Maximising single connection TCP goodput by trading bandwidth for BER[1]

Nedo Celandroni, Francesco Potortì

ISTI, Institute of National Research Council (C.N.R.)
Area della ricerca, via Moruzzi 1, I 56124 Pisa
Phone: +39 050 315 2988/ 3058 - Fax: +39 050 313 8091
E-mail: {Nedo.Celandroni, F.Potorti}@cnuce.cnr.it

SUMMARY

All other conditions being equal, the end-to-end throughput of a TCP connection depends on the packet loss rate at the IP level. This is an issue when IP runs on a wireless link, where the bit error rate is variable and typically much higher than it is on fixed links. Especially on physical links where the bandwidth delay product is high, TCP performance is significantly impaired by apparently low values of the bit error rate. Generally speaking, on a wireless link bandwidth can be traded for information quality (error rate), the simplest method being to change the type or parameters of forward error correction. On this basis, we show a general method of taking advantage of this trade-off in order to maximize the throughput of a TCP connection.

Keywords: wireless, TCP, FEC, BER, packet loss, throughput, goodput

1. INTRODUCTION

The transmission of TCP traffic over wireless data links poses a problem that is not usually apparent on terrestrial links. TCP behaviour is very sensitive to packet loss, which is interpreted as a congestion signal, and consequently as a reason to throttle the data rate. The common behaviour of the wireless data link is to discard errored data packets, which are not even made available to the superjacent TCP/IP stack. This means that the TCP/IP stack cannot distinguish packet loss due to data corruption from packet loss due to other reasons that should be interpreted as congestion signals.

In order to improve the efficiency of TCP over wireless links, where improving the error rate is generally expensive (if indeed possible at all), a number of techniques are in common use. These include various types of spoofers, which may or may not preserve the TCP end-to-end semantics [1], and automatic repeat request (ARQ) [2],[3]. These techniques, each operating at different levels of the protocol stack, exploit local knowledge of the wireless hop characteristics in order to add a shorter delay control loop underneath the end-to-end control loop in the TCP connection. This gives a prompter reaction to packet loss and consequently improves the end-to-end performance at the expense of local buffering of packets to retransmit in case of loss and increased complexity.

Methods have been postulated for a number of different techniques operating at different levels. For example, link-level forward error correction (FEC) [4] operates below TCP. Some methods, such as explicit loss notification (ELN) [5], which make the TCP stack aware of packet losses due to link errors, need some sort of coupling between TCP and the link level, and require TCP

[1] This work is supported by the Ministero dell'Istruzione, Università e Ricerca of Italy under the "TANGO" FIRB project.

This is a preprint of an article published in the International Journal of Communication Systems, Vol. 16, No. 1, pp. 63-79, February 2003,
Special issue on wireless access to the global internet: mobile radio networks and satellite systems

modifications at the end points. Other methods proposed involve changing the TCP stack at the end points, for example TCP-Peach [6].

The idea introduced in this paper is orthogonal to all these techniques (apart from link-level FEC [4], a similar but less general concept). It operates at the physical or link level, by trading the bandwidth of the wireless hop for packet loss rate. It does not interfere in any way with the normal behaviour of the TCP stack, but simply entails the wireless link parameters being appropriately tuned at the physical or link levels. Generally speaking, for any given wireless trasmission method, a number of parameters are chosen to obtain a target performance in terms of bit error rate (BER) and information bit rate (IBR). For a given available radio spectrum, antenna size and maximum transmission power, the selection of a modulation scheme and various FEC types allow a usually wide range of choices. Commonly used wireless systems make such choices in a static fashion, by possibly permitting the user to manually change some of them, or in some cases by dynamically switching between a limited number of preset parameters [7]. Even in those cases where a wide range of IBR values is available, such as the ITU V.34 telephone modem standard, where the symbol rate and the modulation scheme can be changed, deciding whether to switch between them only depends on the perceived channel quality [8]. The reason for switching from one set of parameters to another is that most wireless links have highly variable physical characteristics: satellite links for example are subject to variable atmospheric attenuation of the signal; low and medium earth orbit satellite constellations suffer from variable signal attenuation due to changing slant path, blocking due to obstacles in the line of sight, multipath fading and changing satellite distances; and finally, indoor systems suffer severely from interference from similar systems as well as all the previous reasons. In addition, the bandwidth itself on a given link varies with time, because of dynamic bandwidth allocation among different users. We argue that it is possible to obtain a better performance for TCP connections by jointly choosing the BER and IBR of the wireless links that maximise a TCP connection goodput, i.e. the end-to-end transfer rate.

2. MINIMUM BER VERSUS AVAILABLE INFORMATION BIT RATES

Let us suppose that, for a given wireless transmission system, some of the parameters are dynamically tunable, so that the channel BER can be traded for the IBR (information bit rate). One possibility is to change the modulation scheme, another, which is much more flexible and usually cheaper, is to change the type of FEC (forward error correction) coding used. Most modern transmission systems provide for variable information bit rates by changing the FEC redundancy, some of them for each individual packet. FEC can be applied at different levels: using rate-compatible punctured convolutional codes [9] it is possible to seamlessly change the FEC while maintaining bit time synchronisation of the data stream. Reed-Solomon codes can be applied at the channel level, typically as *outer codes* with respect to an *inner* convolutional code. A similar effect can be obtained at the link level by splitting the TCP packets into smaller link units to which an erasure code is then applied prior to transmission [4],[10].

For a given transmission system, it is always possible to define a family of graphs of minimum BER versus IBR, characteristic of the considered equipment. These indicate how much gain can be obtained in the BER for a given channel condition by sacrificing the IBR, i.e. by increasing the redundancy of transmitted data using one or more of the aforementioned methods. A member of such a family of graphs would be a non decreasing function of BER versus IBR.

Let us define $\boldsymbol{C}$ as the set of possible channel conditions $\boldsymbol{c}$ for some given communication equipment. For each element $\boldsymbol{c}$ of $\boldsymbol{C}$, a BER vs. IBR graph can be drawn that represents the minimum BER that can be achieved by varying the different transmission parameters. In other

words, for a given channel condition $\boldsymbol{c}$, the graph associated with $\boldsymbol{c}$ shows the best link quality achievable versus the redundancy applied to the transmitted data.

In order to describe the available trade offs for a given communication system, the minimum BER vs. IBR graph should be computed or measured for each element $\boldsymbol{c}$ of the set $\boldsymbol{C}$. In practical situations, $\boldsymbol{C}$ can be reduced to a manageable set of easily measurable channel conditions. For example, in a geostationary satellite communication system, $\boldsymbol{C}$ may be reduced to the span of atmospheric attenuations in the system operating range; in a wireless system, $\boldsymbol{C}$ could include the possible combinations of distances from a base station and amounts of multipath fading.

