GOODPUT OPTIMISATION OF LONG-LIVED TCP CONNECTIONS IN A FADED SATELLITE CHANNEL[1]

(EXTENDED VERSION)

Nedo Celandroni, *Member, IEEE*, Erina Ferro, Francesco Potort

Abstract **The optimization of the end-to-end throughput of a TCP connection over geostationary satellite links is a challenging research topic because the high delay-bandwidth product, together with a non-negligible random loss of packets, are conditions which differ considerably from the original environment for which TCP was originally designed. As a result, TCP performance is significantly impaired by the channel bit error rate. In this paper we investigate the application of different FEC (forward error correction) types/rates and different bit rates, for the optimization of TCP goodput, in transmissions over a rain-faded geostationary satellite channel, provided that the end-to-end protocols are left unaltered. We compare physical-level FEC techniques, such as convolutional encoding/Viterbi decoding and Reed Solomon, link-level erasure codes and their combinations, over a wide field of signal-to-noise conditions of the satellite channel. The case of multiple connections per link is also analyzed, in addition to that of a single connection per link. In order to evaluate the throughput of a TCP long-lived connection we used a method that is midway between analysis (fluid model) and simulation.**

Index Terms **Long-lived TCP connections, rain fading, satellite channel, FEC techniques.**

I. INTRODUCTION

IN order to improve the efficiency of TCP over geostationary satellite channels, where improving the error rate is generally expensive, in our study we operated at the physical or link OSI levels by trading the bandwidth of the satellite link for packet loss rate. This technique does not interfere in any way with the normal behaviour of the TCP stack, but simply entails the choice of transmission techniques over the satellite link and the appropriate tune-up of relevant parameters. Generally speaking, for any given satellite transmission 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 (forward error correction) types usually allows a wide range of choices. Commonly used satellite systems make such choices in a static fashion, by possibly permitting the user to change some of them manually, or in some cases by dynamically switching between a limited number of preset parameters [1]. The reason for switching from one set of parameters to another is that most satellite links have highly variable physical characteristics: for example, they are subject to a variable atmospheric attenuation of the signal. In addition, the bandwidth itself on a given link may vary with time, due to dynamic bandwidth allocation among different users of the same channel In this paper we argue that, given an available radio spectrum, antenna size and transmission power, the selection of a modulation scheme and a FEC type allows choosing the link s BER and IBR that maximize the throughput of a TCP connection (also called goodput), i.e. the end-to-end transfer rate. This optimization can be made for different channel quality conditions whose variability is due, for instance, to the variable atmospheric attenuation of the signal. In order to make the proposed method usable in practice, optimal transmission parameters, for each channel condition, can be reported in look-up tables and dynamically applied in an adaptive fashion. A similar approach was followed by Barakat and Altman in [2], where the FEC is operated at link-level by using a block erasure code and the packet losses are assumed to be both independent and correlated [3]. In [9] we presented a method based on the point-byp-oint optimization of the transmission parameters by operating with a modem that had the ability to switch between BPSK (binary phase shift keying) and Q (quadrature) PSK and with a convolutional/Viterbi encoder/decoder. In this paper we assumed, as in [9], to face with additive white Gaussian noise (AWGN) so that we can consider independent packet losses. This assumption is reasonable when operating with geostationary satellite links and fixed user antennas. Here we extended the results obtained in [9], by introducing Reed Solomon and block erasure codes other than mixed techniques (concatenated codes). A wide scenario of the various techniques, including the one used in [2], was produced for different channel quality conditions, and the relevant comparison was made possible. We also considered the case in which multiple TCP connections share the same satellite link.

II. MINIMUM BER VERSUS AVAILABLE INFORMATION BIT RATES

Let us suppose that, for a given satellite transmission system, several of the parameters are dynamically tunable, so that the channel BER can be traded for the IBR. One possibility is changing the modulation scheme; another, which is much more flexible and usually cheaper, is to change the type of FEC coding used. Most modern transmission systems provide for variable information bit rates by changing

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

the FEC redundancy; some do so for each individual packet. FEC can be applied at different levels: using rate-compatible punctured convolutional codes [4] it is possible to seamlessly change the FEC while maintaining bit time synchronization 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 [2], [5].

For a given transmission system, it is always possible to define a family of graphs of minimum BER versus IBR, which are characteristic of the considered equipment. These indicate how much can be gained in the BER for a given channel condition by sacrificing the IBR, i.e. by increasing the redundancy of transmitted data by 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 representing the minimum BER that can be achieved by varying the different transmission parameters. 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 equipment considered. For a given $\boldsymbol{c}$, each $\boldsymbol{p}$ will correspond to a point in the BER-IBR plane. The staircase that connects 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 1 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 performed 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 pre-computed for each possible bandwidth share. With reference to Figure 1, we want to discover, for each channel condition $\boldsymbol{c}$, which is the best-circled point (and, consequently, the optimum transmission parameters $\boldsymbol{p}$) on the IBR-BER plane, from the viewpoint of maximizing the goodput of a TCP connection.

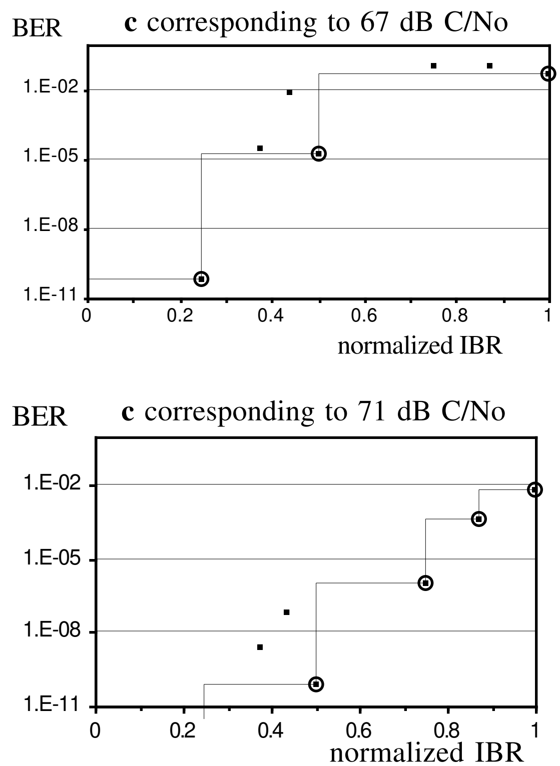


Figure 1. Minimum BER vs. IBR graphs for two channel conditions **c**.

Since we are referring to a geostationary satellite, by far the most important parameter for describing the channel conditions is the signal-to-noise ratio, so we only consider this parameter, 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. Thus, the set $\boldsymbol{C}$ consists of 13 possible different channel conditions $\boldsymbol{c}$. For each $\boldsymbol{c}$ we should build the minimum BER vs. IBR graph, as shown in Figure 1.

III. TCP GOODPUT EVALUATION

A. *TCP Congestion Control Mechanism*

For the reader's convenience we report here a brief description of the TCP Reno congestion control mechanisms. Details can be found in [8-10].

<u>*Some parameter definitions*</u>

- TCP segment: the TCP header plus 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;
- Congestion Window (*cwnd*): a TCP state variable that limits

the flight size;
Fligth size: the amount of outstanding data (not yet acked);
Slow start threshold (*ssthresh*): a threshold beyond which the congestion avoidance phase begins.

TCP Reno congestion control rules

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

The mechanism followed by a TCP sender to ensure the reliable transfer of data is based on the reception of acknowledgements (ACK) from the receiver side and retransmission of lost segments. When a destination TCP receives a segment with a certain sequence number, it sends back an ACK specifying the next expected sequence number which acknowledges all preceding segments. A segment is considered to be lost either after the retransmission timeout expires and no ACK is received, or after three duplicate acknowledgements (four acknowledgements indicating the same sequence number) are received. The slow start and congestion avoidance algorithms are used by a TCP sender to control the amount of outstanding data which is injected into the network. The minimum of the congestion window (*cwnd)*, the receiver window (*rwnd*) and the transmission buffer size limit the flight size. The slow start threshold (*ssthresh*), is used to determine whether the slow start or congestion avoidance algorithm should be used to control data transmission. In the following, the words *segment* and *packet* are used interchangeably.

