results that demonstrate the effect of the high bit error rates of the wireless link on TCP performance. We discuss and illustrate the problems caused by the mobility of hosts using a graph tracing packets between fixed and mobile hosts. We then present a survey of the research done to improve the performance of TCP over mobile wireless networks. We classify the proposed solutions into three categories: link layer, end-to-end and split. We discuss the intuition behind each solution and present example protocols of each category. We discuss the protocols functionality, their strengths and weaknesses. We also provide a comparison of the different approaches in the same category and on the category level. We conclude this survey with a recommendation of the features that need to be satisfied in a standard mobile TCP protocol.

Categories and Subject Descriptors: A.1 **[General Literature]**: Introductory and Survey; C.4 **[Computer Systems Organization]**: Performance of Systems—*Reliability, availability, and serviceability*; C.2.2 **[Computer—Communication Networks]**: Network Protocols; K.6.3 **[Management of Computer and Information Systems]**: Software Management—*Software development, software maintenance, software selection*; K.6.4 **[Management of Computer and Information Systems]**: System Management—*Quality assurance*

General Terms: Design, Performance, Standardization

Additional Key Words and Phrases: standard TCP, mobile TCP, wireless TCP, TCP performance, wired networks, mobile wireless networks, mobility, base station, mobile host, comparison of TCP implementations, link layer, end-to-end, split TCP, snoop, Reno, New-Reno, SACK, I-TCP, WTCP, MTCP, M-TCP, WAP

1. INTRODUCTION

The explosive growth of wide-area cellular systems and local-area wireless networks and the emergence of home area radio networks and personal area body networks are just the beginning of "the wireless revolution." The ultimate goal—uncompromised connectivity and performance for mobile computing devices—requires that we meet the challenge of creating fully integrated, seamless, fault-tolerant and heterogeneous networks composed of fully distributed,

Author's address: Department of Mathematics and Computer Science, Stetson University, 421 N. Woodland Blvd., DeLand, FL 32720; email: helaarag@stetson.edu.

energy efficient, and ubiquitous mobile computing platforms. The realization that wireless connectivity profoundly affects the way we compute, communicate, and interact, motivates us to better comprehend all the aspects of the underlying systems and the interactions between them. Making truly tetherless computing possible, demands that we carefully evaluate, enhance, and perhaps re-design our networks, systems, algorithms, and applications.

Mobile networks and their wireless links are fundamentally different from conventional stationary, wired computer networks. Mobile connectivity frees communication from the location constraints of the stationary wireline infrastructure. It will allow users to access information anytime, anywhere. Mobile users would like to use the same applications over the wireless link and with the same quality of service (QoS) they are getting over a wired link.

TCP/IP is very popular in wired networks. Many researchers [Hasegawa et al. 1999; Faber et al. 1999; Bruyeron et al. 1998; Heidemann 1997; Mathis et al. 1997] studied the performance of TCP and suggested improvements ([Rhee et al. 1999; Pazos et al. 1999]). Many others were interested in the modeling and analysis of TCP [Yang 1999; Mo et al. 1999; Paxon 1994]. There are a wide variety of TCP versions used on the Internet.

Mobile wireless is one of the most challenging environments for the Internet protocols and for TCP in particular. One approach to supporting the wireless environment is to use a transport protocol, not TCP, which is specifically adapted to the wireless world. These protocols can account for problems associated with the nature of the mobile wireless environment. WTCP by Sinha et al. [1999] is an example of such a protocol. WTCP is specifically designed for wireless wide area networks. It is a rate based protocol rather than window-based like TCP. It uses a ratio of the average interpacket delay observed at the receiver to the interpacket delay at the sender as the primary metric for rate control, rather than the packet loss and retransmit timeouts used by TCP.

Many researchers proposed solutions to improve the performance of TCP over mobile wireless networks in an attempt to provide seamless interworking between the wired and wireless worlds. In this paper, we adopt this approach. We discuss the effect of mobility on the performance of TCP and the different solutions proposed in the literature to alleviate the problem.

The rest of the paper is organized as follows. Section 2 presents a background for regular TCP and a definition of mobility. Section 3 presents the problems that arise from the nature of mobile (wireless) networks. Section 4 explains why regular TCP is not suitable for mobile hosts by showing simulation results of the effect of high bit error rates of the wireless links on the performance of regular TCP. It also demonstrates the reason for the negative effect on TCP performance by providing a graph tracing the packets between fixed hosts and mobile hosts in a scenario of mobile disconnection. Section 5 presents different protocols to improve the performance of TCP in mobile networks. In this section, we classify these schemes into three categories: link-layer protocols, end-to-end protocols and split connection protocols. We then present three link layer protocols, five end-to-end protocols, and four split connection protocols. We consider these protocols to be the basis for the research in the area of mobile TCP today. We discuss the problems that each protocol addresses, the main idea of each protocol and its advantages and disadvantages. Section 6 summarizes the features and problems handled by the protocols discussed in this paper. It also compares the different categories. Finally, section 7 presents the conclusion of this paper, which recommends the features that need to be satisfied in a standard mobile TCP.

2. BACKGROUND

2.1. Regular TCP

TCP is a connection-oriented protocol. It has been tuned to perform well for

networks composed of wired links and stationary hosts. It is a reliable transport protocol that adapts to the network requirements. It regulates the number of packets it sends by inflating and deflating a window. To do that the TCP sender uses the cumulative acknowledgements (ACKs) sent by the receiver. TCP also adapts to problems on the wired link. The main problem is the delay caused by packet losses due to congestion. The congestion control scheme in regular (Tahoe) TCP in [Jacobson 1988] implementation has three main parts:

1. Slow-start
2. Congestion avoidance
3. Fast Retransmit.

The Slow-start algorithm works as follows: the TCP sender starts with a congestion window (*cwnd*) that is equal to 1. For each received ACK, TCP exponentially increases the window until it is equal to a threshold (*ssthresh*), then it enters the congestion avoidance phase where it continues to increase its *cwnd* linearly until it reaches the receiver's maximum advertised window.

TCP continually measures how long acknowledgements take to return to determine which packets have reached the receiver, and provides reliability by retransmitting lost packets. For this purpose, it maintains a running average of this delay (round trip delay) and an estimate of the expected deviation from this average. If the current delay is longer than the average by more than four times the expected deviation (*timeout interval)*, TCP assumes that the packet was lost. TCP then retransmits the lost packet.

TCP also assumes that the packet was lost if the sender receives a number of duplicate acknowledgements (usually three). This is because the receiver acknowledges the highest *in-order* sequence number. If it receives *out-of-order* packets, it also generates acknowledgements for the same highest *in-order* sequence number and that results in duplicate acknowledgements. TCP then activates the Fast Retransmit algorithm. The Fast Retransmit algorithm assumes that the missing packet starts with the sequence number that is equal to the number acknowledged by the duplicate ACKs, and thus retransmits it.

TCP reacts to any packet lost by:

1. Dropping *ssthresh* into half the current window or 2 (whichever is larger) to reduce the amount of data.
2. Resetting its transmission (congestion) window size to 1, thus activating the slow-start algorithm to restrict the rate at which the window grows to previous levels.
3. Resetting the retransmission timer to a backoff interval that doubles with each consecutive timeout according to Karn's exponential timer backoff algorithm [Karn 1991]. This also results in the reduction of the traffic load at the intermediate links and therefore controls the congestion in the network.

Details of the TCP protocol are provided in Stevens [1994] and Comer [1991].