Since the set of available channel parameters is generally discrete, the graphs will be staircases. Let us define $\boldsymbol{P}$ as the set of possible parameter settings $\boldsymbol{p}$ of the considered equipment. For a given $\boldsymbol{c}$, each $\boldsymbol{p}$ will correspond to a point in the BER-IBR plane. The staircase connecting the minimum points is the graph relative to $\boldsymbol{c}$, and the corresponding $\boldsymbol{p}$ points are the parameters that define the operating points relative to the graph. Figure 2 shows two examples of such graphs, where the parameter settings useful for obtaining the minimum BER are circled.

Armed with the graphs for the channel conditions $\boldsymbol{c} \in \boldsymbol{C}$, we want to find the optimum operating point for each $\boldsymbol{c}$, from the viewpoint of a TCP connection. Since the set of graphs is known a priori and only depends on the characteristics of the equipment, the search for the optimum operating point is done only when tuning some given equipment, so its computational complexity is not an issue. The result of the tuning operation is a lookup table that is used during normal operations to get the transmission parameters $\boldsymbol{p}$ given the channel condition $\boldsymbol{c}$. In the case of dynamic bandwidth allocation, such a lookup table should be precomputed for each possible bandwidth share. With reference to Figure 2, we want to find out, for each channel condition $\boldsymbol{c}$, which is the best circled point on the IBR-BER plane and, consequently, the optimum transmission parameters $\boldsymbol{p}$. Sections 4 and 5 deal with this problem.

3. TCP OVER WIRELESS LINKS

The Transmission Control Protocol (TCP) is a connection-oriented, end-to-end reliable transport protocol working between hosts in packet-switched networks of any possible topology. The first version of TCP is described in RFC 793 [11].

TCP is designed to operate over a wide spectrum of communication systems, ranging from wired to wireless and satellite networks. However, TCP performance may be severely impaired in environments where a high delay-bandwidth product is associated with a non negligible packet loss due to data corruption. This occurs for example in satellite networks, or when the path between two TCP end-nodes consists of a high number of hops and at least one of them is wireless. Research in this area has long since highlighted this problem, and the literature is rich both in the performance evaluation area and in improvement suggestions [12], [13], [14], [15].

The performance problems of TCP over wireless links, especially on satellite links, are becoming more and more important as wireless links become more widespread and their capacity increases. In future years, we can foresee an undiminished importance of satellite links, because of their intrinsic advantages with respect to wired links, which we briefly list [16] as follows:

- *Ease of scalability*: a new user can join a satellite network by acquiring the appropriate transceiver, whose installation is generally much quicker and cheaper than cabling.
- *Resilience to terrestrial damage*: natural or war disasters do not affect a satellite network as much as a wired network.
- *Multicasting efficiency*: since the satellite is inherently a broadcasting tool, broadcasting and multicasting services are efficiently obtained with simple protocols.

- *Ease of setup*: one individual large-scale organization can set up a completely private worldwide network, for reasons of privacy, security and provider independence.

TCP congestion control mechanism

Table I explains some terms used when describing the TCP Reno congestion control mechanisms. Most of the information is taken from [17].

A segment is considered as being lost either after the retransmission timeout expires and no acknowledgement is received [11], or after three duplicate acknowledgements (four acknowledgements indicating the same sequence number) are received. Any segment received with its ACK bit set is considered as an acknowledgement. Hereafter, such a segment is sometimes called "an ACK".

The slow start and congestion avoidance algorithms are used by a TCP sender to control the amount of outstanding data being injected into the network. The minimum of *cwnd*, *rwnd* and the source buffer size limits the flight size. Another state variable, the slow start threshold (*ssthresh*), is used to determine whether the slow start or congestion avoidance algorithm should be used to control data transmission, as detailed below.

Table I. Definitions of some TCP Reno parameters.

Parameter	Definition
TCP segment	The TCP header and data, i.e. the IP payload.
Maximum Segment Size (*MSS*)	Size of the largest segment usable for the current connection.
Receiver Window (*rwnd*)	The most recently advertised receiver window.
Flight size	The amount of outstanding data, i.e. sent by the sender for which an acknowledgement has not yet been received.
Congestion Window (*cwnd*)	A TCP state variable that limits the flight size.
Initial Window (*IW*)	Value of *cwnd* at the end of the three-way handshake.

Slow Start (cwnd < ssthresh)

During the slow start phase the sender's throughput increases exponentially with time, in order to avoid an initial sudden burst of packets which would potentially cause network congestion. The initial window *cwnd* is set to be not greater than $2 \cdot MSS$ bytes, although an experimental TCP extension [18] allows a larger window to be used (typically $4 \cdot MSS$, provided that *IW* is lower than 4380 bytes). During a slow start, a TCP increases *cwnd* by at most *MSS* bytes for each acknowledgement received that acknowledges new data. The slow start phase ends when *cwnd* exceeds *ssthresh* or when congestion is detected.

Congestion Avoidance (cwnd > ssthresh)

When *cwnd* exceeds *ssthresh*, the sender enters the congestion avoidance phase. Congestion avoidance continues until congestion is detected. During this phase *cwnd* is increased by $1/cwnd$ per each acknowledgement received, which translates into a linear throughput increase with respect to time of about one *MSS* per round trip time.

When a TCP sender detects a segment loss because of a retransmission timeout, the value of *ssthresh* is set to $\max\{FlightSize/2, 2 \cdot MSS\}$. Furthermore, upon retransmission timeout, *cwnd* is set to *MSS*, regardless of the value of *IW*. Therefore, after retransmitting the dropped segment, the TCP sender uses the slow start algorithm to increase *cwnd* up to the new value of *ssthresh*, then congestion avoidance takes over again.

Fast Retransmit (three duplicate ACKs)

When a TCP receiver receives an out-of-sequence segment, it immediately sends an ACK indicating the next sequence number expected. An acknowledgement sent with the same expected sequence number as the previous ACK is called a "duplicate ACK". Duplicate ACKs can be generated because of a number of network problems, such as packet reordering, duplication or dropping. After receiving three duplicate ACKs, the sender retransmits what appears to be the missing segment, without waiting for the retransmission timer to expire. This is called fast restransmit.

Fast Recovery

Immediately after a fast retransmit, new data are sent in accordance with the fast recovery algorithm, until a non-duplicate ACK is received. The reason for not performing slow start instead is that the receipt of each duplicate ACK indicates that a new segment has been received, and thus has left the network.

Implementation of fast retransmit and fast recovery

1. Upon reception of the third duplicate ACK, *ssthresh* is set to max{*FlightSize*/2, 2·*MSS*}.
2. The lost segment is retransmitted and *cwnd* set to *ssthresh*+3·SMSS. This helps to keep the pipe full by compensating for the fact that the duplicate ACKs, while indicating that three packets left the network, did not move the right edge of the transmission window.
3. On each further duplicate ACK received, *cwnd* is increased by *MSS*. The rationale is the same as in the previous point. Without this temporary inflation of *cwnd*, a burst of segments would be sent out when an ACK with a different sequence number is eventually received.
4. During fast recovery a segment is transmitted, as usual, when allowed by the current value of *cwnd* and *rwnd*.
5. When the next ACK arrives that acknowledges new data, *cwnd* is set to *ssthresh* (the value set in step 1).