B. TCP Model

The following fluid model is valid for high delay-bandwidth product links, e.g. for connections with an average flight size of at least ten segments. Losses of segments are due to both congestion and data corruption. We consider a Reno TCP implementation without the SACK [11] and with the Window Scale [12] options, so that *rwnd* never limits the flight size, and the transmission window always results equal to the congestion window *cwnd*. The analysis considers long-lived connections where the length of the Slow Start phase is negligible with respect to the length of Congestion Avoidance phases. Further assumptions are that the transmitting side always has data to transmit, and that all packet loss detections occur as the result of receiving three duplicate ACKs, never because of a retransmission timeout. As in [8], in our system 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 ω goes from a value ω_s to a value ω_e that 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 (see [13] for a detailed illustration), we can observe that ω reaches the value $\mu T+B,$ that we denote by $\omega_{\max}$, when the segment loss is due to congestion, while it reaches lower values when the segment loss is due to random data corruption
In the steady-state analysis of the behavior of TCP, we limit ourselves to considering only the Congestion Avoidance and Fast Retransmit / Recovery phases. This simplification is allowable when connections are sufficiently long and timeout events are rare.

1) Single connection per link.
Let us consider the case in which $B\le\mu T$ $(\beta\le 1)$ first. In the Congestion Avoidance phase the window size ω is increased by the TCP sender by a value $1/\omega$ on each ACK receiving. Denoting by b the number of segments acknowledged by each ACK (b = 2 if the delayed ACKs option is used) we have [16] the following approximated relations: $\frac{d\omega(t)}{dt}=\frac{1}{\omega}\frac{da}{dt}$, where $\frac{da}{dt}$ is the arrival rate of the ACKs. If $\omega\le\mu T$, $\frac{da}{dt}=\frac{\omega}{bT}$, otherwise $\frac{da}{dt}=\frac{\mu}{b}$. Thus we have

$$\frac{d\omega(t)}{dt}=\begin{cases}\frac{1}{bT}, & \omega\le\mu T\\ \frac{\mu}{b\omega}, & \omega\ge\mu T\end{cases}\qquad(1)$$

In the absence of data corruption losses the bottleneck link buffer remains empty during the first phase (linear window growing phase); it then starts filling up to the value B, when a segment is lost due to buffer overflow (congestion). The control dynamics is the same as if the window reached the size $\omega_{\max}=\mu T(1+\beta)$ and were then immediately halved. From the first of (1) we have

$$\omega(t)=\omega_s+t/(bT)\qquad(2)$$

As ω segments are sent in each interval T, denoting by $n(t)$ the number of segments sent from the beginning of the cycle, and considering (2), we have

$$n(t)=\int_0^t\frac{\omega(z)}{T}\,dz=\frac{\omega_s}{T}\,t+\frac{t^2}{2bT^2}\qquad(3)$$

which, inverted, gives

$$t(n)=bT\left(\sqrt{\omega_s^2+2n/b}-\omega_s\right) \tag{4}$$

From (1), for the second phase we have

$$\omega(t)=\sqrt{\omega_s^2+2\mu t/b)} \tag{5}$$

where t is the time from the beginning of the second phase. When the second phase is adjacent to the first one, in (5) $\omega_s=\mu T$. The rate of segment sending is μ, so the number of segments sent is

$$n(t)=\mu t \tag{6}$$

Denoting by t_a and t_b the temporal durations of the first and second phases, when the cycle begins with $\omega_s \le \mu T$, respectively, by setting in (2) $\omega(t_a)=\mu T$, we get $t_a=bT(\mu T-\omega_s)$, and from (6), by setting $\omega(t_b)=\omega_{\max}=\mu T(1+\beta)$, we get $t_b=b\mu T^2\beta(2+\beta)/2^{\circ}$. Denoting by n_a and n_b the number of segments sent in the first and second phases, respectively, we get n_a from (3) as $n_a=n(t_a)$ and n_b from (6) as $n_b=n(t_b)$.

Let us consider the case in which the packet loss is due to congestion only. All cycles are equal and the starting value of the window in each cycle is $\omega_s^{(c)}=\omega_{\max}/2=\mu T(1+\beta)/2$, thus, if $B\le\mu T\,(\beta\le 1)$, the duration of the first phase is $t_a^{(c)}=\mu T^2(1-\beta)/2$. The average goodput can be computed as the sum of segments sent in a cycle divided by the temporal length of the cycle itself. Denoting by λ_c the goodput for congestion losses we have

$$\lambda_c=\frac{n_a^{(c)}+n_b}{t_a^{(c)}+t_b+1/\mu}=\frac{3(1+\beta)^2\mu^3T^2}{4(2+b(1+\beta+\beta^2)\mu^2T^2)}, \tag{7}$$

where $n_a^{(c)}=n(t_a^{(c)})$ from (3) with $\omega_s=\omega_s^{(c)}$.

The term $1/\mu$ at the denominator of (7) takes into account the time spent to resend the segment lost as a result of congestion. Denoting by q_c the segment loss rate due to congestion, and imposing $n_a^{(c)}+n_b=(1-q_c)/q_c$, we get

$$q_c=\frac{8}{8+3b(1+\beta)^2\mu^2T^2} \tag{8}$$

If $B\ge\mu T\,(\beta\ge 1)$, the linear window growing phase does not occur, because $\omega_s\ge\mu T$, so relation (7) becomes

$\lambda_c=\dfrac{3b(1+\beta)^2\mu^3T^2}{8+3b(1+\beta)^2\mu^2T^2}$, which tends to μ for $\mu^2T^2>>1$. The relation (8) is valid for any value of β. We outline the importance of the buffer presence. In fact, from relation (7) we see that the normalized goodput λ_c/μ, for a value of τ=0.5s, which is the link latency for a geostationary satellite, is about 75% without buffer (β=0). Instead, a value of β=0.8 is sufficient to reach 98.5% of the bottleneck rate, even for values of μ as low as 10 segments/s.

When the loss is due to data corruption, it generally occurs before ω reaches $\omega_{\max}$ and each cycle still begins with a value of ω_s which is half the value attained in the previous cycle. In order to compute the average goodput in this case, we wrote a program in *Mathematica*™ language, which is midway between an actual simulator and an analytical fluid model. We assumed that the packet loss follows a geometric distribution with an average loss rate q. We denote by $n_{\max}$ the maximum number of segments in each cycle; $n_{\max}=n_a+n_b$ if $\omega_s<\mu T$, $n_{\max}=\dfrac{b\left(\mu^2T^2(1+\beta)^2-\omega_s\right)}{2}$ otherwise. Starting with the value ω_s, at the beginning of each cycle, we draw the number n of successful segments between two successive losses. We can have $n\le n_{\max}$, or $n>n_{\max}$ In the first case the cycle is prematurely ended by a random loss; otherwise the cycle is ended by a congestion loss. We compute the time duration of each cycle by using relations (2-6) and incrementing such a duration by the time necessary to resend the lost segment. For a number of cycles, the cycle duration and the number of packets sent in relevant cycles are accumulated into the variables t_{tot} and n_{tot}, respectively; the average goodput λ is then computed as $\lambda=n_{tot}/t_{tot}$. In order to determine the confidence interval for the mean of the random variable λ, a sufficient number of independent tests are repeated until the mean falls within the desired interval, which is computed using parameters given by the Student-t distribution.

