Improving End-to-End Performance of TCP Using Link-Layer Retransmissions over Mobile Internetworks

Jackson W.K. Wong and Victor C.M. Leung

Department of Electrical and Computer Engineering
The University of British Columbia
2356 Main Mall, Vancouver, B.C., Canada, V6T 1Z4
E-Mail: vleung@ece.ubc.ca

ABSTRACT **TCP does not perform well in networks with high packet error rates, such as those with wireless links, since TCP assumes network congestion to be the major cause for packet losses. Wireless losses make TCP unnecessarily initiate its congestion control mechanism which results in poor performance in the form of low throughput and high interactive delay. We investigate, through computer simulations, the end-to-end effects of link-layer retransmissions on TCP Reno over a low-data-rate wireless link. Our results show that, by using the more effective selective-reject ARQ at the link layer, the problem of competitive retransmissions between TCP and link layer is much less serious than previously reported. We show that a non-sequencing link layer in combination with fragmentations of datagrams at the base stations and mobile hosts can be employed without significantly degrading TCP performance. We propose link-layer modifications for best-effort retransmissions to reduce possible adverse effect of link-layer resets on TCP.**

I. INTRODUCTION

TCP (Transmission Control Protocol) [1] was originally designed for environments with a low packet error rate (PER). However, in wireless networks, noise bursts, multipath fading and interference result in a much higher PER due to relatively frequent transmission errors. Worst still, TCP interprets wireless losses as signs of network congestion and unnecessarily initiates the combined slow-start with congestion avoidance algorithm [2] to slow down the traffic sent over the connection. The exponential retransmit timer back-off scheme [2] for time-out retransmission can further delay the loss recovery. As a result, TCP suffers from significant throughput degradation and unacceptable interactive delay over wireless networks.

A *link-layer scheme* may be employed to hide wireless losses from TCP by retransmitting lost packets over the wireless link using an automatic repeat request (ARQ) protocol. Examples of commercial systems that use the link-layer approach are CDPD, which uses a LAPD-derived protocol called MDLP [3], and the CDMA wireless system [4]. The main advantage of employing a link-layer protocol for recovery of wireless losses is that its implementation is confined to the data link layer at the base stations and mobile hosts, so that it fits naturally into the layered structure of TCP/IP without requiring changes to TCP.

The main problem about the link-layer approach is the possibility of competing retransmissions between TCP and link layer. In [5], analysis of a TCP-like transport protocol with a stop-and-wait link-layer protocol by simulations shows that link-layer retransmissions improve TCP throughput only when the PER is higher than 10%. At smaller PERs, the TCP retransmission timer times out before the link layer recovers a wireless loss, so that both TCP and link layer retransmits the lost data. These unnecessary duplications due to competing retransmissions contribute to the waste of valuable capacity in the wireless link and significant degradation of end-to-end TCP throughput. Using the OPNET network simulation tool [6], we evaluate the competing retransmissions problem for TCP Reno [7] (the current de facto standard for TCP) over a slow speed wireless link typical in a mobile data network, and determine how link-layer retransmissions can be fine-tuned to minimize competing retransmissions in TCP. We consider only high performance link-layer retransmission schemes based on selective reject ARQ. Our results show that link-layer recovery effectively increases the round trip time (RTT) estimates and the retransmission time-out (RTO) values of TCP, which reduces the likelihood of TCP retransmission time-outs and hence the possibility of competitive retransmissions with the link layer.

Our investigation uncovered the problem of frequent link-layer resets caused by poor channel condition in some existing link-layer protocols like CDPD MDLP [3]. We propose a best-effort approach in which the link layer does most of the recovery for wireless losses and TCP does the rest, without resetting the data link connection. Simulation results show that with a suitable choice of the maximum number of link-layer retransmissions for each packet, the TCP throughput can be maximized.

