TULIP: A Link-Level Protocol for Improving TCP over Wireless Links

Christina Parsa J.J. Garcia-Luna-Aceves

Computer Engineering Department
Baskin School of Engineering
University of California
Santa Cruz, California 95064
chris, jj@cse.ucsc.edu

Abstract— **We present the transport unaware link improvement protocol (TULIP), which dramatically improves the performance of TCP over lossy wireless links, without competing with or modifying the transport- or network-layer protocols. TULIP is tailored for the half-duplex radio links available with today's commercial radios and provides a MAC acceleration feature applicable to collision-avoidance MAC protocols (*e.g.,* IEEE 802.11) to improve throughput. TULIP's timers rely on a maximum propagation delay over the link, rather than performing a round-trip time estimate of the channel delay. The protocol does not require a base station and keeps no TCP state. TULIP is exceptionally robust when bit error rates are high; it maintains high goodput, i.e., only those packets which are in fact dropped on the wireless link are retransmitted and then only when necessary. The performance of TULIP is compared against the performance of the Snoop protocol (a TCP-aware approach) and TCP without link-level retransmission support. The results of simulation experiments using the actual code of the Snoop protocol show that TULIP achieves higher throughput, lower packet delay, and smaller delay variance.**

I. Introduction

With the need to support end-to-end communication services to mobile hosts, wireless networks are quickly becoming an integral part of the Internet and reliable protocols such as TCP [21] must be supported over these networks. Mobile users requiring remote access to corporate LANs, file access and Web transfers over wireless links must rely upon TCP to support their transactions. Unfortunately, although TCP works very well for wired networks with minimal losses other than those due to congestion, wired and wireless networks are significantly different in terms of bandwidth, speed, propagation delay, and channel reliability. In particular, wireless channels suffer from bursty error losses that reduce TCP's throughput, because TCP incorrectly interprets packet loss as a sign of congestion that forces TCP to back off from further transmission and reduce its congestion window. As a result, the overall throughput of the connection is drastically reduced. Methods to hide these losses from TCP is an active area of research [3][4][6][16][8][9].

Maximum throughput occurs in a TCP connection when the TCP congestion window is as large as the bandwidth-delay product of the connection. Current versions of TCP react to losses differently and adjust the TCP congestion window in various ways. Wireless channels present an additional challenge in that losses do not generally occur in isolation, *i.e.*, wireless channels are often characterized by periods of fading in which several losses occur in succession. All versions of TCP are unable to gracefully recover from this situation and must resort to a timeout whenever more than one loss occurs per window of outstanding data. This becomes the predominant shortcoming of TCP over wireless links: the connection suffers long idle periods in which the sender is idle waiting for a timeout, and when the packet is finally retransmitted and recovered, the congestion window is reduced to one segment, thereby reducing throughput until the congestion window again grows to its optimal size.

[0]This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grants DAAB07-95-D157 and DAAH04-96-1-0210

II. Related Work

The quest to solve the ills of TCP over lossy wireless links is an area of active research. Solutions at lower protocol levels attempt to recover losses by using forward error correction(FEC) at the physical layer. FEC is generally considered to be a limited approach (although it remains an area of active research) as it can reverse only a limited number of bit errors, and also is costly in terms of packet delay due to computation time and power consumption in an environment in which power use must be minimized. Solutions based on higher-level protocols attempt to fool TCP by hiding the lossiness of the wireless link and fall into three major categories: Link Layer, Split Connection, and Proxy.

The AIRMAIL protocol [3] provides a reliable link layer in conjunction with forward error correction(FEC). In this approach, in order to conserve bandwidth and power the base station sends an entire window of data before an ACK is returned by the mobile receiver. Unfortunately, a consequence of this approach is that there is no opportunity to correct errors until the end of an entire window, which can cause TCP to time out if the error rate is large or cause a large variation in delay depending upon the position of the loss in the window.

DeSimone *et al.* [2] conclude that introducing reliability at the link layer introduces unnecessary and redundant retransmissions, because of competing retransmission strategies between the transport and link layers. However, this conclusion was reached based on an analysis that did not take into account the very generous timeout value calculated by TCP nor its granularity of 500ms, but rather an ideal case in which a timeout occurs at the estimated round-trip time value. Balakrishnan *et al.* [5] demonstrate that link-layer protocols that fail to provide in-order delivery to the application essentially compete with the upper layers by duplicating retransmissions.

