TCP-Cognizant Adaptive Forward Error Correction in Wireless Networks

Benyuan Liu [1], Dennis L. Goeckel [2], Don Towsley [1]
[1] Department of Computer Science
University of Massachusetts
[2] Department of Electrical and Computer Engineering
University of Massachusetts

Abstract— **Wireless links are characterized by high bit error rates and intermittent connectivity. This can result in significant degradation in the performance (goodput) of TCP over wireless networks since non-congestion related packet losses can be misinterpreted by TCP as indications of network congestion, resulting in unnecessary congestion control. In this paper, we propose a technique, TCP with adaptive forward error correction (TCP-AFEC), to improve TCP performance over wireless networks. TCP-AFEC combines the well-established performance characterization of TCP with an understanding of the link layer error control scheme to dynamically select the forward error correction (FEC) that maximizes TCP goodput according to the current channel condition. The benefit of coupling a characterization of TCP performance with link layer FEC to improve TCP goodput is demonstrated by comparing the performance of TCP-AFEC against those of TCP-SACK and Snoop. Simulation results show that TCP-AFEC outperforms TCP-SACK and Snoop for a wide range of wireless channel conditions.**

I. Introduction

Recently there has been substantial activity in the area of mobile wireless data networks. TCP is the prevalent reliable transport protocol in today's Internet and has been widely used in many application layer protocols. Thus it must be supported in the wireless regime in order to make wireless networks integral parts of the Internet. However, TCP is well known to suffer severe performance degradation in wireless networks [1]. Packet loss due to bit corruptions over wireless links can be misinterpreted by TCP as indications of network congestion. This will unnecessarily trigger the TCP congestion control mechanism, resulting in a reduced throughput. Therefore, it is crucial to improve the performance of TCP over wireless links to support the fast adoption and deployment of wireless data networks. In the past few years, there have been many proposals to improve TCP performance in wireless networks, for example, [2][3][4][5][6][7][8]. The performance of some of these proposals were studied in [1].

In this paper, we propose to couple TCP with an adaptive forward error correction protocol (AFEC) in order to improve TCP's performance in a base station oriented wireless network. The approach integrates the TCP performance characterization and the link layer forward error correction (FEC) performance by employing a well-established TCP throughput formula [9] to compute the forward error correction (FEC) code that maximizes TCP goodput. In particular, we analyze the effect of adding FEC to TCP packets between the base station and mobile host on TCP goodput and present an algorithm to select the code to be used for a given channel condition. The benefit of combining TCP performance characterization with link layer FEC to improving TCP goodput over wireless networks is demonstrated through comparison with several other proposals. We show that TCP combined with adaptive FEC, TCP-AFEC, significantly improves TCP performance over TCP-SACK across a wide range of wireless channel conditions. We also compare TCP-AFEC with Snoop [5], a proposal that provides the biggest improvement over TCP Reno of all approaches under study in [1]. Our simulation results show that TCP-AFEC achieves a goodput comparable to that of Snoop at low bit errors rates and considerably higher goodput than Snoop when bit error rates are high, and thus is more robust than Snoop over a broad range of wireless channel conditions.

Note that a similar approach has been independently proposed in [10]. However, the development in this paper is more closely tied to the physical layer assumptions (as opposed to the packet error process model assumed in [10]) and is expected to yield better performance. Furthermore, we consider a number of important practical issues.

The rest of the paper is organized as follows. Section II describes the TCP-AFEC protocol and highlights the characteristics of the approach. In Section III, we evaluate the performance of TCP-AFEC, and compare the goodput of TCP-AFEC with that of TCP-SACK and Snoop. Conclusions are presented in Section IV.

