Wireless Networks 5 (1999) 279–286

An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols

Eytan Modiano *

MIT Lincoln Laboratory, Lexington, MA 02420-9108, USA

We develop an algorithm that allows an ARQ protocol to dynamically optimize the packet size based on estimates of the channel bit-error-rate. Our algorithm is particularly useful for wireless and satellite channels where the bit-error-rates tend to be relatively high and time variable. Our algorithm uses the acknowledgment history to make estimates of the channel bit-error-rate, based on which the optimal packet size can be chosen. We develop a Markov chain model for the analysis of the system, under static channel conditions, and show that the algorithm can achieve close to optimal performance using a history of just 10,000 bits. We also use the Gilbert–Elliott two-state Markov channel to model dynamic channel conditions. We show, through simulation, that the algorithm performs well even under rapidly changing channel conditions. Finally, we discuss a maximum likelihood approach for choosing the packet size, which performs almost optimally but is much easier to implement.

1. Introduction

Automatic Repeat Request (ARQ) protocols are designed to remove transmission errors from data communications systems. When used over relatively high bit-error-rate links (e.g., 10^{-5} or higher) such as wireless or satellite links, their performance is sensitive to the packet size used in the transmission. This is because when too large a packet size is employed, there is an increased need for retransmissions, while too small a packet size is inefficient because of the fixed overhead required per packet. When an ARQ scheme is to be used at the link layer over a relatively high error-rate link, the packet size should be chosen based on the error-rate. When a perfect retransmission algorithm is employed,[1] the optimal packet size to be used by the data link protocol is given by [5,8]

$$k_{\text{opt}} = \frac{-h\ln(1-p) - \sqrt{-4h\ln(1-p) + h^2\ln(1-p^2)}}{2\ln(1-p)}, \tag{1}$$

where p is the known channel bit-error-rate and h is the number of overhead bits per packet.[2] Unfortunately, however, it is often not possible to know the channel bit-error-rate in advance; further, for some channels the bit-error-rate varies with time. This is particularly the case for radio and satellite channels where signal fading and interference are unpredictable and time varying. There has been some work on developing link layer ARQ schemes that are particularly efficient for high-error-rate channels [6,7]. In [9] a Go-Back-N ARQ scheme is developed that alters the mode of operation based on estimates of channel conditions. The approach in [9] uses a multiple copy transmission mode when the channel appears to be error prone and ordinary Go-Back-N when the channel conditions appear to be good. Another common approach to this problem is to vary the channel transmission rate based on the quality of the link, so that a constant, acceptable, bit-error-rate is obtained. Here we are concerned with situations where that approach is not available and we focus on adapting the size of the packets based on estimates of the channel conditions.

Designers of data link protocols for wireless channels typically choose a packet size that would work with the worst acceptable bit-error-rate. Figure 1 shows the optimal packet size for different bit-error-rates. As can be seen from the figure, with a bit-error-rate of 10^{-3} the optimal packet size is about 200 bits. If a much larger packet size were used the efficiency of the protocol would drop dramatically. Therefore, in order to operate at a bit-error-rate of 10^{-3} a packet size of a few hundred bits should be used.

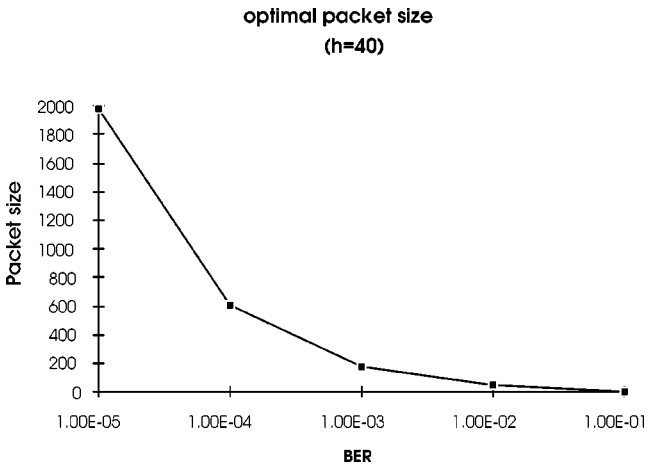

Figure 1. Optimal packet size vs. bit-error-rate.

* Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Air Force.

[1] A perfect retransmission algorithm is one that only retransmits packets that are in error and can continuously transmit new packets as long as no errors occur. The selective repeat protocol [1] is an example of a perfect retransmission algorithm.

[2] These bits are used for control, error detection, and framing (e.g., flags).

Unfortunately this packet size makes inefficient use of the channel when the channel bit-error-rate is much lower.

The algorithms developed in this paper choose the packet size based on the acknowledgment history of the most recently transmitted packets. Given the number of packets that required retransmission, an estimate of the channel bit-error-rate is made, based on which a packet size is chosen to maximizes the expected efficiency of the data link protocol. All of the algorithms in this paper assume that the sender uses a framing mechanism that accommodates variable packet sizes, as is the case for most link-layer framing mechanisms used in practice [1]. For example, the High-Level Data Link Control protocol (HDLC) uses flags for framing and the packet size can be arbitrary [8]. The receiver determines packet boundaries by searching for successive flags in the data. Hence, no additional communication is required between the sender and the receiver for the purpose of coordinating the packet size. Of course, a framing overhead is always incurred and is accounted for as a part of the fixed packet overhead (h).