In the split-connection approaches, the TCP connection is split between the source and base station and then between the base station and the wireless receiver [4][8]. I-TCP [4] runs at the base station, buffers and sends ACKs to the source for packets that have not yet been acknowledged by the receiver. The drawback to this approach is that it violates the semantics of TCP, and cannot therefore be easily deployed in the Internet. M-TCP [8], on the other hand, preserves TCP semantics and aims to improve throughput for connections which exhibit long periods of disconnection. M-TCP is not a complete solution and the authors state the algorithm still requires a good link-layer protocol to recover losses.

In the proxy approach, a proxy is inserted between sender and receiver TCP hosts to help TCP's performance. A well-known example of this approach is the Snoop protocol [6], which runs at the base station; Snoop retransmits lost packets and suppresses duplicate TCP ACKs by sniffing all packets entering an interface before they are passed on to IP. Retransmissions are performed when it detects two duplicate ACKs for any packet it has seen previously and stored in its buffer. Snoop has been shown to improve TCP performance over wireless links with bit-error rates up to 15 bits per millions [6]. Snoop must maintain state for all TCP sessions going through it.

It has also been suggested that TCP-SACK [11] can be used to improve TCP performance over wireless links [5]. TCP-SACK provides for end-to-end selective acknowledgment (SACKs) of received TCP segments. TCP-SACK would certainly expedite the discovery of lost packets on wireless segments if no underlying link-level recovery were used; however, the algorithm would still incorrectly interpret lost segments on the wireless link as a sign of congestion.

III. Protocol Overview

In this paper, we present the transport unaware link improvement protocol (TULIP), and show by simulation studies (for a more detailed protocol description and extensive simulation results see [10]) that TULIP allows TCP to operate efficiently over wireless networks, with no changes to the hosts and TCP's semantics, and without requiring proxies between sender and receiver TCP. TULIP is *service-aware* in that it provides reliability for only those packets (frames) that require such service, but it is not *protocol-aware*, *i.e.*, it does not know any details of the particular protocol to which it provides its reliable service. More specifically, TULIP provides reliable service for packets carrying TCP data traffic, and unreliable service for other packet types, such as UDP traffic (*e.g.,* routing table updates and DNS packets) and TCP acknowledgments (ACKs). TULIP doesn't provide reliable service to TCP ACKs because subsequent cumulative ACKs supersede the information in the lost ACK. The receiver buffers packets and passes them up to the next layer in order, thereby preventing TCP from generating duplicate ACKs in the event that a packet is missing from the expected sequential packet stream. This approach eliminates the need for a transport-level proxy [6], which must keep per-session state to actively monitor the TCP packets and suppress any duplicate ACKs it encounters. An important feature of TULIP is its ability to maintain local recovery of all lost packets at the wireless link in order to prevent the unnecessary and delayed retransmission of packets over the entire path and a subsequent reduction in TCP's congestion window. Flow control across the link is maintained by a sliding window, and automatic retransmission of lost packets is accomplished by the sending side's link layer. Lost packets are detected at the sender via a bit vector returned by the receiver as a part of every ACK packet. This allows for quick and efficient recovery of packets over the link and helps to keep delay and delay variance low. However, TULIP is designed for efficient operation over the half-duplex radio channels available in today's commercial radios by strobing packets onto the link in a turn-taking manner. We introduce a new feature, MAC Acceleration, in which TULIP interacts with the MAC protocol to accelerate the return of link-layer ACKs (which are most often piggybacked with returning TCP ACKs) without renegotiating access to the channel. TULIP causes no modification of the network or transport layer software, and the link layer is not required to know any details regarding TCP or the algorithms it uses. TULIP maintains no TCP state whatsoever, and makes no decisions on a TCP-session basis, but rather solely on a per-destination basis. This approach greatly reduces the overhead of maintaining state information when multiple TCP sessions are active for a given destination (as is common with Web traffic). From the transport layer's point of view, the path to the destination through a lossy wireless link simply appears to be a slow link without losses and TCP simply adjusts accordingly.