2.2. Mobility

A host is mobile if it is allowed to move freely around a local or wide area network. This allows users to access electronic data and services anywhere and anytime. A user should not be able to differentiate the operation and performance between a mobile and a fixed host (FH). Generally, mobile networks are composed of a wired backbone network and a wireless network. A cellular network infrastructure is used to connect mobile users to the Internet. The wireless network is geographically divided into cells, each of which contains a base station (BS) that provides a connection end-point for roaming mobiles. The base stations are connected to the wired infrastructure. They provide a gateway for communication between the wireless network and the backbone interconnect. As a *mobile host* (MH) travels between wireless cells, the task of forwarding data between the wired network and MH must be transferred to the new cell's BS. This is called *handoff*. Figure 1 illustrates a typical mobile network topology.

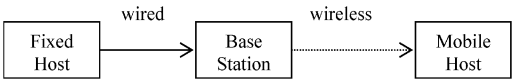

Fig. 1. Network topology.

It is necessary to implement TCP protocols for mobile environments that will provide mobile hosts with the same services that are offered to fixed hosts. Before implementing mobile TCP protocols, we need to know what problems mobile hosts and their wireless links introduce.

3. PROBLEMS WITH WIRELESS AND MOBILE NETWORKS

The successful use of mobile computing entails several challenges, among them:

1. *High bit error rates*: wireless links are susceptible to high bit error rates. This leads to the loss of data packets or acknowledgements.
2. *Disconnections*: disconnections can happen due to several reasons:
   - When a mobile moves from one cell into another, the new base station takes over—this is called handoff. During handoff, there is a brief disconnection period.
   - When a mobile host moves out of the reach of other transceivers.
   - When radio signals are blocked by buildings and other similar objects.
   - When a cell contains a large number of users and the bandwidth is not enough to satisfy their needs.
3. *Limited and variable bandwidth*: the available bandwidth depends on the location and the number of users in the cell.
4. *Cell size*: determining the suitable cell size requires careful design. Small cell sizes provide high-bandwidth connections, but result in small cell residence time, which leads to frequent disconnections.
5. *Power scarcity*: mobile computers are battery-operated and as a result the power resource is limited. Therefore, it will be helpful if the transmitting and receiving time is minimized.
6. *Dynamic network topology*: the topology of the network changes rapidly due to the movement of mobile hosts

To implement a mobile TCP, we have to consider the above problems. We also have to keep in mind the following:

1. *Non-congestion delay*: Long delays and lost packets in mobile environments are not necessarily due to congestion. Congestion control algorithms need to be used only in the event of genuine network congestion. Note that Jacobson [1988] assumes that packet loss due to damage in transit is rare, hence most probably packets get lost due to network congestion and not due to damage. In Jacobson [1988], it has been stated that the congestion control scheme is insensitive to damage loss. High loss rates due to damage of one packet per window (e.g., 12–15% for an 8 packet window) degrades TCP throughput by 60%. The additional degradation from the congestion avoidance window shrinking escalates the problem.
2. *Serial timeouts:* Frequent disconnections cause a condition called serial timeouts at the TCP sender. This happens when the retransmission timer at the sender is doubled with each unsuccessful retransmission attempt, in order to reduce the transmission rate. Thus, when the mobile is reconnected, TCP will take a long time to recover from such a reduction and data will not be transmitted for a period of time.
3. *Packet size variation*: packet size over wireless links is typically much smaller than the packet size over wired links. As a result, each packet on the wired networks gets fragmented when transmitted over the wireless link. Therefore, finding the optimal packet size on the wireless link is a key issue for performance.

4. IS REGULAR TCP SUITABLE FOR MOBILE HOSTS?

Regular TCP was not designed with mobile hosts in mind. Thus we cannot expect regular TCP to perform well in

Table I. Effect of BER on Performance of TCP

	BER = 10^{-5}	BER = 10^{-6}
Throughput (pkts/sec)	39.439	87.455
Success Probability	0.9892	0.999
Transfer time of 5000 pkts. in secs.	123.847	58.032

a mobile network. One reason for this (among many others, as we mentioned earlier) is that wired links have low bit error rates (BER), as opposed to wireless links that suffer from high bit error rates. Echhardt and Steenkiste [1996] indicate that BER of wireless links can reach 10^{-5}. When a packet is lost, regular TCP assumes that it is due to congestion. If regular TCP is used on a mobile network, it will encounter packet losses that may be unrelated to congestion. Nonetheless, these losses will trigger congestion control procedures at the fixed host. These procedures will result in significant reductions in throughput and unacceptable interactive delays for active connections, thus severely degrading performance.

To illustrate this we ran experiments to study the effect of BER on Tahoe TCP. We used the NS [NS] simulator provided by Lawrence Berkeley Labs. We used the network topology shown in Figure 1. This topology has been adopted in many experiments of TCP performance over wireless mobile links; see for example ElAarag and Bassiouni [2001] and Balakrishnan et al. [1997]. The simulation environment used by Fall and Floyd [1996] to compare Reno, New- Reno and SACK TCP on a wired network is also similar to that of Figure 1 but with BS replaced by a finite-buffer drop-tail gateway and MH replaced by a wired data receiver.

The wired link has a bandwidth of 1.5 Mb and delay of 10 ms, while the wireless link has a bandwidth of 0.8 Mb and delay of 100 ms. Bulk data are transferred via ftp from FH (the source) to MH (the sink). BS merely transfers the data between FH and MH. The wireless link is lossy and suffers from high bit error rate. Bit errors on the wireless links are usually bursty and frequent. The wireless link error model is characterized by two states: a good state and a bad state. In the good state, the wireless link does not suffer any losses, while, in the bad state the transmitted packets get corrupted. Both states are exponentially distributed with means α_g and α_b, respectively. The values of α_g and α_b are in packets and are subject to the following relationship:

$$\alpha_b = BER * packet\ size\ in\ bytes * 8 * \alpha_g \quad (1)$$

For ACK parameters, the packet size is replaced by ACK size in relation (1) above. Note that the choice of the exponential distribution allows for error bursts. This is attributed to the fact that the standard deviation of the exponential distribution is equal to its mean. There are several occasions where consecutive packets are lost. The TCP implementation used here is Tahoe TCP (or TCP-Tahoe). For TCP parameters, we set the window size to 50; ssthresh to 32. We chose packet sizes to be 1024 bytes and ACK sizes to be 40 bytes. We used DropTail queues at BS and MH. We chose the queue sizes to be equal to 50 to avoid the drop of packets due to buffering and to focus only on the loss of packets due to the wireless link errors.

