Bandwidth tradeoff between TCP and link-level FEC [†]

Chadi Barakat and Eitan Altman
INRIA
Email : {cbarakat,altman}@sophia.inria.fr

Abstract— **Forward Error Correction (FEC) is widely used for the improvement of the quality of noisy transmission media as wireless links. This improvement is of importance for a transport protocol as TCP which uses the loss of packets as an indication of network congestion. FEC shields TCP from losses not caused by congestion and helps it to improve its throughput but on the other hand it consumes a part of the available bandwidth that could be used by TCP. In this paper we study in detail this bandwidth tradeoff between TCP and FEC. By analysis and simulations we show how TCP performance varies as a function of the amount of FEC.**

I. Introduction

TCP provides a reliable transport service to many of todays Internet applications. It uses two algorithms, Slow Start and Congestion Avoidance, to avoid and control the congestion in the network [13], [22]. Flow control in TCP is based on a window that limits the maximum number of packets the source can send before the receipt of any acknowledgment from the receiver. For TCP the loss of a packet is an indication that the network is congested. The lost packet is retransmitted and the window is reduced in order to alleviate the congestion of the network. But, this strategy in the detection of congestion results in a poor performance of the protocol when packets are lost in the network for other reasons than congestion [1], [4], [5]. Transmission errors on a bad quality link (e.g., wireless link) form the main source for non-congestion losses. A TCP packet corrupted while crossing a noisy link is discarded before reaching the receiver which results in an unnecessary window reduction at the TCP source. In the following we will focus on transmission errors on wireless links and we will call the corrupted TCP packets non-congestion losses or link level losses since they appear at a level below IP.

Many solutions have been proposed to improve the performance of TCP when operating on paths with non-congestion losses [1], [4], [5]. Some of these solutions consist in enhancing TCP with additional mechanisms to help it to recover from non-congestion losses without reducing its window (Explicit Loss Notification [4], loss predictors [6], etc.). Other solutions (e.g., I-TCP [3]) propose to shield the sender from these undesirable losses by splitting the TCP connection at the entry of the lossy part of the network (e.g., at the base station in case of wireless networks). A special transport protocol well tuned to a lossy environment (e.g., STP [12]) is then used across the lossy part. Although they improve the overall performance, these solutions break the end-to-end semantics of TCP. A packet is acknowledged before arriving at its real destination. To solve this problem, some other solutions propose to retransmit lost packets on behalf the source within the network without splitting the TCP connection (e.g., Snoop protocol [4]). The retransmission is done either at the link level or at a level above IP (e.g., by a transport agent at the base station). The main problem with the latter solutions is the interference of the local retransmissions with the TCP retransmissions. TCP retransmission timer may expire while the lost packet is being retransmitted locally over the lossy part. The time taken by a local retransmission as well as the number of tries have be to small otherwise TCP timer would frequently expires. This interference makes the local retransmission of packets an inappropriate solution for long delay noisy links as satellite links. It is however proposed for terrestrial wireless networks where the propagation time over the wireless link is small compared to the end-to-end delay.

A simpler solution, that does not require any modification to existing TCP and that does not interfere with its error recovery mechanisms, consists in improving the quality of the lossy part of the network with Forward Error Correction (FEC) codes [4]. The idea behind FEC is to send, in addition to the original data, some redundant information so that a packet, corrupted while crossing a wireless link, can be reconstructed at its output without requiring any retransmission. One can see FEC as sending, together with original packets, copies of them so that the copy can

[†] A shorter version of this paper has appeared in proceedings of the IEEE International Conference on Networking, Colmar, France, July 2001.

be used when the original packet is lost, of course if the copy itself is not also lost. This improves the quality of the noisy link while consuming some extra bandwidth. The other drawback of FEC is that it requires some processing time for coding and decoding the redundant information. However, the advantages of FEC are numerous and make it interesting despite its cost. The corrupted packets are reconstructed on runtime which eliminates the fluctuations of round-trip time caused by local retransmissions. Also, packets are delivered in the same order with which they are transmitted at the input of the wireless link. This avoids the duplicate ACKs that would be generated if the lost packet is locally retransmitted and subsequent packets from the same connection are forwarded to the destination. In case of local retransmissions, it has been proposed [4], [7] to stop these duplicate ACKs at the entry of the lossy link in order to not trigger the retransmission of the packet at the TCP source. Thus, FEC eliminates any interference with TCP error recovery mechanisms. Given this transparency of FEC, it is recommended for the improvement of the quality of bad transmission media especially those of long propagation delay [1]. For example convolutional coding, Viterbi decoding, together with interleaving techniques and Reed-Solomon encoding, are widely used in our days to render satellite links and wireless links as clean as terrestrial ones.

In this work we are interested in the bandwidth tradeoff between link level FEC and TCP congestion control. While consuming some extra bandwidth, FEC improves the throughput of TCP by shielding it from non-congestion losses. However, much FEC may steal some of the bandwidth used by TCP. The question that we ask here is, given a certain link with certain characteristics (bandwidth, error rate, burstiness of errors), how to choose the amount of FEC so that to get the maximum gain in TCP performance. The aim of this work is to understand this relation between the bandwidth consumed by FEC and that gained by TCP. A mathematical analysis and a set of simulations are used to this end. The simulator used is `ns`, the Network Simulator developed at LBNL [17]. Note that even though our work focuses on wireless links, it is useful for any other transmission medium between two IP routers presenting some non-congestion losses. A typical example could be the losses on an ATM-UBR virtual circuit connecting two adjacent IP routers.

In the next section, we describe the model we used in our analysis. Two models are defined, one for non-congestion losses and another for FEC. In Section III we show how to calculate the throughput of TCP. In Section IV we analyze mathematically and with simulations the interaction between FEC and TCP for a memoryless wireless link. Section V studies the effect of correlation of errors on TCP performance and on the capacity of FEC. The work is concluded in Section VI.

II. The model

Consider a TCP connection that crosses a network including a noisy wireless link of rate μ packets/s. We suppose that the quality of the lossy link is improved by a certain amount of FEC. In the next two subsections, we define our models for the loss process over the wireless link as well as for the FEC added to improve its quality.

A. The model for non-congestion losses

Most of the works on TCP performance [15], [16], [19] make the simplistic assumption that the loss process of TCP packets is not correlated. Packets are assumed to be lost independently with the same probability P. We know that this does not work for wireless links where transmission errors tend to appear in bursts [7], [8], [9], [14]. The model often used in the literature to represent correlated losses on a wireless link is the one introduced by Gilbert [7], [8], [11], [14]. It is a simple ON/OFF model. The lossy link is supposed to be in one of two states: 0 for Good and 1 for Bad. A packet is lost if it leaves the link while it is in the Bad state, otherwise it is supposed to be correctly received. We use such model in our work. A discrete time Markov chain (Figure 1) with two states (Good and Bad) models the dynamics of the wireless link. We focus on the loss process of link level packets called also *transmission units*. We suppose that a TCP packet is transmitted over the wireless link over multiple small transmission units [7], [8]. A transmission unit can be a bit, a byte, an ATM cell, or any other kind of link level blocks used for the transmission of TCP/IP packets.