The dynamics of the above algorithm are exhaustively illustrated in [19].

4. PERFORMANCE OF A SINGLE TCP CONNECTION OVER A BANDLIMITED LINK

TCP performance is significantly influenced by the packet loss rate. In order to find an optimal operating point for the equipment, we need to compute the throughput of a TCP connection given the link bandwidth and the packet loss, either analytically, by simulation or using measurements. This computation can then be used to choose the optimal wireless transmission parameters depending on the current channel condition. In the following, we use analytic methods to illustrate the operation of the proposed procedure.

In 1997, many researchers realized that a simple and elegant formula (the *square root formula*) could capture the steady-state behaviour of TCP, assuming a low (less than 1%) segment loss rate [20]. The square root formula connects the maximum throughput on an unlimited bandwidth channel with the round trip time *RTT* and the segment loss rate *q*:

$$\text{throughput} = \frac{K}{RTT\sqrt{q}} \text{ [segments/s]},$$

where *K* is a constant whose value is $\sqrt{3}/2$ in the case of periodic segment losses and increases with the burstiness of the packet loss process [21]. When not using delayed acknowledgements, as we will do in the following, the value of *K* is multiplied by $\sqrt{2}$.

This is a preprint of an article published in the International Journal of Communication Systems, Vol. 16, No. 1, pp. 63-79, February 2003,
Special issue on wireless access to the global internet: mobile radio networks and satellite systems

In [4], Barakat and Altman compute the TCP throughput at the end nodes (sometimes called *goodput*) as the minimum between the square root formula and the channel bandwidth, with the simplifying assumption that the square root formula holds for a bandlimited channel with random loss.

The analytic computation in this section derives from the approach used by Lakshman and Madhow [22], who proposed a method for computing the TCP throughput for bandlimited channels which is more precise than the square root formula. We only consider the case of a single TCP connection, leaving the case of multiple TCP connections for further study. There are small differences between our procedure and the one in [22], which are detailed in the appendix; the following description applies equally well to both.

We consider a Reno TCP implementation without the SACK option [23]. As far as the size of the receiver window is concerned, we assume that the Window Scale option [24] is used, and that the window size advertised by the receiver is never a limiting factor, which means that it is considered infinite for our purposes. Since we also assume that the transmission buffer is sized accordingly to the receive window, the consequence is that the transmission window is always equal to the congestion window *cwnd*. Further assumptions are that the receiving side does not use delayed ACKs (for the sake of simplicity), that the transmitting side never runs out of data to transmit, and that all packet loss detection happens as a result of receiving three duplicate ACKs, never because of a retransmission timeout. All the results assume a steady state TCP connection, i.e. a long-running connection where the length of the Slow Start phase is negligible with respect to the length of Congestion Avoidance phases.

In Table 2 we summarize the procedures assumed for the analysis of the congestion mechanism.

Table 2. Reno TCP congestion control rules.

Slow Start (*cwnd* < *ssthresh*)	*cwnd* = 2; *ssthresh* = *FlightSize* non dupACK → *cwnd* = *cwnd*+1
Congestion Avoidance (*cwnd* ≥ *ssthresh*)	non dupACK → *cwnd* = *cwnd* + 1/*cwnd* third dupACK → go to Fast Retransmit
Fast Retransmit	*ssthresh* = max(*FlightSize*/2, 2); *cwnd* = *ssthresh* + 3·*MSS*; go to Fast Recovery
Fast Recovery	dupACK → *cwnd* = *cwnd*+1 non dupACK → {*cwnd* = *ssthresh*, go to Congestion Avoidance}
Retransmission timeout	*cwnd* = 1; *ssthresh* = max(*FlightSize*/2, 2); go to Slow Start

We consider a system where a buffer of capacity B is associated with the bottleneck link, whose transmission rate is μ segments/s. Denoting by T the complete round-trip delay, consisting of the link latency τ plus the segment service time, we have $T = \tau + 1/\mu$. We also define the normalized buffer size $\beta = B/(\mu T)$ and denote by ω the congestion window size *cwnd* expressed in segments. We define a *flow control cycle* as the system evolution between the beginning of two consecutive Congestion Avoidance phases; during such a cycle n segments are transmitted, and ω goes from a minimum value ω_{min} to a maximum ω_{max}, where the latter value is reached when a segment loss is detected, that is, when the first of four equally numbered ACKs is received. If we momentarily neglect the temporary *cwnd* inflation during the Fast Recovery phase, we can observe that ω_{max} reaches the value $\mu T + B$ when the segment loss is due to congestion, but lower values when the segment loss is due to random data corruption.

Using the above notations, and denoting by q the segment loss rate, the goodput λ can be computed as (see Appendix)

$$\lambda = \sum_{n=1}^{n_{a\max}} p(n) \frac{n}{\left(-\omega_l + \frac{1+2n+\omega_l^2}{\sqrt{2n+\omega_l^2}} \right)} + \sum_{n=n_{a\max}+1}^{n_{\max}-1} p(n) \frac{n\,\mu}{1+n-n_{a\max}+\mu^2T^2-\mu T\omega_l} + \frac{(1-q)^{n_{\max}}\,\mu n_{\max}}{1+\mu\, t_{\max}} \quad (1)$$

where, for β in [0; 1]:

$$p(n) = (1-q)^n q\,, \qquad \omega_l = \begin{cases} \sqrt{\frac{2(1-q)}{3q}}, & \frac{1-q}{q} \le \frac{3}{8}\mu^2T^2 \\ \min\left\{ \sqrt{\frac{1}{2}\left(\frac{1-q}{q} + \frac{\mu^2T^2}{8} \right)}, \frac{\mu\, T\,(1+\beta)}{2} \right\}, & \frac{1-q}{q} > \frac{3}{8}\mu^2T^2 \end{cases},$$

$$n_{a\max} = \frac{1}{2}(\mu^2T^2 - \omega_l^2)\,, \quad n_{\max} = \frac{1}{2}\left((1+\beta)^2\mu^2T^2 - \omega_l^2 \right), \quad t_{\max} = \frac{1}{2}T\left(\left(2+2\beta+\beta^2\right)\mu\, T - 2\omega_l \right).$$

Table 3 is a comparison of our results with those obtained in [22] and with the simple approach adopted in [4], which yields $\lambda/\mu = \min\left[1/\left(\tau\sqrt{2q/3}\right), \mu\right]/\mu$.