II. TCP-AFEC Protocol

A. TCP Throughput Formula

Despite the complex behavior of TCP due to its various mechanisms such as slow start, congestion control, timeout, etc, it has been shown in [9] that the throughput of a TCP connection is a simple expression of packet loss rate (p) and average round trip time (RTT). The TCP goodput, G_f, can simply be obtained by scaling the throughput by a factor of $(1-p)$:

$$G_f \approx \min\left(W_{max}, \frac{1}{\sqrt{\frac{2bp}{3}} + T_0 \min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)}\right)\frac{(1-p)}{RTT} \quad (1)$$

This research has been supported in part by NSF under awards ANI-9809332, EIA 0080119, NSF ITR-0085848, NCR-9714597, and CCR-9875482. Any opinions, findings, and conclusions or recommendations expressed in this laterial are those of the authors and do not necessarilty reflect the views of NSF.

where W_{max} is the maximum congestion window size of the TCP sender, b represents the effect of delayed ack, and T_0 is the TCP retransmission timeout value.

The above formula assumes congestion-related packet losses and it has been shown to accurately predict TCP goodput over a wide range of packet loss rates [9][11]. Furthermore, our simulation results show that the formula also yields good predictions for random packet loss scenarios.

B. Effect of Forward Error Correction

Forward error correction (FEC) has been widely used in wireless data communication systems to combat transmission errors at the link layer. In FEC, parity-check bits are added to the data to form a codeword, and the codeword is transmitted. The parity bits are used by the receiver to attempt to recover from errors that may have occurred on the wireless link.

Consider the effect of FEC on TCP goodput. On one hand, FEC can reduce the packet error rate using its error correction mechanism. According to the TCP goodput formula (1), this leads to a larger achievable TCP goodput. On the other hand, part of the link bandwidth in a system employing FEC is used to carry parity bits, resulting in a smaller effective channel bandwidth for the real payload.

If the effective channel bandwidth is larger than the achievable TCP goodput (obtained from (1)), the real TCP goodput should be well approximated by (1). If the effective channel bandwidth is not large enough to meet the requirement of the achievable TCP goodput, the TCP sender can achieve at most the effective link bandwidth. Therefore, the real TCP goodput can be approximated by the minimum of the achievable TCP goodput and the effective channel bandwidth. Increasing the level of FEC redundancy increases the achievable TCP goodput but decreases the effective channel bandwidth. Since the achievable TCP goodput is an increasing function of the level of FEC redundancy while the effective channel bandwidth is the opposite, the TCP goodput is maximized when the effective channel bandwidth becomes equal to the achievable TCP goodput.

In TCP-AFEC, we use the TCP goodput formula to analyze the tradeoff between the gain of the TCP goodput and the reduction of effective channel bandwidth through the application of FEC. We will provide an algorithm to compute the optimum FEC code that maximizes TCP goodput.

C. Physical Layer Assumptions

For wireless links, the received signal strength is affected by three major factors: path loss, signal shadowing, and multipath fading. In this paper, we assume the presence of an average signal to interference plus noise ratio (SINR) measurement, where the averaging is over the multipath fading, at the base station. Thus, this average SINR indicates the path-loss and shadowing that the wireless link is undergoing, but does *not* assume knowledge of the multipath fading, which varies at a much more rapid rate. This average SINR measurement is generally available in wireless communication systems and is currently employed for power control [12], handoff [13], and adaptive rate control [14]. The variation of the channel between the time SINR measurements are made and when they are employed is an important consideration in adaptive systems. Whereas this presents a significant problem when estimates of the multipath fading are employed [15], the problem is much less severe for measurements of the path-loss/shadowing and can be compensated for using the statistical model developed in [16].

Given the average SINR (or, more generally, any measurement that is correlated to link quality) of the wireless link, it is critical for our work to establish the packet loss rate as a function of the wireless system parameters; in particular, it is important to be able to characterize the packet error rate as a function of the code rate and the average SINR measurement. There are a number of ways to construct such a mapping, but the difficulty lies in dealing with the averaging over the multipath fading process, for which the statistics might not be known. If the fading affecting bits (or symbols) is assumed to be independent, the formulas follow easily, but this is unlikely to be a good assumption in most systems. Instead, **our** system builds a set of look-up tables that provides the optimal code rate based on the current average SINR measurement and user velocity. Note that the velocity can be well-estimated [17][18] and provides the parameter for parameterized versions of the autocorrelation function. Spectral estimate techniques, of course, can also be employed.