The rest of this paper is organized as follows. Section 2 describes the simulation model. In section 3.1, we show the improvement of end-to-end TCP performance using link-layer retransmissions. Section 3.2 compares TCP performance with and without in-sequence delivery at the link layer. In section 3.3, we present the best-effort modifications for link-layer protocols and analyze the relations between link-layer reliability and TCP performance. Section 4 concludes the paper.

This work was supported by a grant from Motorola Canada Ltd., and by Mil3 Inc. through University Consortium OPNET licences.

II. NETWORK MODEL

We consider a TCP connection between a mobile host and a fixed host incorporating a wireless data link (Fig. 1). Previous work [5] had focused on wireless LANs, which have data rates of 1-2Mbps. With increasing use of laptop computers and portable digital assistants, there is a growing demand for wireless Internet access over a wide area, using mobile data networks which typically employ slow speed wireless links with data rates from 2.4Kbps to 19.2Kbps. We consider a wireless link data rate of 19.2Kbps in this paper.

The characteristics of the congestion losses in the Internet are modeled by the Fritchman model [8], which captures the burstiness of congestion losses using a two-state Markov model (Fig. 2). In the Congested state, datagrams routed through the Internet are lost due to congestion. In the Clear state, datagrams routed through the Internet are successfully delivered. The transition rates from the Clear state to the Congested state and vice versa are denoted as α and β, respectively, which can be derived from the average congestion duration, average number of congestion periods per second, and datagram loss rate in the Internet.

As link-layer protocols employing stop-and-wait and go-back-N have poor efficiency, we consider two link-layer protocols employing selective-reject ARQ: the CDPD Mobile Data Link Protocol (MDLP) [3] which is modified from HDLC, and the Radio Link Protocol (RLP) derived from [9]. In RLP, the receiver periodically sends status feedback frames to cumulatively acknowledge all packets received in-sequence or request retransmissions of specific packets indicated by a bitmap when losses are detected. The use of a bitmap in the retransmission request reduces the overheads of individual negative acknowledgments. The periodic status feedback minimizes the adverse effect of errors in the return channel.

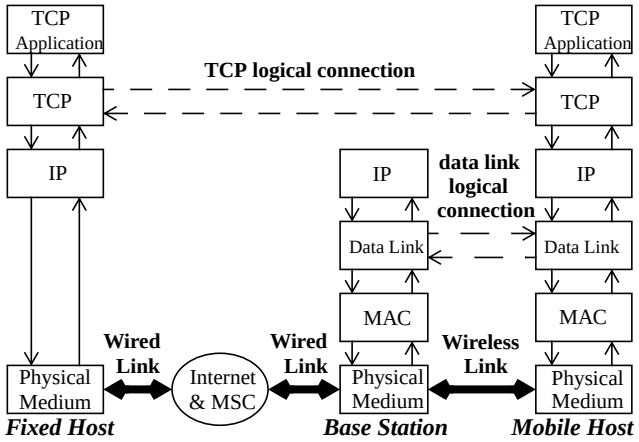
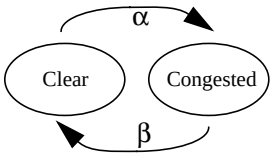


Fig. 1 Communication Architecture of a TCP connection between a fixed host and a mobile host



$$\text{Average congestion duration} = \frac{1}{\beta}.$$

$$\text{Average number of congestion periods per second} = \frac{\alpha \cdot \beta}{\alpha + \beta}.$$

$$\text{Datagram loss rate} = \frac{\alpha}{\alpha + \beta}.$$

Fig. 2 Fritchman binary error model for congestion losses

We have modified the library module of TCP in the OPNET [6] simulator to model TCP Reno [10] and used the parameters in Table 1. Although TCP Reno uses a 16384-byte receiver window, a smaller window size is used here because a large window gives a large round trip time over a low data-rate wireless link, increasing the end-to-end TCP delay and delaying TCP error recovery. Most TCP implementations use a 200-ms Maximum ACK Delay and a 536-byte Maximum Segment Size (MSS) [10]. A 512-byte MSS is used here so as to make it an integral fraction of the window size. The values of RTT and RTO parameters are as suggested in [2].