We ran the simulation for 1000 seconds. The results are shown in Table I. Table I shows the performance of TCP with BER of 10^{-5} and 10^{-6}. In the former case for TCP packets (BS to MH) $\alpha_g = 100$, $\alpha_b = 8$ and for ACKs (MH to BS) $\alpha_g = 1000$, $\alpha_b = 3$. In the latter case, for TCP packets $\alpha_g = 1000$, $\alpha_b = 8$ and for ACKs $\alpha_g = 10000$, $\alpha_b = 3$. We used three performance measures. Throughput is the number of packets MH receives per second. Success probability is an indication of network utilization, and we calculated it as the amount of useful data received by MH to that sent by FH. Transfer time is the time needed to transfer 5000 packets. From Table I, we notice the negative impact on TCP performance caused by higher BER. Decreasing BER by one

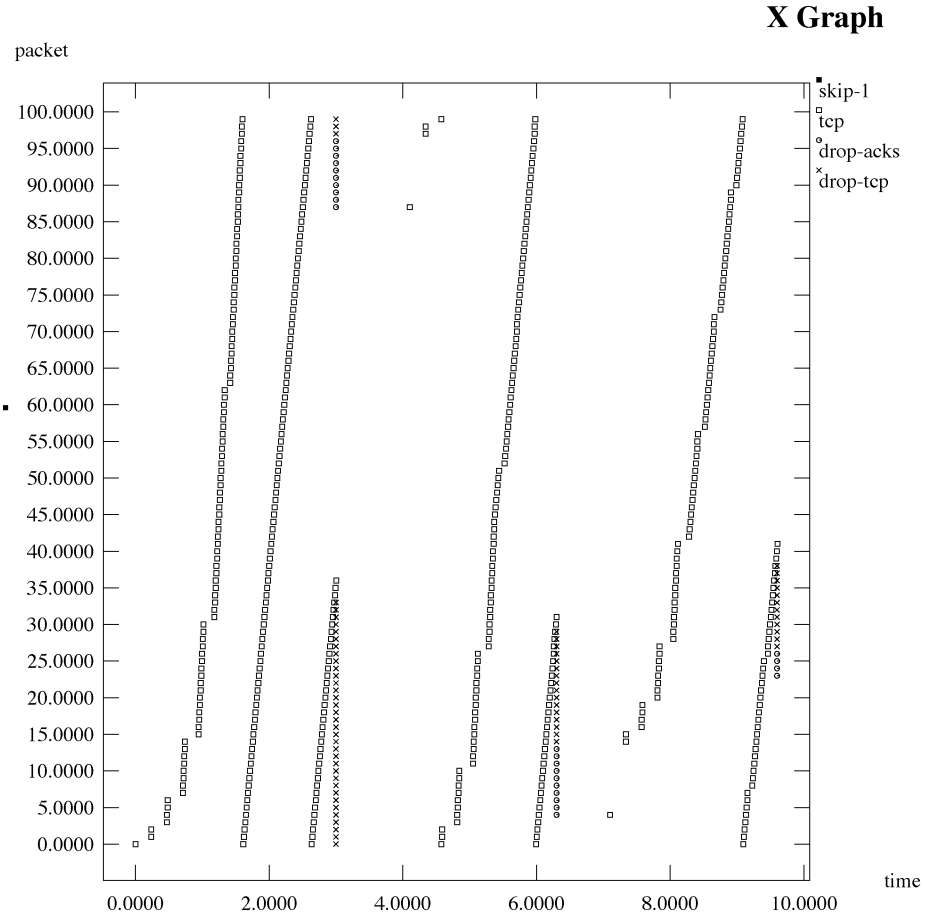

Fig. 2. Effect of mobile disconnection on TCP.

order of magnitude significantly enhanced the performance of TCP. We can notice from Table I that the TCP throughput at BER $= 10^{-6}$ is double that at BER $= 10^{-5}$. Higher BER causes the sender to have smaller windows, which results in low throughput. The success probability increased with lower BER. The lower the value of BER, the fewer packets/ACKs get corrupted and retransmitted, which results in better network utilization. With one order of magnitude less BER, TCP was able to send 5000 packets in almost half the time.

Figure 2 illustrates the effect of mobile disconnections on TCP. It demonstrates the non-congestion delay and timeout problem incurred by TCP in mobile environments. We ran a simulation using the topology and the parameters stated previously. This time we considered that the wireless network suffers only mobile disconnections. We traced the packets and ACKs between FH and MH. The x-axis shows the time of arrival and departure of the packets. The y-axis shows the packet number *mod* 100. The square on the graph represents the enqueuing of packets at FH. The circle represents lost ACKs. All other ACKs are assumed to be received by FH. The "x" on the graph represents the dropped packets. Dropped packets and ACKs are due to the disconnection of MH.

For illustration, Figure 2 shows a zoom on the first 10 seconds of the simulation. The simulation starts with the mobile connected. The sender runs the slow start algorithm with ssthresh $= 50$, which

increases the window exponentially from 1-50 sending packets 0-112. Then it enters the congestion avoidance algorithm, which increases the window linearly, sending packets 113–235. At time 3.0 sec, the connection to the mobile is lost. This causes ACKs 187–196 and packets 197–233 to be lost. The mobile is connected at time 3.3; nevertheless, the sender will not send any packets till time 4.105 sec, when it times out. The sender then assumes that the packets are lost due to congestion. It reacts by setting ssthresh to the value

max (min (cwnd, window), 2)

then enters slow start, shrinking its cwnd to 1. It resends packet 187. Although, the receiver successfully received packets 187–196, the sender had no way to know about it because the corresponding ACKs the receiver sent were lost. The receiver then acknowledges packet 196 (last one it received) causing the sender to increase its window to 2 and sends packets 197, 198. The sender continues to increase its window size exponentially until it reaches ssthresh (which is 25 now), then linearly until the mobile disconnects again at time 6.3. This causes ACKs 304–313 and packets 314–329 to be dropped.

When the mobile disconnects, a number of contiguous packets and ACKs are lost. The TCP sender had to wait until it timed out because there were no duplicate ACKs received. As a result, it did not execute the fast retransmission algorithm. Consequently, it had to recover by dropping its ssthresh to half current window size, its cwnd to 1, and entering slow start, causing a negative impact on its performance. Now if the mobile disconnects frequently, it will cause the serial timeout condition. According to Karn's algorithm, the retransmission timer exponentially backs off, resulting in long timeout periods. Hence, FH can stay in a long timeout period even after the reconnection of the mobile. In worst-case scenarios, we even found that the mobile can disconnect and reconnect several times while FH is still in a timeout period. Once FH times out and catches the mobile in a connection period, it takes the TCP sender a long time, using slow start, to resume the transmission rate to its level before the disconnection occurred. As a matter of fact, we found that the TCP sender never recovers its big congestion window size due to the continuous halving of the ssthresh. More experiments can be found in ElAarag [2000] and ElAarag and Bassiouni [2001].

5. PROTOCOLS TO IMPROVE THE PERFORMANCE OF TCP ON MOBILE NETWORKS

The increasing interest in mobile computers caused researchers like ElAarag and Bassiouni [2001], Chandran et al. [2001], Goff et al. [2000], Ludwig and Rathonyi [1999], Xylomenos and Polzos [1999], Chan et al. [1997], Wang and Tripathi [1998], Samaraweera and Fairhurst [1998] to be interested in the performance and the improvement of TCP in wireless environments.

The Performance Implications of Link Characteristics (PILC) working group has been working on related issues. Dawkins et al. [2001] have been studying the impact that heterogeneous links can have on the performance of end-to-end protocols like TCP. Inamura et al. [2001] suggested mitigations to improve the performance of TCP over 2.5G and 3G wireless networks.