This paper is organized as follows: In section 2 we describe an algorithm for choosing the packet size in such a way that the efficiency of the protocol is maximized. In section 3 we develop a Markov chain analysis for the algorithm assuming static channel conditions and in section 4 we analyze the performance of the algorithm using a time varying channel model. Finally, in section 5, we discuss a simpler algorithm for choosing the packet size based on maximum likelihood estimates of the channel bit-error-rate.

2. A maximum efficiency approach to choosing the packet size

Given the retransmission history we wish to choose the packet size such that the conditional efficiency of the protocol is maximized. To do so we must first derive an expression for the efficiency of the ARQ protocol given the retransmission history. The expressions derived in this section assume the use of an "optimal" ARQ protocol in that only packets containing errors are retransmitted. This assumption makes for a reasonable approximation when using a protocol that attempts to retransmit only packets containing errors, such as the Selective Repeat Protocol (SRP) [1]. For a given channel error rate p, the efficiency of a protocol that uses packets of size k is given by [5]

$$\text{EFF} = \left(\frac{k}{k+h}\right)\frac{1}{(1-p)^{-(k+h)}}, \tag{2}$$

where k is the number of information bits, h is the number of header bits in the packet and p is the channel bit-error-rate. The first term of the above expression represents the ratio of information bits to total bits in a packet, while the second term represents the average number of transmission attempts per packet. We can express the expected efficiency of the protocol given, R, the number of retransmission requests out of the last M packet transmissions, by averaging the above expression over all possible values of p and using the conditional distribution of p given R (assuming that p is constant over the period of interest). The resulting expression is given by

$$\text{EFF}_R(k) = \int_p \frac{k(1-p)^{k+h}}{(k+h)} P(p \mid R), \tag{3}$$

where $P(p \mid R)$, is the conditional probability of p given that R out of the last M packets required retransmission. We now wish to choose the value of k that maximizes EFF_R. To do so we must first express the conditional probability of p given R.

The conditional probability of p given R, $P(p \mid R)$, can be expressed as follows:

$$P[p \mid R] = \frac{P[p, R]}{P[R]} = \frac{P[R \mid p]P[p]}{P[R]}. \tag{4}$$

Solving for the above conditional probability requires knowledge of a prior distribution of p. In the absence of a prior, we assume a uniform prior; that is $P[p] = 1$. Intuitively this amounts to having no prior knowledge of p. This approach, in essence, is the same as a maximum likelihood approach where a uniform prior is assumed, except that here we associate a cost function with the estimates of p. With this approach we get

$$P[R] = \int_p P[R \mid p]P[p] = \int_p P[R \mid p]$$

and so

$$P[p \mid R] = \frac{P[R \mid p]}{\int_p P[R \mid p]}. \tag{5}$$

Given p, the probability that R packets contain errors and therefore require retransmission is the probability that R out of M packet are in error. Since packet errors are independent from packet to packet, this probability can be expressed according to the binomial distribution with parameter E. E is the probability that a packet contains errors and is given by

$$E = 1 - (1-p)^{\hat{k}+h}, \tag{6}$$

where $\widehat{k}$ is the packet size used in the previous M transmissions. Therefore $P[R \mid p]$ can now be expressed as

$$P[R \mid p] = \binom{M}{R} E^R (1-E)^{M-R}. \tag{7}$$

Combining equations (3)–(7), we can express the expected efficiency of the protocol for a given value of R by

$$\text{EFF}_R(k) = \int_p \left[\frac{k(1-p)^{k+h}}{(k+h)} \times \frac{\binom{M}{R}E^R(1-E)^{M-R}}{\int_p \binom{M}{R}E^R(1-E)^{M-R}}\right]. \tag{8}$$

It is now possible to choose the value of k, the block size to be used in future transmissions, so that the efficiency of the protocol is maximized. This can be done by choosing the value of k that maximizes equation (8) for a value of R that is equal to the number of retransmission requests that

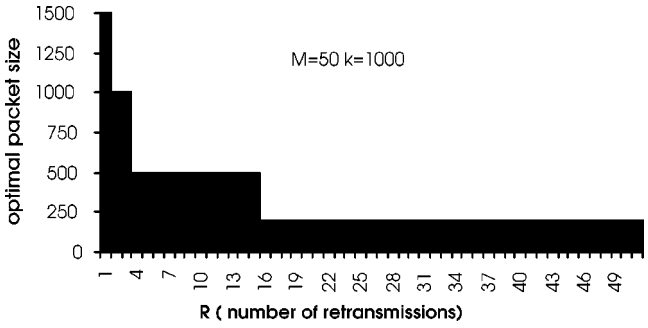

Figure 2. Optimal packet size based on retransmission history of the 50 previous 1000 bit packets.

occurred during the previous M transmissions. Of course, the value of E used in equation (8) should be chosen in accordance with equation (6) using the packet size $\widehat{k}$ used in the previous M transmissions.