IV. Implementation

We have implemented TULIP and Snoop [6] in the C++ Protocol Toolkit (CPT) [7][1]. A key feature of our simulation is that it is based on the exact same source code that runs in the WING prototypes (which are wireless IP routers) [15], and in hosts attached to the WINGs. The IEEE 802.11 specifications are used at the physical layer to emulate the broadcast medium. CPT simulates the wireless and wired transmission media with specific parameters and channel characteristics specified through script files read at runtime. Our implementation of TULIP runs on top of FAMA-NCS [13]. TULIP in turn interacts with IP [20] and the wireless Internet routing protocol (WIRP) [19] for packet forwarding. The code of the Snoop protocol [6] was modified from the on-line FreeBSD implementation[2] to run in CPT.

[1] We thank Rooftop Communications Corporation for donating the toolkit.

[2] We thank H.Balakrishnan for providing on-line source code at ftp://daedalus.cs.berkeley.edu/pub/snoop/

V. PERFORMANCE

TCP's performance over wireless links is analyzed through simulation for networks subject to low and high bit-error rates, burst losses, and fading. We evaluate TCP's performance under three different situations: when there are no underlying link layer retransmissions, when the Snoop protocol [6] is used at the base station, and when TULIP is used. The simulation experiments assume a simple configuration, shown in Figure 1, with a base station and a single wireless host connected to the base station. We compare TULIP's performance to the performance of the Snoop protocol, for which very good performance-improvement results have been reported in the base station configuration.

The following parameters are fixed for all experiments: the channel capacity from the wired source to the base station is 10Mbps, the wireless transmission rate is 1Mbps, file transfers are 10Mbytes with 1400 byte data packets, TULIP's window size is 8 packets, and the offered traffic is a Poisson source with an average rate of 1Mbps. These specifications are the same as those used in the Snoop experiments [6], except for the radio characteristics (including a lower transmission rate for our experiments).

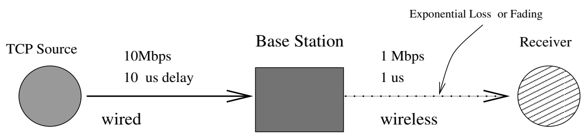


Fig. 1. Topology for Experiments.

A. Low Error Rates

Figure 2(a) shows the average throughput for three cases: TCP with no underlying link-layer retransmissions (labeled no LL), TCP with the Snoop protocol and with TULIP. Bit error rates are varied from 0 to 15 bits/million and the receiver window size is 42Kbytes. When the link errors are zero the graphs show that MAC Acceleration provides a 4 – 5% improvement in throughput. For error rates under 0.5 bits/million, TCP is able to keep up with the losses and does not show appreciable performance degradation. However, as losses begin to rise, it is apparent that traditional TCP can no longer keep up with the losses and the throughput degrades considerably. Both the Snoop protocol and TULIP show decreased throughput as the BER rises; however, their overall throughput is consistently better than unassisted TCP. The reduction in throughput is due to the numerous retransmissions that occur as error rates increase.

Figure 2(b) shows the average packet delay with TULIP is lower than the Snoop protocol in every instance, and as the error rates increase, TULIP's standard deviation is also lower than the Snoop protocol. TULIP's delay is smaller because TULIP has a faster retransmission mechanism. This becomes apparent as the error rate increases and Snoop's variance jumps up at a BER of 15 bits per million when it has trouble with scattered losses within a window and with the occasional loss of retransmitted packets. The end-to-end delay in this figure is reduced by more than half over a session with a 42K receiver window because the queueing at the base station is reduced substantially. This leads to our recommendation that mobile nodes should, in general, advertise a smaller window size to reduce delay and lessen the possibility of congestion at the base station.

B. High Error Rates

Some public wireless communication providers [14][18] report typical wireless one-hop packet loss rates between 10 - 40%. For this reason, we increase the bit error rates on the channel to very high values, *i.e.*, from 15 to 75 bits per million (corresponding to packet loss rates from 16 to 85%).