In this work, we assume the use of block codes for FEC where the level of redundancy can be adjusted. In particular, we consider an (N, K) Reed-Solomon code, where $(N - K)$ parity symbols are added to K data symbols to form a codeword of size N. The number of information symbols per codeword, K, is fixed and the code length N is varied to adjust the redundancy level of the code. Here a symbol is the basic information unit used in a Reed-Solomon code, and is composed of a certain number of bits. Assume a symbol carries m bits; then the length of the code will not exceed 2^m, i.e., $N < 2^m$.

D. TCP-AFEC Protocol

Now we describe the TCP-AFEC protocol. Consider a TCP connection between a host in a wired network and a mobile host via a base station, where the wireless link is the bottleneck of the connection. In TCP-AFEC, a link layer agent is added to the base station and the mobile host respectively to improve the TCP goodput. At the wireless hop of the TCP connection, the link layer agent at the upstream node of the data flow estimates the packet error rate (PER) and TCP session RTT. For each data packet passing by, the agent divides the packet into frames, computes and constructs the optimal FEC code for each frame that maximizes TCP throughput, adds appropriate fragment headers, and then transmits the frames over the wireless link. At the downstream node of the TCP data flow, the frames are assembled and delivered to the transport layer if the number of errors in each frame is correctable by FEC. Otherwise, the whole packet is discarded. Note that our TCP-AFEC protocol does not attempt to retransmit the error frames here. However, the performance of a protocol that supports frame retransmission is an interesting avenue for future research. The size of TCP acknowledgment packets is very small (about 40 bytes), with FEC, these ack packets are much less prone to bit errors than the large data packets. In this work, we assume these ack packets are not subject to errors in the network.

A typical IP header is about 20 bytes. For a packet of 1500 bytes, the typical MTU for wired LAN environment (IEEE 802.3), the header only constitutes a small portion (1.3 %) of the original packet. However, if the normal fragment header operation is to be used in our approach, which fragments a data packet into small frames, the header overhead will not be negligible. For a frame with data size of 175 bytes, the IP header constitutes 11.4% of the total frame size. TCP/IP Header compression [19][20] can reduce the header size by an order of magnitude down to 3-6 bytes while yielding a performance very close to ideal case across a wide range of bit error rates. In this work, we employ the header compression technique in the fragmentation process and assume that it can achieve the ideal performance, where the correct header information can always be constructed for each packet.

Given the estimate of PER and RTT, we can compute the achievable TCP goodput $G_f(N)$ using (1). Assume the raw link bandwidth of the wireless channel is B_c, the effective link bandwidth, G_c, is computed as the portion of bandwidth that is used to carry real payload scaled by the percentage of successful transmissions ($1 - PER$). Since each TCP packet is fragmented into frames of size N, of which K symbols are used for real data. The effective link bandwidth is

$$G_c(N) = B_c \frac{K}{N}(1 - PER) \tag{2}$$

The real TCP goodput (T_{tcp}) is the minimum of the achievable TCP goodput (G_f) and effective link bandwidth (G_c), i.e.,

$$G_{tcp}(N) = \min(G_c(N), G_f(N)) \tag{3}$$

The optimal Reed-Solomon code (N_0, K) is the code that maximizes TCP goodput and is computed as follows.

$$N_0 = \arg\max_N(G_{tcp}(N)) \tag{4}$$

As explained earlier, the TCP throughput is maximized when the achievable TCP throughput equals the effective channel bandwidth. Therefore, ideally N_0 is just the solution to the equation $G_c(N) = G_f(N)$. Note that G_c is a decreasing function of N while G_f is an increasing function of N. Hence, there is a unique solution N_0 to $G_c(N) = G_f(N)$. However, for an (N, K) Reed-Solomon code, N can only take integer numbers within a certain range. For trellis-based codes, the code rate can only be chosen from an even smaller set of values, for example, rates 3/4, 2/3, 1/2, 1/3 and etc. The protocol uses a look-up table generated *a priori* to find the code that yields the largest goodput as a function of the SINR and velocity estimates.

