INTERNATIONAL JOURNAL OF SATELLITE COMMUNICATIONS
Int. J. Satell. Commun. 2001; **19**:429–442 (DOI: 10.1002/sat.713)

TCP-Peach for satellite networks: analytical model and performance evaluation

Ian F. Akyildiz[1,*,†], Giacomo Morabito[1] and Sergio Palazzo[2]

[1]*Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, U.S.A.*
[2]*Istituto di Informatica e Telecommunicazioni, University of Catania, V.le A. Doria, 6, Catania, 95125, Italy*

SUMMARY

Current TCP protocols have low throughput performance in satellite networks mainly due to the effects of long propagation delays and high link error rates. TCP-Peach is a new congestion control scheme for satellite IP networks based on the use of low priority segments, called dummy segments. The sender transmits dummy segments to probe the availability of network resources. Dummy segments are treated as low priority segments thus, they do not effect the throughput of actual data segments. In this paper, TCP-Peach is presented along with its analytical model which is used to evaluate the throughput performance. Experiments show that TCP-Peach is robust to high link error rates as well as long propagation delays, and outperforms other TCP schemes for satellite networks. Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: TCP protocols; satellite networks; analytical model

1. INTRODUCTION

Current TCP protocols have low throughout performance in networks with long propagation delays and relatively high link error rates such as satellite networks [1–5]. In fact,

- The long propagation delays cause longer duration of the *Slow Start* phase during which the TCP sender may not use the available bandwidth.
- The TCP protocol was initially designed to work in networks with low link error rates, i.e. most of segment losses were due to network congestions. As a result, the TCP sender decreases its transmission rate. However, this causes unnecessary throughput degradation if segment losses occur due to link errors.

The IETF is currently developing a new standard identifying which TCP options should be used in future satellite networks [6]. However, to our knowledge, no concrete solutions are given for the above problems [2].

*Correspondence to: Ian F. Akyildiz, Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, U.S.A.
†E-mail: ian@ece.gatech.edu

Contract/grant sponsor: NASA-Ames; contract/grant number: #NAG 2-1262

In Reference [7], we propose TCP-Peach for satellite networks. The main objective of TCP-Peach is to improve the throughput performance in satellite IP networks. The new scheme is based on the use of *dummy segments*, which are sent by the sender to probe the availability of network resources in the connection path. The dummy segments are treated as low priority segments therefore, they do not effect the throughput of actual data traffic. This requires all the routers in the connection path to support some priority mechanism.

In this paper we present the new scheme and evaluate its throughput performance analytically and through simulation experiments.

The paper is organized as follows. In Section 2, we describe TCP-Peach [7] and introduce its analytical model in Section 3. In Section 4, we evaluate the performance of the new scheme through analysis as well as simulation. Finally, in Section 5, we conclude the paper.

2. THE NEW CONGESTION CONTROL SCHEME

The new scheme is composed by the Sudden Start, the Congestion Avoidance, the Fast Retransmit and the Rapid Recovery algorithms as shown in Figure 1. The Sudden Start and Rapid Recovery are the new algorithms, whereas the Fast Retransmit is the same as in Reference [8] and the Congestion Avoidance differs from Reference [8] because it needs to deal with the ACKs for dummy segments. In Sections 2.1–2.3, we present the basics of TCP-Peach. Further details can be found in Reference [7].

2.1. Dummy segments

Dummy segments are low priority segments used by the senders to probe the availability of network resources. If a router on the connection path is congested, then it discards the IP packets

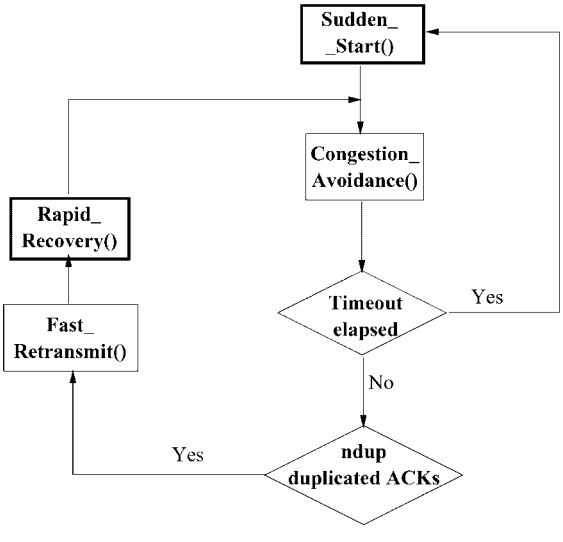

Figure 1. The TCP Peach scheme.

carrying dummy segments first. Consequently, the transmission of dummy segments does not cause a decrease of throughput of actual *data segments*, i.e. the traditional segments. If the routers are not congested, then the dummy segments can reach the receiver which sends ACKs back. The sender interprets the ACKs for dummy segments as the evidence that there are unused resources in the network and accordingly, can increase its transmission rate. Dummy segments do not carry any new information to the receiver. They are generated by the sender as a copy of the last transmitted data segment. Note that dummy segments may produce some overhead, but we outline that they use resources which otherwise would be unutilized.

TCP-Peach requires that all routers in the connection path support some priority discipline. In fact, it injects dummy segments into the network regardless of the current traffic load. As a consequence, dummy segments may congest routers and effect data segment throughput if a router on the connection path does not apply any priority policy. Note that in traditional IP [9] networks the IP *type of service* (TOS) can be used for this purpose. In fact, one of the eight bits of the TOS field in the IP header gives the priority level of the IP packet [9]. Instead, more recent IP versions, e.g. IPv6 [10], explicitely provide several priority levels.