Given a 512-byte MSS in TCP, with the addition of a 20-byte TCP header and 20-byte IP header [1][11], the maximum size of an IP datagram is 552 bytes. The PER generally increases with the packet length and may be unacceptably high for large IP datagrams. However, using a small MSS increases the relative overhead of the TCP/IP headers. As a compromise, large IP datagrams are fragmented over the wireless link. Since IP provides a fragmentation and reassembly mechanism [11][12], we choose to perform fragmentation of IP datagrams at the base station and mobile host, with a 1024-bit Maximum Transfer Unit (MTU) in IP. While not considered here, header compression can further reduce overhead and improve efficiency.

TCP throughput and TCP end-to-end delay are the measures used for performance evaluation. *TCP Throughput* is defined as the number of bits transferred by TCP divided by the total time taken. In order to measure the maximum sustainable throughput, TCP is provided with application data whenever it needs so as to fill up the transmission pipe as much as possible. In each simulation, at least 40 Mbits of bulk data are sent from the source application to the remote application. *TCP End-to-end Delay* is defined as the difference between the time when a TCP segment is first transmitted by the source host and the time when the TCP segment is successfully received by the destination host.

TABLE 1 TCP RENO PARAMETERS

Receive Window Size at fixed host	4096 bytes
Receive Window Size at mobile host	4096 bytes
Maximum ACK Delay	0.2 second
Maximum Segment Size (MSS)	512 bytes
RTT Gain	0.125
RTT Deviation Gain	0.25
RTT Deviation Coefficient	4.0
Initial SRTT	1.0 second
Initial SRTT Deviation	1.0 second
Minimum RTO	2.0 seconds
Maximum RTO	240.0 seconds

III. Simulation results and discussion

We perform simulations for a TCP connection in the presence of link-layer retransmissions for a wireless link with a data rate of 19.2Kbps, over a wide range of PERs from 0 to 60%. The effects of different schemes and parameters in link-layer retransmissions are evaluated.

A. Behavior of TCP with Link-layer Recovery

To investigate the cause of competing retransmissions between TCP and link layer, we consider the effect of link-layer recovery on RTT estimations in TCP. TCP estimates RTT via a running average of the measured time for a segment to be acknowledged. The RTO value of the TCP retransmit timer is set to the smoothed RTT plus four times its mean deviation [2]. In Fig. 3, without link-layer recovery, the average RTT decreases with PER. Note that RTT estimates are updated only by ACKs of those segments that have been transmitted once, in accordance with Karn's algorithm [13]. If Karn's algorithm was not employed, RTT would increase with PER as an increasing number of TCP retransmissions is needed to successfully deliver a segment. In contrary, with link-layer recovery, average RTT increases steadily with PER as the link layer on average takes a longer time for error recovery before delivering the TCP segments and ACKs in either directions.

Fig. 4 shows that the average RTO exhibits the same behavior as RTT, but increases at a faster rate than the RTT estimate in the case of TCP with link-layer recovery. Fig. 5 shows that without link-layer recovery, TCP throughput drops rapidly as PER increases since loss of TCP segments reduces the congestion window and slows down TCP transmissions. Worse, sender time-outs also increase, giving the steep increase in average end-to-end TCP delay shown in Fig. 6. On the other hand, when link-layer recovery is employed, the two figures show that TCP throughput decreases approximately linearly with PER and the average end-to-end delay of TCP increases at a much slower rate.