A general, closed form, solution to the above maximization problem is difficult to obtain; however, for specific values of M, R and $\widehat{k}$ equation (8) can be solved numerically. An optimal value for k can now be found using numerical search algorithms. Since the numerical evaluation of this integral is very intensive, a comprehensive search for the optimal value of k is not practical. Instead, a restricted search using select values for k can be performed. Such a search, for example, can consider values of k that are a multiple of 100; thereby significantly reducing the complexity of the search. Such a restricted search has little impact on the performance of the protocol since values of k that are within 100 bits of the optimal block size should result in near-optimal performance.

As an example, in figure 2 we plot the optimal packet size when a history of 50 previously transmitted 1000 bit packets is considered.

As can be seen from the figure, when the previous fifty transmissions resulted in no errors the packet size can be increased to 1500 bits. When one, two or three errors occurred the packet size can be kept at 1000 bits and when more than three errors occur the packet size is reduced.

3. The performance of the algorithm

It is interesting to compare the performance of this algorithm to that obtained when the channel error probability is known and the optimal packet size is chosen accordingly. The performance of a single iteration of the algorithm can be easily evaluated as follows. For a given value of p, the distribution of R is expressed by equation (7) and the performance of the algorithm can be evaluated. Let $k_{\text{opt}}(R)$ be the optimal value of k chosen by the algorithm for a given value of R. The efficiency of the algorithm with that value of k can be computed according to equation (2). It can then be averaged over the distribution of R given p to yield the performance of the algorithm for a given value of p. Figure 3 shows the performance of the algorithm with various values of M and p, and a previous packet size of

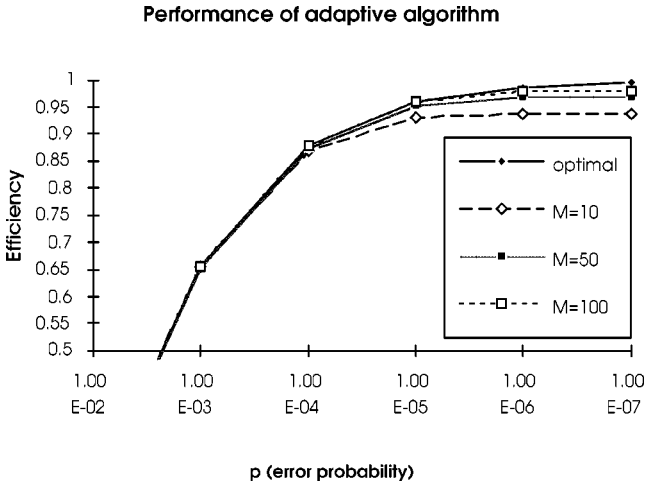

Figure 3. Performance of algorithm for various values of p.

1000 bits. The curve labeled optimal represents the performance of an algorithm that uses the optimal value of k for the given value of p.

As can be seen from the figure the algorithm produces near-optimal performance for values of p that are greater than 10^{-4}, with a history of just 10 packets. This is because when p is large even a history of 10 packets (10,000 bits) yields a sufficient number of errors to provide a reasonable estimate of p. When p is smaller, more of a history is required to obtain a reasonable estimate. With values of p around 10^{-7} a history of 100 packets is required to obtain performance that is close to the optimal. However, it is important to note that the performance of ARQ algorithms is much more vulnerable to the packet size when the probability of error is high. That is, when the probability of error is high the use of a large packet size can have a disastrous effect on the performance of the algorithm; however when the probability of error is low, small variations in the packet size from the optimal size have minimal effects.

The above analysis considers only a single iteration of the algorithm. Figure 3 shows the performance of the algorithm with a previous packet size and a history of 10, 50 and 100 packets. Of course after this single iteration a new packet size would be chosen and the next iteration would use that new packet size. While the above analysis shows the efficiency of the protocol with the new packet size, it does not take into account the next iteration of the algorithm. That is, the above analysis does not compute the steady state performance of this algorithm.

To evaluate the steady state performance of the algorithm we develop a Markov chain model for the system. The state of the system is described by the packet size being used. Since packet sizes can change during every iteration, state transitions occur at the end of every iteration. That is, a state transition can occur after a history of M packets is observed. In order to keep the Markov chain finite, we must limit the range of packet sizes. This, in fact, poses no real limitation on the performance of the algorithm because in practice a limit on the packet size is

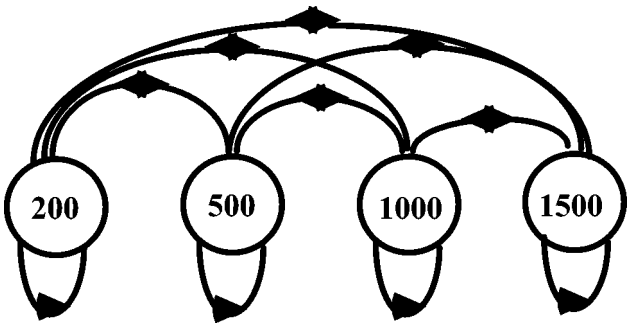

Figure 4. The system Markov chain with four states representing packet sizes of 200, 500, 1000 and 1500 bits.

