INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Improving TCP/IP over Geostationary Satellite Links

Nesrine Chaher — Chadi Barakat — Walid Dabbous — Eitan Altman

N° 3573

December 1998

THÈME 1

Rapport de recherche

ISSN 0249-6399

Improving TCP/IP over Geostationary Satellite Links

Nesrine Chaher , Chadi Barakat , Walid Dabbous , Eitan Altman

Thème 1 — Réseaux et systèmes
Projet Mistral

Rapport de recherche n˚ 3573 — December 1998 — 20 pages

Abstract: We focus in this paper on the undesirable phenomenon of frequent losses during slow-start when TCP operates in large bandwidth-delay product networks such as those including geostationary satellite links. This phenomenon, already identified in [14, 7], results in a degradation in TCP throughput due to a double consecutive slow-start phases per TCP cycle. Given the high cost and the scarcity of satellite links, it is of importance to find solutions to get rid of these losses. We propose two simple modifications to TCP algorithms and illustrate their performance by mathematical analysis and simulations. First, we decrease the slow-start threshold (with respect to current TCP) to less than the window size at which these losses occur, and second, we space the transmission of packets sent during slow-start.

Key-words: TCP, Slow-Start, Congestion Avoidance, Satellite Links, Modeling, Simulations, Performance Evaluation.

All the correspondence to be sent to Dr. Eitan Altman, INRIA B.P. 93, 2004 Route des Lucioles, 06902 Sophia Antipolis Cedex, France. Tel: (33) 4 92 38 77 86; Fax: (33) 4 92 38 77 65.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Amélioration de TCP/IP sur des liens passant par des satellites geostationnaires

Résumé : Dans ce papier, on s'intéresse au phénomène de pertes de paquets dans la phase slow-start du protocole TCP lorsque ce dernier opère dans des réseaux ayant un grand produit délai bande passante comme ceux contenant des liens passant par des satellites geostationnaires. Ce phénomène, déjà identifié dans [14, 7], réduit le débit de la connexion TCP à cause de deux phases slow-start consécutives dans chaque cycle TCP. Étant donné le coût élevé et la rareté des liens satellites, il est très important de trouver des solutions à ces pertes indésirables. Afin de résoudre ce problème, on propose deux simples modifications des algorithmes de TCP puis on étudie, avec un analyse mathématique et des simulations, leur effet sur la performance du protocole. Tout d'abord, on réduit le seuil du slow-start, par rapport à TCP standard, à une valeur inferieure à la taille de la fenêtre de congestion lorsque les pertes se produisent. Ensuite, on espace les paquets transmis pendant la phase slow-start.

Mots-clés : TCP, Slow-Start, Congestion Avoidance, Liens Satellites, Modélisations, Simulations, Évaluation de Performance.

1 Introduction

TCP, the window-based reliable Transmission Control Protocol of the Internet [11, 18], uses two algorithms *Slow-Start* and *Congestion Avoidance* to control the flow of packets in a network. With slow-start, the window of the sender is set initially to one segment and it is increased by one for every acknowledgment (ACK) received. The result is an exponential growth of the connection window, therefore of the throughput, which prevents the sender from overwhelming the network with an inappropriately large burst of traffic. This process continues until we reach the slow-start threshold. At this point, the source moves to congestion avoidance where the window is increased slower by one segment for every window's worth of acknowledged packets, hence every round trip time (RTT). This slow growth aims to probe the network for extra bandwidth.

TCP is very widely used by most of today Internet applications. However, it is known not to perform well in new networks having a large bandwidth-delay product such as those including satellite links. In its present form, TCP is unable to use efficiently the available bandwidth in these networks. Another problem with TCP is its weakness to recover from frequent losses in a wireless environment. For TCP, any packet loss is a congestion indication and consequently it cuts back its window. Due to the high bit error rate in satellite links, such behavior leads to a significant deterioration in TCP throughput as shown in [14]. In the following, we list some of the proposed solutions to these problems.

- **Path MTU Discovery [5]:** To reduce the overhead due to the packet header, a large number of data bytes must be sent in a TCP segment. However, large packets will be fragmented if they encounter a network with small MTU (Maximum Transfer Unit) which increases considerably the overhead. This proposed solution lets TCP detect the largest packet that can cross the Internet without incurring the cost of fragmentation and reassembly. Also, because the TCP sender increases its window by segments, using large packets yields a faster growth of the throughput.

- **Forward Error Correction (FEC) [5]:** To recover from losses due to corruption rather than congestion without the intervention of TCP mechanisms, the satellite link must be made reliable. Error recovery mechanisms based on retransmission (ARQ) takes a lot of time due to the long propagation delay. It is therefore important to improve the link quality by adding some FEC. This solution improves the bit error rate of a satellite channel but it consumes some of the available bandwidth. Also, it has the drawback of increasing the burstiness of errors which requires additional mechanisms to restore the random error nature of the satellite channel.

- **Using Large Windows [12]:** Even if the network has a high bandwidth and the receiver has a large buffer, the throughput achieved by TCP is limited by:

$$\text{throughput} = \frac{\text{max window size}}{\text{RTT}} \tag{1}$$

This follows since a TCP source cannot send more than the window size of packets in an RTT time. Using the standard maximum TCP window of 64Kbytes (coded on 16 bits) and a GEO satellite link of RTT 560ms, this limits the throughput to 117 Kbytes/s. To send at higher rates, a window scale option has been added to TCP. It consists in multiplying the old window field by a scale factor coded on 14 bits. This lets TCP use larger windows, up to 2^{30} Bytes.

- **Increasing the initial window of TCP [4]:** Due to the large RTT, the congestion window grows quite slowly in satellite links. This increases the transfer duration and results in a low utilization of the bandwidth during the early RTTs of the connection life. Starting at an initial window W_I saves $log_2(W_I)$ RTTs at the beginning of the connection. The proposed value for W_I is:

$$\min\{4 * \text{MSS}, \max\ (2 * \text{MSS}, 4380 \text{ octets})\} \tag{2}$$

  where MSS is the Maximum Segment Size. To not alter the congestion control in TCP, W_I is only used at the beginning of the connection. This solution also avoids the problem of undesirable timeouts that occur in TCP versions using Delayed ACKs option[1].