The state of the wireless link state is observed upon the arrivals of transmission units at its output. We suppose that units cross continuously the link. If no real units exist, *fictive units* are inserted. In other words, transitions of the Markov chain associated to the wireless link happen at deterministic moments. The time between two transitions is equal to the transmission time of a unit on the wireless link.

Let p denote the probability that the link passes from Good state to Bad state when a transmission unit arrives at its output. Let q denote the probability that it stays in the Bad state. According to what transmission units mean, p and q can be one of the quantities used to measure error rates in real networks (Bit Error Ratio, Cell Loss Ratio, etc.). q represents how much the loss process of transmission units is bursty. A q close to zero means that losses are isolated and a q close to 1 means that long bursts are

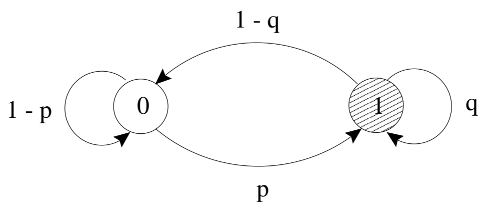

Fig. 1. The Gilbert loss model

dominant. Now, a q equal to p brings us to the case of an memoryless loss process (Bernoulli loss process).

It is useful here to calculate the stationary probabilities that the link is in the Bad and Good states. We suppose that $p, q \in (0, 1)$ so that the Markov chain associated to the lossy link is ergodic and that the stationary probabilities exist and are unique. These probabilities, denoted by π_B and π_G respectively, are used in the following to study what happens to a unit regardless of what has happened to the preceding one. We have,

$$\pi_B = \frac{p}{1-q+p}, \qquad \pi_G = \frac{1-q}{1-q+p}.$$

It is interesting also to calculate the average lengths of Good and Bad periods in terms of transmission units. We denote these lengths by L_B and L_G respectively. A simple calculation shows that,

$$L_B = \frac{1}{1-q}, \qquad L_G = \frac{1}{p}. \tag{1}$$

The expressions of L_B and L_G permit us to calculate the average loss rate as a function of p and q. Denote this rate by L. Intuitively, it is equal to the probability that the link is in the Bad state regardless of what has happened to the previous unit. We have,

$$L = \frac{L_B}{L_B + L_G} = \frac{p}{1-q+p} = \pi_B \tag{2}$$

We use the expression of L in the following to change the burstiness of losses while maintaining the same loss rate. It is clear that the increase in q stretches the duration of the Bad state which increases the burstiness of losses. For a certain loss rate L and in order to increase the burstiness, we vary q from 0 to 1. Then for each q, we use the expression of L to calculate the value of p that keeps the loss rate unchanged.

B. The FEC model

The most common code used for error correction is the *block* code [20], [21]. Suppose that data is transmitted in units as in our model for the lossy link. Block code FEC consists in grouping the units in blocks of K units. Then, a codec adds to every block a group of R redundant units calculated from the K original units. The result is the transmission of blocks of total size $N = K + R$ units. At the receiver, the original K units of a block are reconstructed if at least K of the total N units it carries are correctly received. This improves the link quality since a block can now resist to R losses without being discarded.

Block FEC can be implemented in the physical layer to recover from the loss of bits (e.g., Reed-Solomon codes). This technique, together with interleaving, is used in satellite links to correct the bursts of bit errors that result from the utilization of Convolutional coding/Viterbi decoding. Block FEC can be also implemented at a higher level to recover from the loss of frames or packets. This latter utilization of FEC is known as the recovery from packet errors or simply from erasures [20], [21]. An example of erasure block FEC is the one proposed in [18] for the recovery from lost ATM cells. The difference between the two implementations of FEC is that in the first case a frame contains the redundancy and the original data. However in the second case, the redundancy is included in other frames. Adding redundancy to other frames helps the higher layers to recover from losses not only caused by corruption but also by other phenomena such as congestion of switches or of the MAC layer.

In our work we consider a block FEC code implemented in the layer of transmission units. We ignore any FEC code that may exist below this layer. The input to our study is the loss process seen by transmission units which is assumed to follow the Gilbert model. The FEC layer at the entry of the lossy link (codec) groups the transmission units in blocks of size K, then it adds R redundant units to every block. The total number of units in a block becomes equal to $N = K + R$. At the output of the lossy link, another FEC layer (decodec) takes the N units in every block, eliminates the redundancy, and hands the original block to the upper layer. A block is delivered if at least K of its N units are correctly received, otherwise it is supposed to be discarded. In what follows, we will show how much the parameters of the FEC scheme impacts the performance of TCP.

III. The approximation of TCP throughput

Consider the throughput of the connection as the performance measure that indicates how well TCP behaves over the wireless link. Different models exist in the literature for the calculation of TCP throughput [2], [15], [16], [19]. These models assume long life connections and find the expression of the throughput as a function of the probability that a TCP packet is lost in the network. Denote this probability by P. In case of bursty losses, P represents the inverse of the average number of packets correctly received between two bursts of losses [19]. In other words,

P represents the probability that a packet is the first loss in a burst of losses. This is because the new versions of TCP (e.g., SACK [10]) are designed in a way to divide the window one time by two for a burst of packet losses. The throughput of TCP has been shown to be inversely proportional to the square root of P as well as to the round-trip time. The difference between the different expressions of TCP throughput is in the number of factors they consider. The simplest expression [16] considers only the linear increase multiplicative decrease part of TCP congestion control and makes the assumption that the moments at which the window of TCP is reduced are equally separated. This simple expression of the throughput is called the *square root formula* [16]. It has been shown that this simple expression gives good performance when P is small [19] and when the times between the moments at which the window of TCP is reduced do not vary too much [2]. The other expressions of the throughput consider more factors as the timeout phenomenon, the receiver window, the possibility of other distributions of losses, etc.