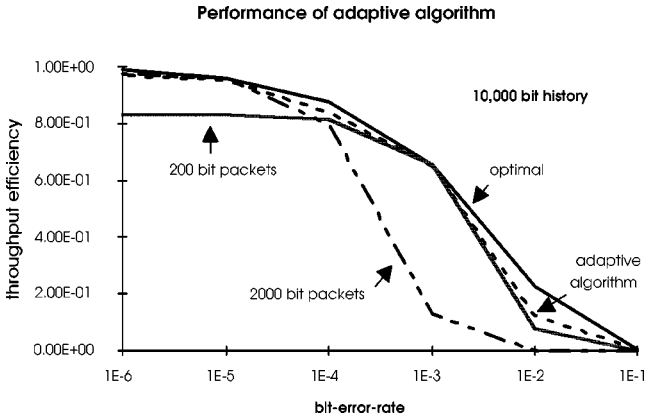

Figure 5. Performance of adaptive algorithm for a 10,000 bit observation history.

usually imposed. Figure 4 shows an example of the system state Markov chain with 4 states representing a packet size range of 200 bits to 1500 bits. As can be seen from the figure, a transition is possible between any two states.

The transition probability between any pair of states can be easily obtained from equations (3)–(7), where the present state is used for the previous packet size and the probability of moving to any other state is equal to the probability of having a value of R that would result in the corresponding new packet size. So, for example, looking at figure 2 which represents transitions from state 1000, the probability of a transition from state 1000 to state 1500 is the probability of having 0 or 1 packet errors in the previous 50 transmissions. That probability is easily computed from equation (7). Once the transition probabilities between the different states is determined the steady-state probability of being in the different states can be easily computed by solving the steady-state equation $\overline{P} \times p_{\rm s}$, where $\overline{P}$ is the N by N state transition probability matrix, $p_{\rm s} = (p_1 \cdots p_N)$ is the steady-state probability distribution of being in the different states, and N is the number of states in the system. This equation is solved by solving the above N linear equations together with the fact that the sum of the state probabilities is equal to 1. Once this distribution is known the steady-state performance is given by

$$\text{EFF}_{\text{steady-state}} = \sum_{i=1}^{N} p_i \left(\frac{k_i}{k_i + h} \right) \frac{1}{(1-p)^{-(k_i+h)}}, \qquad (9)$$

where p_i is the probability of being is state i, k_i is the corresponding packet size, p is the channel bit-error-rate and h is the size of the fixed packet overhead.

This Markov model can now be used to analyze the steady-state performance of the algorithm We apply this analysis to a system with packet sizes ranging from 10 to 2000 bits in 10 bit increments. We limit the packet size to 2000 bits because there is very little performance improvement from using packets larger than 2000 bits. In fact, 2000 bit packets are optimal for a bit-error-rate of 10^{-5} and result in a throughput of at least 96% when the bit-error-rate is smaller than 10^{-5}. The Markov chain for this system has 200 states, corresponding to packet sizes in the range of 10 to 2000 bits. We analyzed the steady-state behavior of this Markov chain for bit-error-rates between 10^{-1} and 10^{-6}. Of course the performance is also a function of the amount of transmission history that we observe during each iteration. In figure 5 we plot the system performance vs. the channel bit-error-rate for a transmission history of 10,000 bits and in figure 6 we plot the system performance for a transmission history of 100,000 bits.[3]

As can be seen from both figures the adaptive algorithm performs very close to optimally with a history of just 10,000 bits. We notice that when the bit-error-rate is very low or very high the adaptive system performs slightly worse than optimal. While when the error-rate is in the "middle range" the performance of the algorithm is nearly optimal. We attribute this difference at the very low error-rate to the fact that at such error-rates it is difficult to make a reliable estimate of the error-rate with just 10,000 or even 100,000 bits of observation. At the high error-rate range the difficulty in making a reliable estimate is that packets are too large to make such an "unbiased" estimate. This is because every packet contains 40 bits of header information and so a packet cannot be made smaller than 50 bits. Nonetheless, we see that the system performs very well even in the absence of very good bit-error-rate estimates at the high and low error-rate ranges. This is because the performance of the algorithm in these ranges is not very sensitive to the choice of packet size. For example, the optimal packet size with an error-rate of 10^{-5} is 2000 bits while with a rate of 10^{-6} it is about 6000 bits. However, when a 2000 bit packet is used with an error-rate of 10^{-6} the resulting efficiency is still better than 98%. Consequently, it is not necessary to obtain a very good estimate of the error-rate but rather it is sufficient to establish that the error-rate is in the low range or the high range. This can be observed by looking at figure 7 where the average packet size used by the algorithm is plotted against the error-rate.

[3] We express the size of the transmission history in bits rather than packets because in different states the packet size changes. We wanted to maintain the amount of time between state transitions constant and so we fixed the transmission history in bits. The number of packets observed between transitions, M, is equal to the smallest integer greater than the transmission history divided by the packet size in the corresponding state.