- **Modifying the window dynamics in TCP (Byte counting) [3]:** To make faster the window growth especially in case of Delayed ACKs and ACK losses, it has been suggested to increase the congestion window based on the number of segments acknowledged instead of the number of ACKs received. This modification is proposed for the initial slow-start phase and for those following a timeout.

- **Selective Acknowledgments [15]:** Current TCP uses cumulative ACKs which carry only the sequence number of the next packet expected by the receiver. This limited information makes the TCP sender unable to recover from more than one lost segment per RTT. This is known to cause special problems when multiple losses occur in the same window of data as with the bursty loss pattern in satellite links. To overcome this deficiency, the use of SACKs has been proposed [15]. SACKs consist in the destination informing the sender of the identity of packets correctly received since the last in-order one. With this information, the source detects the gaps in the receiver buffer and, therefore, it becomes able to retransmit more than lost packet in the same RTT which results in a faster recovery.

- **Virtual sources and destinations:** A well known way to get rid of problems concerning the long delay in a control loop is to split it into several small loops. This idea is well established in other contexts in telecommunications, such as the control of ABR traffic in ATM [1]. A similar idea has been proposed for TCP [2]. The satellite link is isolated from the rest of the Internet using two routers (a virtual destination and a

[1]This option lets the receiver acknowledge every other packet but it prohibits it from delaying an ACK for than a timeout.

virtual source). The sender first transmits to the virtual destination that will take care of the acknowledgments and retransmissions. The router acknowledges the data to the sender as if the receiver has got the data; it sends then the data at a proper (bounded) rate through the satellite link, which serves as a simple pipe without congestion. At the other end, the other router (virtual source) transmits to the receiver, and if a segment is later lost this router will retransmit it. This means that it needs to have a copy of the data for retransmissions. The advantage is that all this is transparent to the sender. However, this solution breaks the end-to-end semantics associated with the TCP protocol.

- **Opening several connections at a higher application level (XFTP) [6]:** The idea is to open several TCP connections in order to send a single file. The advantage is that this does not need any modification to standard TCP. Using many connections simultaneously increases the maximum and the initial windows. Also, it makes faster the growth of the throughput. XFTP uses an adaptive algorithm to find the appropriate number of connections so to not overload the network. The result is a more aggressive TCP, hence a higher throughput.

In this paper, we investigate a problem that occurs when TCP operates in a network having a small buffering capacity compared to its bandwidth-delay product. It is the problem of loosing packets during slow-start before fully utilizing the available bandwidth. These losses are due to the high rate at which the TCP source sends bursts of packets during slow-start. Indeed, an incoming ACK triggers the transmission of a burst of two packets which gives a sending rate twice the available bandwidth[2]. If the network buffers are not designed to absorb this high rate, they will overflow and losses will occur before the source gets in congestion avoidance. The window size when these losses are detected is a wrong estimation of the network capacity. But TCP considers it as the maximum reachable window which results in a throughput degradation. This problem, known as the double slow-start phenomenon, has been previously studied in [14, 7].

Due to their high propagation delay, this problem is very likely to appear in geostationary satellite links. Given the high cost and the scarcity of these links, a solution to the double slow-start phenomenon is required. We propose in this paper two possible changes to TCP in order to solve this problem of throughput degradation. Using a fluid approach, the effect of these improvements on TCP performance is mathematically analyzed. The validity of the results is then proved by a set of simulations using **ns** the Network Simulator [16].

The first proposition consists in modifying the reduction factor one half used by standard TCP to calculate the slow-start threshold after a loss detection. A similar idea has already been proposed in [10] for the operation of the initial threshold so as to improve the performance of short data transfers. Because this problem of bursts with high rate do not occur

[2] Of course if the receivers acknowledges every packet and if the ACKs are not lost on the return path.

in congestion avoidance, leaving slow-start early before the buffer overflow solves the problem. In our analysis, we find an expression for the required factor to get rid of the double slow-start given the network parameters. A study of the throughput as a function of this factor is also performed.

Secondly, we propose to reduce the transmission rate during slow-start in order to avoid the buffer overflow and then the undesirable losses. Instead of sending immediately a burst of two packets in response to an ACK, the source inserts a certain delay before the second packet transmission. This proposal is similar in spirit to the one proposed by Partridge [17] of spacing the acknowledgments which requires, however, changes in the behavior of the destination or adding intelligence to the switches. We present an analysis of the spacing of packets sent in slow-start.

In the next section, we outline the fluid model used in our analysis. Some results found in [7, 14] concerning the double slow-start phenomenon are mentioned. In section 3 and 4, the first and the second proposition are respectively analyzed. In section 5, we present our simulation model and the obtained results. Section 6 gives conclusions and some open questions.

2 The model

The network is modeled as a single bottleneck node of bandwidth μ and of a buffer with capacity B having a FIFO service discipline. We suppose that the TCP source is connected to the bottleneck via a high speed link. When leaving the bottleneck, TCP packets cross the satellite link to the destination where they are acknowledged. The ACKs return to the source via a non congested path. The round trip time between the transmission of a packet and the receipt of its ACK is modeled as the sum of a constant component τ, the service time at the bottleneck $1/\mu$ and the waiting time in the buffer. τ represents the propagation delay plus the processing time at the other nodes of the network. As in [7], we suppose that the TCP connection shares the bottleneck with an uncontrolled exogenous traffic of rate λ ($\rho = \frac{\lambda}{\mu} < 1$ for stability requirement). The different parameters of the model are shown in figure 1.

We assume also that the source has always packets to send and that the receiver acknowledges every new packet.