2) Multiple connections per link:

For the case in which multiple TCP connections share the same link, we consider they have the same link latency τ and segment loss rate q. We also assume that all links between the TCP senders and the common bottleneck buffer have a capacity much higher than the capacity of the bottleneck link and that all connections have the same segment size. In this case, we denote by N the number of TCP connections and by ω the value of the aggregated *cwnd* which is the sum of the *cwnd*s of all connections, while we add the index i to ω, n and n_{tot} to denote the values relevant to each individual connection.

For $\omega\le\mu T$ the window evolution of the connection i is

$$\frac{d\omega^{(i)}(t)}{dt}=\frac{1}{\omega^{(i)}}\frac{da^{(i)}}{dt}=\frac{1}{\omega^{(i)}}\frac{\omega}{bT}\frac{\omega^{(i)}}{\omega}=\frac{1}{bT} \tag{9}$$

In fact, the global ACKs' arrival rate is $\omega/(bT)$ and $\omega^{(i)}/\omega$ is the share that arrives to the sender i. Relation (9) gives rise to

$$\omega^{(i)}(t)=\omega_s^{(i)}+t/(bT), \tag{10}$$

and adding up both sides of (10) for all connections we get

$$\omega(t)=\omega_s+\frac{N}{bT}t \tag{11}$$

Then, similarly to (3) and (4) we obtain $n(t)=\dfrac{\omega_s}{T}\;t+\dfrac{N}{2bT^2}t^2$, and

$$t(n) = bT\left(\sqrt{\omega_s^2 + 2nN/b} - \omega_s\right)/N \quad (12)$$

The number of segments sent by the sender i is

$$n^{(i)}(t) = \frac{1}{T}\int_0^t \omega^{(i)}(z)dz = \omega_s^{(i)}t + t^2/(2bT). \quad (13)$$

Substituting t given by (12) in (13) we get the number of segments sent by the sender i when n segments are totally injected in the bottleneck link.

For $\omega \geq \mu T$ the window evolution of the connection i is

$$\frac{d\omega^{(i)}(t)}{dt} = \frac{1}{\omega^{(i)}}\frac{da^{(i)}}{dt} = \frac{1}{\omega^{(i)}}\frac{\mu}{b}\frac{\omega^{(i)}}{\omega} = \frac{\mu}{b\omega}. \quad (14)$$

Adding up both sides of (14) for all connections we get $d\omega/dt = N\mu/(b\omega)$, which gives:

$$\omega(t) = \sqrt{\omega_s^2 + 2N\mu t/b}, \text{ and } \omega(n) = \sqrt{\omega_s^2 + 2Nn/b}, \quad (15)$$

where t and n are the time and the number of segments totally injected in the bottleneck link from the beginning of the phase, respectively. Substituting $\omega(t)$ given by the first of (15) in (14), after integration we get:

$$\omega^{(i)}(t) = \omega_s^{(i)} + \left(\sqrt{\omega_s^2 + 2N\mu t/b} - \omega_s\right)\Big/N = \omega_s^{(i)} + \Delta\omega(t)/N,$$

$$\text{and } \omega^{(i)}(n) = \omega_s^{(i)} + \left(\sqrt{\omega_s^2 + 2Nn/b} - \omega_s\right)\Big/N = \omega_s^{(i)} + \Delta\omega(n)/N$$

(16)

from which we see that the increment of the aggregated window, denoted by $\Delta\omega$, is evenly spread over all connections. Denoting by Q the number of segments in the buffer, the number of segments sent by each connection can be computed as

$$n^{(i)}(t) = \int_0^t \frac{\omega_{(i)}(z)}{T + Q(z)/\mu}dz = \mu\int_0^t \frac{\omega_{(i)}(z)}{\omega(z)}dz \quad (17)$$

Then, substituting the first of (15) and (16) in (17), after integration and considering (6) we get

$$n^{(i)}(n) = \left(n + \frac{\Delta\omega}{N\mu}(N\omega_s^{(i)} - \omega_s)\right)\Big/N \quad (18)$$

The procedure for computing the goodput is similar to the one used in the single-connection case. At the beginning of each cycle, we draw the number n of successful segments between two successive losses and we check in which case it falls: $n \leq n_{\max}$, or $n > n_{\max}$, where $n_{\max}$ is computed similarly to the single-connection case. In the first case the cycle is prematurely ended by a random loss; in order to discover which connection had the loss we consider that the loss probability of each connection is equal to $\omega^{(i)}/\omega$. If n falls in the second case, the cycle is ended by a congestion loss. Thus, during an entire round-trip-time period, all connections experience a loss in Round Robin fashion because, when the buffer is full, the ACK that triggers the sending of two segments back-to-back causes an overflow, and it occurs for all connections. We compute the time duration of each cycle by using relation (12) for the linear window growing phase and (6) for the second phase, respectively. The cycle duration is incremented by the time necessary to resend one or N lost segments, according to the event of random or congestion loss, respectively. All cycle durations and the total number of segments sent in relevant cycles are accumulated into the variables t_{tot} and n_{tot}, respectively; then the average aggregated goodput λ is computed as $\lambda = n_{tot}/t_{tot}$. The number of segments sent by each connection is accumulated in the variables $n_{tot}^{(i)}$ and the average individual goodputs $\lambda^{(i)}$ are then computed as $\lambda^{(i)} = n_{tot}^{(i)}/t_{tot}$.

C. TGEP validation

In [14] the *Mathematica*™ code of the procedure shown above, called TCP goodput evaluation program (TGEP), is available. TGEP allows the estimation of the goodput within a chosen confidence interval at a chosen confidence level. The limit of TGEP is that it does not consider TCP timeouts and it does not model multiple losses within any interval T, but in cases in which $q \ll 1$, which are significant cases when the delay-bandwidth product is high, TGEP performs satisfactorily.

Unless otherwise specified all results of TGEP in the rest of the paper have been obtained with a confidence interval of –1% at 99% level. TGEP can be easily modified to introduce different packet loss distributions. In order to obviate to the non-consideration of TCP timeouts, we can assume the following expression of the throughput [15], valid for unlimited bandwidth link

$$T_g = \sqrt{\frac{3}{2}}\frac{1}{\mu\left(\sqrt{bq}T + T_o\min\left[1, \sqrt{3bq/8}\right]q(1+32q^2)\right)}. \quad (19)$$

In relation (19) T_o is the value of the timeout, which can be assumed as max{1, 4 RTT} [16], where RTT is the minimum round trip time experienced by the connection. In Table I we report the comparison of TGEP, relation (19), and ns2 simulation outputs, for a single connection; the value of T_o has been set to 1s, because this value is computed by the ns2 version we used. In Table II we report the comparison of TGEP, relation (19) and ns2 simulation for 5 connections. We can observe a good agreement between ns2 and relation (19) for low values, and between ns2 and TGEP for high values of the goodput; this follows our expectations, because relation (19) considers TCP timeouts and unlimited bandwidth, while TGEP considers the bottleneck rate of the link and does not keep timeouts into account. A suitable threshold to use when selecting one or the other procedure can be assumed to be equal to 50%, even if TGEP agrees with relation (19) (and with ns2 simulation) also for much lower values of the goodput.

D. Some considerations concerning the TCP goodput

By using TGEP for different values of the random loss rate q, β=0.8 and τ=0.5, we have computed the average goodput λ and divided it by the value of λ_c given by (7); then we have reported in Fig. 2 the ratio between these goodput values as a function of the ratio between q and q_c given by (8). The result shows that what is widely supported in discussions about TCP performance on satellite links (see e.g. [17]), which entails making the packet loss rate due to data corruption negligible with respect to loss due to congestion, should be reviewed. In fact, even if the goodput always increases by lowering the random loss, it is sufficient to make an average random loss equal to one-third of the congestion loss to get 99% of the goodput due to the loss for congestion only. Considering that lowering the random loss, which means lowering the bit error rate (BER) over the link, is always expensive in terms of energy and/or information rate, it is easy to infer that the calibration of the BER, i.e. the choice of the FEC type/rate to maximize the TCP goodput, admits an optimum value of the BER for each channel quality condition [9].