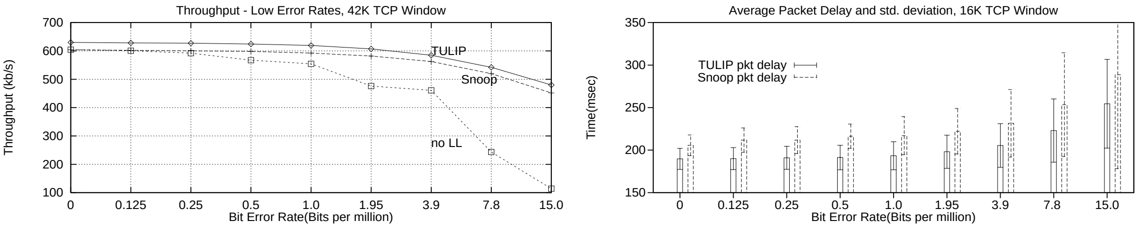
TCP's throughput is shown in Figure 3(a) for a 16Kbyte receiver window. The graph clearly shows that, as the bit error rate increases, all three protocols show a reduced turn in throughput. With a BER of 15 bits/million the TCP connection with no link-layer retransmissions (labeled no LL) has degraded significantly and comes to a near standstill for error rates above 30 bits/million. Beyond a BER of 30 bits/million Snoop's throughput[3] drops off quickly and diverges from TULIP's throughput, which decreases slowly as error rates increase. These error rates are characterized by many multiple losses per window of data. TULIP can easily recover from these episodes and the connection simply appears increasingly slower to the transport layer.

Figure 3(b) shows the average end-to-end packet delay and standard deviation for the corrected Snoop and TULIP. This plot shows that the delays and deviation with the Snoop protocol are significantly larger than with TULIP once error rates exceed 35 bits/million. The larger delay is because the Snoop protocol has trouble recovering multiple losses per window and also recognizing when retransmissions are also lost. Snoop must rely heavily on its timer and cumulative ACKs for retransmissions and gets stuck trying to retransmit the first packet in a series of losses. The deviation is high here because losses are tackled one by one and in order, *i.e.*, once the first loss is recovered, then the next is tackled and so on. TULIP, on the other hand, creates a retransmission list upon the first receipt of an ACK and knows exactly which packets are missing because each ACK specifies the complete state at the receiver's buffer. In addition, if retransmitted packets need to be again retransmitted, TULIP is able to do this as soon as it has received any ACK. The deviation is smaller in TULIP because, with multiple losses per window, it is often the case that errors further down in the window are recovered before the first error. Therefore, by the time the first error is recovered, all the packets can be released to the higher layer in sequence.

C. Burst Losses and Fading

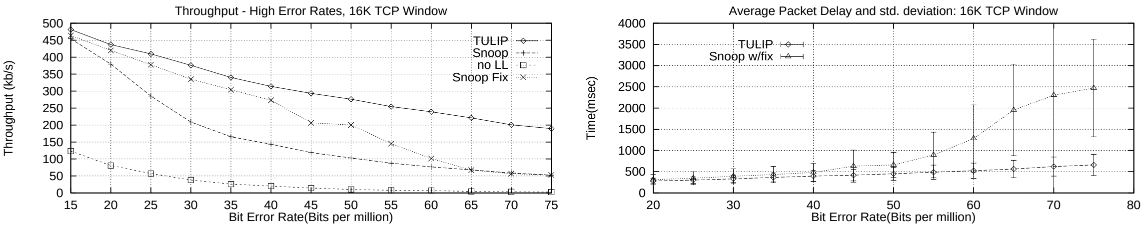
In this section we examine TCP's performance in the presence of packet burst losses and fading on the wireless link. Fad-

[3] Two lines are shown for Snoop: in the one labeled Snoop w/fix we have fixed a coding bug in the on-line source release code.



(a) (b)

Fig. 2. Low Error Rates (a)Throughput (b)End-to-end delay and delay variation



(a) (b)

Fig. 3. High Error Rates (a)Throughput (b)End-to-end delay and delay variation

ing causes periods of silence on the channel, during which time neither sender nor receiver can hear each other. Fading is often caused by the movement of mobile nodes, but can also be caused by objects which move in front of and around a mobile node.

C.1 Uniform Distribution of Burst Losses

In the method used by Balakrishnan et. al. [5], burst losses of a specific size are distributed uniformly over the run of the experiment. The results when bursts of sizes 2,4 and 6 data packets are spread every 64Kbytes of data are shown in Table I. Because of FAMA's handshake, the sender would not send more than one packet into the channel during a fading period, because it would not receive the necessary CTS to send any more data packets. The same would the case for DWFMAC [1]. However, this experiment is still interesting to show, because it indicates that TULIP provides smaller delays and slightly better throughput than Snoop, even in such rare cases in which the MAC layer manages to send multiple packets into the channel that reach the receiver in error, even though the corresponding RTS and CTS packets did not.