Compared to other related work, one of the key features of TCP-AFEC is that the protocol takes a formula-based approach to analytically derive the optimal FEC that maximizes TCP goodput. The required modifications to implement TCP-AFEC include some link layer operations at the base station and mobile host. Since the modified link layer operations are transparent to the TCP at the end hosts, the end-to-end semantics of TCP is preserved.

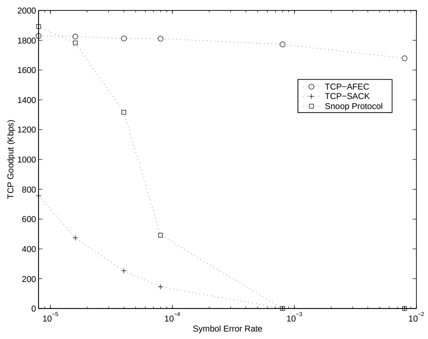


Fig. 1. Goodputs of TCP-AFEC, TCP-SACK and Snoop with no FEC

III. Performance Evaluation

A. Simulation Model

In our simulation model, a fixed host in the wired network sends data to a mobile host via a base station. The path from the fixed host to the base station is modeled as a link with a bandwidth of B_1 and one-way propagation delay of d_1. The wireless link is modeled as an erroneous link with bandwidth of B_2 and delay of d_2. Since our focus is on the erroneous nature of wireless links, we assume there is no loss on the wired link and the buffer at the base station is large enough that there is no buffer overflow. We also assume the bottleneck of the TCP session is not in the wired part, but lies on the wireless link, i.e., $B_2 < B_1$.

In the experiments, a TCP source at the fixed host has an infinite amount of data to send. Each scenario was simulated for one hour and the average TCP goodput was then measured for the duration. The symbol length is set to be 8 bits, i.e., a byte. The number of data bytes for each frame is chosen to be 175 bytes. We use a compressed TCP/IP header of 5 bytes for each frame. The total frame size N can vary from 180 to 255 bytes.

B. Simulation Results

We now present simulation results for two different wireless channel scenarios. The first scenario assumes that the symbol errors in the wireless channel form an i.i.d. Bernoulli process with an average rate of symbol error rate (SER), when conditioned on SINR. This approximates a rapidly varying mobile radio channel and provides a baseline for more realistic rapid varying channel scenarios. The second scenario adopts more modest mobility assumptions for the user, which results in a correlated SER process. Both shadow and multipath fading effects are modeled by the well-established results in wireless communications[21]. The signal strength due to the shadowing is assumed to have marginal distributions that are lognormal, and the autocorrelation function of the underlying normal process (used to generate the log-normal shadowing) is assumed to be exponential [22]. The multipath fading is modeled as a complex Gaussian random process with zero mean and autocorrelation function given by the standard “Jakes” model [23], [24]; note that this implies a Rayleigh fading channel.

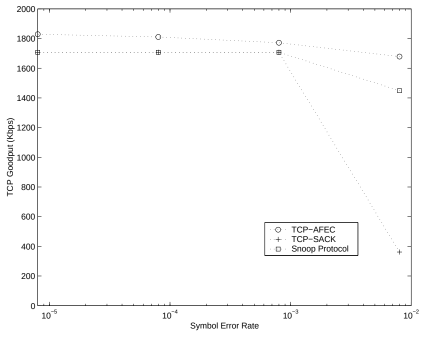
1) High-Mobility Channel: In this scenario, for the link in the wired network, we set the bandwidth (B_1) and one-way propagation delay (d_1) to be 10 Mbps and 40 ms, respectively. For the wireless channel, we set the bandwidth (B_2) to be 2 Mbps and one-way propagation delay (d_2) to be 4 ms. A wide range of symbol error rates, from 8×10^{-6} to 8×10^{-3}, were used in the experiments. The range of symbol error rates is typical for wireless links, ranging from good to bad conditions. The packet size is fixed at 1460 bytes.