Remark 1 *Acknowledging packets at lower frequency as in the Delayed ACKs TCP option [8] alleviates the problem of double slow-start by reducing the source sending rate. However, it has the drawback of increasing the time taken by slow-start, hence degrading the performance given the long propagation delay in satellite links [3]. The modification of TCP window dynamics mentioned in section 1 gives on average the same behavior as acknowledging every packet. Indeed, Delayed ACKs arrive at the source at half the available bandwidth. Increasing the congestion window by the number of segments acknowledged*

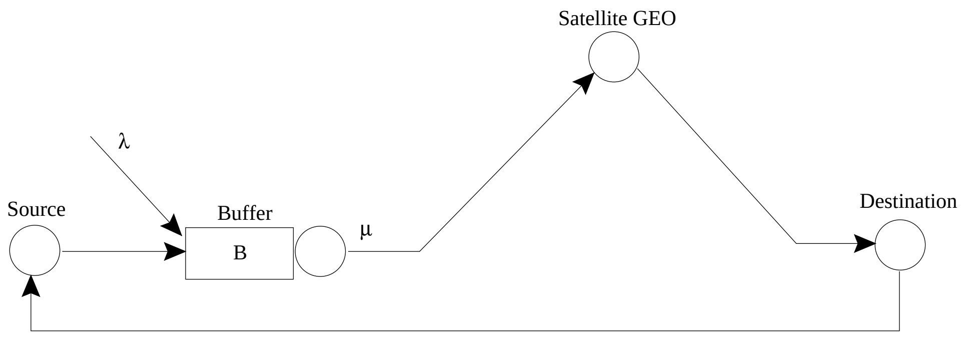

Figure 1: The Network Model

makes every Delay ACK trigger the transmission of a burst of four packets at the source, therefore, on average, the transmission at twice the available bandwidth as in our case.

Let us now define the different parameters used to model the TCP connection. The source has always a congestion window[3] W which is the maximum allowed number of unacknowledged packets. We suppose that the receiver advertised window is large enough so that it doesn't affect the flow control. Starting at a window of one segment, the source increases W by one for every received ACK until it reaches the slow-start threshold W_{th}. Here, it switches to congestion avoidance which yields an increase in W by $1/\lfloor W \rfloor$ when an ACK is received, therefore by one segment every RTT. Note that, in practice, W is stored in bytes and it is incremented by $\frac{MSS*MSS}{W}$ bytes for every incoming ACK [18]. This gives an increase slightly less than one MSS per RTT. Now, if we ignore the queuing time in front of the long propagation delay, RTT can be considered as constant and consequently the window increases linearly during congestion avoidance. If we don't consider the random errors on the satellite link (i.e . by adding enough FEC), this linear growth, which is slower than the exponential one during slow-start, continues until we reach the maximum reachable window W_{max}. This is the maximum number of TCP packets that can be fit into the pipe and the buffer. Here, a loss occurs due to buffer overflow. According to TCP-tahoe [11] used in this paper, W_{th} is set to half W_{max} and W to 1. A new slow-start and then a new TCP cycle begins. We give W_{max} the value found in [7]

$$W_{max} = B\frac{\mu - \lambda}{\mu} + \tau(\mu - \lambda) + 1 \tag{3}$$

We note here that the addition of the exogenous traffic reduces the share of the TCP connection in the buffer and in the bandwidth.

What happens in the double slow-start case is that a loss occurs due to a buffer overflow

[3]In our analysis, W is measured in segments rather than in bytes.

before reaching W_{th}. Therefore, a new slow-start begins with a threshold less than W_{th}. The source gets in congestion avoidance at a smaller window and then takes more time to reach W_{max}. This leads to a degradation in TCP performance.

According to the analysis in [7, 14], the slow-start can be divided into mini-cycles of duration τ. At the beginning of mini-cycle n, a burst of 2^{n-1} ACKs arrives at the source at rate $\mu - \lambda/2$ and another burst of 2^n TCP packets leaves it at rate $2\mu - \lambda$. Thus, the queue at the bottleneck builds up at rate μ (the difference between the total input rate 2μ and the output rate μ). The buffer overflows when we reach a certain window W_B and we are still in slow-start. W_B is given by

$$W_B = \frac{B}{\mu}(2\mu - \lambda) \tag{4}$$

The condition to get a double slow-start is simply $W_B \leq W_{th}$. If we define β as the normalized buffer capacity, we get the following condition found in [7]

$$\beta = \frac{B}{\tau(\mu - \lambda) + 1} \leq \frac{1}{3 - \lambda/\mu} \tag{5}$$

For $\lambda = 0$, this condition becomes similar to the $\beta \leq 1/3$ found in [14].

When the loss occurs at W_B, it isn't immediately detected. We must wait a complete RTT before the reception of the third duplicate ACK if we suppose that the Fast Retransmit algorithm [18] is supported. During this time, the source receives W_B new ACKs. Therefore, the window grows from W_B to a value W_D which depends on the position of W_{th} with respect to W_B and $2W_B$. The maximum value of W_D is $2W_B$ but it can be equal to W_{th} if we get in congestion avoidance before detecting the loss. We can write

$$W_D = min(W_{th}, 2W_B) \tag{6}$$

The second slow-start begins with a threshold $W'_{th} = W_D/2$. Because this new threshold is less than the overflow window, we get in congestion avoidance at the end of this phase. As in the previous case, the congestion avoidance lasts until $W = W_{max}$.

It is clear that to avoid the double slow-start, we must always maintain $W_B > W_{th}$. This can be accomplished by either increasing W_B or decreasing W_{th} if the network parameters satisfy equation (5). Our propositions treat theses two possibilities.

Using TCP-tahoe doesn't mean that this problem cannot occur with the other versions of TCP. We see it every time the slow-start algorithm is called due to a fast recovery [18] failure. Also, we encounter it at the beginning of the connection. Moreover, because a large number of packets is lost between the first loss and its detection, it is difficult for the other versions of TCP to recover from these losses [10]. A timeout occurs and a second slow-start is finally called with a threshold one fourth W_D. This leads to poorer performance particularly for short life connections.

3 The first proposition: Decreasing the slow-start threshold

The idea is to decrease W_{th} to get below W_B. This makes the source enter congestion avoidance with one slow-start. Because W_{max} is imposed by network parameters, the only possible way is to change the reduction factor one half which is used in determining W_{th}. Therefore, we take $W_{th} = \gamma W_{max}$ with $0 < \gamma \leq 1/2$. A γ greater than $1/2$ violates the congestion control principle of TCP.