Many other researchers were interested in the behavior of TCP in mobile ad hoc networks (MANETs). Ad hoc networks are multihop wireless networks consisting of a large number of radio-equipped nodes that may be as simple as autonomous sensors, to laptops mounted on vehicles or carried by people. These types of networks are useful in any situation where temporary network connectivity is needed, such as disaster relief, or in the battlefield. Although MANETs suffer from some of the problems of wireless mobile networks like the high BER, MANETS introduce a new set of problems. Node connectivity in MANETs tends to change over time. The rate at which the connectivity changes depends on the number of nodes, their velocity, transmission range, and obstacles in the environment that may

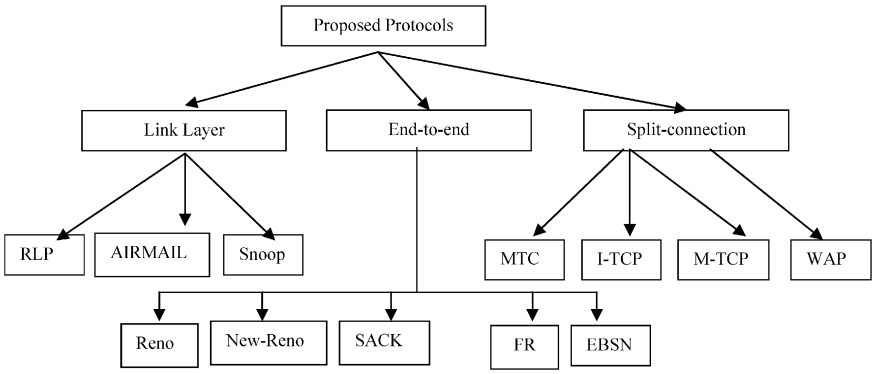

Fig. 3. Protocols to improve the performance of TCP on mobile networks.

create shadow. There are two effects of this change in node connectivity. First, nodes may need to recompute routes to some destinations. Second, it is likely that the ad hoc network may be temporarily partitioned due to node mobility. A protocol that improves the performance of TCP in MANETs has to take into consideration the route recomputations and network partitions problems. Such protocols are beyond the scope of this paper. Some pointers to the literature in this area include Liu and Singh [2001], who designed a thin layer between Internet protocol and standard TCP called ATCP to improve the performance of TCP over MANETs; Chandran et al. [2001], who proposed a feedback-based scheme where the source is sent a Route Failure Notification (RFN) when the route is disrupted. Jiang et al. [2001] studied the performance of TCP Reno, NewReno, SACK, and Vegas in MANETs Kim et al. [2000] proposed the TCP-BuS protocol where each node can buffer packets during the route disconnection and reestablishment. Additionally, they incorporate new measures to deal with the reliable transmission of important control messages.

In the rest of this section, we summarize some of the first protocols that have been proposed to improve the performance of TCP over wireless networks. We believe that these protocols lay the foundation for all subsequent research in this area. We can classify the different proposed protocols into three categories: link layer protocols, end-to-end protocols, and split-connection protocols. Figure 3 shows the protocols that we will discuss.

5.1. Link Layer Protocols

This approach tries to increase the quality of the lossy wireless link. Thus, it hides the characteristics of the wireless link from the transport layer and tries to solve the problem at the link layer. The intuition behind link layer protocols is that the problem is local, and hence should be solved locally. They use techniques like forward error correction (FEC) and automatic repeat request (ARQ). There have been several proposals for link-layer protocols. Radio Link Protocol (RLP) was proposed by Nanda et al. [1994], AIRMAIL by Ayanoglu et al. [1995] and Snoop by Balakrishnan et al. [1995]. In this section we shall discuss these three link layer protocols.

• *Radio Link Protocol (RLP)*:
Nanda et al. [1994] proposed a point-to-point automatic repeat request (ARQ) for radio channels. Their protocol exploits in-order delivery of link-layer packets over the radio link. In their basic protocol, a packet is retransmitted only if the transmitter is sure that it was not received. This makes the protocol very efficient in the sense that the receiver gets no more than one copy of any packet. Feedback

packets from the receiver together with sequence number of packets and a send sequence number at the transmitter are used to determine whether the packet was received or not. In the basic protocol, the channel may be forced to be idle during periods when all retransmissions have been completed. An enhanced version of their protocol preemptively retransmits unacknowledged packets during this time. This enhancement results in higher throughput and lower delays. Also, the basic protocol requires frequent full receiver state feedback, which is inefficient if user data is to be carried in the reverse direction. So, the enhancement protocol piggybacks partial receiver state in the reverse channel on user data packets.

• *AIRMAIL*:
Ayanoglu et al. in [1995] proposed a protocol named AIRMAIL (AsymmetrIc Reliable Mobile Access In Link-layer). The protocol is asymmetric in the sense that the base station is the side responsible for making decisions, whether it is transmitting or receiving. This is because the mobile host has limited battery power and smaller processing capability. Thus the asymmetry places the bulk of the intelligence at the base station with the goal of reducing the processing load at the mobile. The reliability is established by using a combination of automatic repeat request (ARQ) and forward error correction (FEC). The protocol requires the base station to send periodic status messages, while allowing the mobile to combine several acknowledgements into a single one to conserve power. The mobile acknowledgement, unlike the base station, is event driven to reduce the processing load on the mobile. There are three possible levels of FEC: bit-level which is achieved in hardware at the physical layer, byte-level which is done by a per-packet cyclic redundancy check (CRC), and packet-level which is done by allocating some packets for correction which are used for recovery of lost packets without retransmission. Ayanoglu et al. showed that a different level of FEC is needed depending on the characteristics of the mobile channel. Therefore, they designed an algorithm that adaptively uses the three levels of channel coding. Thus the bandwidth expansion due to FEC is minimized. The authors also handle handoff by window management and state transfer.

• *The Snoop Protocol*:
Balakrishnan et al. [1995] aimed to achieve the goal of improving TCP performance without changing the existing TCP implementation in the fixed network. They introduced a module, called Snoop, at the base station that monitors every packet that passes through the connection in either direction. The Snoop module maintains a cache of TCP packets sent from the fixed host that have not yet been acknowledged by the mobile host. A packet loss is detected either by the arrival of duplicate acknowledgment or by a local timeout. To implement the local timeout, the module has its own retransmission timer. The Snoop module retransmits the lost packet if it has it in the cache. Thus, the base station hides the packet loss from the fixed host, hence avoiding its invocation of an unnecessary congestion control mechanism. The authors improved the performance of the Snoop module [Balakrishnan et al. 1996] by adding selective retransmissions from the base station to the mobile host.

The authors also handled the case when the packet losses are from the mobile host to the base station (which is more likely [Balakrishnan et al. 1999]). In order to have the sender identify whether the loss happened on the wireless link or on the wired link due to congestion, they kept track of the packets that are lost at the base station. The base station then generates negative acknowledgements for those packets back to the mobile. This is very useful for bursty errors; when more than one packet is lost in the same window.

Balakrishnan et al. also used a routing protocol to handle handoff. Basically, the base station anticipates that a handoff is going to occur. The neighboring base stations form a multicast group; they receive all packets destined to the mobile. A Snoop module is started at each base

station of the multicast group. It starts with an empty cache and slowly builds up the cache.

• *Comparison of link layer protocols*:
Link layer protocols focus on the problems that arise from lossy wireless links by reducing their high bit error rate. While both RLP and AIRMAIL use ARQ where the mobile host requests the retransmission of corrupted messages, RLP assumes circuit-mode data, while AIRMAIL assumes packet-mode data. In RLP, unacknowledged packets are preemptively retransmitted during the periods when the channel is idle. That is, when the unused circuit bandwidth would otherwise be wasted. Furthermore, AIRMAIL deals with handoff and uses adaptive forward error correction and the concept of asymmetry. It also takes into consideration power scarcity and small processing capability of the mobile. Snoop alleviates the problems caused by high-bit error rates by caching unacknowledged packets at the base station and locally retransmitting them.

The authors of the Snoop protocol noticed that the throughput of Snoop's handoff protocol will be poor until the caches are built up at the neighboring base stations. This will cause serious performance degradation in the presence of frequent handoffs. To solve this problem the authors had the mobile host receive data from the old base station until the new base stations build up their caches.