As previously outlined in [8], and in agreement with relation (19), for high bandwidth-delay product links, the TCP throughput degradation strongly depends on the parameter $\tau^2\mu^2 q$ and it is almost independent of each single parameter if $\mu\tau >> 1$.

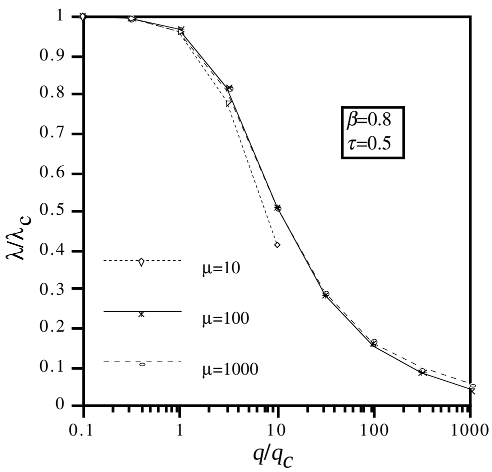


Fig. 2. λ / λ_c ratio versus q/q$_c$ ratio, for different values of the bottleneck rate μ, β=0.8 and τ=0.5.

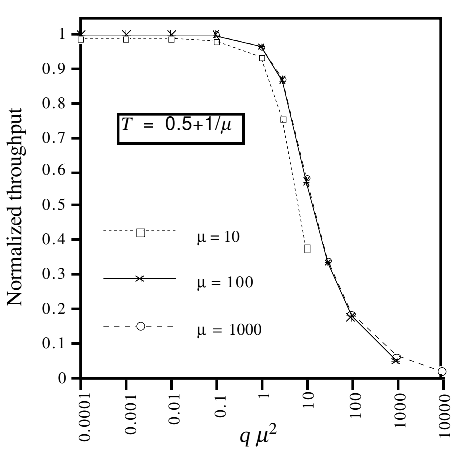


Fig. 3. TCP normalized goodput (λ / μ) versus the parameter q / μ^2, for different values of the bottleneck rate μ, β=0.8 and τ=0.5s.

In Figure 3 we report the normalized goodput versus the parameter $q\mu^2$ for τ=0.5s, β=0.8, and for different values of the bottleneck rate. Note that the degradation becomes relevant only for values of $q\mu^2$ which are considerably greater than unity, and that the dependence on μ is practically null, for high values of μ (e.g. in the order of 100). Plots reported in Fig. 3 are useful for quickly estimating the goodput degradation in a very wide set of cases of TCP over geostationary satellite links. We point out that, for high values of μ, q values that are inversely proportional to μ^2 produce the same value of the goodput; that is, higher bottleneck rates require lower random loss rates to get the same value of the goodput. Lower values of μ (e.g. 10 in the figure) induce some difference in the goodput, mainly due to the variation that μ induces in the parameter T(indeed $T = \tau + 1/\mu$). In order to reduce this difference it is possible to find a better parameter that depends on the value of τ. In the case of τ=0.5s we found the parameter p that makes the goodput insensitive by individual parameters q, μ and T is $p = q\mu^2 T^6$, even for values of μ as low as 10.

An important issue to consider is choosing the optimum packet length, that is the value of *MSS* that maximizes 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 4 it is apparent that long packets should be used in order to maximize 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*.

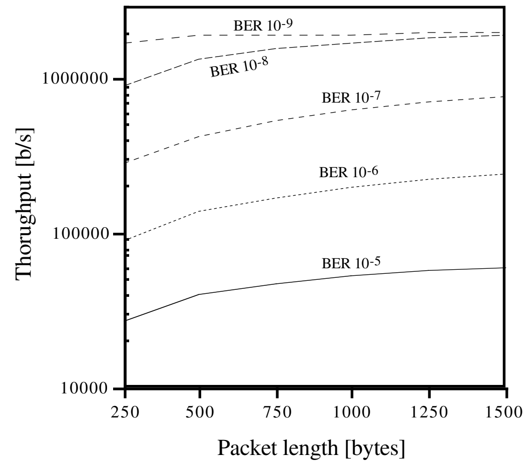


Fig. 4. TCP throughput versus segment size for different bit error rates.

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.

IV. PERFORMANCE OF VARIOUS FEC TYPES

In this Section we show the performance of several of the most widely used FEC techniques, in terms of segment error rate (SER) as a function of the signal-to-noise ratio available at a satellite earth station receiver. We assume operating over an AWGN channel and assume a negligible inter symbol interference. This hypothesis is widely accepted for non-mobile users of geostationary satellite links.

A. Convolutional Coding

Let us begin by showing the characteristics of a convolutional coder/Viterbi decoder over BPSK/QPSK modulated symbols (Fig. 5). The BER data are taken from the Qualcomm Viterbi decoder data sheet [18] (standard NASA 1/2 rate with constraint length 7 and derived punctured codes), while the average error burst length (*ebl*) was obtained through software simulation by using the Phil Karn streaming codes [19]. Since we needed to evaluate the BER and error burst characteristics for BER values even less than 10^{-9}, we resorted to extrapolation for some points, which are marked with a circle on the Figure. The complete set of data is plotted versus E_c/N_0, that is, the ratio of channel bit energy to the one-sided noise spectral density, which is equal to $C(\text{carrier power})/N_0$, divided by the modem channel bit rate.

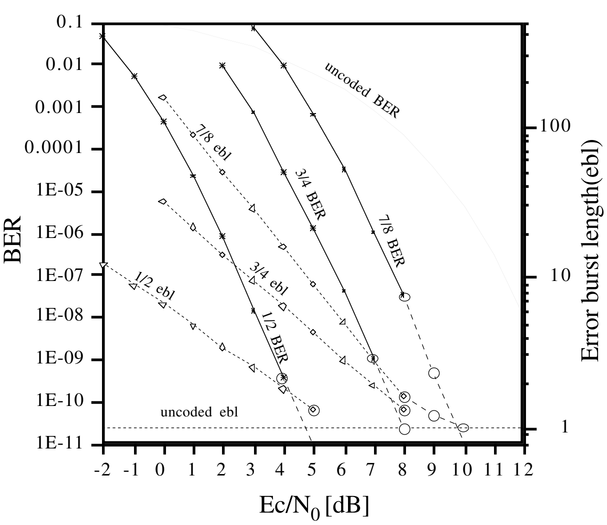


Fig. 5. BER and average error burst length versus E_c/N_0, for different values of the convolutional coding rate. Extrapolated values are circled.

B. *Reed Solomon Coding*

The Reed Solomon (RS) codes [20-23] are linear block codes with an alphabet size of $Q=2^m$, m being the number of bits in a symbol. An RS code is specified as RS(n, k), which means that k m-bit symbols of information data are added to $r=n-k$ parity symbols to form an n-symbol codeword. The block (codeword) maximum length is $n=Q-1$ symbols [21], but it can be extended to Q and $Q+1$ [20]. If an RS code has r redundant symbols, the code is able to correct any pattern of t symbol errors and s symbol erasures for which

$$2t + s < d = r + 1. \quad (20)$$

An erasure occurs when the position of an erred symbol is known, and d is the minimum Hamming distance between codewords. Thus, this code is able to correct up to $t_0 = r/2$ errors when there are no erasures and up to $s_0 = r$ erasures when there are no errors. The rate of an RS code is k/n; anyway, codes can be shortened by assuming sending a number z of zeroes (that are not obviously transmitted) and reiserting them at the decoder. This allows reducing the coding rate at will, which becomes $(k-z)/(n-z)$. Large values of r dramatically decrease the block error rate, but require a very high computational power. These codes are particularly advantageous when used in the presence of bursts of errors as occurs on the output of a Viterbi decoder (see Fig. 5), because an error sequence of length *ebl* can corrupt $\lfloor ebl/m \rfloor$ or $\lfloor ebl/m \rfloor+1$ symbols at maximum. This makes the concatenation of RS codes (*outer codes*) with convolutional codes (*inner codes*) a very powerful FEC technique. The performance of such concatenated codes is enhanced by inserting a symbol interleaver/deinterleaver between the two coders, which reduces the correlation of the erred symbols.