Table 3. Steady-state normalized throughput using TCP Reno. The segment loss rate q depends on the segment size, the bit error rate, and the error process characteristics.

Link utilization (λ/μ) μ=100 [segments/s], τ = 1 [s], β = 0.8 (values as in [22])				
q	Square root formula [4]	Results from [22]		Analisys as in Appendix
		Analysis	Simulation	
10^{-1}	0.038	0.035	0.019	0.028
10^{-2}	0.121	0.116	0.108	0.112
10^{-3}	0.383	0.369	0.381	0.367
10^{-4}	1	0.952	0.911	0.923
10^{-5}	1	0.994	0.989	0.994
10^{-6}	1	0.996	0.994	0.996
0	1	0.996	0.994	0.996

It is worth noticing that the implementation complexity of the procedure proposed in this paper does not depend on the choice between the simplified approach taken in [4] – that is, using the square root formula – and the more complex one taken in [22] that we have described in this section. In fact, during the tuning phase of some given equipment, any of these analytical methods as well as simulation or measurement based methods could be used. On the other hand, during normal operation, the implementation of the proposed procedure only employs precomputed lookup tables.

5. OPTIMUM OPERATING POINT FOR A GIVEN CHANNEL CONDITION

Using the results in Section 4 and the notation defined in Section 2, we can now find, for each channel condition ***c***, the point on the relative graph where the TCP throughput is maximum. We define this point as the optimum operating point ***O***(***c***) for a given ***c***. Since the graph relative to ***c*** is dependent on the equipment characteristics alone, it is possible to compute the complete set of the

operating points, $\boldsymbol{O}(\boldsymbol{C})$, once and for all for some given wireless transmission equipment. Taking Figure 2 as a reference, the optimum operating point will be chosen from the circled points on the graph. The choice will be made using formula (1). Since each $\boldsymbol{O}(\boldsymbol{c})$ corresponds to certain transmission parameters $\boldsymbol{p}(\boldsymbol{O}(\boldsymbol{c}))$, we will have a mapping of the optimal transmission parameters on all the considered channel conditions $\boldsymbol{C}$, which was our objective. In the case of dynamic bandwidth allocation, one such mapping should exist for each possible per-user allocation.

We can informally observe that a non-trivial optimum operating point for a given $\boldsymbol{c}$ does indeed exist. Let us suppose that, given a fixed radio spectrum, antenna size and maximum transmission power, BER is a non decreasing function of IBR. If we let IBR increase arbitrarily, BER increases up to its limiting value 0.5, corresponding to a limiting goodput value of 0. If we let BER decrease arbitrarily, IBR decreases down to its limiting value 0. Being IBR an upper limit for the goodput, the border conditions correspond to a goodput approaching 0. Being the goodput a positive function of the channel conditions, there exists a point of maximum for it.

Bottleneck link rate

Let us suppose that the wireless link has a channel bit rate R, that the transmission uses a FEC with redundancy r, $r \geq 1$: in this case the information bit rate IBR will be R/r. If the packets transmitted over the wireless link have a TCP/IP overhead of H bits and a payload of MSS bits, we compute the bottleneck link rate as

$$\mu = \frac{R}{r} \cdot \frac{1}{H + MSS} \text{ [segments/s].}$$

The same formula applies for whichever method is used for obtaining a higher data redundancy on the channel, for example by changing the modulation scheme.

Error bursts

When dealing with convolutional coding types of FEC, the resulting bit error rate is not a sufficient measure for characterising the data path behaviour. In fact, after decoding a convolutionally encoded data stream, errors appear in bursts [25]. Assuming the error burst occurrences as a binomial process [26], we denote by b the error burst probability, that is, the expected number of error bursts in a sequence of a given bit length divided by this length. We further denote by $p(l)$ the probability that the length of a given burst is exactly l bits, and we assume that $\mathrm{E}\{l\} << 1/b$, that is, we assume that the bursts are spaced apart, which is the case of a small BER. For greater BER values, the TCP dynamics are dominated by the retransmission timeouts, and formula (1) does not hold.

Under the above assumptions, the expected number of packets corrupted as a consequence of a burst of errors is

$$1 + (\mathrm{E}\{b\} - 1) / (H + MSS).$$

Since for all cases of practical interest, i.e. for all BER values smaller than 10^{-3}, the average burst length $\mathrm{E}\{b\}$ is much smaller than the packet length, it is safe to assume one errored segment per error burst.

Apart from the first and last bits in an error burst, which are errored by definition, the others are errored with probability $\theta \approx 0.6$ [26]. Denoting with e the number of errored bits in an error burst, the segment loss rate due to channel noise is thus simply

$$q = \mathrm{BER}/\mathrm{E}\{e\} \cdot (MSS+H).$$

This is a preprint of an article published in the International Journal of Communication Systems, Vol. 16, No. 1, pp. 63-79, February 2003,
Special issue on wireless access to the global internet: mobile radio networks and satellite systems

Optimum packet length

One more issue to consider is choosing the optimum packet length, that is the value of *MSS* that maximises the TCP throughput on the wireless link. We considered a 2.048 Mb/s IBR channel at different values of the BER. The error distribution is assumed to be a binomial process, τ is 0.5s and β is set at 0.8. From Figure 1 it is apparent that long packets should be used in order to maximise the throughput on a channel with errors. For each packet, a TCP/IP header of 40 bytes is considered in addition to the payload length *MSS*.

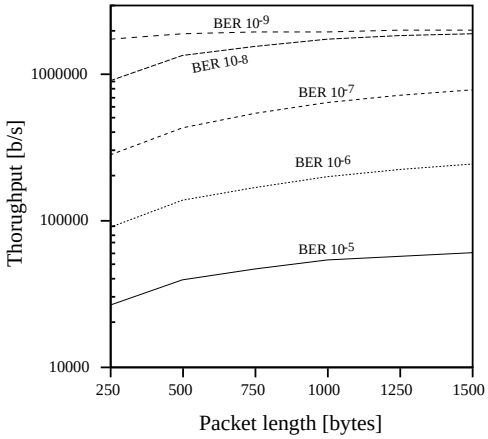


Figure 1. TCP throughput versus segment size for different bit error rates.

One effect explaining why longer segments are more convenient derives from the fact that the slope of the throughput during Congestion Avoidance is proportional to *MSS*. This means that a flow control cycle is shorter, consequently the probability of losing a segment because of random corruption is lower, and the frequency of flow control cycles whose throughput does not reach the maximum allowable by the bottleneck link is lower.