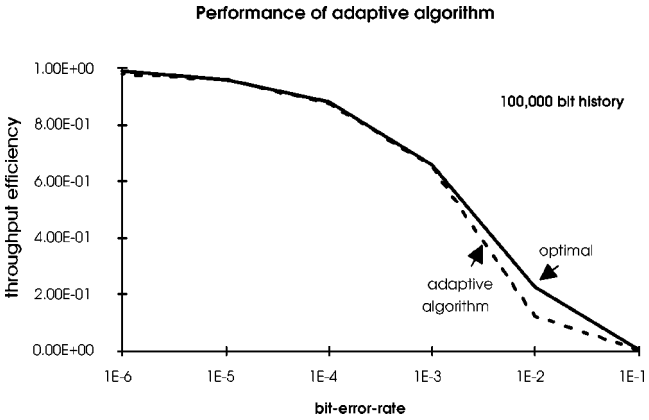

Figure 6. Performance of adaptive algorithm for a 100,000 bit observation history.

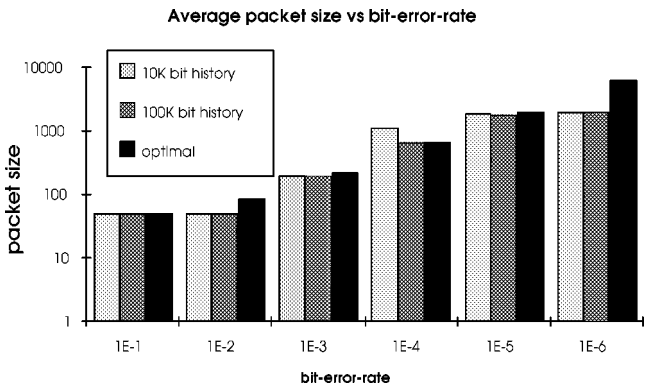

Figure 7. Average packet size used by the adaptive algorithm.

Of course, these figures only represent the system's performance when the bit-error-rate is not changing. However, in a wireless environment channel conditions are often time-varying. In the next section we analyze the performance of the algorithm under time varying channel conditions using a two-state Markov channel model.

4. Performance analysis under changing bit-error-rate conditions

The performance analysis of the previous section assumed that the channel bit-error-rate remained constant. However, over a wireless medium channel conditions are likely to be time varying due to fading. We model the time varying channel using the well known Gilbert–Elliott two-state Markov model [3,4]. The Gilbert–Elliott channel is a two-state Markov chain, where each state represents a binary symmetric channel (BSC), as shown in figure 8. In the "good" state the BSC cross-over probability, $P_{\rm G}$, is low and in the "bad" state the cross-over probability, $P_{\rm B}$, is high. Transitions between the two states occur according to an exponential random process of rate $\mu_{\rm G}$ for transitions from the "good" state to the "bad" state and $\mu_{\rm B}$ for transitions from the "bad" state to the "good" state. That is, the amount of time that the channel remains in a given state is exponentially distributed with an average value of $1/\mu_{\rm B}$

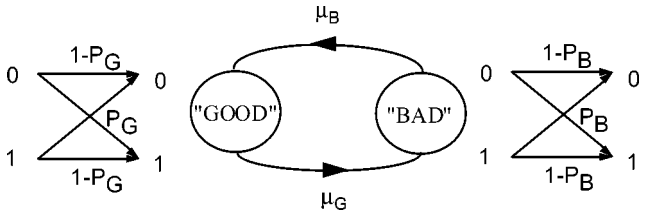

Figure 8. The Gilbert–Eliott channel model.

for the "bad" state and $1/\mu_{\rm G}$ for the "good" state. This model is commonly used to model fading channels such as the Rayleigh channel [2,10]. A high bit-error-rate while in the "bad" state is used to represent a channel fade, and a lower error rate in the "good" state represents the channel under normal conditions. The transition rates between the "good" and "bad" states can be chosen according to the statistics of the actual channel being modeled, where the average amount of time spent in the "bad" state equals the average duration of a fade and the average amount of time spent in the "good" state equals the average amount of time between fades. This approach has been shown to accurately model the performance of a fading channel [10].

The Gilbert–Elliott model accounts for the time-correlation between packet errors that results from the channel being in a fade. Within a given channel state packet errors are independent and equations (6)–(8) hold; however, they do not hold when the channel transitions between states. This is because when the channel is allowed to vary packet errors are no longer independent. Nonetheless, the algorithm of the previous section and equations (6)–(8) can still be used to select a packet size. Of course, when errors are not independent using this approach will sometimes (e.g., during state transitions) yield poor error rate estimates and result in the use of a sub-optimal packet size. Therefore, it is reasonable to expect that as long as channel transitions do not occur too frequently the algorithm will usually predict an appropriate packet size. It is during channel transitions that the algorithm is likely to use a sub-optimal packet size. In this section we wish to analyze the impact of these channel transitions on the overall performance of the algorithm.