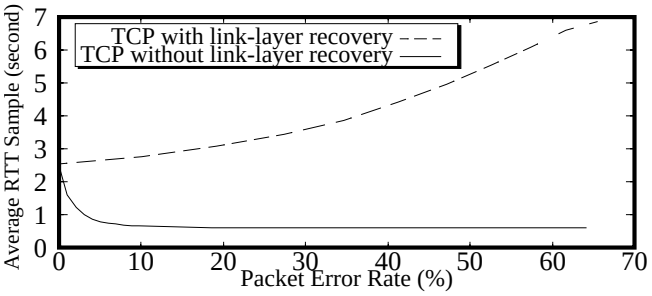


Fig. 3 Average value of RTT in TCP with and without link-layer recovery

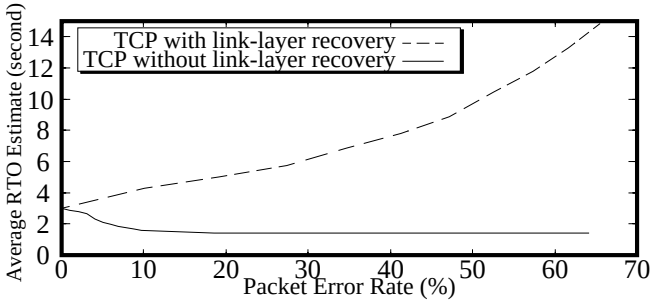


Fig. 4 Average value of RTO in TCP with and without link-layer recovery

It is apparent that link-layer retransmissions can improve the performance of TCP by shielding it from wireless losses as much as possible. What is seen by TCP is not the lossy wireless link but a more reliable link with a longer and more variable delay. One possible problem is that TCP may not adapt well to the increased delay and initiate many retransmissions, before the link layer can recover a packet loss. In [5], a serious problem of competitive retransmissions between TCP and link layer was reported. However, this problem is not obvious here, because TCP is able to adjust by increasing its RTO, which reduces the likelihood of a premature TCP retransmission time-out. The major differences between our simulation model and the one employed in [5] are as follows.

- We consider TCP Reno while a simplified model of an older version of TCP is used in [5].
- A lower data rate (19.2Kbps) typical for a mobile data network is considered in this paper, compared to the higher data rate used in [5] typical for wireless LANs.
- We use a highly efficient selective-reject ARQ in the retransmisson schemes, while a very inefficient stop-and-wait protocol is used in [5].
- We use a RTT Deviation Coefficient [2] of 4, as recommended for TCP Reno [7], instead of 2 in [5]. Using a higher RTT Deviation Coefficient gives a higher RTO estimate and makes TCP time-outs less likely.

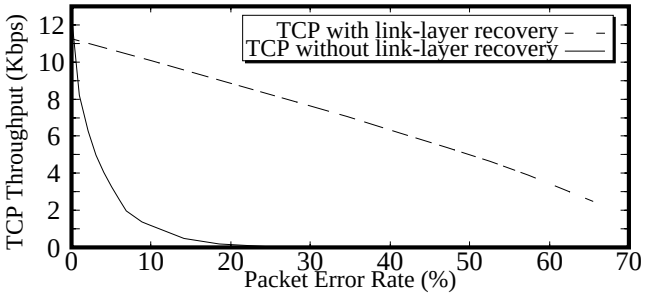


Fig. 5 Throughput of TCP with and without link-layer recovery versus packet error rate

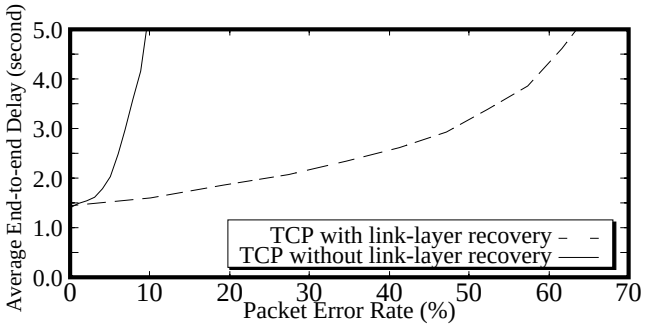


Fig. 6 Average end-to-end delay of TCP with and without link-layer recovery versus packet error rate

B. *Effects of Link-layer Re-sequencing on TCP Performance*

Link-layer recovery employing selective reject ARQ can be designed to preserve or not to preserve at the receiver the sequence of packets delivered to it at the sender. We discuss the effects of these two approaches to link-layer recovery on TCP performance, taking into account of IP datagram fragmentations before transmissions over a wireless link.