6. AN APPLICATION EXAMPLE

The proposed scenario is composed of a number of users sharing a wireless channel, each user thus seeing a variable bandwidth wireless link; a typical example would be a wireless access point to the fixed network accessed by a variable number of nomadic users. The wireless transceivers are assumed to have been tuned appropriately, and a set of lookup tables generated such that, for any channel share R dedicated to a given user and for any channel condition $\boldsymbol{c}$, the transceivers choose the best set of transmission parameters $\boldsymbol{p}$, such as modulation scheme, channel level FEC and link level FEC. This choice is dynamically updated for each user, by referring to the lookup tables whenever the values of R or $\boldsymbol{c}$ change. The whole procedure is done only on the wireless transceivers, and is independent of the end nodes and of the traffic on the link. In other words, the wireless transceivers optimise the link as seen by each individual user for the case of a single steady-state TCP connection traversing it, irrespectively of the type of traffic that actually traverses the link.

Let us now illustrate the detailed procedure for building the lookup tables, that is, for computing the set $\boldsymbol{O}(\boldsymbol{C})$ of the transmission parameters on a wireless link that is optimal for TCP. We will produce numerical results for a simple case. Consider a TDMA (Time Division Multiple Access)

modem for geostationary satellites built around a hardware Viterbi decoder [27] for the standard NASA 1/2 rate code with constraint length 7, capable of working at 4.096 Msymbols/s either with BPSK or QPSK (Binary / Quadrature Phase Shift Keying) modulation schemes at coding rates 1, 7/8, 3/4, 1/2, using punctured coding where necessary. The spectrum occupancy is the same both in BPSK and in QPSK, and their redundancy *r* is 2 and 1 respectively, while the redundancy for the four convolutional coding rates is 1, 8/7, 4/3 and 2, respectively. A dynamic bandwidth allocation scheme is considered, where the user gets one of three different channel shares, corresponding to bit rates on the channel *R* equal to 0.512, 2.048 and 8.192 Mb/s, respectively.

*Table 4. The transmission parameters **p** and the corrensponding redundancy r and information bit rate IBR [Mb/s] values for the three cases of R equal to 0.512, 2.048 and 8.192 Mb/s.*

p	no coding	7/8 code	3/4 code	1/2 code
BPSK	r = 2 IBR = 0.256,1.024,4.192	r = 2.29 IBR = 0.224,0.896,3.584	r = 2.67 IBR = 0.192,0.768,3.072	r = 2 IBR = 0.128,0.512,2.048
QPSK	r = 1 IBR = 0.512,2.048,8.192	r = 1.14 IBR = 0.448,1.792,7.168	r = 1.33 IBR = 0.384,1.536,6.144	r = 2 IBR = 0.256,1.024,4.192

For each of the three values of *R* a lookup table should be computed during the tuneup phase of the wireless equipment. Apart from the modulation scheme and the convolutional coding rate, no other transmission parameters can be varied, so we have a set of eight ***p*** transmission parameters for each of the three cases, that is, two choices for the modulation scheme and four choices for the convolutional coding rate. Table 4 summarises the values of redundancy and IBR = *R*/*r* for the eight possible transmission parameters.

Since we are referring to a geostationary satellite, the most important parameter for describing the channel conditions is by far the signal to noise ratio, so we consider this parameter only, expressed in decibels as the carrier power to the one-sided noise spectral density ratio (C/N_0 [dB]). Let us further suppose that our modem is capable of working in a 12 dB range for C/N_0, from 66 to 78 dB, and that the granularity for the measurement of the attenuation is 1 dB. The set ***C*** consists thus of 13 possible different channel conditions ***c***. For each ***c*** we should build the minimum BER vs. IBR graph, as shown in Figure 2.

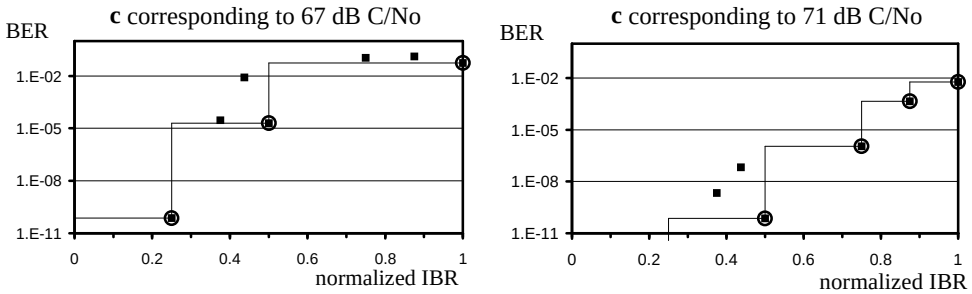
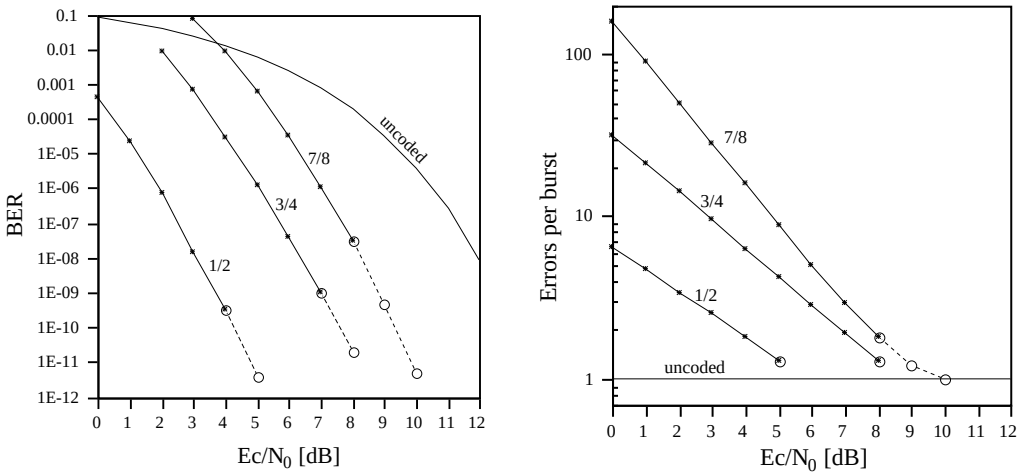


*Figure 2. Minimum BER vs. IBR graphs for two channel conditions **c**.*

The data used in the plots in Figure 3 are taken from the Qualcomm Viterbi decoder data sheet [27]. The complete set of data is plotted in Figure 3a as BER versus E_c/N_0, that is, the ratio of channel bit energy to the one-sided noise spectral density, which is equal to C/N_0 divided by the modem channel bit rate. In order to work with a more significant range of data, we extrapolated the numbers from the data sheet. The extrapolated data are marked on the plot with a circle.

This is a preprint of an article published in the International Journal of Communication Systems, Vol. 16, No. 1, pp. 63-79, February 2003,
Special issue on wireless access to the global internet: mobile radio networks and satellite systems

Figure 3b shows the average number of errors per error burst, obtained through software simulation using the Phil Karn streaming codes [28]. Since we needed to evaluate error burst characteristics for BER values much less than 10^{-9}, we resorted to extrapolation for some points, which are marked with a circle on the plot.