Currently, some routers in the Internet do not apply any priority policy. However, in the near future, Internet will support quality of service through the *Differentiated Service Model* (DiffServ) [11], which requires all routers to support multiple service classes. As a consequence, all recent commercial routers, e.g. Cisco series 7000 and 12 000 [12], support at least the IP TOS.

In the following sections we will show that the ACKs for the dummy segments transmitted during the Sudden Start and Rapid Recovery are received during the Congestion Avoidance. Consequently, in TCP-Peach, the Congestion Avoidance needs some modifications. We introduce the variable *wdsn*. Upon receiving an ACK for a dummy segment, the sender checks the value of *wdsn* and

- If $wdsn = 0$, then the congestion window, *cwnd*, is increased by one, i.e. $cwnd := cwnd + 1$.
- If $wdsn \neq 0$, then the *wdsn* value is decreased by one, i.e. $wdsn := wdsn - 1$, and the congestion window value, *cwnd*, is not changed.

The variable *wdsn* is used in order to match the behaviours of TCP-Peach and TCP-Reno [8] when the network is congested. In the beginning of a new connection *wdsn* is set to zero.

2.2. The Sudden Start

The Sudden Start is executed in the beginning of a new connection in order to avoid the low throughput performance of Slow Start in long propagation delay networks. The variables cwnd and wdsn are initially set to one and zero, respectively. The Sudden Start lasts for one round trip time (RTT) then the sender enters the Congestion Avoidance phase. During the Sudden Start, the sender transmits one data segment and (rwnd − 1) dummy segments, where rwnd is the maximum value allowed for the congestion window size, i.e. cwnd $\leqslant$ rwnd, given by the receiver. The dummy segments reach the receiver and thus, their ACKs arrive to the sender only if there are unused resources in the network. The sender will receives these ACKs when the Sudden Start is over and the Congestion Avoidance is running as shown in Figure 1. Therefore, since wdsn is equal to zero, the sender will increase its congestion window, cwnd, all the times it receives an ACK for a dummy segment. Accordingly, if n_{Absorbed} is the number of dummy segments that the network is able to absorb, then, at the end of the Sudden Start, the transmission rate for the new connection suddenly jumps from 1/RTT to n_{Absorbed}/RTT. In Figure 2 we compare the behaviour

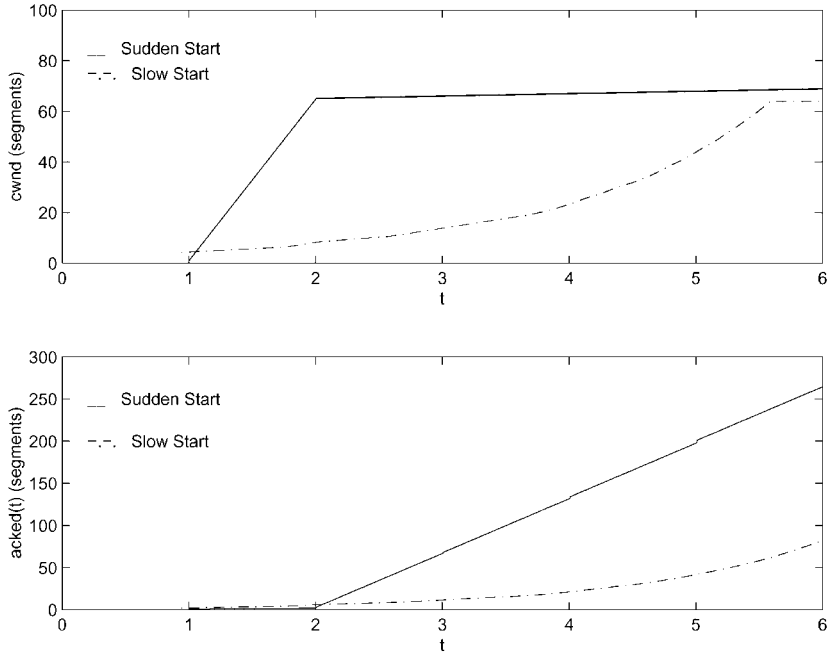

Figure 2. Comparison of the initial behaviour of TCP-Peach (solid lines) and TCP-Reno (dashed lines).

of TCP-Peach (solid lines) and TCP-Reno (dashed lines) in the beginning of a new connection. We assumed that rwnd = 64 segments; the unit for the time axis is the round trip time (RTT). In the upper plot of Figure 2, the congestion window, cwnd, for the TCP Peach reaches its final value within two round trip times since the beginning of the connection, while much more time is needed by traditional TCP implementations [13,8]. This implies that in the beginning of a new connection, the sender transmits data segments more rapidly than in traditional TCP implementations [13,8], as shown in the bottom plot of Figure 2.

2.3. The Rapid Recovery

The *Rapid Recovery* substitutes the classical Fast Recovery algorithm [8] with the objective of solving the throughput degradation problem due to link errors.

As shown in Figure 1, when a segment loss is detected through ndup duplicated ACKs, we use the original Fast Retransmit algorithm [8]. After completing the Fast Retransmit algorithm we apply the Rapid Recovery algorithm, which will terminate at the time when the ACK for the lost data segment is received. Consequently, the Rapid Recovery lasts for RTT. Then, the TCP sender will enter the Congestion Avoidance phase as depicted in Figure 1.