Figure 1 shows the goodputs of TCP-AFEC, TCP-SACK and Snoop as a function of symbol error rate. Here no FEC is used in TCP-SACK and Snoop. Comparison of TCP-AFEC with TCP-SACK and Snoop of two fixed coding overhead follows shortly after in this section. The symbol error rates, ranging from 8×10^{-6} to 8×10^{-3}, produce packet error rates ranging from 1.2 % to 99 % in TCP-SACK and Snoop.

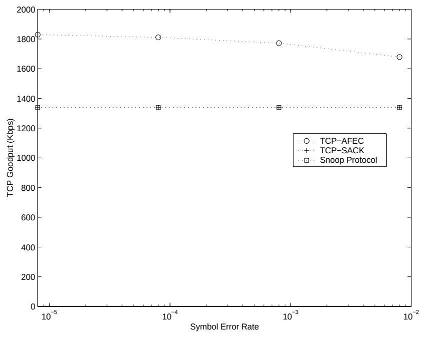
For a symbol error rate of 8×10^{-6}, TCP-SACK achieves an average goodput of 756.3 Kbps, while TCP-AFEC and Snoop yield similar goodputs around 1.8 Mbps, an improvement of 140 % over TCP-SACK. Note that in this case both TCP-AFEC and Snoop achieve a goodput close to the link bandwidth of 2 Mbps. As the symbol error rate increases to 8×10^{-5}, the packet error rate for TCP-SACK and Snoop grows to 11.3 %. In this case, the goodputs of both TCP-SACK and Snoop drop drastically to 146 Kbps and 492 Kbps, respectively, while Snoop still outperforms TCP-SACK by a large margin. For the same symbol error rate, the goodput of TCP-AFEC only suffers a slight degradation, providing an eleven-fold gain over TCP-SACK and nearly a three-fold gain over Snoop. When the symbol error rates become even higher (8×10^{-4}, 8×10^{-3}), the transmission of both TCP-SACK and Snoop stalls due to the TCP congestion control. However, TCP-AFEC still maintains a high goodput above 1.6 Mbps, much larger than those of TCP-SACK and Snoop. The optimal FEC codes of the four SER are (175, 198), (175, 188), (175, 184), (175, 182), respectively. Note that as the channel becomes less erroneous, less redundancy is need to achieve maximum TCP goodput.

In the above comparison, no FEC is employed by TCP-SACK and Snoop. Now we investigate the performance of TCP-SACK and Snoop with a fixed FEC. We considered two different coding overheads, 5 % and 30 %, representing light and medium levels of code redundancy. The resulting Reed-Solomon codes are (190, 175) and (255, 175), respectively.

The goodput of TCP-SACK and Snoop under the two fixed coding overhead are compared with the goodput of TCP-AFEC in Figure 2. From Figure 2 (a), we observe that the 5 % overhead FEC significantly improves the goodput of TCP-SACK and Snoop for all four symbol error rates. The goodputs of TCP-SACK and Snoop for symbol error rates 8×10^{-6}, 8×10^{-5}, 8×10^{-4} are now only about 6.7 % smaller than those of TCP-AFEC. However, for the symbol error rate of 8×10^{-3}, TCP-AFEC still outperforms Snoop and TCP-SACK by 13.7 % and 78.5 %, respectively. This is because the packet error rate of TCP-SACK and Snoop is still 3.5 % after the application of 5 % overhead FEC. The packet errors cause a substantial performance degradation for TCP-SACK. Snoop is able to provide considerable improvement, but still falls below TCP-AFEC. For symbol error rates from 8×10^{-6} to 8×10^{-4}, the 5 % overhead FEC reduces the packet errors rates to such small values that goodputs of TCP-SACK and Snoop have reaches the limit of effective link bandwidth.



(a) Reed-Solomon code(175, 190)



(b) Reed-Solomon code(175, 255)

Fig. 2. Goodputs of TCP-AFEC with TCP-SACK and Snoop with fixed FEC

The goodputs of TCP-SACK and Snoop under 30 % overhead FEC are shown in Figure 2 (b). In this case, the redundancy level is large enough to account for the high symbol error rate 8×10^{-3}. However, the large coding overhead is an overkill for the low symbol error rates, resulting in smaller goodputs than those under 5 % overhead FEC.