Figures 3a and 3b. BER and average number of errors per burst versus E_c/N_0 (channel bit energy over single sided noise spectral energy ratio), for different values of the coding rate. Extrapolated values are circled.

Following the procedure outlined in the previous sections, we obtain the values summarised in Figure 4 using a segment size *MSS* of 960 bytes (7680 bits), a link latency τ of 0.5 s and $\beta = 0.8$. The plot shows the TCP throughput computed using formula (1) for the 13 different channel conditions $\boldsymbol{c}$ in the case of R = 2.048 Mbit/s. The set $\boldsymbol{O}(\boldsymbol{C})$, defined in Section 5, is the envelope of the curves plotted in Figure 4, that is, the set of the maximum throughputs for each channel condition.

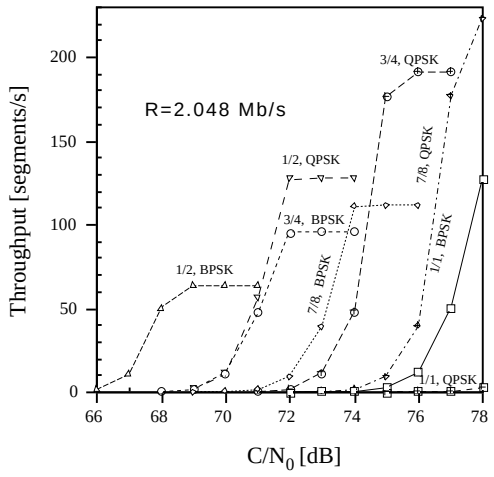


Figure 4. TCP throughput versus C/N_0, for the complete set of transmission parameters for R = 2.048 Mbit/s.

This is a preprint of an article published in the International Journal of Communication Systems, Vol. 16, No. 1, pp. 63-79, February 2003,
Special issue on wireless access to the global internet: mobile radio networks and satellite systems

The set of optimal points obtained in Figure 4 only needs four transmission parameters ***p*** out of the possible eight. It is interesting to note that ***p*** corresponding to uncoded QPSK ($r = 1$) is never used, even though it provides the maximum information bit rate. In the C/N_o range considered, in fact, uncoded QPSK provides at best a packet loss rate of 0.24, which is practically unusable for TCP.

Figure 5 compares between the results obtained with the proposed method and those obtained with a *maximum tolerable BER* criterion for the three channel shares *R* considered.

The maximum tolerable BER criterion is defined so that, for any given channel condition ***c***, the transmission parameters ***p*** should be chosen in a way that maximises the IBR subject to the constraint that the bit error rate does not exceed a predefined threshold [29]. This is a reasonable criterion for choosing the transmission parameters without focusing on TCP performance. The plots in Figure 5 highlight that no choice of a single threshold for the BER provides the same performance as our method for the whole range ***C*** of channel conditions in the three cases.

The curves corresponding to the BER thresholds occasionally exhibit non monothonic behaviour; this happens, for example, at 76 dB for $BER \le 10^{-6}$. The reason is that, by imposing a maximum BER threshold, the maximum tolerable BER criterion often requires the use of transmission parameters that actually provide a much better BER. In fact, the points in Figure 5 at $C/N_0 = 76$ dB for the curve $BER \le 10^{-6}$ correspond to transmission parameters ***p*** that incidentally provide a BER of exactly 10^{-6}, while the points on the same curves at $C/N_0 = 75$ dB and 77 dB correspond to transmission parameters for which the BER is smaller, and consequently the TCP throughput is higher. This apparently strange behaviour is one consequence of the fact that the maximum tolerable BER criterion does not account for the peculiarities of TCP.

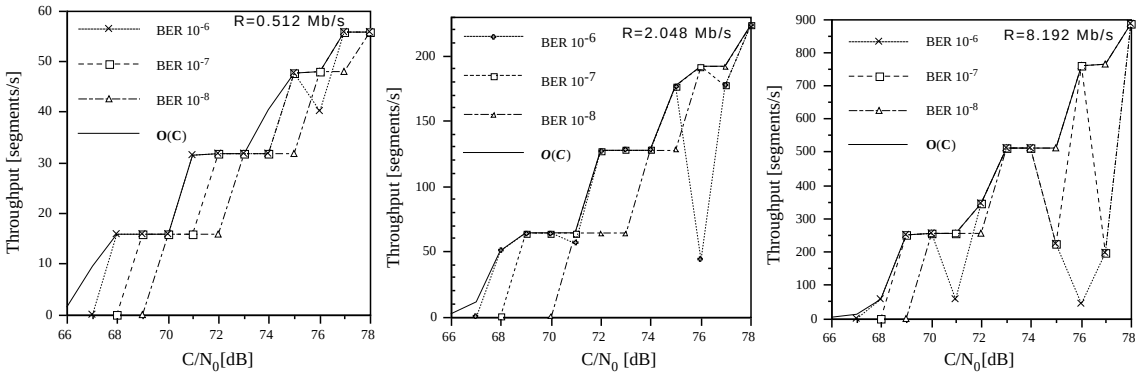


*Figure 5. Comparison of the maximum tolerable BER criterion using three different BER thresholds (10^{-6}, 10^{-7}, 10^{-8}) with **O(C)** computed using the proposed method. The three cases for R set at 0.512, 2.048 and 8.192 Mb/s are depicted, respectively.*

Another reasonable criterion for choosing the transmission parameters, which is often cited in TCP performance discussions, entails making the wireless channel transparent from the TCP point of view. This means that the packet loss rate introduced by errors on the channel should be negligible with respect to that introduced by congestion [13]. In other words, the packet loss rate caused by BER alone should be much less that that caused by the TCP dynamics due to IBR restriction alone.

In contrast, for the limited set of cases in our examples, corruption on the channel and congestion give similar contributions to the total packet loss in the points of maximum goodput. This observation will be the subject of further research.

This is a preprint of an article published in the International Journal of Communication Systems, Vol. 16, No. 1, pp. 63-79, February 2003,
Special issue on wireless access to the global internet: mobile radio networks and satellite systems

7. CONCLUSIONS

Wireless is a transmission environment where improving the bit error rate on a link is generally expensive, and may not even be possible at all. We have outlined a general framework for tuning some given transmission equipment in order to maximize the throughput of a single TCP connection passing through the wireless link by trading information bit rate for bit error rate. The optimal operating point is dependent on the channel conditions, and should be precomputed once for some given equipment. During normal operation the wireless equipment, which must be able to measure the channel conditions in real time, chooses its transmission parameters by using a simple lookup table, or multiple lookup tables in the case of dynamic bandwidth allocation.