The Rapid Recovery first keeps the classical Fast Recovery conservative assumption that all segment losses are due to network congestion because the TCP layer does not know anything about the exact causes for the losses, i.e. due to network congestion or due to link errors [2]. Accordingly, the TCP sender halves its congestion window, cwnd, as in TCP-Reno [8], and sets the value of the variable wdsn equal to cwnd.

Thus, if the segment loss was detected when cwnd was equal to cwnd_0, then it becomes $\text{cwnd} = \text{cwnd}_0/2$, which means the sender will transmit $\text{cwnd}_0/2$ data segments approximately during the Rapid Recovery.

Moreover, in order to probe the availability of network resources, the TCP sender transmits cwnd_0 dummy segments. The ACKs for these dummy segments will be received after the ACK for the lost data segment, i.e. they will be received when the sender is in Congestion Avoidance.

If the packet loss is due to congestion, then the congested router can serve cwnd_0 packets per round trip time, approximately. As a result, the network will accommodate the $\text{cwnd}_0/2$ data packets, which have high priority, and only $\text{cwnd}_0/2$ among the cwnd_0 dummy segments transmitted during the Rapid Recovery. Each time the sender receives the ACK for a dummy segment, it controls the value of the variable wdsn, which is higher than 0. As a result, the congestion window, cwnd, is not increased due to the transmission of the dummy segments. In other words, TCP-Peach behaves like TCP-Reno [8] when a segment loss occurs due to network congestion. If the network is not congested and all dummy segments are ACKed to the sender, then the congestion window, cwnd, reaches the value it had before the packet loss was detected, i.e. $\text{cwnd} = \text{cwnd}_0$.

3. ANALYTICAL FRAMEWORK

Let X be a connection using TCP-Peach. We assume that the SACK option [14,15] is implemented. As a result, the loss recovery mechanisms are triggered only once for all packet losses within the same congestion window [14,15].

We also assume that the network can accommodate at most W_{MAX} segments per round trip time (RTT) from X. Note that if $b_{\text{Available}}$ is the available bandwidth for X, then $W_{\text{MAX}} \approx (b_{\text{Available}} \times \text{RTT})$.

We divide the time into *cycles*. As shown in Figure 3, a cycle is the time period between two consecutive recovery phases. For the nth cycle, we give the following definitions:

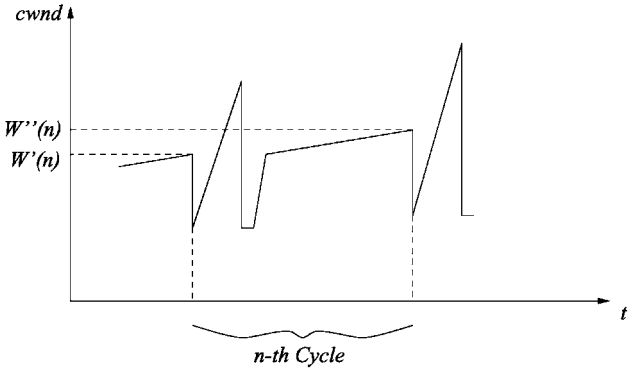

Figure 3. Cycle representation.

- $W'(n)$ is the congestion window, cwnd, when the cycle begins.
- $W''(n)$ is the congestion window, cwnd, when the cycle ends.

Obviously, the following relationship holds:

$$W''(n) = W'(n+1) \quad \forall n > 0 \tag{1}$$

Let $N(t)$ be the number of new data segments transmitted in the time interval $[0, t]$. The throughput is given by

$$\gamma = \lim_{t \to \infty} \frac{N(t)}{t} \tag{2}$$

Now let $Nc(t)$ represent the number of cycles in the time interval $[0, t]$. Accordingly, the throughput given in equation (2) can be obtained as

$$\gamma = \lim_{t \to \infty} \frac{N(t)}{Nc(t)} \times \lim_{t \to \infty} \frac{Nc(t)}{t} \tag{3}$$

Note that

- The term $\lim_{t \to \infty} N(t)/Nc(t)$ is the mean value of the number of new data segments transmitted in a cycle, $E\{N^{(\text{Cycle})}\}$.
- The term $\lim_{t \to \infty} Nc(t)/t$ is the reciprocal of the mean duration of a cycle, $1/E\{T^{(\text{Cycle})}\}$.

It follows that the throughput, γ, is

$$\gamma = \frac{E\{N^{(\text{Cycle})}\}}{E\{T^{(\text{Cycle})}\}} \tag{4}$$

Let $E\{N^{(\text{Cycle})}(w_1, w_2)\}$ represent the mean value of new data segments transmitted during a cycle characterized by the 2-tuple (w_1, w_2), i.e. $W'(n) = w_1$ and $W''(n) = w_2$. Applying the theorem of the total probability [16], we can write the mean value of the number of data segments successfully transmitted during a cycle, $E\{N^{(\text{Cycle})}\}$, as

$$E\{N^{(\text{Cycle})}\} = \sum_{w_1=1}^{W_{\text{MAX}}+1} \sum_{w_2=1}^{W_{\text{MAX}}+1} E\{N^{(\text{Cycle})}(w_1, w_2)\} P\{W'(n) = w_1,\ W''(n) = w_2\} \tag{5}$$

Analogously, let $E\{T^{(\text{Cycle})}(w_1, w_2)\}$ represent the mean value of the duration of a cycle characterized by the 2-tuple (w_1, w_2), i.e. $W'(n) = w_1$ and $W''(n) = w_2$. Applying the theorem of the total probability [16], we obtain

$$E\{T^{(\text{Cycle})}\} = \sum_{w_1=1}^{W_{\text{MAX}}+1} \sum_{w_2=1}^{W_{\text{MAX}}+1} E\{T^{(\text{Cycle})}(w_1, w_2)\} P\{W'(n) = w_1,\ W''(n) = w_2\} \tag{6}$$

In the following, first we evaluate $E\{N^{(\text{Cycle})}(w_1, w_2)\}$, then $E\{T^{(\text{Cycle})}(w_1, w_2)\}$, and finally $P\{W'(n) = w_1, W''(n) = w_2\}$.