In order to analyze the performance of the adaptive algorithm with the above channel model we resort to simulation. For the simulation we assume that in the "good" state the bit-error-rate is 10^{-5} ($P_{\rm G} = 10^{-5}$) and in a "bad" state the bit-error-rate is 10^{-3} ($P_{\rm B} = 10^{-3}$). We also assume that the rate of transition between the two states is the same in both directions ($\mu = \mu_{\rm G} = \mu_{\rm B}$) with an average amount of time between state transitions $\rho = 1/\mu$.[4]

In figure 9 we plot the performance of the algorithm using a history of 50 packets ($M = 50$) and for values of ρ between 0 and 5 seconds over a channel with transmission rate $R = 100,000$ bits-per-second (bps). We should point out, however, that although the channel transition rate ρ is

[4] With an exponential state transition rate of μ the average amount of time spent in a state is equal to $1/\mu$.

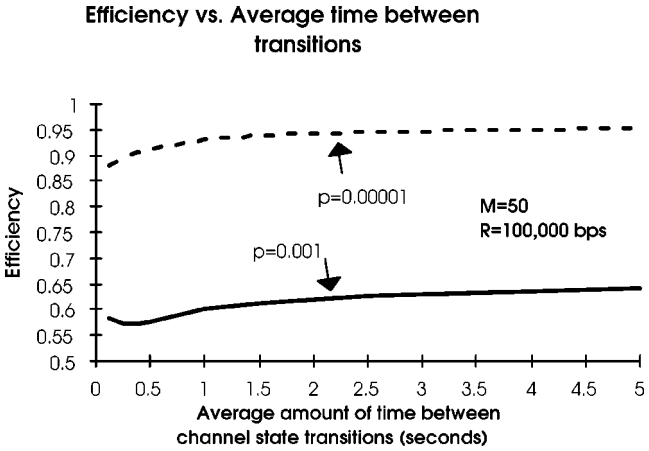

Figure 9. The efficiency of the protocol vs. the average amount of time between channel state transitions.

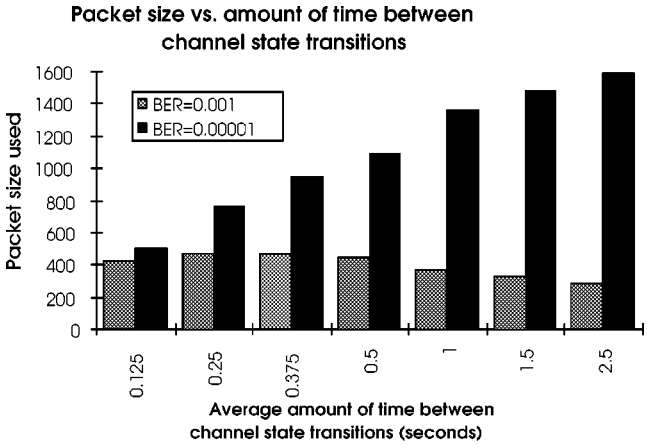

Figure 10. Average packet size used by the algorithm.

expressed in seconds, the parameter that impacts the performance of the algorithm is actually the average number of bits between state transitions (b). Of course, for a channel rate R_c, the average number of bits between channel transitions is simply $b = \rho R_c$. Hence, the results shown in the figure also apply to other combinations of b and R_c that yield the same value of $\rho = b/R_c$. For example, $\rho = 1$ second over a 100,000 bps channel represents 100,000 bits between channel state transitions and is equivalent to $\rho = 10$ seconds over a 10,000 bps channel or $\rho = 0.1$ seconds over a 1,000,000 bps channel. The performance of the algorithm is therefore the same for any combination of R_c and ρ that yield the same value of b. Consequently, we can make the interesting observation that the algorithm reacts better to channel variations with higher transmission rates.

Clearly, the throughput efficiency that the algorithm achieves depends on the state of the channel. That is, when the channel is in the "good" state we expect a much higher throughput efficiency than when the channel is in the "bad" state. We, therefore, plot the throughput efficiency for each of the two states. As can be seen from the figure the algorithm performs reasonably well even when the channel bit-error-rate is changing rapidly (e.g., $\rho = 0.5$ seconds). For comparison notice, from figure 3, that even under static channel conditions and perfect knowledge of the channel error-rate the efficiency of an optimal ARQ protocol is limited to about 0.65 with a bit-error-rate of 10^{-3} and 0.96 with a bit-error-rate of 10^{-5}. The results in figure 9 show efficiency values that approach the optimal as ρ increases. When the rate of change decreases (ρ increases), the performance of the algorithm improves as expected. The surprising result is that the overall performance of the algorithm appears reasonably good (not far from optimal performance) even when channel conditions are changing rapidly.

In figure 10 we plot the average packet size used by the algorithm vs. the average time between transitions. Again for comparison notice, from figure 1, that under static channel conditions and perfect knowledge of the channel error-rate the optimal packet size is about 200 bits with a bit-error-rate of 10^{-3} and 2000 bits with a bit-error-rate of 10^{-5}. As can be seen from the figure as ρ increases the packet size used while the channel is in the different states approaches the optimal. When the channel is in the "good" state (bit-error-rate of 10^{-5}) the average packet size used by the algorithm increases with average time between transitions. This is because as the amount of time the channel remains in the "good" state increases the packet size is able to approach the optimal packet size for the given channel error-rate. However, when the bit-error-rate changes rapidly the algorithm yields smaller packet sizes reflecting the effect of error-rate changes during some of the observation intervals. Similarly, when the bit-error-rate is 10^{-3}, the packet size decreases with average transition times. These results reflect the fact that when the average transition time is high, the algorithm can yield estimates of a stable bit-error-rate and consequently use an appropriate packet size. However, again, when the average transition time is low the bit-error-rate is likely to vary during an observation interval and consequently result in selecting a packet size that is not optimal.