This paper is a description of the concept, in fact it avoids dealing with many difficulties which themselves are the subject of ongoing research. We make no attempt to consider the effect of multiple different wireless links on the path of a single TCP connection, despite the fact that locally computed optimality does not necessarily mean global optimality. We do not consider what happens when multiple TCP connections share the same link. We do not attempt to model short-lived TCP connections. None of these issues can be disregarded, and they should be studied in order to assess the practical feasibility of the concept.

Additional dimensions in the space of transmission parameters are worth considering besides the modulation type and convolutional coding used in our example, such as Reed-Solomon channel coding and link-level erasure codes.

Once the above issues have been tackled, simulation studies will be required to prove that the concept is applicable in practice. This will be left for further research.

APPENDIX

In the steady state analysis of the behavior of TCP we do not consider the Slow Start phase, which produces a single initial transient. This is justified if the chance of a retransmission timeout is negligible, which is a reasonable assumption for packet loss rates less than 10^{-3} [20]. The dynamics of the flow control is thus limited to the Congestion Avoidance and Fast Retransmit / Recovery phases. In the former phase, if losses are only due to congestion, the window size ω starts at a minimum value $\omega_{\min}$ that is set at the end of the previous cycle and increases linearly up to the value μT. From this point on the bottleneck link is saturated, the arrival rate of the ACKs[2] is equal to μ, and the rate of increase of the congestion window with time becomes equal to the inverse of its size. A fluid-model description of the described behaviour is

$$\frac{d\omega(t)}{dt} = \begin{cases} \dfrac{1}{T}, & \omega \le \mu T \\ \dfrac{\mu}{\omega}, & \omega \ge \mu T \end{cases}. \tag{A1}$$

The bottleneck link buffer stays empty during the linear phase, then it starts filling up to the value B, when a segment is lost due to buffer overflow (congestion). At this point the sender does not know about the loss yet, so it keeps sending a new segment for each ACK it receives[2]. According to points 2 and 3 in Section 3, the control dynamics are the same as if the window reached the size $\omega_{max} = \mu T + B$ and then was immediately halved, apart from the missed opportunity of transmitting one new segment, because TCP must retransmit the lost one.

[2] We assume no delayed ACKs.

When the loss is due to data corruption, the above described event occurs before ω reaches ω_{max}. Each cycle begins with a value of ω which is half the value attained in the previous cycle. The following discussion is taken from [22], with small differences which improve the accuracy of the results for some values of q.

The basic approximation is that, for each cycle, the congestion window starts at a fixed value ω_l that is computed by considering the number of segments sent in each cycle (i.e. between two consecutive losses), which is [3] $(1-q)/q$; we then define t_l as the fixed time duration of a cycle in the linear window growth phase. Integrating the first of (A1) we have

$$\omega(t_l) = \omega_l + t_l / T = 2\omega_l \tag{A2}$$

and then

$$t_l = \omega_l T . \tag{A3}$$

In the time interval T, a number of segments equal to the window size is sent, so the segment rate is $\omega(t)/T$. The number of segments sent in a cycle is

$$n(t_l) = \int_0^{t_l} \omega(t)/T \, dt \;=\; \frac{1}{T}\left(\omega_l t_l + \frac{t_l^2}{2T}\right). \tag{A4}$$

Substituting (A3) in (A4) we get

$$\omega_l = \sqrt{\frac{2n(t_l)}{3}} = \sqrt{\frac{2(1-q)}{3q}} . \tag{A5}$$

Equation (A5) is valid during the linear window growth phase. At the end of this phase, ω_l equals $\mu T/2$, and the number of segments sent is $n_l = 3/8 \cdot \mu^2 T^2$.

If q is such that $(1-q)/q > 3/8 \cdot \mu^2 T^2$, then[4] let us consider the non-linear phase, that is, the time from when the congestion window exceeds μT and the time when a packet is lost; during this phase, whose length we call Δt, the link is saturated, the buffers fills up, the window grows non linearly and n_s segments are transmitted, with

$$n_s = \mu \Delta t = \frac{1-q}{q} - \frac{3}{8}\mu^2 T^2 . \tag{A6}$$

By integrating the second of (A1) in the non-linear phase, during which the throughput is constant and equal to μ, we obtain

$$\frac{1}{2}\left(\omega^2(\Delta t) - \mu^2 T^2\right) = \mu \Delta t \tag{A7}$$

and, combining (A6) and (A7), we get

$$\omega_l = \omega(\Delta t)/2 = \sqrt{\frac{1}{2}\left(\frac{1-q}{q} + \frac{\mu^2 T^2}{8}\right)}. \tag{A8}$$

Denoting by p(n) the probability that n segments be transmitted without any loss, and assuming independent losses, we have p(n) = $(1-q)^n q$.

For each value of n we then compute the relevant throughput dividing n by the time interval $t(n)$ needed to send n segments. The function $t(n)$ depends on the window growing phase. The

[3] In [22] the approximate value $1/q$ is used.

[4] In [22] the non-linear phase in not considered, and (A5) is alwas used.

throughput λ is then obtained by averaging over all values of n up to the value n_{max}, when a congestion loss is experienced.

During the linear phase the function $t(n)$ is obtained by solving relation (A4) with respect to t_l. This phase lasts up to the time $t_a = \mu T^2 - \omega_l T$, obtained by (A2) by setting $\omega(t_l) = \mu T$. The value $n_{a\max}$ is obtained as $n(t_a)$ from (A4). The time interval $t(n)$, needed to send n segments, is then incremented by the time to send one more segment, to take the missed transmission opportunity into account[5].

Considering that ω segments are sent during the interval T, and considering (A2), the average time to send one segment can be computed as $T/\omega(n) = T^2/(T\omega_l + t(n))$. For segment sequences greater than $n_{a\max}$ the system enters the non-linear phase. The time interval $t(n)$, needed to send n segments, is then equal to $t_a + (n - n_{a\max})/\mu$, while the time to send one more segment is $1/\mu$. The number of segments $n_{\max}$ which cause a congestion loss is equal to $n_{a\max}$ plus the maximum number of segments in the non-linear phase, which is obtained from (A7), by substituting $\omega(\Delta t)$ with the maximum window size $\mu T(1+\beta)$. We thus obtain $n_{\max} = n_{a\max} + \mu^2 T^2 \beta(2 + \beta)/2$, while the duration of the whole cycle is $t_{\max} = t_a + (n_{\max} - n_{a\max})/\mu + 1/\mu$. The probability to send at least $n_{\max}$ consecutive segments is $(1-q)^{n_{\max}}$. Making all the necessary substitutions and simplifying we get[6] relation (1).

REFERENCES

[1] J. Border, M. Kojo, J. Griner, G. Montenegro, Z. Shelby, “Performance enhancing proxies intended to mitigate link-related degradations”, IETF, RFC 3135, June 2001.

[2] A. Chockalingam, M. Zorzi and V. Tralli, “Wireless TCP performance with link layer FEC/ARQ”, Proceedings of the ICC99, 1999, pp. 1212-1216.