Segment losses can occur for two reasons:

- *Link errors*: Assume that a segment loss due to link errors was detected at time t_0 when the congestion window, cwnd, is equal to w. In the next Rapid Recovery phase, which has the duration from t_0 to (t_0 + RTT), the sender transmits approximately ($w/2$) data segments and w dummy segments.
  - If $w \leqslant (2\,W_{\text{MAX}}/3)$, then the network can accommodate all data segments and dummy segments transmitted during the Rapid Recovery phase. The time required to completely recover from the data segment loss is (3 RTT/2). In this time interval the sender transmits w new data segments [7].
  - If $w > (2\,W_{\text{MAX}}/3)$, then the network will discard $(3\,w/2 - W_{\text{MAX}})$ dummy segments. At time $(t_0 + 3\,\text{RTT}/2)$ the congestion window, cwnd, will be equal to $(W_{\text{MAX}} - w/2)$. Between time t_0 and $(t_0 + 3\,\text{RTT}/2)$ the sender transmits $(W_{\text{MAX}} - w/2)$ new data segments.
- *Network congestion*: When the congestion window, cwnd, exceeds the value W_{MAX}, the network is not able to accommodate all segments transmitted by the sender. As a result, a congestion occurs. Suppose that a data segment loss due to network congestion is detected at time t_0. As explained in Reference [7], between time t_0 and (t_0 + RTT) the sender transmits $(W_{\text{MAX}}/2)$ new data segments. At time (t_0 + RTT) the congestion window, cwnd, is equal to $W_{\text{MAX}}/2$, as in the case of TCP-Reno [8].

Based on the above discussions it follows that $E\{N^{(\text{Cycle})}(w_1, w_2)\}$ and $E\{T^{(\text{Cycle})}(w_1, w_2)\}$ are determined as follows:

$$E\{N^{(\text{Cycle})}(w_1, w_2)\} = \begin{cases} \sum_{w=w_1}^{w_2} w & \text{if } w_1 \leqslant 2\,W_{\text{MAX}}/3 \quad \text{and} \quad w_1 \leqslant w_2. \\ \sum_{w=W_{\text{MAX}}-w_1*2}^{w_2} w & \text{if } w_1 > 2\,W_{\text{MAX}}/3 \quad \text{and} \quad W_{\text{MAX}} - w_1/2 \leqslant w_2 \end{cases} \tag{7}$$

and

$$E\{T^{(\text{Cycle})}(w_1, w_2)\} = \begin{cases} (3/2 + w_2 - w_1)\text{RTT} & \text{if } w_1 \leqslant 2\,W_{\text{MAX}}/3 \text{ and } w_1 \leqslant w_2 \\ (3/2 + w_2 - W_{\text{MAX}} + w_1/2) & \text{if } w_1 > 2W_{\text{MAX}}/3 \\ & \text{and } W_{\text{MAX}} - w_1/2 \leqslant w_2 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The probability $P\{W'(n) = w_1, W''(n) = w_2\}$ is given by

$$P\{W'(n) = w_1, W''(n) = w_2\} = P\{W''(n) = w_2/W'(n) = w_1\}\, P\{W'(n) = w_1\} \tag{9}$$

It is easy to demonstrate that $P\{W''(n) = w_2/W'(n) = w_1\}$ can be evaluated as follows:

$$P\{W''(n) = w_2/W'(n) = w_1\} = \begin{cases} [\prod_{w=w_1}^{w_2-1}(1-P_{\text{Loss}})^w][1-(1-P_{\text{Loss}})^{w_2}] \\ \quad \text{if } w_1 \leqslant 2W_{\text{MAX}}/3 \quad \text{and} \quad w_1 < w_2 < W_{\text{MAX}}+1 \\ [\prod_{w=W_{\text{MAX}}-w_1/2}^{w_2-2}(1-P_{\text{Loss}})^w][1-(-P_{\text{Loss}})^{w_2}] \\ \quad \text{if } 2W_{\text{MAX}}/3 < w_1 \leqslant W_{\text{MAX}}+1 \quad \text{and} \quad w_1 < w_2 < W_{\text{MAX}}+1 \\ \prod_{w=w_1}^{W_{\text{MAX}}}(1-P_{\text{Loss}})^w \\ \quad \text{if } w_1 \leqslant 2W_{\text{MAX}}/3 \quad \text{and} \quad w_2 = W_{\text{MAX}}+1 \\ \prod_{w=W_{\text{MAX}}-w_1/2}^{W_{\text{MAX}}}(1-P_{\text{Loss}})^w \\ \quad \text{if } w_1 > 2W_{\text{MAX}}/3 \quad \text{and} \quad w_2 = W_{\text{MAX}}+1 \\ 0 \quad \text{otherwise} \end{cases} \tag{10}$$

where P_{Loss} is the probability that a segment is dropped due to link errors.