Without loss of generality, we consider in our work the simple square root formula for TCP throughput. One can simply use other more sophisticated expressions of the throughput. Suppose that the receiver acknowledges every data packet. Let T denote the average number of packets between losses ($T = 1/P$). Recall that in case of bursty losses, T denotes the average number of packets between bursts of losses. Let RTT denote the average round-trip time seen by the connection. Thus, we can write the throughput of TCP in terms of packets/s as [16]

$$Thrp == \frac{1}{RTT}\sqrt{\frac{3}{2}T} = \frac{1}{RTT}\sqrt{\frac{3}{2P}}.$$

Suppose that the wireless link is the bottleneck on the path of the connection. We make this assumption because we want to optimize the amount of FEC so that a TCP connection becomes able to fully utilize the available bandwidth on the wireless interface. Thus, in the absence of FEC, the throughput of TCP is upper bounded by μ and we write it as follows

$$Thrp = \min\left(\frac{1}{RTT}\sqrt{\frac{3}{2}T}, \mu\right)$$

Our objective is to express the throughput of TCP as a function of the parameters of the loss process of transmission units (p, q) and the parameters of the FEC scheme (N, K). We already have the expression of the throughput as a function of what happens at the packet level (P). What we still need to do is to relate the loss process of transmission units to the loss process of TCP packets. But, TCP packets can be lost in other parts of the network not only on the wireless interface. The loss process of TCP packets is then the sum of multiple processes. To simplify the analysis we consider the best case when packets are only lost on the wireless interface if the wireless bandwidth μ is not fully utilize. When the wireless interface becomes fully utilized, losses may appear in other parts of the network but this is not important since the throughput of TCP does not change and remains equal to the available bandwidth on the wireless interface. Note here that adding a certain amount of FEC when TCP packets can be lost in other parts of the network gives less gain in TCP performance as when packets can be only lost on the wireless link. This is simply because the increase in the total packet loss probability P is less important in the first case.

Using our above assumptions, we calculate T as a function of the different parameters of the Gilbert and FEC models. This gives us the expression of the throughput of TCP. In the case transmission units are lost independently from each other ($p = q$), the calculation of T is straightforward. $P = 1/T$ is no other than the probability that a TCP packet is lost while crossing the wireless link. This case is studied in the next section. The difficulty appears when transmission units are lost in bursts. The correlation of losses at the unit level causes the correlation of losses at the TCP packet level. T must be then calculated as the average number of TCP packets transmitted between bursts of packet losses. But, because of redundancy and because of the notion of blocks and units, the calculation of T in the bursty case is quite difficult. Some assumptions must be made at the unit level and at the packet level to ease the analysis. This is done in Section V. Note here that our aim from the analysis of the correlation case is to study the effect of burstiness in transmission errors on the efficiency of a given FEC scheme, and hence on the throughput of TCP. We are not interested in the study of the effect of correlation between packet losses at the TCP level on end-to-end performance.

Now, even though it increases T, the addition of FEC consumes some bandwidth and decreases the maximum throughput TCP can achieve. Instead of μ, we get $K\mu/N$ as a maximum TCP throughput. If we denote by S the size of a TCP packet in terms of transmission units, the throughput of TCP in presence of FEC and in terms of units/s can be written as,

$$Thrp(N, K) = \min\left(\frac{S}{RTT}\sqrt{\frac{3}{2}T(N, K)}, \frac{K}{N}\mu\right) \quad (3)$$

In the sequel all rates and throughputs will be expressed in terms of transmission units per second. For a given

Gilbert model parameters we will focus on the calculation of $T(N,K)$, and hence the throughput $Thrp(N,K)$, as a function of the amount of FEC.

IV. The case of non-correlated losses

In this section we suppose that transmission units are lost independently of each other with probability p ($p = q$). Thus, TCP packets are also lost independently of each other but with probability $P(N,K)$ which is a function of the FEC scheme (N,K) we are using on the wireless link. The throughput of the TCP connection can be approximated by substituting T in formula (3) by $1/P(N,K)$. First, we state our analysis of TCP performance, the analytical results as well as their interpretations. Next, simulation results are presented.

A. The analysis

Suppose that TCP packets are of the size of one link level block ($S = K$ units). Given a certain block size (K) and a certain FEC rate (K/N), the choice of the size of the TCP packet in terms of blocks is another problem that we will not address in this paper. It is a problem of TCP algorithms rather than of the amount of FEC we are using on the link. However, we noticed that with the values of K we are using in this paper, larger packets give approximately the same performance as single block packets.

A packet is lost when more than R of its units are lost due to transmission errors. This happens with a probability,

$$P(N,K) = \sum_{i=0}^{K-1} \binom{N}{i} (1-p)^i p^{N-i}.$$

The throughput is obtained by plugging this value of $P(N,K)$ into equation (3).

It is clear that the addition of FEC (i.e., N¿K) at the link level reduces the loss probability of TCP packets and thus increases the throughput. This happens whenever the first term of the minimum function in equation (3) is smaller than the second term. This improvement of the throughput continues until the two terms of the minimum function become equal. At this point, the quantity of FEC added to the wireless link is sufficient to eliminate the negative effect of non-congestion losses on TCP. We say here that the FEC has *cleaned* the link from TCP point of view. Any increase in N beyond this point results in a throughput deterioration. There will be more FEC than what is needed to clean the link. Thus, the appropriate quantity of FEC is the one given by the equality of the two terms of the minimum function. Given a link of bandwidth μ, transmission blocks of size K, and a unit loss probability p, the optimum quantity of FEC from TCP point of view is the

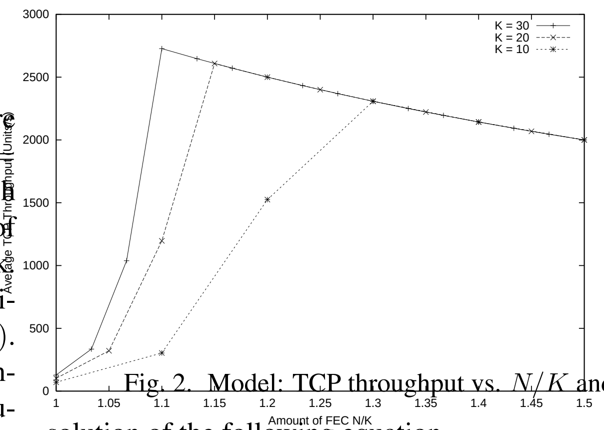

Fig. 2. Model: TCP throughput vs. N/K and K

solution of the following equation,

$$\frac{N}{RTT}\sqrt{\frac{3}{2P(N,K)}} = \mu. \tag{4}$$

Note that such quantity of FEC is appropriate only for TCP connections. UDP transfers as real time audio and video flows may require another quantity of FEC function of the loss rate they tolerate.

B. Analytical results

We show in Figure 2 how the throughput of TCP varies as a function of the ratio N/K (the FEC code rate) for different values of K (10, 20, and 30 units). RTT is taken equal to 560 ms and the wireless link bandwidth to 3000 units/s. We can consider this scenario as the case of a mobile user downloading information from the Internet through a satellite link. This value of μ is approximately equal to the maximum ATM cell rate on a T1 link (1.5 Mbps). p is set to 0.01.