5.2. End-to-End Protocols

In the end-to-end approach, the TCP sender attempts to handle the losses in a way that improves the performance over regular TCP. Therefore this category maintains the end-to-end semantics of TCP. In this section, we discuss the following end-to-end protocols: Reno, New-Reno, SACK, Fast Retransmission (FR) scheme and Explicit Bad State Notification (EBSN) mechanism. Reno, New-Reno and SACK are different TCP implementations that were initially designed to improve the performance of regular TCP in wired networks. Some researchers for example Balakrishnan et al. [1997] considered them for wireless networks. Allman et al. [1999] considered them for networks with satellite channels, which share some of the problems of networks with mobile wireless links. If these implementations were to be used on wireless networks, they will have the advantage that no recompilation of new software will be needed at the fixed hosts. Thus they can overcome the disadvantage of end-to-end protocols, which we will discuss later. Nevertheless, their improvement of performance over regular TCP in wireless networks is expected to be limited, since they were not designed specifically to overcome the problems of the wireless networks. Comparisons of Reno, New-Reno and SACK in wired networks can be found in Fall and Floyd [1996], and in wireless networks can be found in ElAarag and Bassiouni [2001] and ElAarag [2000].

• *Reno*
Reno TCP, [Jacobson 1990; Stevens 1997] is like regular (Tahoe) TCP except it includes *fast recovery*. The TCP sender enters fast recovery if it times out or receives three duplicate acknowledgments. The sender then retransmits the lost packet and reduces *ssthresh* by half. Unlike TCP Tahoe, the sender then does not enter slow start. It reduces the value of the congestion window (*cwnd*) by half, then increments it by one for each duplicate acknowledgement received. When a "new" ACK is received, the sender exits fast recovery, sets *cwnd* to *ssthresh* and enters the congestion avoidance phase where it increases the window linearly. A "new" ACK is an ACK with a value higher than the highest seen so far — a non-duplicate ACK.

• *New-Reno*
This implementation, proposed by Hoe [1996], is basically a modification over Reno TCP. New-Reno helps improve the performance of Reno when multiple packets are lost from the same window. To achieve that, it modifies the action taken upon receiving a "new" ACK. Unlike Reno, New-Reno does not exit fast recovery

unless all data transmitted at the start of the fast retransmission phase have been acknowledged. Thus, it only exits fast recovery when it receives an ACK for the highest sequence number sent. Thus, a "new partial" ACK (an ACK that represents some, but not all, of the outstanding packets) is used as indication that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. When multiple packets are lost in the same window, Reno recovers by waiting for serial retransmission timeout. This is because in Reno, the reception of a new partial ACK causes the sender to exit fast recovery by "deflating" the window.

• *SACK*
SACK was defined in RFC 2018 by Mathis et al. [1996], and later extended in RFC 2883 by Floyd et al. [2000]. SACK TCP further improves TCP performance by allowing the sender to retransmit packets based on the selective ACKs provided by the receiver. The implementation constitutes a SACK field that contains a number of SACK blocks. The first block reports the most recently received packets. The additional blocks repeat the most recently reported SACK blocks.

The SACK uses the basic congestion control algorithms and uses retransmit timeouts as a last option for recovery. The main difference is the way it handles the loss of multiple packets from the same window, in fast recovery. Like Reno, SACK enters fast recovery upon receiving duplicate ACKs. It then retransmits a packet and cuts its congestion window in half. In addition to that, SACK has a new variable called the *pipe*, and a data structure called the *scoreboard*. The *pipe* is incremented when the sender sends a new or a retransmitted packet. It is decremented when the receiver receives a new packet. This is indicated when the sender receives a duplicate ACK with a SACK option. The *scoreboard* stores ACKs from previous SACK options, allowing the sender to retransmit packets that are implied to be missing at the receiver. Like New-Reno, the sender exits fast recovery when all the outstanding data are acknowledged.

• *Fast retransmission scheme*:
Caceres and Iftode [1995] were among the first to address the impact of mobility and wireless networks on the performance of TCP. They used the fast retransmission scheme, employed by current implementations of TCP, to provide smooth handoffs on networks that lose packets during handoff. They made minimal changes to the mobile IP software on the mobile host to signal the completion of the handoff. Consequently, the TCP software on the mobile host forwards the signal to the fixed host. The TCP on the fixed host then invokes the retransmission scheme. This forces the fixed host to reduce the congestion window to half and retransmit one segment immediately. Experiments showed that this scheme causes connection to resume communication after 50 ms as opposed to 650 ms using regular TCP. In this scheme, if triplicate acknowledgements are used to invoke the fast retransmission procedure, then no changes in the TCP software is required. Otherwise, minimal changes to TCP are needed if a specially marked TCP acknowledgement packet containing the sequence number of the last data packet successfully received by the mobile host is used.

Caceres and Iftode also recommended the use of small cells with little or no overlap between them. They relied on the fact that little or no overlap between small cells provides high aggregate bandwidth, support low-poweredmobile transceivers, and provide accurate location information.

• *EBSN*:
Bakshi et al. [1997] studied the effect of local error recovery and explicit feedback by the base station. Their idea is that although local recovery by the base station using link layer protocols is found to improve performance, timeouts can still occur at the fixed host, causing redundant packet retransmissions. Their experiments showed that explicit feedback from the base station to the fixed host can completely eliminate the possibility of timeouts occurring at the fixed host, while

the wireless link is in a bad state. When a wireless link is in a bad state, little data is able to reach the mobile, causing packets from the fixed host to be queued at the base station. It would appear that using an existing feedback mechanism, *like Explicit Congestion Notification ECN* would solve the problem. Sending an ECN from the base station to the fixed host would decrease the flow of new packets to the base station, and thus prevent unnecessary timeouts. But it does not prevent timeouts of packets *already* on the network. The solution to this problem is to send an *Explicit Bad State Notification* (*EBSN*) that would reset the timeout at the fixed host during local recovery. The EBSN will totally eliminate timeouts even for packets that had already been put on the network before the wireless link entered the bad state, and at the same time not cause the fixed host to decrease its congestion window. EBSN was found to provide up to 100% performance improvement over regular TCP in wide-area networks, and up to 50% in local area networks, which is very close to the theoretical maximum.

Bakshi et al. also investigated the effect of packet size variation. They showed that choosing an optimal packet size could improve performance by up to 30%. They found that the optimal packet size depends on the error conditions of the wireless link. A fixed table could be maintained at the base station that maps a particular wireless link error characteristic to its optimal packet size.

• *Comparison of end-to-end protocols*:
The advantage of Reno, New-Reno, SACK and fast retransmit scheme is that they do not involve base stations. Therefore, they are independent on any mobile networking environment and can be used over an internetwork. Reno, New-Reno and SACK, since they were not designed for wireless mobile networks, do not handle handoff problems. The fast retransmit scheme focused on handoff problems and did not consider other problems mentioned above, like bit error rates of wireless links. Also, their scheme is not adequate to recover from multiple losses per window resulting from long disconnections. This is because the transmitting host will still perform *slow start*, shrinking its window to 1, and retransmitting only one packet in its window. Thus, the effective throughput is limited. In fact, the behavior of this protocol in this case is no different from regular TCP. Similarly, if disconnections are frequent or the wireless environment is lossy, this scheme will cause the sender's congestion window to be repeatedly halved, which is again no different from regular TCP. Further, initiating the fast retransmit procedure on the fixed host, when it is congested, can worsen the congestion.

The advantage of the EBSN scheme is that it eliminates timeouts at the fixed host during local recovery. This will provide great performance improvement, especially in wireless links with high bit error rates. The major disadvantage is that handoff is not considered.