This new factor doesn't change W_B, so the condition to solve the double slow-start remains $W_B > W_{th} = \gamma W_{max}$. If we replace W_B and W_{max} by their values given in equations (4) and (3), we find that γ must be chosen so that

$$\gamma < \frac{\beta(2-\rho)}{\beta(1-\rho)+1} = \frac{W_B}{W_{max}} = \gamma_{max} \tag{7}$$

It is clear that γ_{max} decreases when β becomes smaller due an increase in μ or τ or a decrease in the buffer capacity B. This means that W_B is becoming farther from one half W_{max} and then we must reduce further W_{th} to make the double slow-start vanish. We note also that an increase in λ, thus in ρ, makes γ_{max} move quickly to infinity (Figure 2). This increase is very sharp when ρ approaches 1. Thus, the double slow-start problem becomes less important when the exogenous traffic grows. In fact, the aggressive behavior of TCP during slow-start makes the decrease in W_B smaller than that of W_{th} reducing the likelihood of a buffer overflow during slow-start.

Even if it solves the problem of double slow-start for a β satisfying equation (5), this solution doesn't lead always to a better performance. To show this, we consider, for a given β, the window size at the beginning of the congestion avoidance phase as a means to compare TCP performance for two different γ. Any throughput improvement requires an increase in this window.

First we consider the case of a $\gamma > \gamma_{max}$. Such γ doesn't solve the double slow-start problem, thus the congestion avoidance still starts at W'_{th}, the threshold of the second slow-start phase. By studying the behavior of W'_{th} as a function of γ, we can evaluate how much a faulty reduction factor affects the performance. The following analysis shows that choosing a wrong γ by any adaptive protocol not only doesn't solve the double slow-start but also it deteriorates the throughput with respect to the existing TCP. To calculate $W'_{th}(\gamma)$, two cases must be considered:

- $2\gamma_{max} < 1/2$
  In this case, using (6) and (7), we have $W'_{th}(1/2) = W_B$ before the change in the factor one half. Now, for a $\gamma \geq 2\gamma_{max}$, we get $W_{th} = \gamma W_{max} > 2W_B$, then $W_D = 2W_B$ and $W'_{th}(\gamma) = 2\gamma W_B$. This window is always smaller than $W'_{th}(1/2)$ which results

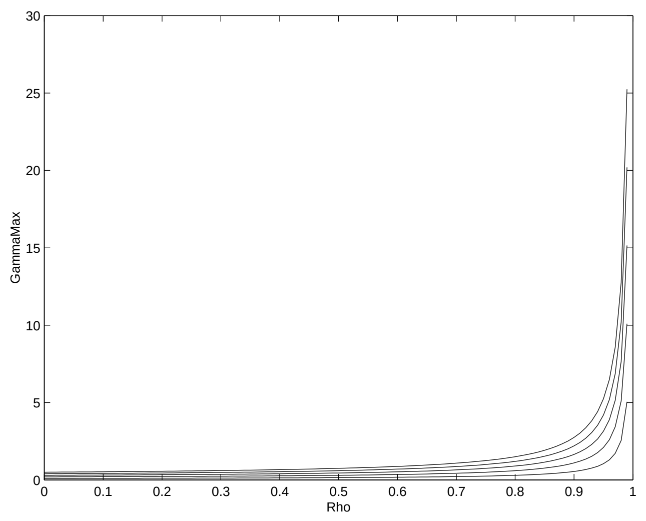

Figure 2: γ_{max} vs. ρ for different network parameters

in a linear performance degradation when γ moves from $1/2$ to $2\gamma_{max}$. Now, for a $\gamma < 2\gamma_{max}$, $W_{th} < 2W_B$, thus $W_D = W_{th}$ and $W'_{th}(\gamma) = \gamma^2 W_{max}$ which is always less than $W'_{th}(1/2)$. This means that moving γ between $2\gamma_{max}$ and γ_{max} decreases quickly the window at the beginning of congestion avoidance and, hence, the performance.

- $2\gamma_{max} \geq 1/2$
  Here, $W'_{th}(1/2) = W_{th}/2$ for standard TCP. However, for a γ between γ_{max} and $1/2$, W_{th} stays less than $2W_B$, therefore $W'_{th} = \gamma^2 W_{max}$ which is a decreasing function of γ always smaller than $W'_{th}(1/2)$. Thus, also in this case, the throughput deteriorates quickly when a wrong γ is chosen between $1/2$ and γ_{max}.

Now, to study the effect of a $\gamma < \gamma_{max}$ on TCP performance, we consider also two cases. Note that such γ solves the problem of double slow-start but it makes the congestion avoidance start at a window γW_{max}.

- $\gamma_{max} > 1/4$
  As we have seen, the loss is detected at $W_D = W_{th}$ for standard TCP. Therefore, the comparison must be done between $W'_{th}(1/2) = W_{max}/4$ and γW_{max}.

- $\gamma_{max} \leq 1/4$
  In this case, to study the performance of TCP, we must compare $W'_{th} = W_B$ to γW_{max}. Note that W_B is greater than $W_{max}/4$ using equation (7).

These two cases show that if the chosen factor is less than $1/4$, it reduces the throughput even if it solves the problem of double slow-start. In contrast, if in the first case we choose

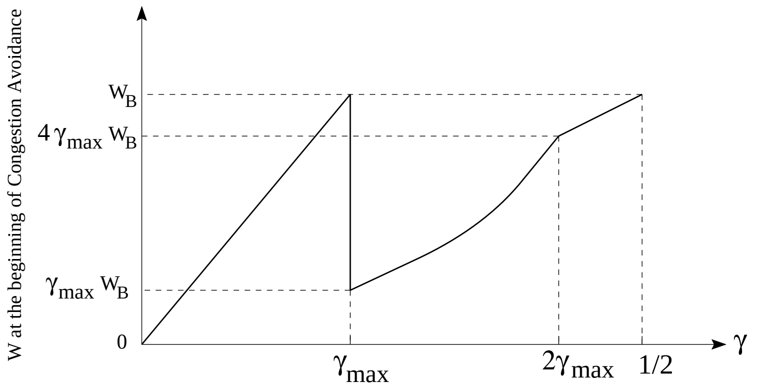

Figure 3: The window size at the beginning of congestion avoidance

a factor between 1/4 and γ_{max}, we get an improvement in TCP performance. This is how the reduction factor must be chosen so that our proposition works properly. To make clear this analysis, we plot in figure 3 the variation of the window size at the beginning of the congestion avoidance phase as a function of γ for the case $\gamma_{max} < 1/4$.