It is clear that the performance improves considerably when FEC is added and this improvement continues until the optimum point given by equation (4) is reached. Beyond this point, any increase in FEC deteriorates the throughput as we explained. Also, we notice that, for a certain quantity of FEC, an increase in K improves the performance. An increase in the block size results in a larger R thus in a better capacity to correct multiple errors per block. At large blocks, FEC can correct the same errors corrected when blocks are divided into small ones but also it has the capacity to correct these errors when they are grouped together. Another reason for performance improvement is that an increase in K results in larger TCP packets, then in a faster growth of the congestion window. TCP window is increased in terms of packets rather than bytes. Thus, the source returns faster to its rate prior to the detection of a packet loss. However, increasing the

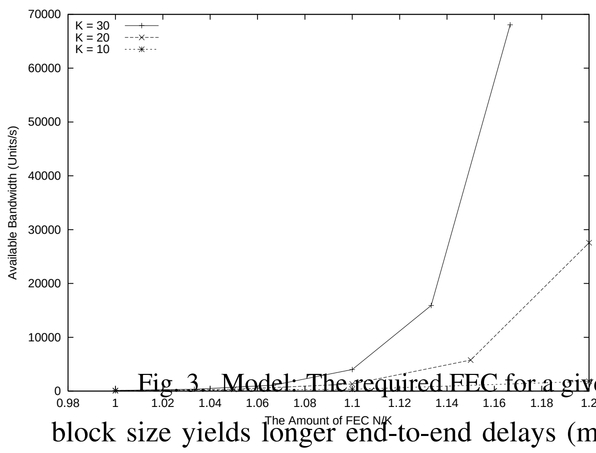

Fig. 3. Model: The required FEC for a given μ

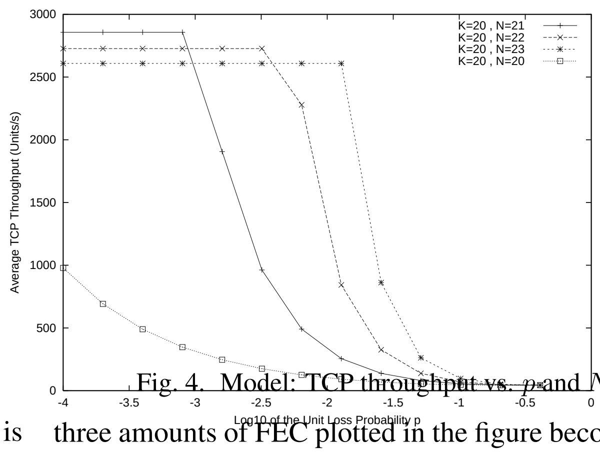

Fig. 4. Model: TCP throughput vs. p and N/K

block size yields longer end-to-end delays (more time is needed to fill a block), more processing time (redundant units are computed as a function of all the original units in the block), and larger memory requirements (the units of a block must be stored at the output before being decoded and handed to upper layers).

In Figure 3, we plot the left hand side of equation (4) as a function of N/K for the same three values of K. These curves provide us with the optimum amount of FEC for a given μ, p, and K. The optimum amount of FEC is obtained by the intersection of the line corresponding to K and the horizontal line corresponding to μ. We see well that any increase in K reduces considerably the amount of FEC needed to clean the wireless link from TCP point of view. Again, this is because the increase in K results in a faster growth of the window and in a better resilience against grouped errors. Given a certain μ, a compromise between K and FEC rate must be done. First, we have to choose the largest possible K then we choose the appropriate amount of FEC.

For $\mu = 3000$ units/s and $K = 20$, we show in Figure 4 how the throughput of TCP varies as a function of the transmission unit loss probability p and this is for different values of N. It is clear that adding just one redundant unit to every FEC block results in a considerable gain in performance especially at small p. Adding more redundancy at small p deteriorates slightly the performance since the link is already clean and the additional redundancy steals some of the bandwidth used by TCP. This is not the case at high p where much redundancy needs to be used in order to get good performance. We see well how the situation changes in the middle of the p axis and how a large N starts to give better performance. Note that even though an excess of FEC reduces the performance of TCP when losses are rare, the reduction is negligible in front of the gain in performance we obtain when losses become frequent. When the link is heavily lossy ($\log(p) > -1.7$), the three amounts of FEC plotted in the figure become insufficient to clean the wireless link and all the curves converge to the same point. This asymptotic behavior is due to the fact that the packet loss probability $P(N, K)$ tends to 1 when p tends to one whatever are the values of N and K.

C. Simulation results

Using `ns` [17] we simulate a simple scenario where a TCP source is connected to a router via a high speed terrestrial link and where the router is connected to the TCP receiver via a lossy wireless link. The Reno version of TCP [10] is used. This version tries to avoid slow start upon recovering from losses and thus it must give close results to the expression of the throughput we used which assumes that TCP stays in the congestion avoidance phase [16]. The TCP source is fed by an FTP application with an infinite amount of data to send. We set the TCP receiver so that it acknowledges all data packets and advertises an infinite window. We added then our FEC model to the simulator. A TCP packet is fragmented at the entry of the wireless link into K units. The FEC layer adds then R redundant units to every packet (block). At the output of the wireless link, the FEC layer reconstructs the original packet if at least K of its N units are correctly received otherwise it rejects it. The transmission units on the lossy link are supposed to be ATM cells of size 53 bytes. We chose the bandwidth of the lossy link in a way to get a service rate μ of 3000 cells/s. RTT is taken equal to 560 ms and the buffer size in the middle router is set to 100 packets. This guarantees that no losses occur in the middle router before the full utilization of the available bandwidth on the wireless interface.

Figures 5 and 6 show the variation of the simulated throughput as a function of the amount of FEC (N/K) and the unit loss probability p. In the first figure p is set to 0.01. We notice clearly the good match between these results and the analytical ones. The small difference is due to the