C.2 Markov Model of Fading

The method to simulate channel fading consists of a two-state Markov model taken directly from the work by Lettieri *et al.*[17], which is also discussed by Wang *et al.* [22] and Swarts *et al.* [12]. Briefly, the model consists of a two-state Markov chain representing *Good* and *Bad* states on the channel, transition probabilities into and out of the states, and error loss probabilities associated with each state. We have performed this experiment for a pedestrian speed of 2km/hr and the results are presented in Figures 4(a) and (b) as the BER in the *Good* state is varied from 0.01 to 100 bits/million and the loss probability in the *Bad* state held constant at 50%. Figure 4(a) shows that TULIP and the Snoop protocol display similar performance as long as the error rates are low; however, Snoop's throughput begins to diverge and fall below TULIP once error rates exceed 10 bits/million. The performance of TCP with no underlying retransmissions degrades once error rates exceed 0.1 bits/million, or approximately 1/8Mbytes. The end-to-end packet delay, depicted in Figure 4(b), shows that the average delay for TULIP and the Snoop protocol are similar; however, the standard deviation for Snoop is again higher. For the highest error rate shown, 100 bits/million, TULIP provides a much lower delay and a tighter bound on the deviation.

VI. Conclusion

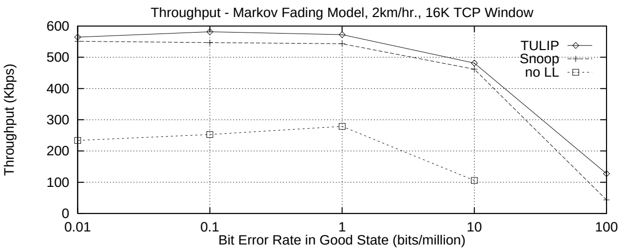
The results of our simulations show that TULIP performs better for any bit-error rate than Snoop and TCP with no underlying retransmissions. In addition, at very high error levels, TULIP's throughput is up to three times higher than both Snoop and TCP with no underlying retransmissions. End-to-end delay becomes a problem as the losses on the link increase; however, reducing the size of the receiver's advertised window can help to alleviate the queuing delay. The end-to-end packet delay with TULIP is significantly lower than the other two approaches and, in contrast to Snoop, the standard deviation of delay with TULIP grows only slightly with increasing error rates. We have examined the effects of burst losses and channel fading and our results show that again the TULIP approach quickly retransmits the dropped packets once the channel is active again, yielding reduced but consistent throughput. The simulations show that during fading TULIP provides higher throughput and lower end-to-end delays compared to both Snoop and TCP with no underlying retransmissions.

The advantage of our approach over other published ap-

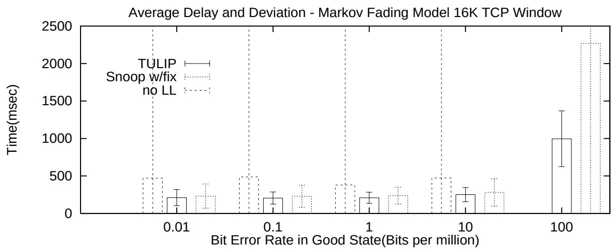
Bursts Distributed every 64Kbytes					
Burst Size #packets	TULIP Throughput(Kbps)	Snoop Throughput(Kbps)	Δ (Kbps)	TULIP Delay $\pm$ dev.(ms)	Snoop Delay $\pm$ dev.(ms)
2	587.3	562.6	24.7	540$\pm$56	582$\pm$60
4	550.0	527.6	22.4	579$\pm$74	621$\pm$84
6	516.1	496.4	19.7	618$\pm$98	660$\pm$114

TABLE I

THROUGHPUT OF TULIP AND SNOOP IN THE PRESENCE OF BURSTS OF LENGTH 2,4 AND 6 PACKETS. BURST PERIODS ARE DISTRIBUTED EVERY 64KBYTES OF DATA. RECEIVER WINDOW IS 42KBYTES.