Note that the memory of the past is lost at the beginning of each cycle. Therefore, the process $W'(n)$ is Markovian and can be characterized by its transition probability matrix, $Q^{(W')}$, whose generic element is given by

$$[Q^{(W')}]_{[w_1,w_2]} = P\{W'(n+1) = w_2/W'(n) = w_1\} \tag{11}$$

From Equation (1) we obtain,

$$[Q^{(W')}]_{[w_1,w_2]} = P\{W''(n) = w_2/W'(n) = w_1\} \tag{12}$$

where the probability $P\{W''(n) = w_2/W'(n) = w_1\}$ is given in Equation (10). We can evaluate $\pi^{(W')}$, which represents the row array whose xth element represents $P\{W'(n) = x\}$, as the solution of the linear system given by

$$\begin{cases} \pi^{(W')}Q^{(W')} = \pi^{(W')} \\ \pi^{(W')} \cdot \mathbf{1} = 1 \end{cases} \tag{13}$$

where $\mathbf{1}$ represents a column array whose elements are all equal to 1.

4. PERFORMANCE EVALUATION

In this section, we first analyse the effects of the round trip time (RTT) and the loss probability, P_{Loss}, on the throughput. For this purpose, we use the analytical paradigm developed in Section 3. We then compare the throughput performance of TCP-Peach and TCP-Reno through simulation.

4.1. Analytical results

In Fig. 4 we show the throughput performance of TCP-Peach for different values of the round trip time, RTT, and the loss probability, P_{Loss}. The throughput has been evaluated using Equation (4),

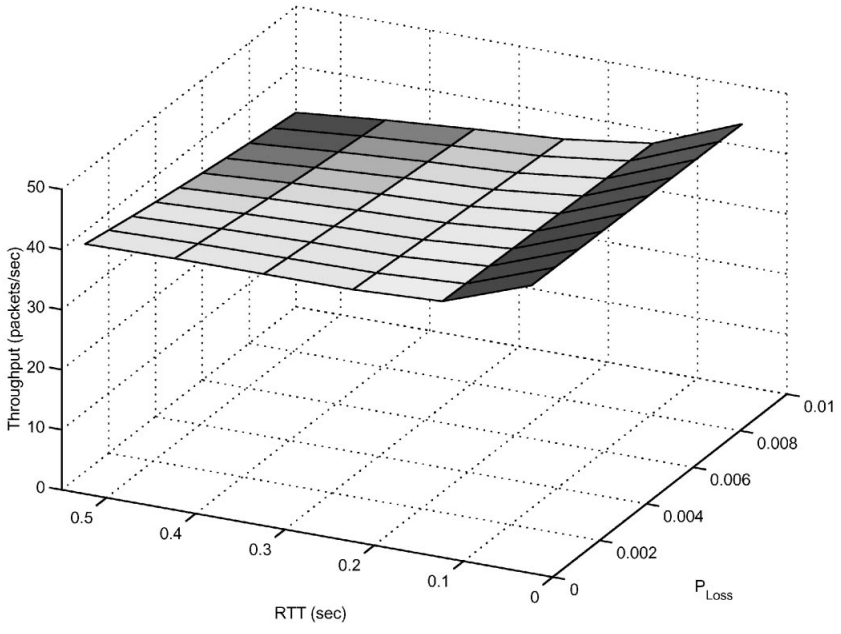

Figure 4. Throughput performance.

where $E\{N^{(\mathrm{Cycle})}\}$ and $E\{T^{(\mathrm{Cycle})}\}$ have been evaluated as explained in Section 3. We assumed that the available bandwidth is equal to 50 segments/s. The round trip time values range between 50 ms (LEO satellites) and 550 ms (GEO satellites) while the P_{Loss} values range between 0 and 10^{-2}. Note that the bit-error rate (BER) in satellite networks can be as high as 10^{-4}, i.e. one bad bit out of 10 000 bits. For TCP segments of 1000 bytes, the BER 10^{-4} gives P_{Loss} higher than 10^{-2} even if powerful error correction algorithms are applied.

As expected, the throughput decreases when the round trip time (RTT) and the loss probability, P_{Loss}, increase in Figure 4. However, note that TCP-Peach is very robust to high values of the round trip time and loss probability. In fact, the throughput obtained when RTT = 550 ms and $P_{\mathrm{Loss}} = 10^{-2}$ is only 26.83 per cent lower than the case RTT = 50 ms and $P_{\mathrm{Loss}} = 0$. Under the same conditions, the throughput performance degradation evaluated in Reference [1] is higher than 80 per cent.

4.2. Simulation results

Now we compare the performance of TCP-Peach and TCP-Reno in the case of several interactive flows. The TCP-Reno implementation considered here is suggested in Reference [17] and is also known as New Reno because it removes some problems of the original Reno [18,19]. Moreover, we assume that both TCP-Reno and TCP-Peach implement the SACK options [14,15].