5.3. Split-Connection Protocols

The main idea behind the split connection approaches is to isolate mobility and wireless related problems from the existing network protocols. This is done by splitting the TCP connection between the mobile host and the fixed host into two separate connections: a wired connection between the fixed host and the base station, and a wireless connection between the base station and the mobile host. In this way the wired connection does not need any changes in existing software on the fixed hosts, and the wireless connection can use a mobile protocol specialized to provide better performance. In what follows, we briefly discuss the following split-connection protocols: MTCP, proposed by Yavatkar and Bhagawat [1995], I-TCP, by Bakre and Badrinath [1997], M-TCP, by Brown and Singh [1997] and WAP, by the Wireless Application Forum.

• *MTCP*:
The basic idea behind MTCP is to protect the long connection over the wired network from the impact of the erratic

behavior of the short connection over the wireless link and also recover quickly from errors over the wireless link. Therefore, Yavatkar and Bhagawat [1995] introduced a session layer protocol called MHP (Mobile Host Protocol), at the base station and the mobile host. The session layer is above TCP and below the socket. They designed this layer such that it compensates for the unreliability and unpredictability of the wireless link using its knowledge about host migration and wireless links characteristics. They proposed two implementations for the session layer. One uses TCP over the wireless link, and the second uses a selective repeat protocol (SRP) over the wireless link. SRP is designed to recover quickly from high and bursty packet losses. The receiver returns a selective ACK (SACK) when an out of sequence packet is received specifying the missing packet. Then, the sender in turn retransmits the missing packet. SRP also can recover more than one packet in one round trip time.

• *I-TCP*:
The same idea was used by Bakre and Badrinath [1997]. If a mobile host need to communicate to a fixed host using I-TCP, a request is sent to the current base station to open a TCP connection with the fixed host on behalf of the mobile host. The mobile host communicates with its base station on a separate connection using a variation of TCP that is tuned for wireless links and is aware of mobility. The I-TCP software consists of two components—one on the mobile host and the other on the base station. The component on the mobile host consists of special library calls that are similar in functionality and interface to the socket calls made by an application using regular TCP. This library makes the communication needed with the base station, transparent to the mobile host. The second component consists of a user level Unix process pumping data from one part of the connection into the other. It also handles handoff support for I-TCP.

• *M-TCP*:
Brown and Singh [1997] focused on the effects of frequent long disconnections and low variable bandwidth on TCP throughput. They also considered the power scarcity of the mobile devices. Thus, they designed a protocol that dynamically assigns a fixed amount of bandwidth to each mobile node based on their changing needs. It performs local error recovery to solve problems resulting from the lossy wireless link. It reduces the power consumption at the mobile node by ensuring that the number of duplicate packets are kept small. The protocol also ensures efficient handoff and deals with the problems caused by long or frequent disconnections. They used a three-layer hierarchical architecture. At the lowest level, there are the mobile nodes and the base station in each cell. Several base stations are controlled by a machine called the supervisor host at a second level of the hierarchy. The supervisor hosts are connected to the wired network at the top level of the hierarchy. The supervisor host handles the routing and maintains connections for mobile users. This protocol is a split connection protocol where the connection between the fixed host and the mobile host is split at the supervisor host. Regular TCP is used on the fixed network (between the fixed host and the supervisor host), while a special version of TCP is used over the wireless link. The TCP client at the supervisor host is called SH-TCP while that on the mobile host is called M-TCP. When the SH-TCP receives data from the sender, it passes it to the M-TCP client, which replies by an acknowledgment. The SH-TCP passes this acknowledgement to the sender. To ensure that the sender does not go into congestion control when the ACK does not arrive because the mobile host temporarily got disconnected, the SH-TCP does not forward the ACK of the last byte to the sender until it knows that the mobile host has disconnected. This forces TCP to go into persist mode by setting the window size to zero. Therefore, TCP will not suffer from retransmit timeouts, nor will it close its congestion window. When the mobile host reconnects, the TCP sender is ready to transmit at full speed. If the mobile host did not disconnect but has little available bandwidth. The

SH-TCP shrinks the sender's window before it exponentially backs off its retransmission timer. At the M-TCP client, when the mobile host disconnects, it freezes all its timers to ensure that the congestion control is not invoked. When it reconnects, it unfreezes all the timers and resumes normal operation, as if it did not lose any data during disconnection.

• *WAP*:
In this model, Internet applications interact with an application gateway to reach the wireless world, and the application gateway uses a wireless transport protocol, and potentially a modified version of the application data to interact with the mobile wireless device. The most common approach is extension of the World Wide Web client into the mobile wireless device, using some form of proxy server at the boundary of the wireless network and the Internet. This is the approach adopted by the *Wireless Application Protocol Forum* (WAP) [WAP]. WAP is an application environment and set of communication protocols for wireless devices, designed to enable manufacturer-, vendor-, and technology-independent access to the Internet and advanced telephony services. WAP is designed in a layered fashion in order to be extensible, flexible, and scalable. With the open system interconnection (OSI) model in mind, the WAP stack is basically divided into five layers. They are:

- Application Layer: Wireless application environment (WAE)
- Session Layer: Wireless session protocol (WSP)
- Transaction Layer: Wireless transaction protocol (WTP)
- Security Layer: Wireless transport layer security (WTLS)
- Transport layer: Wireless datagram protocol (WDP)

The WAP programming model is shown in Figure 4. The gateway acts as a proxy server to a mobile client and translates requests from WAP protocol stacks to protocol stacks employed by the information server. Encoders translate the content coming from the server into compact formats to reduce the size of data over the wireless network.

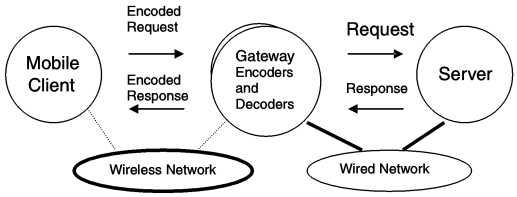

Fig. 4. WAP basic architecture.

• *Comparison of split-connection protocols*:
The difference between MTCP and I-TCP is that the latter concentrates on the effect of motion and handoff while the former concentrates on the effect of small packet sizes and lossy links in the absence of motion. While MTCP uses the SRP as its specialized transport protocol over the wireless link, as described earlier, I-TCP uses TCP with some modification that handles the problems caused by the mobility of the mobile host. When a move by the mobile host is detected, the retransmission timers over the wireless part are cleared, consequently, immediately entering *slow start*. This allows I-TCP to come back to normal speed faster than the regular TCP congestion control mechanism. A similar action is performed when a mobile host reconnects from a disconnection. This prevents disconnections from causing long pauses. Resetting the retransmission timers cannot have any negative effect on fixed network congestion when the mobile host switches cells.

MTCP and I-TCP assume one or two network hops between the base station and the mobile host. Thus the round trip time is very small. Therefore, in case of MTCP/SRP this means that the congestion window will quickly build up to maximum again after being reduced to one.