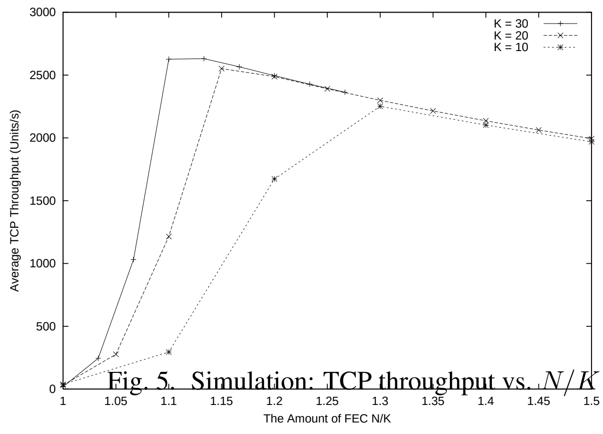

Fig. 5. Simulation: TCP throughput vs. N/K and K

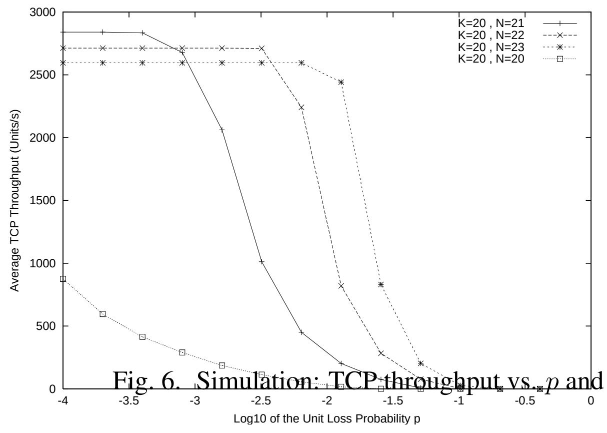

Fig. 6. Simulation: TCP throughput vs. p and N/K

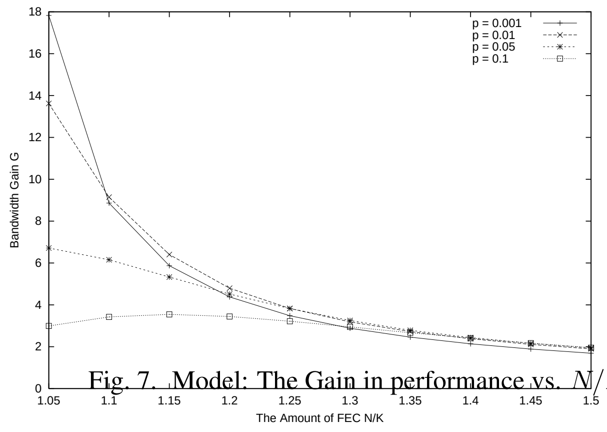

Fig. 7. Model: The Gain in performance vs. N/K and p

fact that the expression of the throughput we used does not consider the possibility of a timeout when multiple packet losses appear in the same TCP window. Also, in our analysis we considered that RTT is always constant which does not hold when the throughput of TCP approaches the available bandwidth.

D. The tradeoff between TCP throughput and FEC cost

We compare in this section the bandwidth gained by TCP to that consumed by FEC. Let G be the ratio of these two bandwidths,

$$\begin{aligned} G &= \frac{Thrp(N,K) - Thrp(K,K)}{Thrp(N,K) \times \frac{N-K}{K}} \\ &= \left(1 - \frac{Thrp(K,K)}{Thrp(N,K)}\right) \times \left(\frac{K}{N-K}\right). \end{aligned} \quad (5)$$

This quantity indicates how much beneficial is the addition of FEC. It can be considered as a measure of the overall performance of the system TCP-FEC. We want to improve TCP performance without paying for FEC more than we gain in TCP performance. A value close to one of this gain means that we pay for FEC as much as we gain in TCP throughput. A negative value means that the link was clean for TCP so that the addition of FEC has reduced the performance instead of improving it.

In Figure 7 we plot G as a function of the amount of FEC for different unit loss probabilities. Again, we take $\mu = 3000$ units/s and $K = 20$. This figure shows that the gain in overall performance is important when the loss probability and the amount of FEC are small. Moreover, with small amounts of FEC, the gain decreases considerably when the loss rate (p) increases. Now, when the FEC rate increases, the curves converge approximately to the same point with a slightly better gain this time for high loss probabilities.

For small p, little FEC is sufficient to clean a link which improves considerably the performance of TCP. On the other hand, this little FEC is not able to clean a link with a high p. The result is a small gain in TCP performance, hence a small G. This explains what we see in the left hand part of Figure 7.

Consider now the right hand part of the figure. A large amount of FEC is able to clean links with a wide range of p. TCP obtains then the same throughput $Thrp(N,K)$ (equal to $K\mu/N$ units/s) for all the values of p considered in the figure. But, the gain we defined in equation (5) is not only a function of the throughput after the addition of FEC. It is also a function of the throughput before this addition. Given that the initial throughput (denoted by $Thrp(K,K)$ in the equation defining of G) decreases when p increases, the gain in overall performance is more important at high p than at small p, and this is when FEC is added in large amounts. The point at which two curves meet gives the required amount of FEC to clean their corresponding links.

E. Number of connections and the gain in performance

We notice in Figure 7 that using a small amount of FEC gives the best gain in overall performance. A small amount of FEC helps TCP to improve considerably its throughput but it does not help it to use all the available bandwidth. Any additional FEC improves further the quality of the link which improves further the throughput of TCP but the revenues are not as important as for the first units of redundancy. Thus, in order to maintain a high gain, one can use a small amount of FEC and share the available bandwidth between multiple TCP connections. The result will be a better utilization of the link bandwidth while using a small amount of FEC. But, in practice one cannot guarantee that there are always enough connections to use the available bandwidth. A TCP connection must be able to use all this bandwidth when it operates alone in the network. For this reason FEC has to be added in large amounts so that to make the lossy link clean from the point of view of a single TCP connection even if the achieved gain is not so important.

To clarify this point, we study the gain in the overall performance when many TCP connections share the lossy link. Suppose that C connections run simultaneously between the source and the destination. Because they see the same RTT and they are subject to the same loss process, we can suppose that they achieve the same throughput [15]. Let $P_I(N,K)$ be the probability that a packet of an individual connection is lost. Using (3), the total TCP throughput can be written as

$$Thrp(N,K) = \min\left(\frac{CS}{RTT}\sqrt{\frac{3}{2P_I(N,K)}}, \frac{K}{N}\mu\right) \quad (6)$$

Because the loss process is not correlated, $P_I(N,K)$ is equal to $P(N,K)$. Thus, the difference in the case of many connections than in the case of a single one is a multiplicative factor C in the first term of equation (3). The second term of this equation remains unchanged. A factor C means a faster improvement of the throughput when the amount of FEC increases as illustrated in Figure 8. This figure corresponds to $p = 0.01$ and $K = 10$. At large C, the link can be cleaned by a smaller amount of FEC than at small C. When the amount of FEC added to a link is not enough to clean it, one can open many TCP connections to use more bandwidth and get more gain in overall performance instead of adding more FEC to im-