We simulate N senders transmitting data to N receivers as shown in Figure 5. The N streams are multiplexed in the Earth Station A, whose buffer can accommodate K segments. Both data and dummy segments may get lost due to link errors with a probability P_{Loss}. We assume that $N = 10$, $K = 50$ segments and rwnd = 64 segments. We also assume that the link

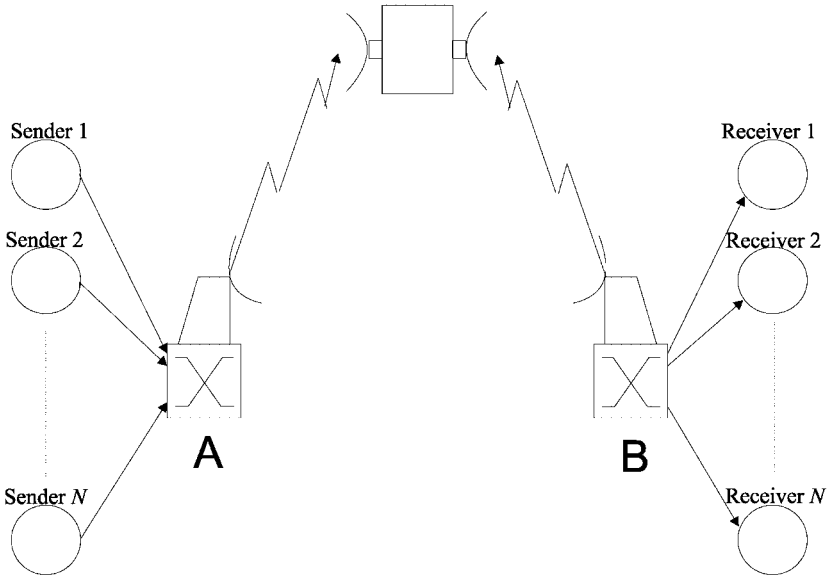

Figure 5. Simulation scenario.

capacity is $c = 1300$ segments/s which is approximately 10 Mb/s for TCP segments of 1000 bytes. The RTT values considered are RTT = 50 ms (LEO systems), RTT = 250 ms (MEO systems) and RTT = 550 ms (GEO systems). All the results shown have been obtained by considering the system behaviour for a time interval equal to $t_{\text{Simulated}} = 500$ s, which is almost 1000 times the highest round trip time value.

In Figure 6, we compare the throughput results of TCP-Reno and TCP-Peach for different values of round trip times, RTT, and loss probabilities due to link errors, P_{Loss}.

We observe that the higher the round trip time (RTT) and the probability P_{Loss}, the lower are the throughput values obtained using TCP-Reno. Although similar behaviour is observed in TCP-Peach, overall TCP-Peach provides higher throughput values than TCP-Reno. Moreover, it is easy to see that the higher the round trip time (RTT) values and the loss probability, P_{Loss}, values, the higher is the throughput gain, g, obtained by the TCP-Peach. The throughput gain, g, can be measured as the ratio between the throughput, r_{Peach}, obtained by TCP-Peach and the throughput, r_{Reno}, obtained by TCP-Reno, i.e.

$$g = r_{\text{Peach}}/r_{\text{Reno}} \tag{14}$$

In all cases we investigated, g always increases with increasing RTT and P_{Loss}.

Note that in our experiments we assumed one hop satellite communication. Although this may be true for GEO cases, it may not be for LEO cases, i.e. the connection from sender to receiver may pass through several LEOs (multihop cases). Consequently, the RTT may become higher and the performance improvements of TCP-Peach may then become much more obvious in LEO cases as well.

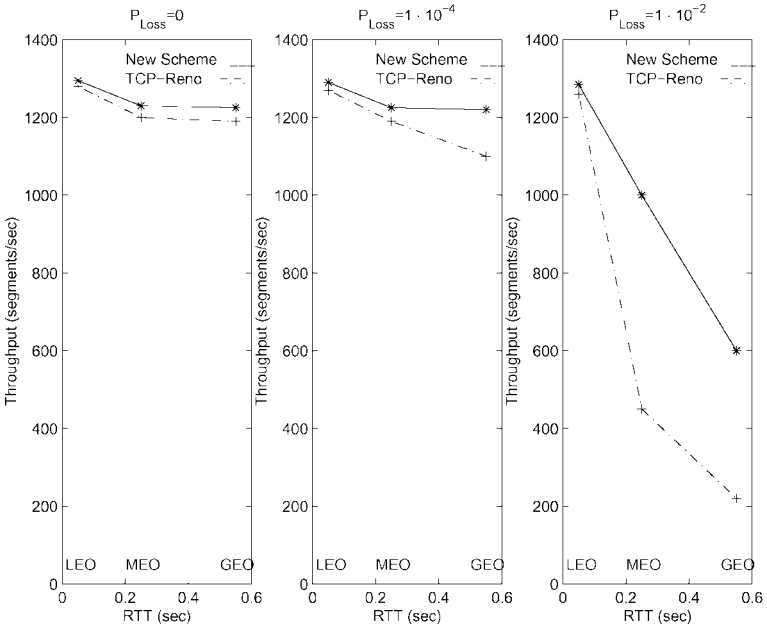

Figure 6. Performance comparison of TCP-Peach (solid lines) and TCP-Reno (dashed lines) when the SACK option is implemented for different values of loss probabilities.