In this subsection we refer to unknown positions of erred symbols (RS codes), while we will analyze in the next subsection the case of known positions of erred symbols (erasure codes).

Denoting by p_s the symbol error probability, the block error probability p_b, in case of independent erred symbols (infinite interleaver), can be easily computed as

$$p_b = 1 - \sum_{j=0}^{t_0} \binom{n}{j} p_s{}^j (1-p_s)^{n-j}. \quad (21)$$

When erred bits can be considered as independent, e.g. in the case of an AWGN channel, the symbol error probability can be computed as $p_s = 1-(1-p)^m$, p being the bit error probability (BER). In case of error train, such as at the output of a Viterbi decoder, simulation is required to estimate p_b. In the past, RS coding was always implemented in hardware; however, the computing power of today's machines allows reasonable benchmark figures for software implementations as well [21].

One of the most popular codes is RS(255, 223) with m=8 and t_0=16. Choosing a segment length of 1115 bytes, which contains 5 223-byte blocks, we evaluated the segment error rate (Fig. 6), for different values of E_c/N_0, and E_b/N_0, by using RS alone, convolutional encoder/Viterbi decoder with different coding rates, and concatenated RS/convolutional codes. The last ones are used both with and without an interleaver which, if present, is assumed as ideal. An ideal interleaver allows us to consider independent erred symbols and relation (21) can be used. In this case an approximate evaluation of p_s, which is good for low values of p, is given by:

$$p_s = p(\overline{ebl} + m - 1)/\overline{ebl}, \quad (22)$$

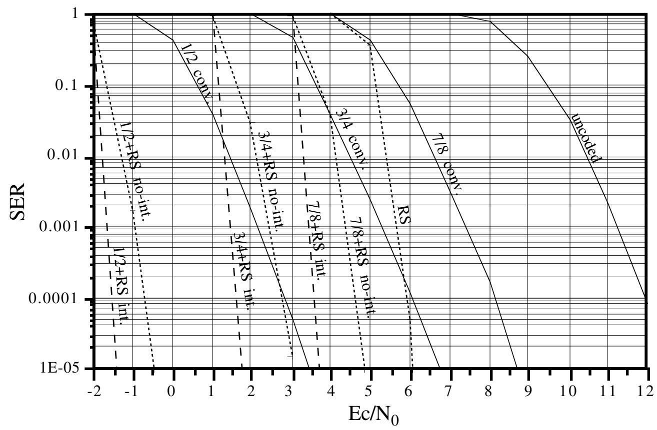
where $\overline{ebl}$ is the average error burst length. The SER is then given by $\text{SER} = 1-(1-p_b)^{n_b}$, n_b being the number of blocks in a segment. For the simulation of the case without interleaver we assumed (see [20] and reference therein) that the error burst length has a geometric distribution

$$\Pr\{ebl = l\} = p_l(1-p_l)^{l-1};\ \ l=1,\ 2,...,\ \ p_l = 1/\overline{ebl}, \qquad (23)$$

and the distance between error bursts has the empirical distribution,

$$\Pr\{D = v\} = p_d(1-p_d)^{v-K-1},\ \ v = K+1,\ K+2,..., \qquad (24)$$

where $p_d = 1/(\overline{D} - K + 2)$, $\overline{D}$ is the average distance between error bursts, which can be assumed as $\overline{D} = p/\overline{ebl}$, and K is the convolutional code constraint length. To compute the SER, when the segment is convolutionally encoded, we can use the output from relation (22), with *m* equal to the segment bit length.



a)

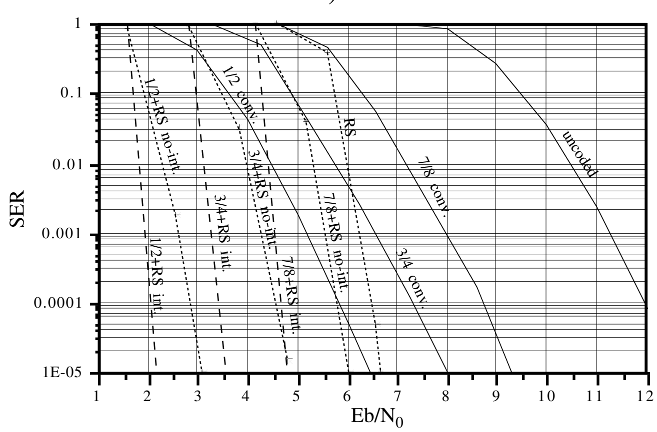


Fig. 6. Segment error rate $/N_0$, (a) and versus E_b/N_0, (b) for different convolutional coding rates and for concatenated RS/convolutional codes, with and without symbol interleaver.

The evaluation of the code that performs better, given the SER value we want achieve, is straightforward if the comparison is made in energetic terms, as usual. Relations that link bit rates and signal to noise ratios are

$$R_i = C_n - E_b; \qquad\qquad R_c = C_n - E_c, \qquad (25)$$

where we denoted by R_i and R_c the information and channel bit rates, expressed in dB, respectively, by E_b and E_c the values of E_b/N_0 and E_c/N_0, respectively, and by C_n the value of C/N_0. From the first of (25) we see that, for a certain SER value, a code that needs a lower E_b/N_0 allows a higher information bit rate with the same C/N_0, if we assume having an ideal modem that is able to vary the bit rate with continuity. From the spectrum efficiency point of view, from the second of (25), a code that needs a higher E_c/N_0 allows a lower channel bit rate with the same C/N_0 and it then occupies a narrower bandwidth. However, the comparison makes sense if the same information bit rate is obtained. Thus, introducing subscripts 1 and 2 for two different codes, we have: $R_{c1} = C_{n1} - E_{c1}$ and $R_{c2} = C_{n2} - E_{c2}$, then:
$R_{c1} < R_{c2}$ *if* $C_{n1} - E_{c1} < C_{n2} - E_{c2}$.
To get the same R_i , $C_{n1} - E_{b1}$ must be equal to $C_{n2} - E_{b2}$, then code$_1$ requires a lower channel bit rate, i.e. a lower channel occupancy, with respect to code$_2$ if $E_{b1} - E_{b2} < E_{c1} - E_{c2}$. To give an example, code RS has a spectrum occupancy that is lower than the 3/4 convolutional one, for SER<0.001.

C. Erasure codes

The term erasure code refers to a block FEC that is able to recover from a corruption or loss of some blocks of information used for sending a data unit. An original piece of information (e.g. a TCP segment) is divided into *k* blocks and added to *r*=*n*-*k* redundant blocks; the resulting *n* blocks are then individually sent in *n* frames. An erasure code that is easy to understand, flexible and efficient to implement, even on a common PC, is presented in [5], together with its implementation performance. In [2] an analysis of TCP throughput has been performed, by using FEC erasures over a lossy channel that is assumed to follow the Gilbert model [3]. Here an AWGN channel is assumed with or without convolutional coding.

In the case of independent block losses the SER can be computed as the output of relation (21) with p_s equal to the block loss rate (*blr*) and t_0=*r*. In order to be able to do this, we must know the position of the lost (or corrupted) blocks; thus, a minimum protocol is needed in this case. For each block we have assumed a cyclic redundancy check (CRC) field of 16 bits, a field to indicate the block sequence number inside the packet, plus the last block information and some spare bits for a total overhead of 32 bits.

We computed the performance of the erasure code in case of independent bit errors (no other FEC used) for 23, 46 and 92 byte block sizes and 48, 24 and 12 blocks per packet, respectively. Such metrics are chosen to send a segment of 1104 bytes, which is very close to the size used in RS coding, in order to allow a performance comparison.