2) Low-Mobility Channel: For the low-mobility channel, the wired link parameters are set as $B_1 = 10Mbps$ and $d_1 = 10ms$. The wireless channel model is characterized by the following parameters: average signal to interference and noise ratio (SINR) to represent the average received signal strength after path loss; mobile speed (v) and correlated distance (D) to characterize the shadowing effect; Doppler frequency (f_D) to characterize multipath fading effect. Note that the mobility of the system is characterized by the model parameter v, while the packet errors account for both the multipath fading and shadowing effect. In the simulation, the parameters used are as follows: average $SINR$ (averaged over both the multipath fading and the path-loss/shadowing) = 25 dB, v = 3 m/s, D =10 m, f_d = 6 Hz, and symbol rate (link bandwidth) B_2 = 20000 symbols/s = 160 Kbps. The propagation delay for the wireless link (d_2) is set to be 80 ms. The packet size is 512 bytes.

To determine the potential of the TCP-AFEC protocol in such a scenario, we assume TCP-AFEC can perfectly adapt to channel conditions at any time scale, i.e., it can choose the code that maximizes goodput for any time interval. We divide a one-hour long symbol error trace obtained from simulation into fixed length segments. We shall refer to these segments as “adaptation intervals”. We then choose, for each adaptation interval the code that maximizes the goodput within that interval. The

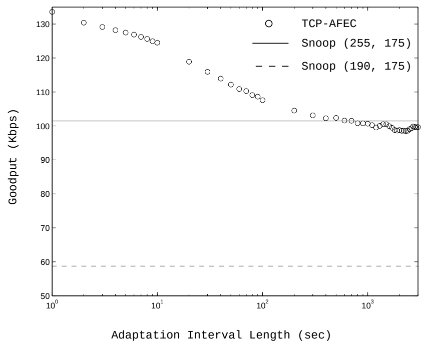


Fig. 3. Goodputs of Idealized TCP-AFEC and Snoop

overall TCP-AFEC goodput is the sum of the optimal goodputs of all the adaptation intervals. Figure 3 shows the goodputs of idealized TCP-AFEC for different values of the length of the adaptation interval and the goodput of Snoop using fixed coding overheads of 5% and 30%.

Since TCP-AFEC maximizes its goodput at the time scale of the adaptation interval, we expect that smaller adaptation interval length results in better performance for TCP-AFEC . This is validated by the simulation results shown in Figure 3. We observe as adaptation interval increases, the TCP-AFEC goodputs decreases.

At 5% coding overhead, Snoop achieves a goodput of 58.2 Kbps, which is much smaller than the goodputs of TCP-AFEC at all adaptation intervals. At 30% coding overhead, Snoop achieves a goodput of 101.4 Kbps. We observe for adaptation intervals larger than 500 seconds, the goodputs of TCP-AFEC and Snoop are similar; for adaptation intervals smaller than 500 seconds, TCP-AFEC outperforms Snoop.

Note that the above results are for an idealized situation, where TCP-AFEC can perfectly adapt to channel conditions at any time. In practice, a base station only has an estimate of the current channel conditions. The performance of TCP-AFEC depends on how well it can predict the channel conditions for the future adaption intervals. We expect that the actual results will be quite close to the idealized performance, particularly for smaller adaptation intervals, which will be on the order of milliseconds rather than seconds. The design and performance of the prediction scheme are parts of the on-going efforts.

IV. Conclusions

In this paper, we presented a novel technique (TCP-AFEC) to improve TCP performance over wireless networks using adaptive forwarding error correction. The protocol integrates the TCP performance characterization at the transport layer with link layer error control schemes. A well-established TCP goodput formula is combined with the error correction power of FEC to select the optimal code that maximizes TCP goodput for different channel conditions. Through simulation, we show that TCP-AFEC outperforms TCP-SACK and Snoop over a wide range of wireless channel conditions. We consider several practical issues in [25], such as the TCP session RTT estimation at the base station, the TCP packet size selection. To demonstrate the benefit of combining TCP performance characterization with link layer error control schemes, we also compared the performance of TCP-AFEC with a physical layer optimization scheme in [25]. Without TCP performance knowledge, the physical layer optimization scheme yields smaller goodputs than TCP-AFEC.