[3] A. F. Canton and T. Chahed, “End-to-end reliability in UMTS : TCP over ARQ”, Proceedings of the IEEE Globecom ’01, 2001, pp. 3473 -3477.

[4] C. Barakat and E. Altman, “Bandwidth tradeoff between TCP and link-level FEC”, Computer Networks, Vol. 39, No. 2, June 2002, pp. 133-150.

[5] H. Balakrishnan and R. H. Katz, “Explicit Loss Notification and Wireless Web Performance”, Proceedings of the IEEE Globecom Internet mini-conference, Sydney (AU), November 1998.

[6] I. F. Akyildiz, G. Morabito and S. Palazzo, “TCP-Peach: a new congestion control scheme for satellite IP networks”, IEEE/ACM Transactions on Networking, vol. 9, no. 3, June 2001, pp. 307-321.

[7] C. B. Cox and T. A. Coney, “Advanced Communications Technology Satellite (ACTS) fade compensation protocol impact on Very Small-Aperture Terminal bit error rate performance”, IEEE Journal on Selected Areas in Communications, Vol. 17, No. 2, February 1999, pp. 173-179.

[8] “A modem operating at data signaling rates of up to 33,600 bps for use on the general switched telephone network and on leased point-to-point 2-wire telephone-type circuits”, ITU-T Rec. V.34, ITU, Geneva, February 1998.

[9] J. Hagenauer, “Rate-compatible punctured convolutional codes (RCPC codes) and their applications”, IEEE Transactions on Communications, Vol. 36, No. 4, April 1988, pp. 389-400.

[10] L. Rizzo, “Effective erasure codes for reliable computer communication protocols”, Computer Communication Review, Vol. 27, No. 2, April 1997, pp. 24-36.

[11] J. Postel, “Transmission Control Protocol, DARPA internet program protocol specification”, IETF, RFC 793, September 1981.

[12] V.G. Bharadwaj, J. S. Baras and N. P. Butts, “An architecture for Internet service via broadband satellite networks”, International Journal of Satellite Communications, Special Issue on IP, Vol. 19, Issue 1, January/February 2001, pp. 29-50.

[13] C. Partridge and T. J. Shepard, “TCP/IP performance over satellite links”, IEEE Network, Vol. 11, No. 5, September/October 1997, pp. 44-49.

[5] In [22] this time increment is neglected.

[6] In [22] the summations in (1) are approximated by integrals which are then numerically solved.

[14] M. Marchese, "Performance analysis of the TCP behavior in a GEO satellite environment", Computer Communications Journal, Special Issue on the Performance Evaluation of Telecommunication Systems: Models, Issues and Applications, Vol. 24, Issue 9, May 2001, pp. 877-888.
[15] M. Allman, S. Dawkinks, D. Glover, J. Griner, T. Henderson, J. Heidemann, S. Ostermann, K. Scott, J. Semke, J. Touch and D. Tran, "Ongoing TCP research related to satellites", IETF, RFC 2760, February 2000.
[16] S. Kota, R. Jain and R. Goyal, "Broadband satellite network performance", Guest Editorial, IEEE Comm. Mag., Vol. 37, No. 7, July 1999, pp. 94-95.
[17] M. Allman, V. Paxson and W. S. Stevens, "TCP congestion control", RFC 2581, April 1999.
[18] M. Allman, S. Floyd and C. Partridge, "Increasing TCP's initial window", IETF, RFC 2414, September 1998.
[19] A. A. Awadallah and C. Rai, "Brief comment on cwnd inflation during fast recovery", online at <URL:http://klamath.stanford.edu/~aaa/tcp/>, June 1997.
[20] M. Mathis, J. Semke, J. Mahdavi and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm", Computing Communications Review, Vol. 27, No. 3, July 1997, pp. 67-82.
[21] E. Altman, K. Avrachenkov, C. Barakat, "A stochastic model of TCP/IP with stationary random losses", Computer Communication Review, Vol.30, No.4, October 2000, pp. 231-242.
[22] 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, Vol. 5, No. 3, June 1997, pp. 336-350.
[23] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, V. Jacobson and R. Braden, "TCP selective acknowledgement options", IETF, RFC 2018, October 1996.
[24] V. Jacobson, R. Braden and D. Borman, "TCP extensions for high performance", IETF, RFC 1323, May 1992.
[25] L. H. Charles Lee, "Convolutional coding: fundamental and applications", Artech House, 1997.
[26] L. J. Deutsch and R. L. Miller, "Burst statistics of Viterbi decoding", NASA Code 310-20-67-62, <URL:http://tmo.jpl.nasa.gov/tmo/progress_report/42-64/64W.PDF>, May and June 1981.
[27] "Q1401: K=7 rate 1/2 single-chip Viterbi decoder technical data sheet", Qualcomm incorporated, September 1987.
[28] P. Karn, "Error correcting codes", < URL:http://people.qualcomm.com/karn/code/fec/>.
[29] B. Vucetic, "An adaptive coding scheme for time-varying channles", IEEE Transactions on Communications, Vol. 39, No. 5, May 1991, pp. 653-663.

AUTHORS' BIOGRAPHY

Nedo Celandroni received the Dr. Ing. degree in Electronic Engineering from the University of Pisa , Italy, in 1973. Since 1976 he has been a researcher with the CNUCE Institute (now ISTI) of the Italian National Research Council (CNR). He worked for the realisation of the Flight Dynamic System of the SIRIO satellite project. Since 1979 he has been involved in the field of digital satellite communications. He participated in several projects in this field: STELLA I/II (Satellite Transmission Experiment Linking Laboratories), FODA (Fifo Ordered Demand Assignment), FODA/IBEA (Information Bit Energy Adaptive), Progetto Finalizzato Telecomunicazioni, Experiments on the satellites Olympus and Italsat. His interest includes rain fade countermeasure systems, data quality estimation, VSAT systems, GEO MEO and LEO satellites for mobile telephony and multimedia systems and wireless networks. He obtained two patents, in 1989 and 1996, for the design of the FODA and FODA/IBEA systems, respectively .

Francesco Potortì received his electronic engineering degree from the University of Pisa, Italy, in 1991. He is currently a full-time researcher at the ISTI-CNR institute in Pisa, Italy, where he has worked since 1989 in the fields of satellite communication protocols and fade countermeasure systems. His research interests include communications protocols and their implementation, wireless and satellite communications, internet technology with regard to integrated services, TCP congestion management and TCP over wireless channels, simulation of communications systems. He coauthored more than thirty refereed scientific papers. He is the real-time software implementor and one of the inventors of the FODA/IBEA system for satellite communications, which has been patented in Italy. He wrote, alone or with others, some telecommunications-related free simulations programs that he uses in his scientific activity. From 1997 to 2001 he has taught computer science and operating systems courses at the Engineering Faculty of the University of Pisa.