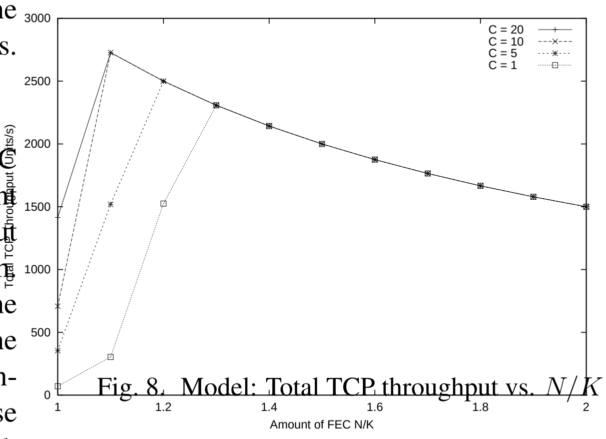

Fig. 8. Model: Total TCP throughput vs. N/K and C

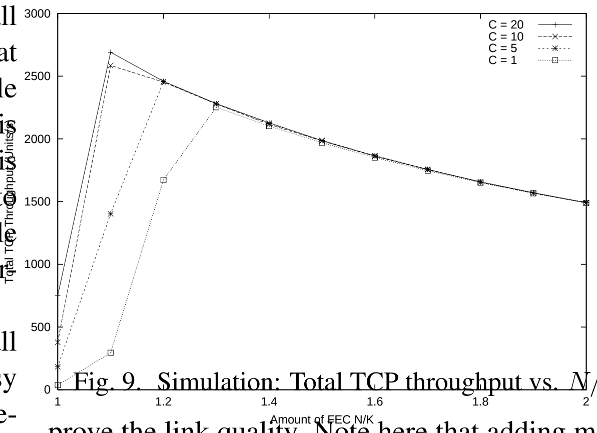

Fig. 9. Simulation: Total TCP throughput vs. N/K and C

prove the link quality. Note here that adding more FEC to clean a link reduces the maximum limit of TCP throughput ($\mu K/N$). As we see in Figure 8, a total throughput of 2700 units/s cannot be obtained when few connections are sharing the wireless link even if enough FEC is added to clean it. However, this total throughput can be obtained when increasing the number of connections and reducing the amount of FEC.

In Figure 9 we show the simulation results that correspond to Figure 8. The match is clear for large amounts of FEC. However for small amounts, the analysis gives larger values. This discrepancy is due to the fact that the expression of the throughput we used does not consider the timeout phenomenon. At small amount of FEC, losses are frequent which results in small windows, multiple losses per window, and an important probability of timeout [19]. This gives poorer throughput than the one given by the analysis. The difference in the results is exacerbated by the factor C which explains the large mismatch in the case of 20 connections.

V. The case of correlated losses

In this section we study the effect of burstiness of transmission unit losses on the efficiency of a FEC scheme. It is clear that when unit losses tend to appear in bursts, more redundant information is needed to clean the link. Packets are hurt by burst of losses and they require a large number of redundant units per packet (R) to be corrected. But, for the same average loss rate (L), the correlation of losses reduces the probability that the link passes to the Bad state (p decreases when q increases). This reduces the probability that a packet is hurt by a burst of losses. TCP throughput may then improve and the amount of FEC may be reduced. An analysis is needed to understand these opposite effects of burstiness.

A. *Performance analysis*

Burstiness at the unit level results in burstiness at the TCP level. In the presence of TCP versions that reduce their windows once in response to a burst of packet losses (e.g., SACK [10]), T in equation (3) represents the average number of TCP packets correctly received between two bursts of packet losses. Let us calculate T, and hence the throughput, as a function of the amount of FEC, K, the average loss rate, the burstiness of losses, and μ.

Calculation of the average number of packets between bursts

Let t be the number of TCP packets correctly received between two separate bursts of losses at the TCP level. The minimum value of t is therefore one packet and its expectation is equal to T. Let Y_n be the state of packet n. 0 is the number of the first good TCP packet between the two bursts. Y_n takes two values B (Bad) and G (Good). We have $Y_0 = G$. The expectation T can be written as,

$$\sum_{n=0}^{\infty} P(t > n | Y_0 = G) = 1 + \sum_{n=1}^{\infty} P(t > n | Y_0 = G).$$

The computation of T is quite complicated since the spacing between the TCP packets varies with the window size. Another complication is that $\{Y_n\}$ does not form a Markov chain. Indeed, if we know for example that a packet, say n, is of type B then the probability that packet $n+1$ is of type G depends also on the type of packet $n-1$. If packet $n-1$ were G rather than B, then the *last units of packet* n are more likely to be those that caused its loss. Hence, the probability that packet $n+1$ is B is larger in this case.

This motivates us to introduce another random variable which will make the system more "Markovian" and will permit us to write recurrent equations in order to solve for T. We use for this purpose the state of the last *transmission unit* received, or fictively received before the nth TCP packet. The knowledge of the state of this unit, denoted by Y_n^{-1} (which may take again the values B or G), fully determines the distribution of the state Y_n of the following TCP packet. We write T as,

$$1 + \alpha P(Y_1^{-1} = G | Y_0 = G) + \beta P(Y_1^{-1} = B | Y_0 = G),$$

where

$$\alpha = \sum_{n=1}^{\infty} P(t > n | Y_0 = G, Y_1^{-1} = G)$$

$$\beta = \sum_{n=1}^{\infty} P(t > n | Y_0 = G, Y_1^{-1} = B)$$

We shall make throughout the following assumption,

Assumption 1:

$$P(Y_1^{-1} = G | Y_0 = G) \approx \pi_G$$

$$P(Y_1^{-1} = B | Y_0 = G) \approx \pi_B$$

Assumption 1 holds when the time to reach steady state for the Markov chain in the Gilbert model is shorter than the time between the beginning of two consecutive TCP packets (either because the TCP packets are sufficiently long, or because the TCP packets are sufficiently spaced). Assumption 1 also holds when π_B and the loss probability of a whole TCP packet are small. Indeed we can write,

$$\begin{aligned} \pi_G &= P(Y_1^{-1} = G | Y_0 = G) P(Y_0 = G) \\ &\quad + P(Y_1^{-1} = G | Y_0 = B) P(Y_0 = B) \\ &\approx P(Y_1^{-1} = G | Y_0 = G) \cdot 1 \\ &\quad + P(Y_1^{-1} = G | Y_0 = B) \cdot 0 \end{aligned}$$

In view of Assumption 1, the probability that the unit preceding a packet is lost can be considered as independent of the state of the previous packet. It follows that,

$$T = 1 + \alpha \pi_G + \beta \pi_B,$$

$$\begin{aligned} \alpha &= (1 - P(Y_1 = B | Y_1^{-1} = G))(1 + \alpha \pi_G + \beta \pi_B) \\ \beta &= (1 - P(Y_1 = B | Y_1^{-1} = B))(1 + \alpha \pi_G + \beta \pi_B) \end{aligned}$$

which gives us,

$$\frac{1}{T} = \pi_G P(Y_1 = B | Y_1^{-1} = G) + \pi_B P(Y_1 = B | Y_1^{-1} = B).$$

The calculation of T, and therefore of the throughput, is simplified to the calculation of the probability that a packet is lost given the state of the unit just preceding it. These are the two probabilities $P(Y_1 = B | Y_1^{-1} = G)$ and $P(Y_1 = B | Y_1^{-1} = B)$ that figure in the above expression of $1/T$. But again it is difficult to find explicit expressions for these probabilities. A TCP packet can be lost by a single long burst as well as by multiple separate small bursts.