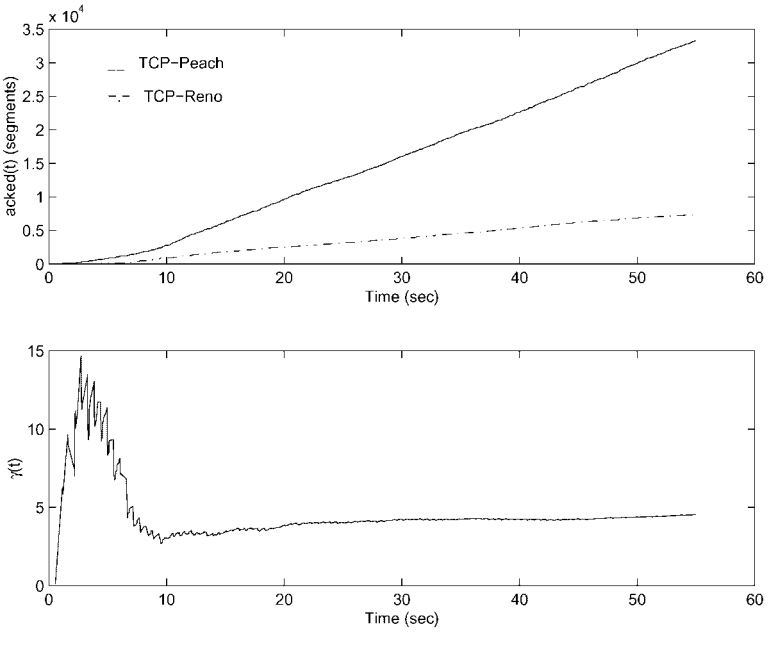

Figure 7. Behaviour of acked(t) for TCP-Reno and the TCP-Peach (upper plot) and of $\gamma(t)$ (bottom plot).

In Figure 7, we show the values for acked(t) and $\gamma(t)$ for cases when the segment loss probability due to link errors is $P_{\text{Loss}} = 1 \times 10^{-2}$ and the round trip time is RTT = 550 ms (GEO system). Note that

- acked(t) is the number of TCP data segments acknowledged in the time interval $[0, t]$.
- $\gamma(t)$ is a measure of the throughput gain in $[0, t]$ achieved using the TCP-Peach:

$$\gamma(t) = \frac{\text{acked}_{\text{Peach}}(t)}{\text{acked}_{\text{Reno}}(t)} \tag{15}$$

In all experiments conducted, we observed that $\gamma(t)$ increases rapidly in the beginning. This is due to the improvement achieved by the Sudden Start (TCP-Peach) compared to the Slow Start (TCP-Reno). For higher values of t, $\gamma(t)$ converges to a value which again depends on the performance improvement achieved by the Rapid Recovery Algorithm in TCP-Peach.

Currently, web applications are very popular in the Internet. Therefore, we simulate the case in which the N senders in Figure 5 are TCP-Peach senders transmitting web pages, each of S segments. As soon as a web page transfer is completed, i.e. all the ACKs for the S segments of one web page are received, the TCP sender begins to transmit a new web page. In Figure 8, we show the average throughput values achieved by TCP-Peach [7] (solid lines), TCP-Reno [8] (dotted lines) and TCP-Reno with the increased initial window (IIW) option [3] (dashed lines) for different values of the round trip time (RTT) and the loss probability, P_{Loss}. In all cases we

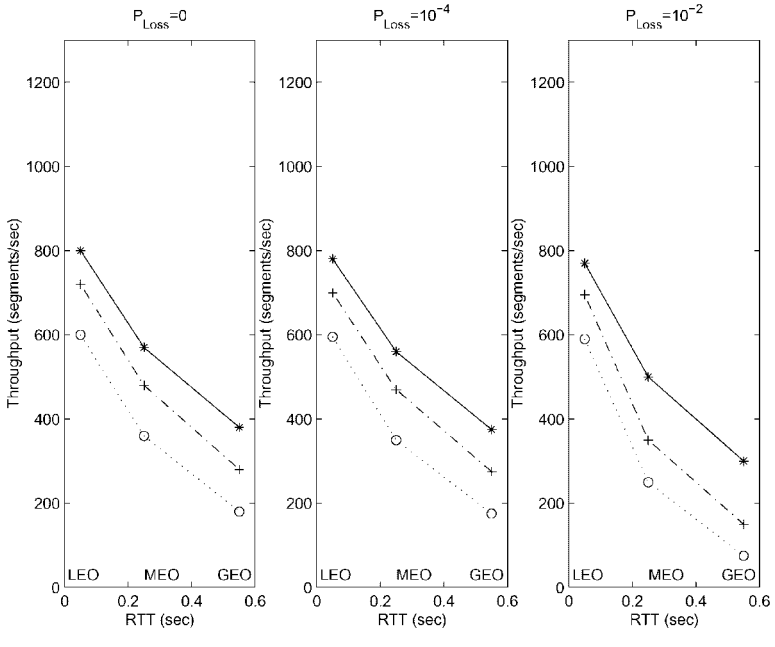

Figure 8. Transfers of files of $S = 50$ segments: average throughput for TCP-Peach (solid lines), TCP-Reno (dotted lines) and TCP-Reno + IIW (dashed lines).