In Figure 7 E_b/N_0 and E_c/N_0 versus redundancy are reported for the three block sizes chosen and for a SER value of 10^{-4}. The same picture also shows both the channel and information bit rates, which would be possible to achieve with the same C/N_0, normalized with respect to the bit rate of the uncoded channel. For example, a segment sent with 48 23-byte blocks, with 16 redundant blocks, by using the same power as when sending an uncoded 1104-byte segment, can be sent at a channel bit rate of 3 times, and getting an information bit rate of 1.9 times, with respect to the uncoded segment, in order to achieve the same SER of 10^{-4}. We can observe that the minimum E_b/N_0 is achieved in all cases with a redundancy of 1/3 and the 23-byte block case exhibits the best performance. The same test for different SER values gives similar results.

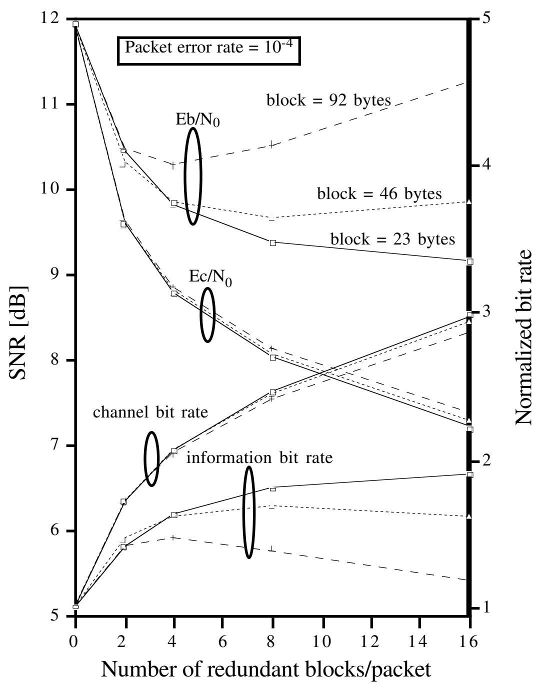


Fig. 7. E_b/N_0, E_c/N_0 and achievable bit rates versus redundancy, for SER=10^{-4} and different block sizes.

V. TCP GOODPUT WITH DIFFERENT FECS AND MODULATION SCHEMES

Let us now return to our original problem, which was to find, for each channel condition $\boldsymbol{c}$, the point on the relative graph where the TCP goodput is maximum. We define this point as the optimum operating point $\boldsymbol{O}(\boldsymbol{c})$ for a given $\boldsymbol{c}$. Since the graph relative to $\boldsymbol{c}$ depends 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 1 as a reference, the optimum operating point will be chosen from the circled points on the graph. The choice is made by using TGEP or relation (19) output illustrated in Section III. It is worth noticing that the implementation complexity of the procedure to evaluate the TCP goodput does not sensibly affect the applicability of the method we are proposing in the practice. In fact, during the tuning phase of some given equipment, any analytical methods as well as simulation or measurement-based methods could be used. On the other hand, during normal operation, the implementation of the procedure proposed in [9] only employs the pre-computed lookup tables already mentioned. Since each $\boldsymbol{O}(\boldsymbol{c})$ corresponds to certain transmission parameters $\boldsymbol{p}(\boldsymbol{O}(\boldsymbol{c}))$, we have a mapping of the optimal transmission parameters on all the considered channel conditions $\boldsymbol{C}$, which is our objective. In the case of dynamic bandwidth allocation, one such mapping should exist for each possible per-user allocation.

A. Single TCP connection per link

We now estimate the throughput of a single TCP connection, for different conditions of channel quality. We assume having a modem, working at 2 Msymbols/s, which is able to switch between BPSK and QPSK modulation schemes. Thus the resulting channel bit rate is 2 and 4Mb/s, respectively. This choice was made both in order to represent a commonly used piece of equipment and because the spectrum occupancy is the same in both cases. We also assume that the channel bandwidth is shared in TDMA (time division multiple access) mode among the users and that the considered TCP connection received an assignment of one-half of the carrier capacity. The C/N_0 ratio available at the demodulator input spans between 66 and 78 dB. C/N_0 depends on the signal attenuation, e.g. due to atmospheric conditions (rain fading). The bit rate of uncoded data is thus 2 or 1Mb/s when the modem works in QPSK or BPSK, respectively. This situation was chosen in order to facilitate the comparison with the case of multiple connections. The decision concerning the selection of the transmission parameters is made at the earth station on the basis of the link quality, which is supposed to be known.

The modulation scheme selection and convolutional coding/decoding is performed at the physical level; RS coding/decoding is preferably performed at the physical level, but it could be made by software at link level, while erasure coding/decoding is performed at link level [5]. The optimal transmission parameters are supposed to be stored in look-up tables that are accessed for instantaneous values of both the C/N_0 ratio and the capacity that is assigned to the link. When the satellite channel capacity is shared in FDMA (frequency division multiple access) mode among the users, to obtain the same situations as in TDMA we must consider a modulation rate of 1Msymbols/s. When a carrier of 2Msymbols/s is entirely assigned to the TCP connection, the results are still valid if we decrement by 3 dB the range of C/N_0 ratio (63-75).

Figure 8 (a-f) reports the results relative to the various coding types employed. All points lying on the envelopement curves represent the maximum goodput obtainable with the available hardware for a given channel quality condition. In [9] it is shown that the adoption of a different criterion, such as the commonly used one which imposes the BER below a certain threshold, produces results that may sensibly differ from optimality, even if the threshold is made variable with the link capacity. Our point-by-point optimization is more complex to estimate; however, the calculus is made only once and it does not add any overhead at working time. All combinations of coding rates-modulation schemes are shown for the convolutional case only, while in all other cases only the combinations that contribute to the envelopment are reported, for the sake of the clarity of the pictures. In the RS case we used shortened codes which derive from the RS(255,223). The erasure code relative to 23-byte blocks performs better than the 46-byte one if the code is used alone. When the code is concatenated with the convolutional one, the 46-byte code performs slightly better than the 23-byte one if no interleaving is used. We chose to represent both concatenated codes without interleaving; the reason for that is justified by the following considerations. As shown in Fig. 6, the interleaver allows a gain of about one dB in a wide range of SER values. As far as the TCP goodput is concerned, the interleaver introduces an additional latency equal to d/μ, where d is the segment depth of the interleaver. The gain due to interleaving is thus reduced in most cases, and we must take into account the interleaving complexity as well.

In order to make comparisons easier, in Figure 9 we reported the envelopments of all cases shown in Figure 8 plus an additional one: the RS(239,223) concatenated with convolutional codes. We note that, as expected, the best performance is achieved by using concatenated codes, in particular by using RS plus convolutional ones. RS code outperforms the convolutional one only at high C/N_0 values, while its performance is worse at low C/N_0 values. Anyway we must underline that the convolutional/Viterbi technique is more flexible and is suitable for any segment length. Instead, block codes need a segment length equal to a multiple of the block length, in order to achieve a high packetization efficiency. Erasure codes exhibit the worst performance; their most significant peculiarity is that a number of redundant blocks in a packet can not only be corrupted but even be lost, to allow a successful segment delivery, and an AWGN channel does not take sufficient advantage of this.

B. *Multiple TCP connections per link*

By using the same point-by-point optimization method, we made a comparison between the goodput achieved by a single connection with the aggregated goodput of 8 TCP connections each of which has its own buffer and a bandwidth share equal to 1/8 of the entire bandwidth. The comparison of the envelopment curves, which express the maximum values of the goodput versus C/N_0 for different coding types, is reported in Fig. 10. The whole channel bandwidth is 4 or 2Mb/s for QPSK or BPSK modulation schemes, respectively. When the bandwidth is divided among 8 connections and is assigned individually to the requesting users, the transmission parameter optimization is made for TCP bottleneck rates of 512 (256) kb/s in the QPSK (BPSK) case. In the other case, the optimization is made for the full rate and a single TCP connection, because all the bandwidth is assigned to a single user. Figure 10 shows that the total goodput sensibly differs in the two cases, for many of the C/N_0 values considered and for all coding types. The difference is null when the selection of the transmission parameters only allows excessively low values of the SER.