To further facilitate the analysis, we assume that bursts of losses at the unit level are separated so that two bursts rarely appear within the same packet. This is formalized in the following assumption,

Assumption 2: The following holds,

$$(1-q)\cdot L\cdot N << 1$$

A TCP packet is supposed to be lost only if it is hurt by a burst larger than R. We don't consider therefore the probability that multiple small and separate bursts at the unit level contribute to the loss of a TCP packet. This is possible when the sum of the average lengths of the Good state (L_G) and the Bad state (L_B) is much larger than the packet length N. Using (1) and (2) yields Assumption 2. If Assumption 2 is not satisfied, many bursts may appear within the same packet leading to a higher loss probability than the one given by our analysis, therefore to a lower throughput. In this case, we expect that our analysis results in an overestimation of the real throughput.

Consider first the case $Y_1^{-1} = B$. In view of Assumption 2, packet 1 is lost if its first $R+1$ units are also lost. Thus,

$$P(Y_1 = B|Y_1^{-1} = B) = q^{R+1}.$$

For the case $Y_1^{-1} = G$, packet 1 is lost if a burst of losses of length at least $R+1$ units appears in its middle. We get,

$$\begin{aligned} &P(Y_1 = B|Y_1^{-1} = G) \\ &= \quad q^R p\left(1 + (1-p) + \cdots + (1-p)^{N-R-1}\right) \simeq Kq^R p \end{aligned}$$

We used here the approximation $(1-(1-p)^{N-R}) \simeq Kp$.

Given a certain average loss rate L and a certain correlation of losses expressed by q, we can find p using equation (2). T can be then calculated for any FEC scheme (N,K) as,

$$\frac{1}{T} = q^{N-K} L\left((1-q)K + q\right). \tag{7}$$

The throughput can be approximated using equation (3).

B. Analytical results

Using (3) and (7), we plot in Figure 10 the throughput of TCP as a function of burstiness and this is for different amounts of FEC. The burstiness is varied by varying q which is called the Conditional Loss Probability in the figure. K is set to 20 and the loss rate L to 0.01. The other parameters of the model are taken as in the previous section. We see well that in case of burstiness (large q), a large

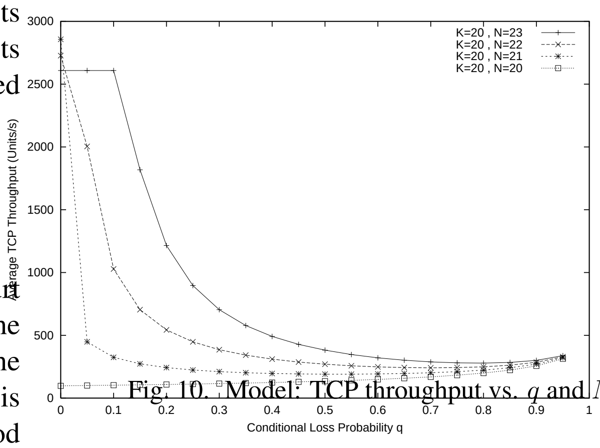

Fig. 10. Model: TCP throughput vs. q and N/K

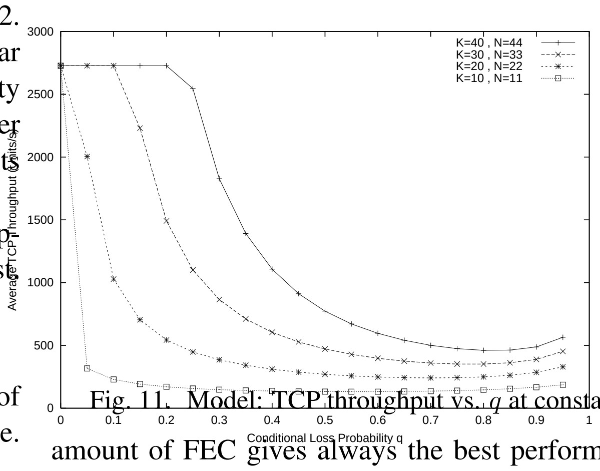

Fig. 11. Model: TCP throughput vs. q at constant N/K

amount of FEC gives always the best performance. The difference in the performance is important for small bursts (small q). When burstiness increases, the throughput decreases drastically for the three FEC schemes we consider in the figure. A large amount of FEC helps the throughput to resist to small bursts but once these bursts become larger than the FEC capacity, the throughput deteriorates quickly. We see also that in the absence of FEC, the throughput improves a little when burstiness increases. We notice this improvement in the throughput for the three other FEC schemes at large q. As expected, all the curves converge to the same point when bursts become very large. Here, the addition of FEC in small quantity has no meaning. Much FEC must be added to clean the link. But, much FEC reduces the throughput when burstiness decreases given the bandwidth it consumes. A compromise must be done between much FEC to resist to bursts and a small amount of FEC to give better performance when correlation decreases.

Now, we show in Figure 11 how the block size K can help TCP to resist to bursts of losses. We take the same amount of FEC ($N/K = 11/10$) and we vary K. Increasing K increases the number of redundant units in a TCP

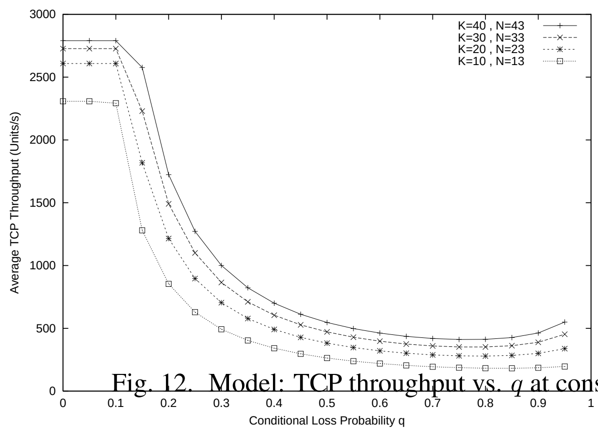

Fig. 12. Model: TCP throughput vs. q at constant R

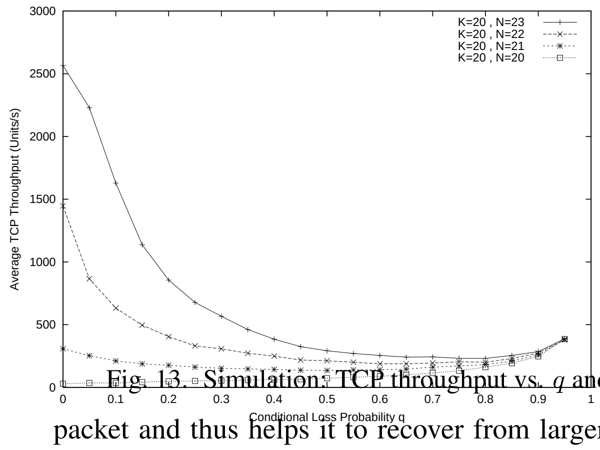

Fig. 13. Simulation: TCP throughput vs. q and N/K

packet and thus helps it to recover from larger bursts. A large block size still gives better performance even at high correlation. As we said before, this is due to the faster growth of the window in terms of units when large packets are used. The problem with large blocks is that they require a long time to code/decode the redundant information.