References

[1] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[2] Ajay Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *15th International Conference on Distributed Computing Systems*, 1995.

[3] A. DeSimone, M. Chuah, and O. Yue, "Throughput performance of transport-layer protocols over wireless LANs," in *Proceedings of Globecom*, 1993.

[4] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin, "AIRMAIL: A link-layer protocol for wireless networks," *ACM Wireless Networks*, vol. 1, no. 1, pp. 47–60, 1995.

[5] Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks*, vol. 1, no. 4, 1995.

[6] David Eckhardt and Peter Steenkiste, "A trace-based evaluation of adaptive error correction for a wireless local area network," *Mobile Networks and Applications*, vol. 4, no. 4, pp. 273–287, 1999.

[7] A. Chockalingam, M. Zorzi, and V. Tralli, "Wireless TCP performance with link layer FEC/ARQ," in *Proceedings of IEEE ICC'99*, 1999.

[8] Karen L. Gray and Daniel L. Noneaker, "The effect of adaptive-rate coding on TCP performance in wireless communications," *Proc. IEEE/AFCEA EuroComm 2000 Conference*, pp. 233–237, 2000.

[9] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, April 2000.

[10] Chadi Barakat and Eitan Altman, "Bandwidth tradeoff between TCP and link-level FEC," in *Proceedings of IEEE International Conference on Networking*, Colmar, France, Jul 2001.

[11] J. Bolliger, T. Gross, and U. Hengartner, "Bandwidth modelling for network-aware applications," in *Proceedings of Infocomm'99*, 1999.

[12] R. Yates, "A framework for uplink power control in cellular radio systems," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1341–1347, 1995.

[13] V. Veeravalli and O. Kelly, "A locally optimal handoff algorithm for cellular communications," *IEEE Trans. Veh. Technol.*, pp. 603–609, Aug 1997.

[14] S. Nanda, K. Balachandran, and S. Kumar, "Adaptation techniques in wireless packet data services," *IEEE Communications Magazine*, Jan 2000.

[15] D. Goeckel, "Adaptive coding for time-varying channels using outdated fading estimates," *IEEE Trans. Commun.*, vol. 47, pp. 844–855, June 1999.

[16] S. Wei and D. Goeckel, "Error statistics for average power measurements in wireless communication systems," *to appear in IEEE Trans. Commun.*, 2002.

[17] R. Narasimhan and D. Cox, "Speed estimation in wireless systems using wavelets," *IEEE Trans. Commun.*, vol. 47, pp. 1357–1364, Sept. 1999.

[18] R. Narasimhan and D. Cox, "Estimation of mobile speed and average received power in wireless systems using best basis methods," in *33rd Asilomar Conf. on Signals, Systems & Computers*, 1999, pp. 300–305.

[19] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links," *RFC 1145*, 1990.

[20] Mikael Degermark, Mathias Engan, Bjorn Nordgren, and Stephen Pink, "Low-loss TCP/IP header compression for wireless networks," in *Mobile Computing and Networking*, 1996, pp. 1–14.

[21] Theodore S. Rappaport, *Wireless Communications: Principle and Practice*, Prentice Hall, 1996.

[22] M. Gudmundson, "Correlation model for shadow fading in mobile radio systems," *Electron. Lett.*, vol. 27, pp. 2145–2146, Nov. 1991.

[23] W. Lee, *Mobile Communications Engineering*, McGraw-Hill, 1998.

[24] W. C. Jakes, Ed., *Microwave Mobile Communications*, IEEE Press, 1994.

[25] Benyuan Liu, Dennis L. Goeckel, and Don Towsley, "TCP-cognizant aaptive forward error correction in wireless networks," Tech. Rep., Computer Science Dept - University of Massachusetts, Amherst, http://gaia.cs.umass.edu/ benyuan, 2001.