5. Using maximum likelihood estimates

In section 2 we selected the packet size based on an estimate of the bit-error-rate that maximized the efficiency of the protocol. That approach, in theory, should yield the optimal results. Unfortunately, however, we were unable to obtain a closed form solution to the optimization problem and had to resort to a numerical evaluation. As a consequence we were forced to look at a restricted set of values for the packet size, somewhat limiting the performance of the algorithm. In this section we present another approach that yields a closed form, but sub-optimal, solution. This approach is based on making a Maximum Likelihood Estimate (MLE) of p, based on which the optimal packet size is chosen according to equation (1).

The MLE for p given R retransmissions is the value of p that maximizes the probability that R retransmissions are required. That is, it is the value of p that maximizes

equation (7). This maximization can be done by taking the derivative of the log of equation (7) and setting it equal to zero. Solving for p shows that, $\widehat{p}$, the maximum likelihood estimate for p is

$$\widehat{p} = 1 - \left(\frac{M-R}{M}\right)^{1/k}. \tag{10}$$

This estimate can now be combined with equation (1) to yield the optimal packet size based on the MLE for p. While this MLE yields a closed form estimate of p, it tends to overestimate p when many errors occur and underestimate p when few errors occur. For example, when no errors occur the MLE for p is zero, this estimate for p would yield a very large packet size (infinite according to equation (1)). The cost function approach of the previous section would result in a more conservative packet size. At the other extreme when all M packets are in error the MLE for p is 1. This estimate is clearly biased because any value for p that results in a probability of retransmission that is close to 1 would result in an estimate of $p = 1$. So, for example, when a packet size of 10^6 bits is used with $p = 10^{-3}$, the probability of a retransmission is close to 1, R would almost always equal N and the MLE for p would be 1. Clearly, using the MLE estimates for p to choose an optimal packet size is not appropriate. However, it may be possible to alter the MLE approach in such a way that the above biases are compensated for by restricting the packet size to a useful range of packet sizes. So, for example, if the MLE of p is 1, the algorithm can use a minimum packet size of say 200 bits and, similarly, the algorithm can eliminate very low channel error estimates. That is, when no packets are found to be in error the MLE of the bit-error-rate is zero. Theoretically, this would imply an infinitely large packet size. Since there is very little benefit from using a very large packet size, the algorithm can be designed to allow a maximum packet size of about 2000 bits which would yield a better than 98% efficiency.

An MLE approach can be used by restricting the packet size to fluctuate within a range of values that is appropriate for the application. In many cases the range of interest is between 10^{-3} and 10^{-7} with packet size range of between 200 bits and 2000 bits. The MLE can be used to provide an estimate for the bit-error-rate based on which a packet size in the above range can be chosen. The packet size is generated according to equation (1) as follows:

$$K_{\text{MLE}} = \begin{cases} 2000, & k_{\text{opt}} > 2000, \\ k_{\text{opt}}, & 200 < k_{\text{opt}} < 2000, \\ 200, & k_{\text{opt}} < 200, \end{cases}$$

where k_{opt} is the packet size generated by equation (1) using the MLE of the bit-error-rate. In figure 11 we plot the packet size vs. the number of retransmission requests. It is interesting to compare this figure to figure 2 where the same was plotted for the maximum efficiency approach. As can be seen from the two figures both approaches yield similar packet sizes and as a result we can expect the performance

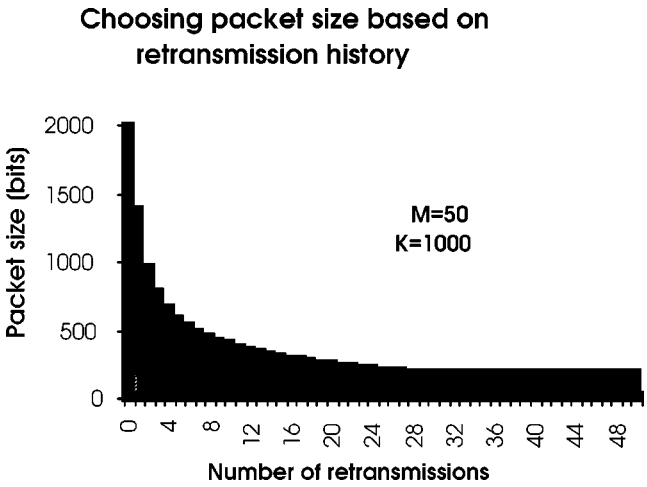

Figure 11. Optimal packet size based on retransmission history of the 50 previous 1000 bit packets.

of this "restricted" MLE approach to compare very favorably with that of the maximum efficiency approach. The main advantage of using this MLE approach is the ease with which packet sizes can be computed and the resulting simple implementation of the algorithm.