If out-of-sequence packet delivery is allowed, the link layer forwards IP fragments to the IP layer without delay. At the destination host, if several out-of-sequence IP fragments are sufficient to reassemble an IP datagram, TCP will receive a segment out of order. This is acceptable since TCP assumes an unreliable lower layer, and is capable of re-sequencing segments. On the receipt of each out-of-sequence segment, TCP returns a duplicate ACK. Reception of three consecutive duplicate ACKs by the TCP sender causes the sender to invoke fast recovery, decreasing the congestion window and hence reducing the TCP throughput. In [14], the problem of a non-sequencing link layer is identified, without taking into account of IP fragmentations. If the number of fragments per datagram is not too small, e.g., 5 in our model, reassembly of IP datagrams provides a partial re-sequencing of IP fragments. To receive an out-of-sequence TCP segment, all constituent IP fragments need to be received earlier than the lost IP fragment. If the link-layer delay is sufficiently small and the protocol ensures that a lost packet is retransmitted at least once before several subsequent packets are sent, the loss will likely be recovered before the destination IP layer reassembles and sends an out-of-sequence segment to the TCP layer.

For a link layer with in-sequence packet delivery, IP fragments may be stored in the link-layer re-sequencing buffer for an extended period of time awaiting recovery of earlier fragments to restore packet sequencing, which may substantially delay TCP segments and the ACKs piggybacked on them. At best, delaying ACKs slows the growth of the transmit window at the TCP sender receiving these ACKs, thus limiting increases in TCP throughput. At worst, delaying ACKs for too long may cause the TCP sender to time out, shrinking the congestion window and reducing throughput. However, allowing out-of-sequence delivery enables faster return of TCP ACKs, which helps to open the transmit window and prevent retransmission time-outs.

The above discussion suggests that in-sequence delivery at the link layer does not necessarily give better TCP throughput performance. Our simulation results (not shown here due to space limitation) confirm this. In the worst case, the non-sequencing data link protocol has a throughput degradation of only 2.5% below the re-sequencing data link protocol. In most cases, the TCP throughputs for both cases are comparable. For TCP/IP traffic, a *non-sequencing link-layer protocol is preferred* because it does not require re-sequencing buffers, thus simplifying the implementation of the data link protocols.

C. *Performance Analysis of TCP with Best-effort Link-layer Retransmissions*

In MDLP [3], if a packet loss cannot be recovered after several retransmissions, the data link connection will reset. When the wireless channel condition is bad, there can be frequent link-layer resets resulting in loss of TCP segments. We propose a best-effort approach in which the link layer does most of the recovery for wireless losses while TCP does the rest, and the data link connection is maintained.

In the original MDLP, if the number of successive time-out retransmissions reaches the maximum allowable value, called **N2**, the data link connection is reset and re-established (Fig. 7). During connection re-establishment, the sender sends a SAMBE command frame to the receiver [3]. Upon receipt of a SAMBE, the receiver returns a UA response frame to confirm the connection request [3]. When the UA reaches the sender, a new connection is established. In our modified MDLP, the SAMBE command is sent as before. However, no queued packets are discarded and sequence numbers are not reset. Upon receipt of a SAMBE, the receiver delivers all packets in the re-sequencing buffer to the higher layer, advances the receive window, and returns a combined UA+ACK response frame indicating the sequence number of the next packet expected in the now empty receive window. When the UA+ACK reaches the sender, it advances the transmit window to start sending packets at the sequence number specified by the UA+ACK frame, after clearing from the transmit buffer those packets with smaller sequence numbers.

Fig. 8 compares the effects of N2 on the performance of TCP with the original MDLP to that with the modified MDLP. At a low PER, the original and modified MDLP give the same TCP throughput performance. At higher PERs, best-effort retransmissions give an improved TCP throughput performance when N2 is small. However, as N2 increases, as expected, the difference in performance diminishes. In poor wireless channel conditions (with PER>30%), best-effort retransmissions can maintain the link-layer connection, so that when the channel condition improves (with PER<10%), the link layer can continue the data transfer without dropping any packets which contain useful TCP segments.