In figure 4, we show how the throughput varies for three values of β. The results are got using the NS simulator. The simulation model is described in section 5. For $\beta = 0.5$, we see always a performance degradation because a double slow-start problem doesn't exist ($\gamma_{max} > 1/2$). For the two other β, we notice an upward jump in the throughput for a certain γ, theoretically equal to γ_{max}. This increase is interesting for $\beta = 0.25$ given that $\gamma_{max} > 1/4$. However, for the smallest β, the fact that γ_{max} is less than 1/4 makes impossible to find a γ that gives a better throughput than that at $\gamma = 0.5$.

The implementation of such solution requires only modifications in the source algorithm. No changes are needed in the destination behavior. Some functions must be added at the source to estimate, on runtime, the network parameters, then the appropriate γ using equation 7. Estimation techniques as Packet-Pair [13] for bandwidth and RTT tracking [9] for buffer capacity can be adopted.

A possible solution could be to register the sequence number of packets sent at the beginning and at the end of every congestion avoidance phase, let X_1 and X_2 respectively. Let W be the window size when X_2 is registered. If $X_2 - X_1 < W$, this means that the loss detected has happened in the slow-start phase due to a buffer overflow. Therefore, a double slow-start problem exists and can be resolved. A simple analysis shows that $X_2 - X_1$ is equal to $2W_B - W$ and that W is simply the old factor (γ^{old})) times W_{max}. This lets us

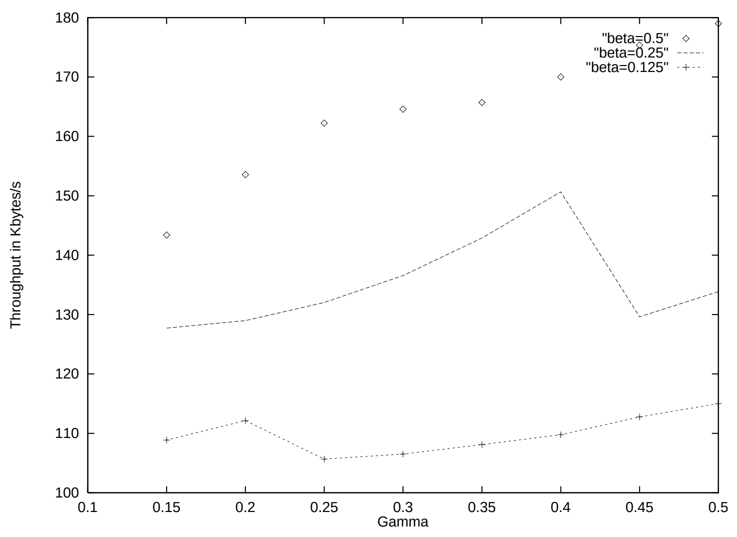

Figure 4: The effect of γ on the throughput

calculate the new factor that must be chosen:

$$\gamma_{max}^{new} = \frac{\gamma^{old}(X_2 - X_1 + W)}{2W} \tag{8}$$

Note that if the loss is detected in slow-start, this means that $W_{th} = \gamma^{old} W_{max} > 2W_B$, therefore $\gamma_{max}^{new} < 1/4$. According to our previous analysis, the throughput degradation due to double slow-start cannot be solved.

4 The second proposition: Spacing the packets

Our idea is to keep W_{th} unchanged and to increase W_B so that to get in congestion avoidance before loosing a packet during slow-start. Hence, the congestion avoidance phase starts at $W_{max}/2$ instead of $W_D/2$ yielding an improvement in TCP throughput. W_B is inversely proportional to the queue building rate, therefore to increase it, we slow the rate at which the TCP source sends the bursts of packets during slow-start. Let R be this rate. To reduce the queue length, R must be slower than $2\mu - \lambda$, the maximum sending rate during slow-start. Also, R must not be less than $\mu - \lambda$, the available bandwidth at the bottleneck. An R slower than $\mu - \lambda$ makes the source the bottleneck and changes also the behavior of TCP during congestion avoidance which must be avoided.

After decreasing R, a larger burst is needed to fill the buffer resulting in a larger W_B.

To find the needed rate, we repeat the analysis done in [7] but now with a constant transmission rate R. Note that changing R doesn't affect the window evolution as a function of time during slow-start because the source is still sending a burst of two packets in response to every received ACK. The only change is that a certain delay (1/R) is introduced between the packets transmitted even if the output interface of the source lets it send at higher rates. Thus, the mini-cycles still have a duration τ and W is still doubling at the end of each mini-cycle. Note that this assumption cannot be true if τ is not large enough to leave a certain time between the transmission of the last packet in a mini-cycle and the receipt of the first ACK in the next one. If this true, the length of a mini-cycle is no longer τ which results in a different window evolution. Due to the small buffering capacity compared to the bandwidth delay product of the network considered in our analysis, we are sure that this change in the duration of mini-cycles cannot occur and, therefore, the growth of W remains the same.