When multiple TCP connections share the same link, it was empirically observed in [8] by making use of simulation, that the throughput is inversely proportional to τ_i^a, where τ_i is the latency of the connection i, and 1†a†2. As the latency introduced by a geostationary satellite is quite high (half a second) it is reasonable to assume that the additional latency introduced by terrestrial hops in the entire link path is negligible with respect to the satellite one. Thus, assuming that all connections have practically the same latency, they obtain an equal share of the link bandwidth. This is strongly supported by our simulations as well (see Table II). We simulated the case in which 8 TCP connections share the same buffer and the same bottleneck link. The results obtained for the aggregated goodput of all connections do not sensibly differ from the one obtained by considering 8 separated links and buffers with 8 connections (reported in Fig 10). Furthermore, the difference of the two results decreases as the size of the buffer increases. This is a useful issue because it allows to reduce the number of the look-up tables necessary to take the optimum transmission parameters. In fact, provided that the buffer employed is sufficiently large, different tables for different number of TCP connections are not required. It is sufficient to divide the allocated bandwidth by the number of connections and use the resulting value to enter the table that gives the optimum transmission parameters as a function of the bandwidth in the current channel condition (C/N_0).

VI. CONCLUSIONS

A rapid simulation tool (TGEP) was developed for evaluating the TCP Reno goodput in case of random and independent segment losses. The program is easily modifiable for different segment loss distributions

We then depicted a scenario that shows the maximum achievable TCP goodput in a realistic situation in which a bandwidth portion of a geostationary satellite transponder is assigned to a user (e.g., in TDMA mode), for single or multiple TCP connections. Several coding techniques, used for two different modulation schemes (B/QPSK), have been applied to TCP connections, in a wide range of the signal-to-noise ratios of the channel that is supposed to be corrupted by AWGN. The case studied closely emulates a situation in which the signal level, which is present at the receiver input, is affected by a variable attenuation due to different atmospheric conditions. The transmission modality of the data packets is chosen in the earth station, according to the instantaneous signal-to-noise ratio and to the assigned value of the bandwidth, by making use of look-up tables whose parameters are pre-computed.

The comparison of the different coding techniques shows the supremacy of the convolutional/Viterbi case, especially when concatenated with RS codes, over the erasure coding. This is the result in an AWGN channel. However, we think that the strong peculiarity of the erasure codes, which allow the recovery of a packet even with some lost blocks, can be better exploited when different types of impairment affect the channel. The evolution of the present work will be an analysis of the coding techniques used in this paper in the presence of long bursts of errors (e.g. using the Gilbert [3] model).

REFERENCES

[1] 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.
[2] C. Barakat and E. Altman, Bandwidth tradeoff between TCP and link-level FEC , *Computer Networks*, vol. 39, no. 2, June 2002, pp. 133-150.
[3] E. N. Gilbert, "Capacity of a Burst noise Channel" *Bell Systems Technical Journal*, vol. 39, (1960), pp. 1253-1266.
[4] 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.
[5] L. Rizzo, Effective erasure codes for reliable computer communication protocols , *Computer Communication Review*, vol. 27, no. 2, April 1997, pp. 24-36.
[6] 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.
[7] 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.
[8] 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.
[9] N. Celandroni, F. Potorti , Maximizing single connection TCP goodput by trading bandwidth for BER , *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 63-79, 2003.
[10] M. Allman, V. Paxson and W. S. Stevens, TCP congestion control , *RFC 2581*, April 1999.
[11] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, V. Jacobson and R. Braden, TCP selective acknowledgement options , *IETF, RFC 2018*, October 1996.
[12] V. Jacobson, R. Braden and D. Borman, TCP extensions for high performance , *IETF, RFC 1323*, May 1992.
[13] 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.
[14] http://wnet.isti.cnr.it/software/tgep.html
[15] J. Padhye, V. Firoiu, D.F. Towsley and J.F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation", *IEEE/ACM Trans. Networking*, vol. 8, pp. 133-145, 2000.
[16] S. Shenker and L. Zhang, "Some Observations on the Dynamics of a Congestion Control Algorithm", *ACM SIGCOMM-CCR*, vol. 20, no. 5, October 1990, pp. 30-39.
[17] C. Partridge and T. J. Shepard, TCP/IP performance over satellite links , *IEEE Network*, vol. 11, no. 5, September/October 1997, pp. 44-49.
[18] Q1401: K=7 rate 1/2 single-chip Viterbi decoder technical data sheet , *Qualcomm incorporated*, September 1987.
[19] P. Karn, Error correcting codes , <URL:http://people.qualcomm.com/karn/code/fec/>.
[20] M. K. Simon, J.K. Omura, R. A. Scholtz and B. K. Levitt, "Spread Spectrum Communications Handbook", revised edition: McGraw-Hill, 1994.
[21] M. Riley and I. Richardson, "Reed_Solomon Codes", online at http://www.4i2i.com/reed_solomon_codes.htm, November 2002.
[22] R. Blahut, "Theory and Practice of Error Control Codes", Reading, MA: Addison-Wesley, 1983.
[23] S. Lin and D.J. Costello, "Error Control Coding: Fundamental and Applications": Prentice-Hall, 1983.

TABLE I
GOODPUT OF A SINGLE CONNECTION - NS2, RELATION (19) AND TGEP COMPARISON

TCP normalized goodput (β=0.8, τ=0.5s, μ=100 segments/s, delayed ACKs, no SACK,

ns-2 simulations and TGEP output confidence intervals = 1% at 99% level)

q (*loss rate*)	ns-2 simulation TCP Reno	ns-2 simulation TCP New Reno	Relation (19) T_o =1 s	TGEP
0.1	0.0344	0.0364	0.0364	0.0578
0.01	0.158	0.165	0.164	0.162
0.001	0.546	0.554	0.535	0.547
0.0001	0.955	0.955	1.697	0.957
0.00001	0.988	0.989	-	0.991

TABLE II
GOODPUT OF 5 CONNECTIONS- NS2 AND TGEP COMPARISON

TCP Reno (no delayed ACKs) goodput of 5 connections sharing a link with a bottleneck rate μ=455 segments/s, τ =0.5 *s*, β=0.8. Confidence intervals are at 99% level.

ns2 simulations

q	Connect. #1	Connect. #2	Connect. #3	Connect. #4	Connect. #5	Total	Relation (19) T_o=2s
10^{-4}	0.198–3.1%	0.199–2.5%	0.199–0.5%	0.196–2.2%	0.202–2.3%	0.994–0.03%	2.656
10^{-3}	0.165–2.4%	0.169–2.6%	0.164–4.3%	0.169–5.3%	0.169–1.3%	0.836–0.67%	0.8345
10^{-2}	0.0469–0.64%	0.0471–2.3%	0.470–1.5%	0.0466–3.1%	0.0468–1.6%	0.234–0.4%	0.2478
10^{-1}	0.0090–3%	0.0090–0.8%	0.0090–0.7%	0.0090–0.6%	0.0092–2%	0.0452–4%	0.0429