Again, the analysis leading to equation (10) assumed independent bit-errors. With a time varying channel bit-errors may no longer be independent and the analysis no longer holds. Using the Gilbert–Elliott model for a time-varying channel yields independent bit-errors within each state, yet errors are correlated over the entire channel. Hence equation (10) holds conditional on the state of the channel (i.e., while the channel remains in one state); however, it does not hold when the channel transitions between states. Nonetheless, the algorithm and equation (10) can be used to provide error-rate estimates and choose a packet size. Of course, when errors are correlated, these estimates will sometimes (e.g., during state transitions) result in the use of a suboptimal packet size. The same situation occurred with the previous algorithm and it is for that reason that we needed to resort to simulation in order to analyze the performance of the algorithm. Since both the MLE approach and the maximum efficiency approach yield similar packet sizes for a given number of retransmissions, their performance is very similar even under time-varying conditions.

6. Conclusion

This paper presents an algorithm that adapts the packet size used by a data link layer ARQ protocol based on estimates of channel conditions. The algorithm was designed to work efficiently with an "optimal" ARQ protocol such as the Selective Repeat Protocol (SRP). An interesting observation that can be made as a result of this work is that while the performance of ARQ protocols is sensitive to the packet size used at different error-rates, it is not necessary to have an accurate estimate of the channel error-rate in order to choose a "good" packet size that yields nearly op-

timal performance. It is this observation that allows our algorithm to perform very well even with a short observation history. In fact, we found that very good performance can be obtained with a history of just 10,000 bits.

An interesting aspect of the performance of the algorithm is the performance with varying channel conditions. We used a Markov channel bit-error-rate model where the channel can be in either a "good" state or a "bad" state. Simulation results indicate that the algorithm performs well even when the channel conditions are rapidly varying. However, the performance of the algorithm is best when channel conditions vary relatively slowly to the observation period. Since an observation period of just 10,000 bits can yield satisfactory results, we can conclude that the algorithm would perform well when the average time between changes in channel conditions is greater than the time it takes to transmit about 50,000 bits. So, for example, with a 9.6 Kbps channel, the bit-error-rate can change every few seconds while still allowing the algorithm to perform well. Even when the bit-error-rate changes more rapidly the algorithm yields stable and satisfactory results but not as good as those obtained under a slower rate of change.

Implementing the algorithm as described in section 2 would require a table lookup approach. This is because choosing the next packet size requires a relatively complicated computation. However, these computations can be done in advance and stored in a table indicating for a given packet size and retransmission history, what the next packet size should be. Alternatively, we found that using a MLE approach to making the error-rate estimates performs nearly as well as the optimal approach. Since the MLE approach yields simple closed form results it can be easily implemented without the need for a table lookup mechanism.

Acknowledgements

The author would like to thank Steve Bernstein of MIT Lincoln Laboratory for his useful comments and discussions regarding this work. This work was sponsored by the Department of the Army under contract F19628-95-C-0002.

References

[1] D.P. Bertsekas and R. Gallager, *Data Networks* (Prentice-Hall, Englewood Cliffs, NJ, 1987).

[2] P. Bhagwat et al., Enhancing throughput over wireless LANs using channel state dependent packet scheduling, in: *Proc. of Infocom '96*, San Francisco, CA (April 1996).

[3] E.O. Elliott, Estimates of error rates for codes on burst-noise channels, Bell System Technical Journal (September 1993).

[4] E.N. Gilbert, Capacity of a burst-noise channel, Bell System Technical Journal (September 1960).

[5] E. Modiano, Data link protocols for LDR MILSTAR communications, Lincoln Laboratory, Communications Division Internal Memorandum (October 1994).

[6] M. Moeneclaey and H. Bruneel, Efficient ARQ scheme for high error rate channels, Electronic Letter 20 (1984) 986–987.

[7] A.R.K. Sastry, Improving automatic repeat request (ARQ) performance on satellite channels under high error rate conditions, IEEE Transactions on Communications 23 (1975) 436–439.

[8] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis* (Addison-Wesley, Reading, MA, 1987).

[9] Y. Yao, An effective Go-Back-N ARQ scheme for variable-error-rate channels, IEEE Transactions on Communications 43(1) (January 1995).

[10] M. Zorzi, R.R. Rao and L.B. Milstein, On the accuracy of a first order Markov model for data transmission on fading channels, in: *Proc. of the 4th International Conference on Universal Personal Communication*, Tokyo, Japan (November 1995).

Eytan Modiano received his B.S. degree in electrical engineering and computer science from the University of Connecticut at Storrs in 1986 and his M.S. and Ph.D. degrees, both in electrical engineering, from the University of Maryland, College Park, MD, in 1989 and 1992, respectively. He was a Naval Research Laboratory Fellow between 1987 and 1992 and a National Research Council Post Doctoral Fellow during 1992–1993 while he was conducting research on security and performance issues in distributed network protocols. He joined the Communications Division of MIT Lincoln Laboratory in 1993 where he has been working on communication protocols for satellite, wireless, and optical networks. He has published over 20 papers on various aspects of data networks including multiple access, queueing systems and distributed protocols. Since 1994 he has also been an adjunct professor in the College of Computer Science at Northeastern University, where he teaches graduate level courses on data networks.