The total input rate at the bottleneck during bursts is $R + \lambda$. Therefore, the queue builds up at rate $R + \lambda - \mu$ and the TCP source must keep sending packets at rate R for a time $\frac{B}{R+\lambda-\mu}$ in order to fill the buffer. As in [7], the number of TCP packets sent during this time is taken as an approximation of the overflow window W_B. Hence,

$$W_B = \frac{BR}{R + \lambda - \mu} \tag{9}$$

We notice here that this window increases when R decreases for any network parameters since $\lambda < \mu$. Also, it moves to infinity when R tends to $\mu - \lambda$, thus making losses during slow-start a rare phenomenon. For a network satisfying equation 5, there is some R between $\mu - \lambda$ and $2\mu - \lambda$ below which we have always $W_B > W_{th}$. This R is given by

$$R < (\mu - \lambda)\frac{1 + \beta(1 - \rho)}{1 - \beta(1 + \rho)} = R_{max} \tag{10}$$

It is clear that any decrease in β due to a smaller buffer B or longer path reduces R_{max} making necessary further reduction in the transmission rate. For a certain μ, a variation of λ between 0 and μ makes R_{max} vary between $\mu\frac{1+\beta}{1-\beta}$ and 0. But, equation 9 shows that the minimum value of W_B when λ varies is B whatever is the value of R. Given that W_{th} moves to zero when λ increases, surely it falls below W_B at a certain point even if the transmission rate is too high. Thus, for networks operating with a high exogenous traffic it is not necessary to insert a delay between TCP packets.

The window growth isn't affected by the delay introduced. Therefore, any R satisfying (10) results in the same performance given that the windows at the beginning and at the end of the congestion avoidance phase don't change. This is illustrated in figure 5 where we show the throughput for β=0.5, 0.25 and 0.125 [4]. Unlike the first proposition, where a solution to

[4]In this figure, the difference in throughputs after the double slow-start disappearance, represented by the upward jump, is due to a difference in W_{max}.

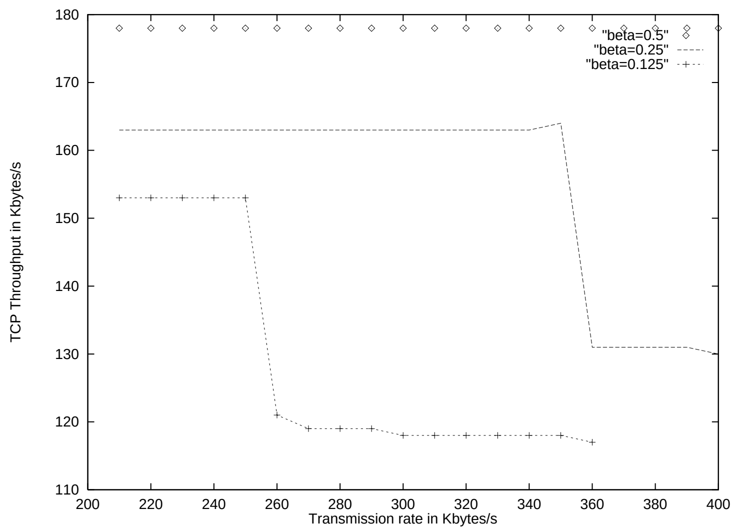

Figure 5: The effect of R on the throughput

the double slow-start phenomenon cannot be found for all the values of β, sending packets at a rate R close to $\mu - \lambda$ solves always the problem. However, transmitting always at exactly $\mu - \lambda$ avoids a queue building at the bottleneck even in congestion avoidance. Thus, a TCP packet isn't lost when $W = W_{max}$. The window continues growing until it reaches that advertised by the receiver. In this case, the number of outstanding packets doesn't exceed $\tau(\mu - \lambda)$. Indeed, even if W lets the source send more packets, the limited transmission rate forces it to wait a time $1/(\mu - \lambda)$. During this time an ACK arrives and the number of outstanding packets remains the same. The solution is to delay only packets sent during slow start and not to add any spacing between packets sent in congestion avoidance. This way, a transmission at $\mu - \lambda$ becomes possible and influences only the slow-start phase of TCP.

The implementation of this solution is easier than the first one since we don't need to follow equation (10). It is enough to transmit at the available bandwidth estimate (i.e. $\mu - \lambda$) to be sure to solve the problem of double slow-start. A solution could be to transmit one TCP packet for each ACK of the burst received in a mini-cycle, then to send the remaining packets at an average rate calculated from the incoming ACKs. The advantage of this proposition compared to the ACKs spacing proposed in [17] is that it does not require any intelligence in the routers nor in the destination behavior. It requires only minor changes at the source level.

5 Simulations

To validate our propositions, a set of simulations has been conducted using **ns** [16]. The network studied is that of figure 1. A one-way TCP connection, having an unlimited data to send and segments of total length 512 bytes, is established between the circles named *Source* and *Destination*. The Tahoe version of TCP is adopted at the source and the receiver is a simple sink acknowledging every incoming packet. The satellite link of capacity 1.5 Mbps (T1 link) and of delay 250 ms is crossed by a background traffic of average rate 100 packets/s. To get the same service time at the bottleneck, background packets have the same length as TCP segments. The TCP source is connected to the satellite via a high speed link of rate 10 Mbps and of delay 30 ms. The bottleneck node is supplied with Drop-Tail buffer of capacity 40 packets. If we express all the parameters in packets, we get

$$\mu = 366 \text{ packets/s}, \; \lambda = 100 \text{ packets/s}, \; B = 40 \text{ packets}, \; \tau = 0.56 \text{ s}.$$

A simple substitution of these parameters in equation 5 shows that a double slow-start must appear. This is illustrated in figure 6. The TCP throughput in this case is 161 packets/s which represents 60.5% of the available bandwidth. We notice here that we get a little in congestion avoidance before detecting the loss in the first slow-start. Therefore, $W_D = W_{th}$ and the congestion avoidance phase starts at one fourth the maximum window.