(a)



(b)

Fig. 4. TULIP and Snoop during Markov Fading Model. Loss probability in bad state is 50% and BER in good state is varied. Pedestrian speed 2km/hr. (a) Throughput (b) End-to-end delay and standard deviation

proaches is that we keep no TCP state and therefore do not need to look into the TCP packet headers. This means that TULIP works correctly with any current or future version of TCP (*e.g.*, TCP-SACK), even if TCP headers are encrypted. TULIP works with both IPv4 and IPv6; in the latter case, TCP data packets can be identified as requiring reliable service from the NextHeader field in the IPv6 header. In addition, because our approach does not restrict the network to the presence of a base station, it can easily be applied to multi-hop wireless networks. Furthermore, by controlling the MAC layer, TULIP conserves wireless bandwidth by piggybacking TCP ACKs with link-layer ACKs and returning them immediately across the channel through MAC Acceleration.

REFERENCES

[1] P802.11–Unapproved Draft: Wireless LAN Medium Access Control (MAC) and Physical Specifications. Technical report, IEEE, January 1996.

[2] A. DeSimone, M.C. Chuah, and O.C. Yue. Throughput performance of transport-layer protocols over wireless LANs. In *Proc. IEEE Globecom '93*, pages 36–46, Houston,TX, Nov. 1993.

[3] E. Ayanoglu, S. Paul, T. LaPorta, K. Sabnani, and R. Gitlin. Airmail: A link-layer protocol for wireless networks. *Wireless Networks*, 1(1):47–60, 1995.

[4] A. Bakre and B. R. Badrinath. I-TCP: indirect TCP for mobile hosts. In *Proc. 15th IEEE Int'l Conf. on Distributed Computing Systems*, pages 136–43, Vancouver, BC, Canada, May 1995.

[5] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 6(5):756–69, Dec. 1997.

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

[7] D. Beyer and B. Nguyen. *The C++ Protocol Toolkit: Overview*. Rooftop Communications Technical Manual, 1995.

[8] Kevin Brown and Suresh Singh. M-TCP: TCP for mobile cellular networks. In *Computer Communication Review*, volume 27 No. 5, pages 19 – 43, Oct., 1997.

[9] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications*, 13(5), 1995.

[10] C.Parsa and J.J. Garcia-Luna-Aceves. Improving TCP performance over wireless networks at the link layer. *To appear, ACM Mobile Networks and Systems Journal*, 1999. Available at http://www.cse.ucsc.edu/research/ccrg.

[11] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe,Reno, and SACK TCP. In *Computer Communication Review*, volume 26 No. 3, pages 5 – 21, July, 1996.

[12] F.Swarts and H.C. Ferreira. Markov characterization of digital fading in mobile VHF channels. *IEEE Transactions on Vehicular Technology*, 43(4):977–985, Nov. 1994.

[13] C. L. Fullmer and J.J. Garcia-Luna-Aceves. Solutions to hidden terminal problems in wireless networks. In *Proc. SIGCOMM'97*, pages 39–49, Cannes, France, Sept. 1997.

[14] C.L. Fullmer. Personal communication, April 1998. Rooftop Communications Corp., Mountain View, CA 94041.

[15] J.J. Garcia-Luna-Aceves, C.L. Fullmer, E. Madruga, D. Beyer, and T. Frivold. Wireless Internet Gateways (WINGS). In *Proc. MILCOM'97*, Monterey,California, Nov. 1997.

[16] U. Madhow H. Chaskar, T.V. Lakshman. On the design of interfaces for TCP/IP over wireless. In *MILCOM '96 Conference Proceedings*, volume 1 No. 3, pages 199–203, Oct. 1996.

[17] P. Lettieri, C. Fragouli, and M. Srivastava. Low power error control for wireless links. In *Proc. Third ACM/IEEE MobiCom Conference*, pages 139–150, Budapest,Hungary, Sept. 1997.

[18] M.Ritter and R.Friday. Personal communication, April 1998. Metricom Inc., Los Gatos, CA 95032.

[19] S. Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal*, 1(2), 1996.

[20] J.B. Postel. Internet Protocol. Technical report, SRI Network Information Center, September 1981. RFC 791.

[21] J.B. Postel. Transmission Control Protocol. Technical report, SRI Network Information Center, September 1981. RFC 793.

[22] H.S. Wang and N. Moayeri. Finite-state Markov channel-A useful model for radio communication channels. *IEEE Transactions on Vehicular Technology*, 44(1):163–171, Feb. 1995.