TGEP output

q	Connect. #1	Connect. #2	Connect. #3	Connect. #4	Connect. #5	Total	Relation (19) T_o=1s
10^{-4}	0.199–3.5%	0.198–0.6%	0.199–5.8%	0.199–3.7%	0.199–1.89%	0.994–0.01%	2.657
10^{-3}	0.169–5.1%	0.168–7.7%	0.166–8.2%	0.171–7.5%	0.171–7.2%	0.845–1%	0.8375
10^{-2}	0.0498–5.1%	0.0509–13.8%	0.0515–8.4%	0.0495–6.7%	0.0494–5.8%	0.248–1%	0.2565
10^{-1}	0.0156–1.3%	0.0157–1.7%	0.0156–1.6%	0.0155–1.5%	0.0157–1.9%	0.0781–1%	0.0568

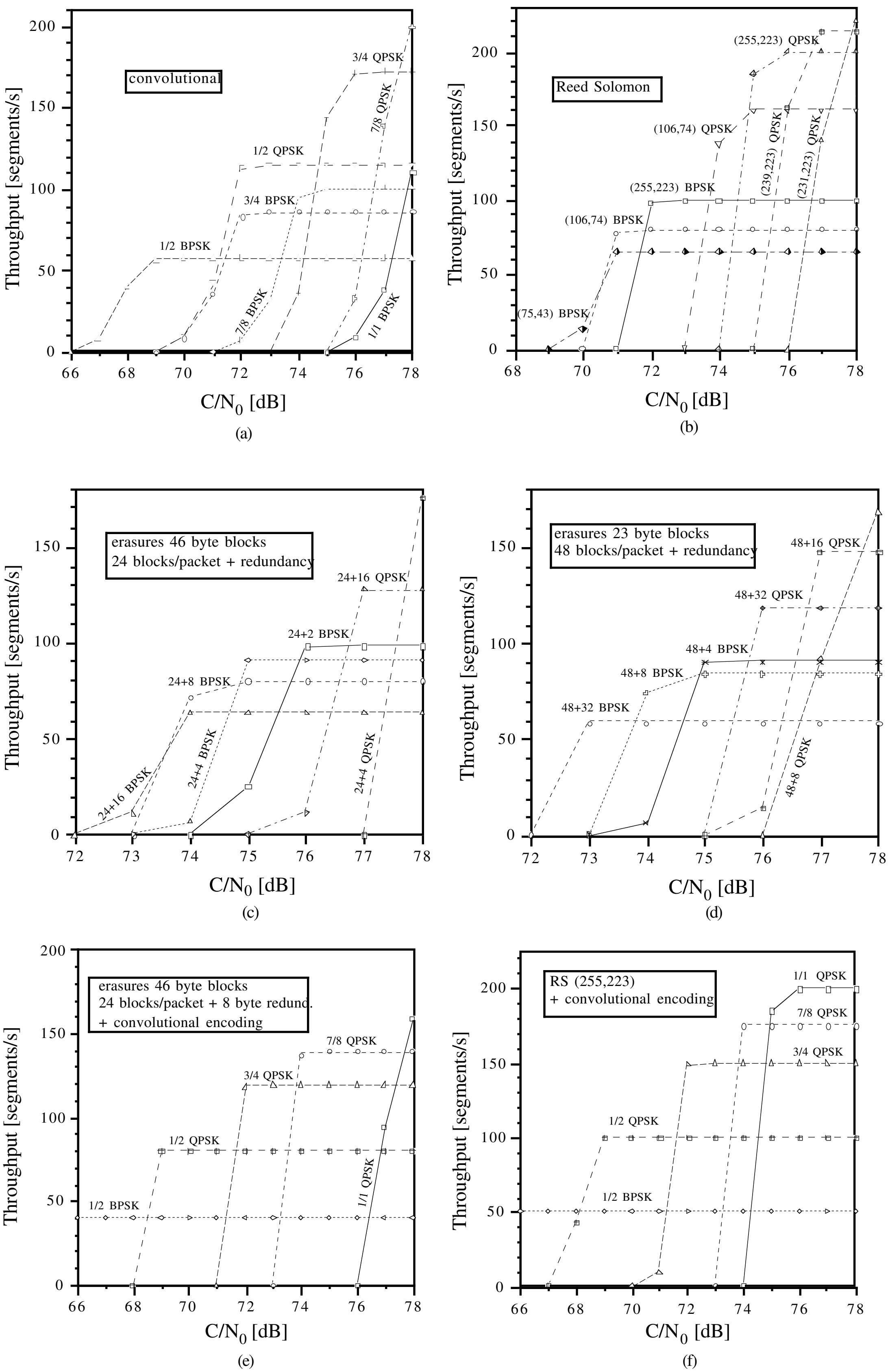


Fig. 8. TCP goodput versus C/N_0 for various coding types.

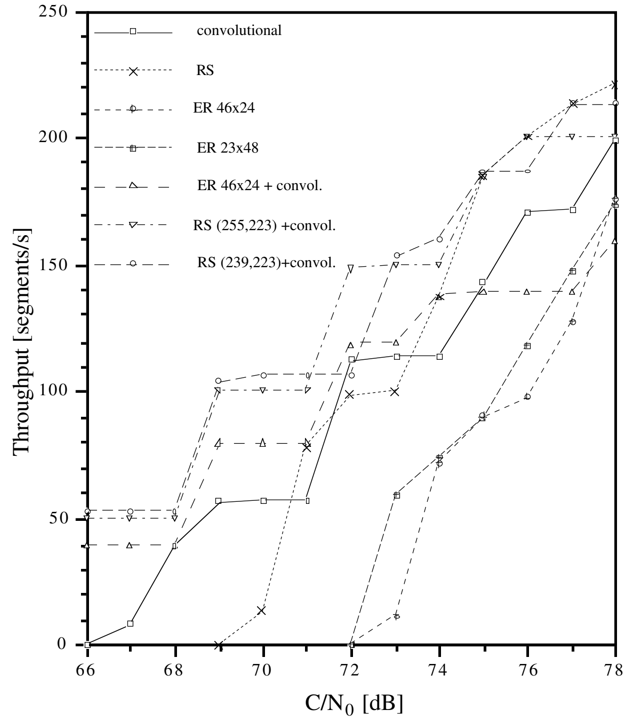


Fig. 9. Maximum goodput versus C/N_0, for different coding types.

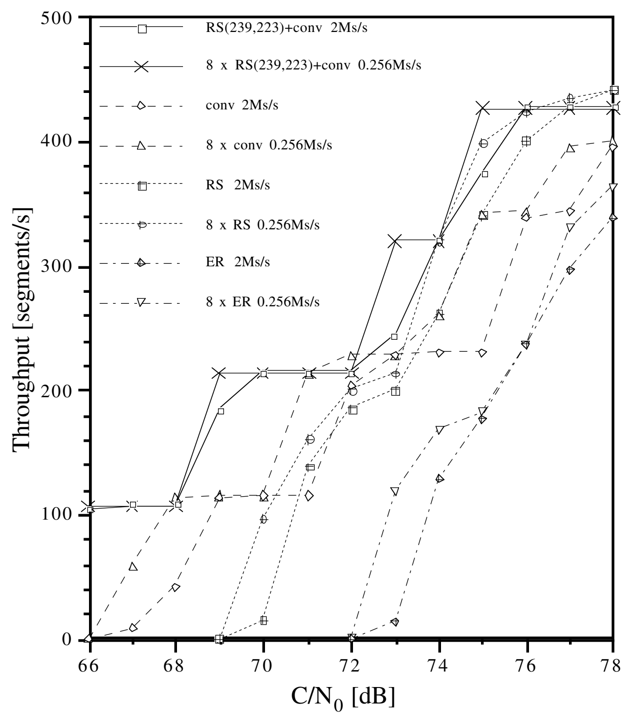


Fig. 10. Comparison between the envelopments of the goodput of one TCP connection and the aggregated goodput of 8 TCP connections that share the same amount of bandwidth, versus C/N_0, for different coding types. The rate of 0.256Msymbols/s is relative to the average allocation to each one of the 8 TCP connections. The RS cases refer to codes as in Fig. 8 b, while the ER (erasure code) cases refer to codes as in Fig 8 c.