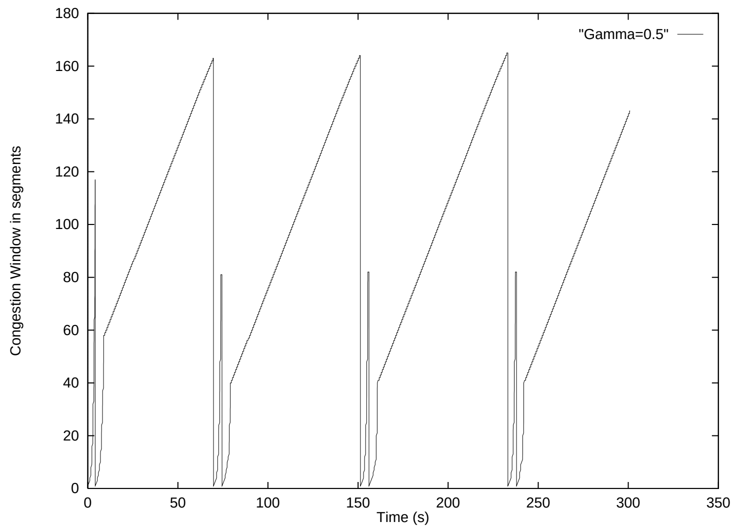

Figure 6: The double slow-start problem

To get rid of this problem, we first try to reduce the factor one half according to our first proposition. Equation 7 indicates that a γ less than 0.38 must be used. Because

$\gamma_{max} = 0.38 > 1/4$, a throughput improvement is possible in this scenario. Taking $\gamma = 0.34$, figure 7 shows well the disappearance of the double slow-start. Therefore, the throughput increases to 172 packets/s, hence 4% more bandwidth utilization. In figure 8, we reduce W_{max} by more than four times when a loss is detected. Although one slow-start is used per TCP cycle, the throughput decreases to 153 packets/s. In this case, it is better to keep the problem unsolved.

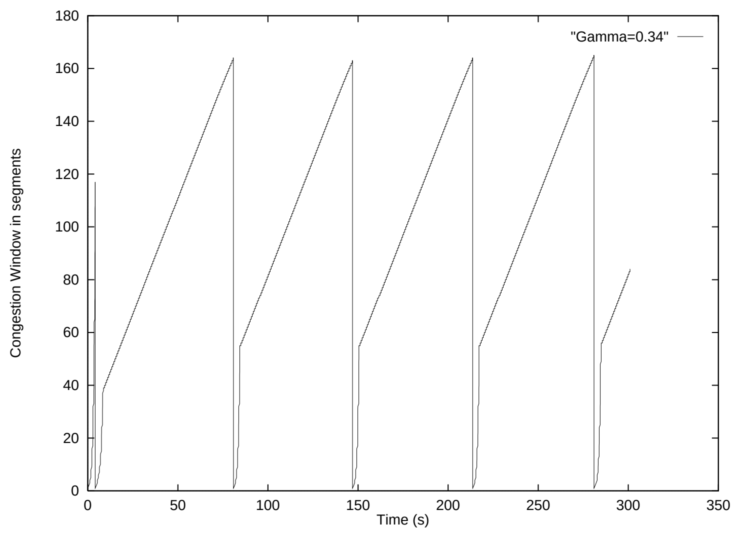

Figure 7: The case of a $1/4 < \gamma < \gamma_{max}$

We try now to improve the throughput by slowing the transmission. For the given parameters, equation 10 indicates that we must send at slower than 480 packets/s to avoid a buffer overflow during slow-start. We conducted two simulations with $R = 440$ and $R = 300$. The figures 9 and 10 show the corresponding results. We remark that these two transmission rates solve the double slow-start problem and give the same window at the beginning of the congestion avoidance phase. As we have said, the throughputs are the same and are equal to 195 packets/s. This gives better performance than the first proposition even for $\gamma \simeq \gamma_{max}$. Indeed, instead of starting at γW_{max} as in the first case, the congestion avoidance, where the bulk of data is transfered, starts at $W_{max}/2$ after the spacing of the packets. Note here that, in all the figures, W_{max} keeps the same value which means that our propositions have no influence on TCP behavior during congestion avoidance.