The advantage of the M-TCP hierarchical architecture is that the mobile node could move from one cell into another but can still remain within the domain of the same supervisor host. Therefore, no transfer of state of control between the two base stations is required, and thus no actual handoff is required for a long period of time, when the mobile host leaves the

Table II. Summary of Proposed Protocols

	RLP	Airmail	Snoop	FR	EBSN	MTCP	I-TCP	M-TCP
High BER	√	√	√		√	√	√	
Bursty error		√	√		√	√		
handoff		√	√	√			√	√
Long disconnections							√	√
Frequent disconnections							√	√
bandwidth	√	√						√
Cell size				√				
Power scarcity		√			√			√
Serial timeouts					√	√	√	√
Packet size variation					√	√		
End-to-end TCP semantics	√	√	√	√	√			√
Compatibility	√	√	√			√	√	√

domain of the supervisor host. This makes it easier to manage the mobile node's connections as the state is transferred infrequently: O ($\sqrt{N}$) per unit time, where N is the number of cells controlled by one supervisor host. As opposed to O (N) per unit time in the case where the base station manages the transport connections, and therefore handoff occurs every time the mobile leaves the cell.

This protocol, unlike other split connection protocols such as I-TCP and MTCP, ensures the end-to-end semantics of TCP because the supervisor host does not acknowledge any data unless the mobile host does. In MTCP and I-TCP, acknowledgements to packets can reach the source even before the packets actually reach the mobile host. This will cause a serious problem if the mobile host gets disconnected before receiving the packets. While this is acceptable for some applications, such as ftp, that use application layer acknowledgements in addition to end-to-end acknowledgements, it is not well suited for applications like telnet, which depend on end-to-end transport layer acknowledgements.

In M-TCP, only a small amount of state is transferred during handoff, unlike MTCP and I-TCP, which require transfer of large buffers and thus suffer from high handoff cost in case of frequent disconnections. On the other hand, M-TCP handles long disconnections efficiently, while MTCP and I-TCP may run out of buffers.

The disadvantage of the M-TCP architecture is the complexity of the supervisor host, as it has to handle connections for all the mobile nodes in all cells in its domain. This protocol assumes that a link layer protocol is used to ensure that the bit error rate at the transport layer is small. Therefore, it was designed to handle disconnections and thus is best used in environments with low bit error rate.

WAP is quite different than the other three split protocols considered in this section. In the WAP approach, the proxy and the server needn't be located on separate machines. A server may include WAP proxy capabilities. This might be suitable for configurations where you need to guarantee end-to-end transaction security. It also has a complete different stack of protocols. The WAP approach basically extends the World Wide Web to a mobile handset. The other three approaches allow mobile devices to access a complete range of Internet-based services. In these approaches, the intent is to allow the mobile wireless device to function as any other Internet-connected device.

6. COMPARISON OF CATEGORIES

In this section, we will make a comparison on the category level. Table II summarizes the problems handled by the protocols discussed in this paper, and their features. However, we did not include Reno, New-Reno or SACK in this table, since they were basically designed for wired network, hence they only satisfy the end-to-end TCP semantics in the above table. Nevertheless, they are included in the

discussion below. From this table and our previous discussion, we can conclude the following:

The main advantages of link layer protocols are that they maintain the end-to-end semantics of TCP. For example, in the case of Snoop, it simply forwards the acknowledgements generated by the mobile host to the fixed host, thus avoiding the unnecessary fast retransmissions and congestion control invocations by the fixed host, since the base station handles the problem locally. They increase the reliability of the wireless link by reducing the bit error rate by one or two orders of magnitude. Therefore, link layer protocols perform real well in environments with high bit error rates. Moreover, they fit naturally into the layered structure of network protocols.

Unfortunately, link layer protocols do not help in the case of long or frequent disconnections. This is because, if the mobile is disconnected, it will not receive the retransmitted packets. Therefore, a good link-layer protocol is necessary, but not sufficient, to reduce the loss rate due to bit error. We need to ensure that we take care of disconnection problems in another layer of the stack. Moreover, it should be tightly coupled with higher-layer protocols. Balakrishnan et al. [1996] proved that simple link-layer protocols could adversely impact TCP performance and do not completely shield the sender from wireless losses for two reasons:

1. Incompatible setting of the timers at the two layers.
2. Effect of link-layer protocol on the TCP fast retransmission mechanism.

End-to end protocols suffer from the major drawback that they require modification to TCP at the fixed host code and thus they do not satisfy compatibility. This is very undesirable because it will lead to recompilation and relinking of existing applications on fixed hosts. Nevertheless, they satisfy the end-to-end semantics of TCP.

The major advantage of split-connection protocols is that they provide backward compatibility with the existing wired network protocols, thus do not require any modifications at the fixed host for accommodating mobile hosts. They shield the mobility and wireless problems from the fixed host. In addition, they can handle disconnections efficiently. Their disadvantage is that they might not provide the end-to-end semantics of TCP, which can cause major problems if not handled properly.

7. CONCLUSION

One of the most challenging and interesting recent trends in computer networks is the integration of mobile communications. With the increasing importance of host mobility, and the popularity of TCP/IP on fixed networks, we are in need of a reliable mobile TCP/IP protocol to be used in mobile networks. Mobile IP has been standardized. It solved the operability problem of integrating mobile networks into the Internet. Now, we need to have a standard mobile TCP. This mobile TCP protocol should be designed with the problems of mobility and wireless connections in mind. In this paper, we illustrated, using simulation results, the problems caused by the mobility of hosts and their wireless links. We demonstrated their negative effect on the performance of regular TCP. We presented a survey of different protocols that improve the performance of TCP on mobile networks. From our simulations and the discussion of the advantages and disadvantages of each protocol, we conclude with the features that need to be satisfied in a standard mobile TCP:

1. Avoid erroneously triggering congestion control mechanisms on the fixed host.
2. Avoid the serial timeout problem on the fixed host.
3. Be reliable, by solving the problems arising from the lossy wireless links and their bursty high BER.
4. Can efficiently deal with handoff.

5. Can handle frequent and long disconnections of the mobile host.
6. Take into consideration the limited bandwidth and power scarcity of mobile hosts.
7. Use a dynamic packet size depending on the dynamic bandwidth available for mobile hosts.
8. Provide end-to-end semantics of TCP.
9. Preferably provide compatibility; that is, do not require any change in software on the fixed hosts.

In order that the mobile TCP protocol performs efficiently, a link layer protocol suitable for mobile networks must be used. The use of link layer protocols to provide an acceptable error performance over the wireless connection is now a standard industrial practice While link layer protocols can efficiently provide reliability, transport layer protocols can be designed to efficiently deal with handoff and disconnections.

ACKNOWLEDGMENT

The author is grateful to the anonymous reviewers for their comments and suggestions that helped improve the quality of this paper.

REFERENCES

ALLMAN, M., GLOVER, D., AND SANCHEZ, L. 1999. Enhancing TCP Over Satellite Channels using Standard Mechanisms, RFC 2488.

AYANOGLU, E. PAUL S., LAPORTA, T. F., SABNANI, K., AND GITLIN, R. 1995. *AIRMAIL*: A link–layer protocol for wireless networks. *Wireless Networks 1*, 47–60.

BAKRE, A. AND BARDINATH, B. R. 1997. Implementation and Performance Evaluation of Indirect TCP. *IEEE Transactions on Computers 46*, 3, 260–278.

BAKSHI, B., KRISHNA, P., VAIDYA, N. H., AND PRADHAN, D. K. 1997. Improving performance of TCP over wireless networks. *17th International Conference on Distributed Computing Systems*, pp. 365–373.

BALAKRISHNAN, H., PADMANABHAN, V., AND KATZ, R. 1999. Effects of asymmetry on TCP performance. *Mobile Networks and Applications 4*, 3, 219–241.

BALAKRISHNAN, H., PADMANABHAN, V., SESHAN, S., AND KATZ, R. 1997. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking 5*, 6, 756–769.

BALAKRISHNAN H., SESHAN S. AND KATZ R. 1995. Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks 1*, 4, 469–481.