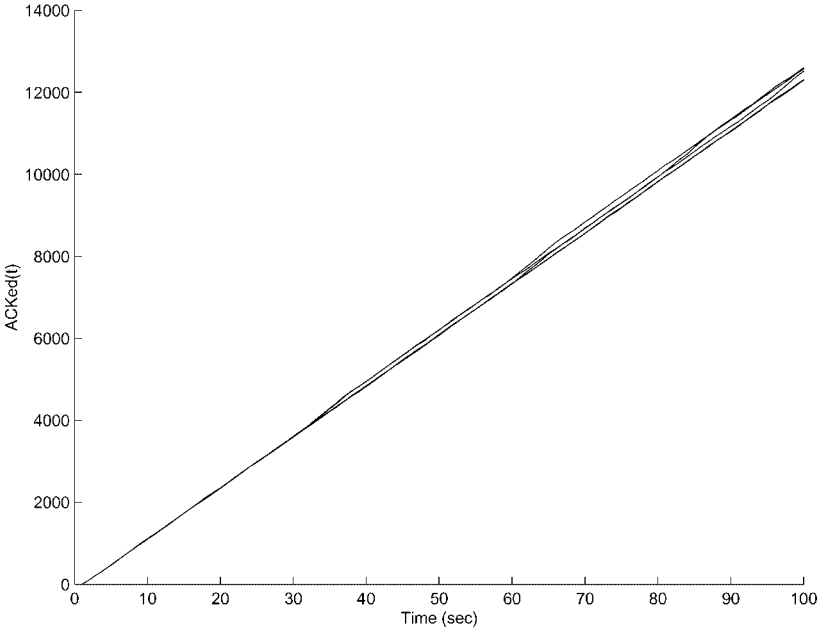

Figure 9. Fairness evaluation.

assumed that the SACK option [14,15] is used. TCP-Peach achieves the highest throughput performance in Figure 8.

Finally, we evaluate the fairness of TCP-Peach. Let $\text{acked}_i(t)$ represent the number of segments acknowledged in the time interval $[0, t]$ for connection i, for $i = 1, 2, \ldots, N$. In Figure 9, we show $\text{acked}_i(t)$ dependent on time t for i from 1 to N, which are obtained by simulating the system in Figure 5 with parameters $N = 10$, $K = 50$ segments, rwnd $= 64$ segments, $c = 1300$ segments/s, $P_{\text{Loss}} = 0$, RTT $= 550$ ms and all connections using TCP-Peach. In Figure 9, at any time t, $\text{acked}_{i'}(t) \approx \text{acked}_{i''}(t)$, for any i' and i''. This means that each TCP-Peach connection is given a fair share of the system resources. We obtained similar results using other values for system parameters.

5. CONCLUSIONS

In this paper we introduced an analytical model of TCP-Peach and evaluated its performance. The TCP-Peach is based on the use of dummy segments which are low priority segments that do not carry any new information to the receiver. Therefore, TCP-Peach requires the routers along the connection to implement some priority mechanism at the IP layer. Priority can be supported at the IP layer by the *Type of Service* option in the traditional IP, whereas IPv6 explicitly supports several priority levels. TCP-Peach is composed of two new algorithms: the Sudden Start and the Rapid Recovery, and the Congestion Avoidance and the Fast Retransmit as introduced in References [13,8].

The main feature of TCP-Peach is that it only requires modifications in the sender behaviour. If the receiver implements the SACK option [14,15], straightforward modifications of TCP-Peach as presented here can allow a further improvement in the throughput performance.

ACKNOWLEDGEMENTS

The work of Ian F. Akyildiz and Giacomo Morabito was supported by NASA-Ames under contract # NAG 2-1262.

REFERENCES

1. Lakshman TV, Madhow U. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking* 1997; **5**(3).
2. Partridge C, Shepard TJ. TCP/IP performance over satellite links. *IEEE Network Magazine* 1997; 44–49.
3. Akyildiz IF, Jeong S-H. Satellite ATM networks: a survey. *IEEE Communication Magazine* 1997; **35**(7):30–43.
4. Metz C. TCP over satellite… The final frontier. *IEEE Internet Computing* 1999; 76–80.
5. Allman M *et al.* Ongoing TCP research related to satellites. *RFC* 2760, February 2000.
6. Allman M, Glover D, Sanchez L. Enhancing TCP over satellite channels using standard mechanism. *Internet RFC* 2488, 1999.
7. Akyildiz IF, Morabito G, Palazzo S. TCP-Peach: a new congestion control scheme for satellite IP networks. *IEEE/ACM Transactions on Networking* June 2001: **9**(3).
8. Jacobson V. Congestion avoidance and control. *Technical Report*, April 1990.
9. Postel J. DoD standard internet protocol. *IETF RFC* 760, January 1980.
10. Deering S, Hinden R. Internet protocol version 6 (IPv6) specification. IETF RFC 2460, December 1998.
11. Bernet Y *et al.* A framework for differential services. *Internet draft*, *draft-ietf-diffserv-framework*-02.*txt*, February 1999.
12. http://www.cisco.com
13. Jacobson V. Congestion avoidance and control. *Proceedings of ACM SIGCOMM'88*, August 1988.
14. Mathis M, Mahdavi J, Floyd S, Romanow A. TCP selective acknowledgement options. *ftp*: *ftp.ietf.cnri.reston.va.us/internet-drafts/draft-ietf-tcplw-sack*-00.*txt*, April 1996.
15. Mathis M, Mahdavi J. Forward acknowledgement: refining TCP congestion control. *Proceedings of ACM SIGCOMM'96*, August 1996; 281–291.
16. Feller W. *An Introduction to Probability Theory and its Applications*. Wiley: New York, 1968.
17. Fall K, Floyd S. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *ACM Computer Communication Review* 1996; **26**(3):5–12.
18. Floyd S. TCP and successive fast retransmits. *ftp://ftp.ee.lbl.gov/papers/fastretrans.ps*, February 1995.
19. Hoe J. Improving the start-up behavior of a congestion control scheme for TCP. *Proceedings of ACM SIGCOMM'96*, August 1996.
20. Allman M, Floyd S, Partridge C. Increasing TCP's initial window. *Internet RFC* 2414, 1998.