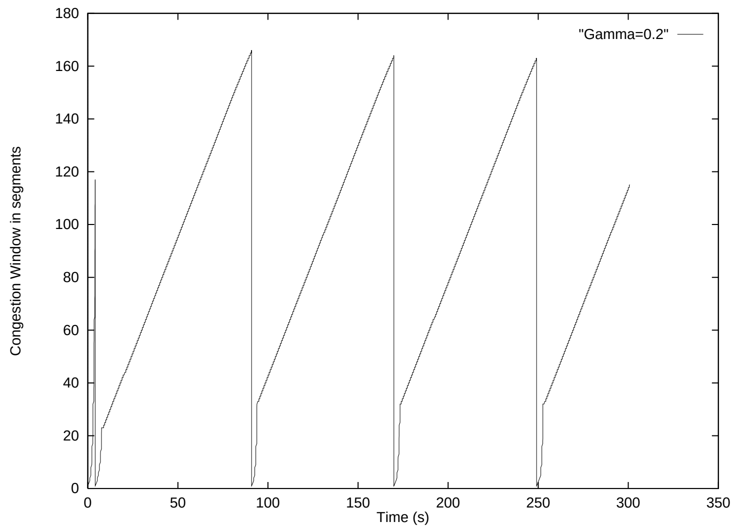

Figure 8: The case of a $\gamma < 1/4$

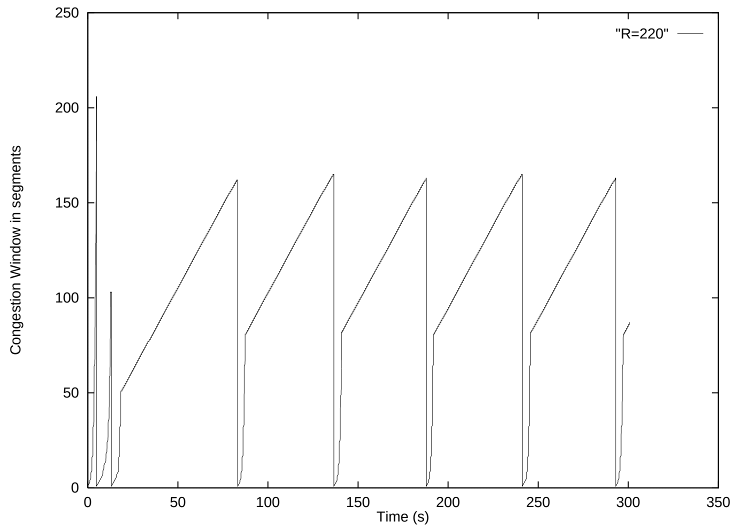

Figure 9: The case of $R = 440$ Packets/s

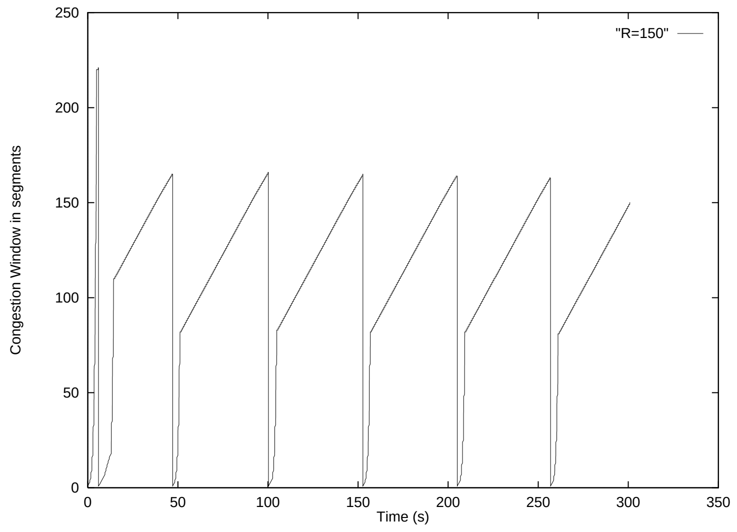

Figure 10: The case of $R = 300$ Packets/s

6 Conclusions

In this paper, we introduced two modifications to standard TCP in order to avoid the losses that appear when the network buffers are not large enough to absorb the bursts of packets sent during slow-start. These bursts make the buffer overflow early resulting in a deterioration in TCP throughput. The problem becomes more pronounced with the prevalence of large bandwidth-delay product networks such as those including GEO satellite links.

Such losses can be avoided if the source gets in congestion avoidance before the buffer overflow. First, we proposed to change the factor one half used in TCP to calculate the threshold of the slow-start following a loss detection. Second, we proposed to space the TCP packets transmitted so that to slow the queue building rate at the bottleneck and then to push farther the losses.

Using mathematical analyses and simulations, we studied the effect of these modifications on TCP performance. For the first one, we found that, although it is able to avoid the losses, it doesn't lead always to throughout improvement. The reduction factor must be chosen carefully in function of network parameters and in some cases it is better to keep the double slow-start unsolved. However, the second modification is less complicated and leads always to a better performance. We showed that transmitting during slow-start at a rate close to the available bandwidth moves to zero the queue building rate at the bottleneck without deteriorating the throughput.

Such solutions require adaptive algorithms at the source to track the network dynamics. The implementation can be easier if some informations are fed back by the network and the receiver to the source. As an example, supplying the network with RED (Random Early Detection) buffers with ECN (Explicit congestion Notification) capacity gives the source an idea on the available buffering space.

Additional work is needed to study the utility of these propositions with the other versions of TCP. The double slow problem occurs when TCP fails to recover from multiple losses in a Window. Thus, such study must consider the bursty errors in satellite links. Also, because random losses forces TCP to stop its congestion avoidance phase at a window less than W_{max}, an analysis must be conducted to evaluate the effect of this event on the performance of our propositions. These random losses affect mainly the first one.

References

[1] The ATM Forum Technical Committee, Traffic Management Specification, 1996.

[2] M. Allman, D. Glover, J. Griner, K. Scott, and J. Touch,"Ongoing TCP research related to satellites", Internet Draft, 1998, Work in Progress.

[3] M. Allman,"On the Generation and Use of TCP Acknowledgments", ACM Computer Communication Review, 28(5), October 1998.

[4] M. Allman, S. Floyd, C. Partridge, "Increasing TCP's Initial Window", September 1998, RFC 2414.

[5] M. Allman and D. Glover, "Enhancing TCP over satellite channels using standard mechanisms", Technical report, NASA Lewis, 1998.

[6] M. Allman, H. Kruse, S. Ostermann,"An Application-Level Solution to TCP's Satellite Inefficiencies", Proceedings of the First International Workshop on Satellite-based Information Services (WOSBIS), Rye, New York, November 13, 1996.

[7] E. Altman, J. Bolot, P. Nain, D. Elouadghiri- M. Erramdani, P. Brown, D. Collange,"Performance Modeling of TCP/IP in a Wide-Area Network", INRIA-France, Research Report N=3142, Mars 1997 (available in http://www.inria.fr:80/RRRT/RR-3142.html). A shorter version in 34th IEEE Conference on Decision and Control, December 1995, New Orleans, Louisiana, (invited paper), pp. 368-373.

[8] R. Braden, "Requirements for Internet Hosts - Communication Layers", October 1989, RFC 1122.

[9] L. Brakmo and L. Peterson, "TCP Vegas: End-to-End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communications, Vol. 13, No. 8, pp. 1465-1480, Oct. 1995.

[10] J. Hoe, “Improving the Start-up Behavior of a Congestion Control Scheme for TCP”, SIGCOMM Symposium on Communications Architectures and Protocols, Aug. 1996.

[11] V. Jacobson, “Congestion avoidance and control”, Proc. ACM Sigcomm’88, Stanford, CA, USA, Aug. 1988.

[12] V. Jacobson, R. Braden, and D. Borman, “TCP Extensions for High Performance”, May 1992, RFC 1323.

[13] S. Keshav, “A control-theoretic approach to flow control”, in Proc. SIGCOMM’91 Symp., Sept. 1991, pp. 3-15.

[14] 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, pp. 336-350, June 1997.

[15] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow,”TCP Selective Acknowledgment Options”, October 1996, RFC 2018.

[16] S. McCanne and S. Floyd, “NS (Network Simulator)”, 1995. URLs http://www-nrg.ee.lbl.gov/ns, http://www-mash.cs.berkeley.edu/ns.

[17] C. Partridge, “ACK Spacing for High Delay-Bandwidth Paths with insufficient Buffering”, Internet Draft, Sep. 1998, Work in Progress.

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

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399