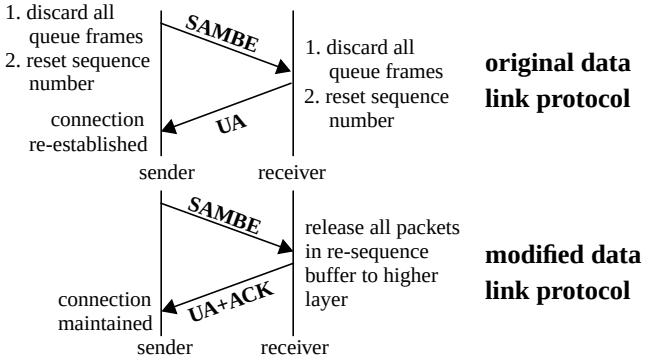


Fig. 7 Comparison between original and modified data link protocol

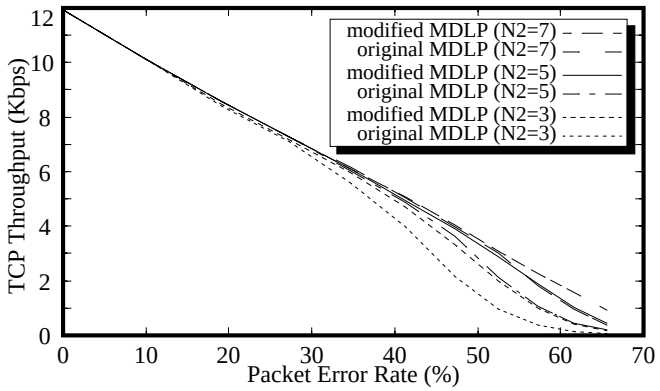


Fig. 8 Throughput of TCP using the original and the modified MDLP

Fig. 9 shows the effect of N2 on the average RTT value of TCP. With N2 = 2, the average RTT increases with PER until PER reaches about 40%. Further increases in PER do not raise the average RTT because the link layer gives up retransmission of lost packets which have already taken 2 retransmissions to recover. With N2 = 5, the average RTT increases with PER through a wider range of PER, from 0 to 60%, because the link layer is given more time to recover losses.

Note that decreasing N2 reduces the reliability of the link layer as seen by TCP, causing TCP to recover more losses. Since TCP considers each loss as an indication of network congestion and hence slows down the traffic, TCP may suffer from reduced throughput performance. On the other hand, a link layer with larger N2 hides from TCP most of the wireless losses, but results in higher RTT and RTO values in TCP. With a higher RTO, TCP takes longer to recover network congestion losses by time-out retransmissions. Fig. 10 considers the situation with a datagram loss rate in the Internet of 5%. With

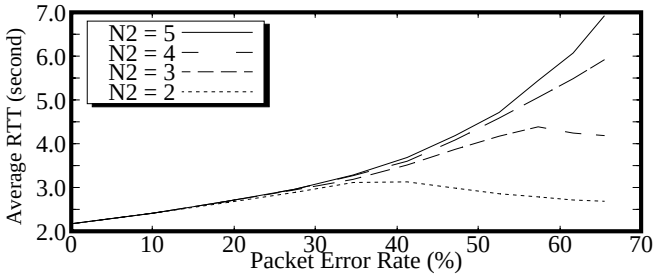


Fig. 9 Average value of RTT in TCP under various values of Maximum No. of Retransmissions (N2)