The benefit of large packets is also illustrated in Figure 12. In this figure we plot for the same number of redundant units per packet (R), the variation of the throughput for different packet sizes. It is clear how a large packet size gives better performance than a small one even though the amount of FEC is smaller. From equation (7), increasing K for the same R decreases T, but this decrease is small compared to the gain we get from the increase in the packet size. In other words, the throughput in terms of packets/s deteriorates when we increase K at a constant R, but in terms of units/s it improves. Which counts is the number of redundant units per packet rather than the total amount of redundancy.

C. Simulation results

We consider the SACK version of TCP [10] which is able to recover from a burst of losses without reducing its window multiple times and without resorting to timeout and slow start. The parameters of the network are not changed. L is set to 0.01 and K to 20. Our intention is to validate by simulation the analytical results we plotted in Figure 10. With these settings, Assumption 2 is satisfied for all the values of q we consider in the figure.

The results are plotted in Figure 13. The curves in this figure show the same behavior as those in Figure 10. But we see some mismatch at low burstiness. This is due to our assumption that a packet can only be lost by a single burst not by multiple small and separate bursts of losses at the unit level. As one must expect, the simulation gives a lower throughput in this region given that we are overestimating T. This mismatch disappears at high correlation. For large q, p is small and then the probability to lose a packet due to separate bursts becomes negligible.

VI. Conclusions

In this paper we studied via mathematical analysis and simulations the interaction between the parameters of a FEC scheme and the performance of TCP congestion control. FEC is a promising approach for the improvement of the quality of noisy transmission media as wireless and satellite links. We showed that TCP throughput improves with the addition of FEC until a certain point where the noisy link becomes clean from TCP point of view. Any increase in FEC beyond this point is not beneficial. It reduces the available bandwidth and deteriorates TCP performance.

At a constant amount of FEC, we showed that the increase in the block size results in an improvement of TCP performance. Large blocks contain large amount of redundancy and are able to better resist to grouped errors. Moreover, large blocks permit to TCP to increase fast its window after the occurrence of losses. Another result of our study is that it is possible to improve further the performance by opening multiple TCP connections to the same destination. With multiple connections, less FEC is required to clean a noisy link and the throughput of TCP can reach higher values.

Concerning burstiness we showed that, even if the average error rate remains unchanged, the increase in burstiness reduces the capacity of FEC and deteriorates the throughput of TCP. An addition of FEC can solve the problem but this FEC becomes unnecessary when burstiness disappears. We found also that an increase in block size improves FEC resilience to bursts of errors without any further addition of FEC. Thus, it is better to use always the largest possible block size. If a large block size is not possible, an alternative could be to implement some kind of adaptive FEC that adjusts the amount of redundancy as a function of the degree of burstiness.

REFERENCES

[1] M. Allman, D. Glover, and L. Sanchez, “Enhancing TCP Over Satellite Channels using Standard Mechanisms”, *RFC 2488*, Jan 1999.

[2] E. Altman, K. Avratchenkov, and C. Barakat, ”A stochastic model for TCP/IP with stationary random losses”, *to appear at ACM SIGCOMM*, Sep 2000.

[3] A. Bakre and B. R. Badrinath, “I-TCP: Indirect TCP for Mobile Hosts”, *International Conference on Distributed Computing Systems*, May 1995.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. Katz, “A comparison of Mechanisms for Improving TCP Performance over Wireless Links”, *ACM SIGCOMM*, Aug 1996.

[5] C. Barakat, E. Altman, and W. Dabbous, “On TCP Performance in a Heterogenous Network : A Survey”, *IEEE Communications Magazine*, Jan 2000.

[6] S. Biaz and N. H. Vaidya, “Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result”, *Seventh International Conference on Computer Communications and Networks (IC3N)*, Oct 1998.

[7] H. Chaskar, T. V. Lakshman, and U. Madhow, “On the design of interfaces for TCP/IP over wireless”, *IEEE MILCOM*, 1996.

[8] A. Chockalingam, M. Zorzi, and R.R. Rao, “Performance of TCP on Wireless Fading Links with Memory”, *IEEE ICC*, Jun 1998.

[9] B. R. Elbert, “The Satellite Communication Applications Handbook”, *Artech House*, Boston, London, 1997.

[10] K. Fall and S. Floyd, “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP”, *Computer Communication Review*, Jul 1996.

[11] E.N. Gilbert, “Capacity of a burst-noise channel”, *Bell Systems Technical Journal*, Sep. 1960.

[12] T. Henderson and R.H. Katz, “Transport Protocols for Internet-Compatible Satellite Networks”, *IEEE Journal on Selected Areas in Communications*, Feb 1999.

[13] V. Jacobson, “Congestion avoidance and control”, *ACM SIGCOMM*, Aug 1988.

[14] A. Kumar and J. Holtzman, “Performance Analysis of Versions of TCP in a Local Network with a Mobile Radio Link”, *Sadhana: Indian Academy of Sciences Proceedings in Engg. Sciences*, Feb 1998.

[15] T.V. Lakshman and U. Madhow, “The performance of TCP/IP for networks with high bandwidth-delay products and random loss”, *IEEE/ACM Transactions on Networking*, Jun 1997.

[16] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, “The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm”, *Computer Communication Review*, Jul 1997.

[17] The LBNL Network Simulator, `ns`, available at http://www-nrg.ee.lbl.gov/ns.

[18] N.C. Oguz and E. Ayanoglu, “Performance Analysis of Two-Level Forward Error Correction for Lost Cell Recovery in ATM Networks”, *IEEE INFOCOM*, Apr 1995.

[19] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP Throughput: a Simple Model and its Empirical Validation”, *ACM SIGCOMM*, Sep 1998.

[20] L. Rizzo, “Effective erasure codes for reliable computer communication protocols”, *Computer Communication Review*, Apr 1997.

[21] N. Shacham and P. McKenney, “Packet Recovery in High-Speed Networks Using Coding and Buffer Management”, *IEEE INFOCOM*, 1990.

[22] W. Stevens, “TCP Slow-Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”, *RFC 2001*, Jan 1997.