BALAKRISHNAN, H., SESHAN, S., AND KATZ R. 1996. A comparison of mechanisms for improving TCP performance over wireless links. *ACM SIGCOMM'96* Palo Alto, CA, Aug. 1996, pp. 256–269.

BROWN, K. AND SINGH, S. 1997. M-TCP: TCP for mobile cellular networks. *Computer Communication Review*, July, 19–43.

BRUYERON, R., HEMON, B., AND ZHANG, L. 1998. Experimentations with TCP selective acknowledgment. *ACM SIGCOMM Computer Communication Review 28*, 2, 54–77.

CACERES, R. AND IFTODE, L. 1995. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications 13*, 5, 850–857.

CHAN A., TSANG D., AND GUPTA S. 1997. Impacts of handoff on TCP performance in mobile wireless computing. *International Conference on Personal Wireless Communications*, pp. 184–188.

CHANDRAN, K., RAGHUNATHAN, S., VENKATESAN, S., AND PRAKASH, R. 2001. A feedback-based scheme for improving TCP performance in ad-hoc wireless networks. *IEEE Personal Communications 8*, 1, 34–39.

COMER, D. 1991. *Internetworking with TCP/IP*, vols. 1–3, Prentice Hall, Englewood Cliff, NJ.

DAWKINS MONTEGRO, G., KOJO, M., MAGRET, V., AND VAIDYA, N. 2001. End-to-end performance implication of links with errors, IETF RFC 3155, Available at: http://www.ietf.org/rfc/rfc3155.txt.

ECKHARDT, D. AND STEENKISTE, P. 1996. Measurement and analysis of the error characteristics of an in-building wireless network. *Proceedings ACM SIGCOMM*, Oct., 243–254.

ELAARAG, H. AND BASSIOUNI, M. 2001. Performance evaluation of TCP connections in ideal and non-ideal network environments. *Computer Communications Journal*, *24*, 18, 1769–1779.

ELAARAG, H. 2000. Transport control protocols for the success of the Internet, Ph.D. dissertation, University of Central Florida.

FABER, T, TOUCH, J., AND YUE, W. 1999. The TIME-WAIT state in TCP and its effect on busy servers. *Proceedings of INFOCOM*, pp. 1573–1583.

FALL, K. AND FLOYD, S. 1996. Simulation based comparisons of Tahoe, Reno, and SACK TCP. *ACM Computer Communication Review 26*, 3, 5–21.

FLOYD, S., MAHDAVI, J., MATHIS, M., AND PODOLSKY, M. 2000. An extension to the selective acknowledgement (SACK) option for TCP, RFC 2883.

GOFF, T., MORONSKI, J., PHATAK, D. S., AND GUPTA, V. 2000. Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, IEEE3, pp. 1537–1545.

HASEGAWA, G., MURATA, M., AND MIYAHARA, M. 1999. Fairness and stability of congestion control mechanisms of TCP. *Proceedings of IEEE INFOCOM*, pp. 1329–1336.

HEIDEMANN, J., 1997. Performance interactions between P-HTTP and TCP implementations. ACM *SIGCOMM Computer Communication Review, 27*, 2, 64–73.

HOE, J. C. 1996. Improving the start-up behavior of a congestion control scheme for TCP. *ACM SIGCOMM*, pp. 270–280.

INAMURA, H., MONTENEGRO, G., LUDWIG, R., GURTOV, A., AND KHAFISOV, F. 2001. TCP over 2.5G and 3G wireless networks, IETF, Internet draft. Available at: http://www.ietf.org/internet-drafts/draft-ietf-pilc-2.5g3g-09.txt

JACOBSON, V. 1988. Congestion avoidance and control. *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 314–329. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z for an updated version.

JACOBSON, V. 1990. Modified TCP congestion avoidance algorithm. Technical Report 30, April 1990. Available at ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

JIANG H., CHENG, S., AND CHEN, X. 2001. TCP Reno and Vegas in wireless ad hoc networks. *IEEE International Conference on Communications 1* (IEEE cat n 1CH37240), pp. 132–136.

KARN, P. AND PARTRIDGE, C. Nov. 1991. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems 9*, 4, 364–373.

KIM D., TOH, C.-K., AND CHOI, Y. 2000. TCP-BuS: Improving TCP performance in wireless ad hoc networks. *IEEE International Conference on Communications 3*, pp. 1707–1713.

LIU J. AND SINGH S. 2001. ATCP: TCP for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications 19*, 7, 1300–1315.

LUDWIG, R. AND RATHONYI, B. 1999. Link layer enhancements for TCP/IP over GSM. *Proceedings of IEEE INFOCOM*, pp. 415–422.

MATHIS, M., MAHDAVI, J., FLOYD, S., AND ROMANOW, A. 1996. TCP selective acknowledgement options October IETF, RFC 2018, January.

MATHIS, M., SEMKE, J., AND MAHDAVI, J. July 1997. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review 27*, 3.

MO, J., LA R., ANANTHARAM V., AND WALRAND J. 1999. Analysis and comparison of TCP Reno and Vegas. *Proceedings of IEEE INFOCOM*, pp. 1556–1563.

NANDA, S., EJZAK, R. AND DOSHI, B. 1994. A retransmission scheme for circuit-mode data on wireless links. *IEEE Journal on Selected Areas in Communications 12*, 8, 1338–1352.

NS THE UCB/LBNL/VINT Network Simulator (NS), URL http://www-mash.cs.berkeley.edu/ns/.

PAXSON, V. August 1994. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking 2*, 4, 316–336.

PAZOS, C., AGRELO, J., AND GERLA, M. 1999. Using back-pressure to improve TCP performance with many flows. *Proceedings of IEEE INFOCOM*, pp. 431–438.

RHEE, I., BALAGURUF, N., AND ROUSKASF, G. 1999. MTCP: Scalable TCP-like congestion control for reliable multicast. *Proceedings of IEEE INFOCOM*, pp. 1265–1273.

SAMARAWEERA, N. AND FAIRHURST, G. April 1998. Reinforcement of TCP error recovery for wireless communication. ACM *SIGCOMM Computer Communication Review 28*, 2, 30–38.

SINHA P., VENKITARAMAN, N., SIVAKUMAR, R., AND BHARGHAVAN, V. 1999. WTCP: A reliable transport protocol for wireless wide-area networks. *Proceedings of ACM Mobicom'99*, Seattle, WA, pp. 231–241.

STEVENS, W. R., 1994. *TCP/IP Illustrated*, vol. 1, Addison-Wesley.

STEVENS, W. R. January 1997. TCP Slow start, congestion avoidance, fast retransmission, and fast recovery algorithms, IETF RFC, January 2001.

WANG, K., AND TRIPATHI, S. 1998. Mobile-end transport protocol: an alternative to TCP/IP over wireless links. *Proceedings of IEEE INFOCOM, 3*, pp. 1046–1053.

WAP, Wireless Application Protocol Forum, available at http://www.wapforum.org.

XYLOMENOS, G. AND POLZOS, G. 1999. TCP and UDP performance over a wireless LAN. *Proceedings of IEEE INFOCOM*, pp. 439–446.

YANG, X. 1999. A model for window based flow control in packet-switched networks. *Proceedings of IEEE INFOCOM*, pp. 423–430.

YAVATKAR, R. AND BHAGAWAT, N. 1995. Improving end-to-end performance of TCP over mobile internetworks. *Proceedings of the Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, pp. 146–152.

Received June 2001; revised December 2001, April 2002; accepted April 2002