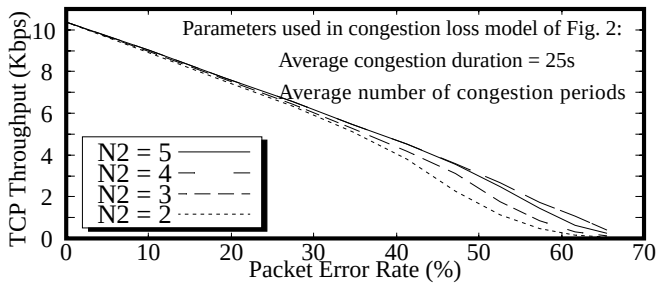


Fig. 10 Throughput of TCP under various values of Maximum No. of Retransmissions (N2)

N2 = 2 and 3, TCP needs to recover more wireless losses that data link layer has given up retransmissions, giving lower throughput when PER is high. With N2 = 5, a higher RTO value at high PER makes TCP recover congestion losses relatively slowly, giving lower throughput than with N2 = 4. In this case it is apparent that the best choice for N2 is 4.

IV. Conclusions

We have performed a detailed analysis of using link-layer retransmissions to improve the end-to-end performance of TCP in wireless networks. Simulation results show that employing existing data link protocols like CDPD MDLP and RLP for local retransmissions over wireless link can help to improve the throughput and end-to-end delay performance of TCP. We have determined that the problem of competitive retransmissions between TCP Reno and link layer is not obvious for slow-speed data links using selective-reject ARQ for link-layer retransmissions. We have also discovered that non-sequencing data link protocols do not necessarily degrade TCP performance. Instead, re-sequencing buffers and complex logic for handling out-of-sequence packets that would otherwise be needed for a sequencing link-layer protocol can be avoided. Finally, we have proposed modifications to link-layer protocols for best-effort retransmissions to overcome the problem of frequent link-layer resets. We conclude that low reliability causes TCP to recover more losses but high reliability may slow down TCP time-out retransmissions for recovery of network congestion losses. With a suitable division of the wireless-loss recovery function between TCP and link layer, the TCP throughput can be maximized.

References

[1] J.B. Postel, “Transmission Control Protocol”, IETF RFC 793, Sep. 1981.
[2] V. Jacobson, “Congestion Avoidance and Control”. *Proc. ACM SIGCOMM*, Aug. 1988.
[3] “Mobile Data Link Protocol”, Part 403, CDPD System Specification, Release 1.1, Jan. 19, 1995.
[4] P. Karn, “The Qualcomm CDMA Digital Cellular System”, *Proc. 1993 USENIX Symp. on Mobile and Location-Independent Computing*, pp. 35-40, Aug. 1993.
[5] A. DeSimone, M.C. Chuah and O.C. Yue, “Throughput Performance of Transport-Layer Protocols over Wireless LANs”, *Proc. IEEE GLOBECOM*, pp. 542-549, Nov. 1993.
[6] MIL 3 Inc., Optimized Network Engineering Tools (OPNET), Release 3.0.B, 1997.
[7] W.R. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”, IETF RFC 2001, Jan. 1997.
[8] B.D. Fritchman, “A binary channel characterization using partitioned Markov Chains”, *IEEE Trans. Information Theory*, pp. 221-227, 1967.
[9] S. Paul, E. Ayanoglu, T.F. LaPorta, K.W.H. Chen, K. K. Sabnani, R. D. Gitlin, “An Asymmetric Protocol for Digital Cellular Communications”, *Proc. INFOCOM*, pp.1053-1062, 1995.
[10] W.R. Stevens: *TCP/IP Illustrated, Vol. 1: The Protocols*. Addison-Wesley, 1994.
[11] J.B. Postel, “Internet Protocol”, IETF RFC 791, Sep. 1981.
[12] D.D. Clark, “IP Datagram Reassembly Algorithms”, IETF RFC 815, Jul. 1982.
[13] P. Karn and C. Partridge, “Improving Round-Trip Time Estimates in Reliable Transport Protocols”, *Proc. ACM SIGCOMM*, Aug. 1987.
[14] H. Balakrishman, V.N. Padmanabhan, S. Seshan and R.H. Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, *Proc. ACM SIGCOMM*, Aug. 1996.