Analysis of TCP in Networks with Small Buffering Capacity and Large Bandwidth-Delay Product

Chadi Barakat, Eitan Altman

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Analysis of TCP in Networks with Small Buffering Capacity and Large Bandwidth-Delay Product

Chadi Barakat — Eitan Altman

N° 3574

December 1998

THÈME 1

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

Analysis of TCP in Networks with Small Buffering Capacity and Large Bandwidth-Delay Product

Chadi Barakat , Eitan Altman

Thème 1 — Réseaux et systèmes
Projet Mistral

Rapport de recherche n° 3574 — December 1998 — 18 pages

Abstract: It is well known that inefficiencies occur in the operation of TCP when the bandwidth delay product of the network is large compared to its buffering capacity. A central reason for that is a cyclic behavior of the protocol in which two consecutive slow-start phases appear in each cycle [5, 1]. This results in low throughput and under-utilization of the available bandwidth. We show in this paper that, for an even larger ratio between the bandwidth-delay product and the buffer size, which is typical for satellite links, a new regime of three consecutive slow-start phases occurs. We need to refine previously studied models in order to explain this phenomenon. We analyze in this paper this phenomenon, predict when it occurs and evaluate the corresponding average throughput of the connection.

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

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

Analyse de TCP dans des réseaux ayant des petits buffers et un grand produit délai-bande passante

Résumé : Il est connu que le fonctionnement de TCP souffre d'une certaine inefficacité lorsque le produit délai-bande passante du réseau est large par comparaison à la capacité de ses buffers. La cause principale de ce problème est un comportement cyclique du protocole avec l'apparition de deux phases slow-start consécutives dans chaque cycle TCP. Ceci résulte en une réduction de l'utilisation de la bande passante disponible. Dans ce papier, on montre que, pour des plus grands rapports entre le produit délai-bande passante et la taille du buffer ce qui est typique pour les liens satellites, un nouveau régime de trois phases slow-start consécutives apparaît. On est obligé d'améliorer les modèles déjà étudiés pour pouvoir interpréter ce phénomène. On analyse dans ce travail ce problème et on prévoit quand est ce qu'il peut apparaître tout en évaluant son effet sur le débit de la connexion.

Mots-clés : TCP, Slow-Start, Congestion Avoidance, Modélisation, Simulations, Évaluation de Performance.

1 Introduction

Most of today Internet applications use TCP to control the flow of their packets and to recover from any information loss. The flow control is based on a congestion window W which limits the maximum number of unacknowledged packets a source can transmit. The receiver sends back cumulative acknowledgments (ACK) to inform the source of the sequence number of the last in-order packet received. At the source, these ACKs trigger the transmission of new packets and provide it with information on the network state. Any packet loss is considered as a congestion indication forcing the source to decrease multiplicatively its window. As long as no losses are detected, the window increases linearly resulting in an increase in TCP throughput. This reaction to losses yields a performance degradation when packets are lost due to corruption not congestion. This problem becomes more prevalent with the insertion in the Internet of wireless and satellite links characterized by a high bit error rate. Several schemes have been proposed to decouple the congestion control of TCP from its error control. They aim to let TCP retransmit the errors without reducing considerably the throughput.

TCP uses two algorithms, slow-start and congestion avoidance [4], to adapt its window to the network state. We assume that W is measured in segments not in bytes. Starting at $W = 1$, slow-start is used to increase W by one segment for every incoming ACK until we reach a threshold W_{th} considered as an estimation of the network capacity. This exponential growth in W prevents the sender from overwhelming the network with an inappropriately large burst of traffic. After W_{th}, the source moves to congestion avoidance where W is increased by 1 segment for every window's worth of acknowledged packets, hence every Round Trip Time (RTT). This slower growth aims to probe the network for extra bandwidth and it continues until a packet is lost. The loss is detected either by timeout or by the receipt of four consecutive ACKs carrying the same sequence number (3 Duplicate ACKs). The second way is called the fast retransmit algorithm [9]. The source assumes that the network gets congested and considers half the window at which the loss is detected (W_{max}) as a good estimation of the available capacity, so it sets $W_{th} = W_{max}/2$. The next action depends on the version of TCP. The Tahoe version is conservative. It supposes that the network is severely congested and thus, it resorts always to slow-start. The Reno version distinguishes between the two ways of loss detection. It keeps the same behavior as Tahoe in case of timeout. However, it considers the receipt of Duplicate ACKs as an indication of light congestion and, therefore, it starts directly a new congestion avoidance phase after recovering from the losses [9]. Although this new behavior is intended to avoid the long time taken by slow-start, it performs badly when multiple losses occur in the same window [2]. A long timeout is required to recover from these losses followed by a slow-start which we try to avoid. This results in a poor performance when TCP Reno operates in networks with lossy links. Selective Acknowledgment (SACK) [6] has been proposed to overcome this deficiency. The receiver feeds the source with a list of packets correctly received. This information helps it to retransmit only the gaps in the receiver buffer while maintaining enough packets in the network in order to not deteriorate the utilization.

A critical parameter in TCP performance is the slow-start threshold. A low W_{th} reduces the throughput by making the source spend a long time to reach the available capacity W_{max}. This is due to the slow growth of the window during congestion avoidance. However, because of the high transmission rate during slow-start (the source sends two packets in response to an incoming ACK which yields a transmission rate twice the available bandwidth), a high threshold may overflow prematurely the network buffers leading to an underestimation of the network capacity. Starting directly a new congestion avoidance after a loss, as in TCP Reno, avoids the slow-start and then the undesirable buffer overflow. But, as we have said, with lossy links it is very likely to pass to a new slow-start. This is always the case with TCP Tahoe. Thus, an understanding of this interaction between W_{th}, network buffers and throughput during slow-start is required.

The threshold of the first slow-start is given a default value at the beginning of the connection. For subsequent slow-starts, it is estimated on runtime in function of network parameters. This estimate increases with the bandwidth-delay product of the network. A well known problem appears when W_{th} is set too large so that the buffers overflow before getting in congestion avoidance at W_{th}. Multiple losses occur forcing TCP to reduce its window and to call an new slow-start with a new threshold less than the network capacity estimate. This problem of double slow-start preceding the congestion avoidance phase has been already identified and analyzed in [5, 1, 8]. It can be seen in any slow-start phase if the buffer capacity doesn't scale with the bandwidth-delay product of the network. It is typical in satellite links, because of their long propagation delay and their bursty losses which make TCP versions resort to slow-start to recover from errors. Also, it can be seen in the first slow-start of the connection. A bad choice of the default value affects the performance especially for short live connections. To reduce this effect, Hoe in [3] proposes a mechanism to find a more appropriate initial threshold.

Our experience in simulations showed that two consecutive slow-start phases are very common in satellite links. However, we also obtained very frequently a phenomenon of a cyclic evolution of TCP window containing *three consecutive slow-start phases* per cycle. This phenomenon had already been observed in the context of terrestrial links in [1], in which it occurred quite rarely. The goals of this paper are to understand and analyze this phenomenon, to predict when it would occur, and to evaluate its effect on TCP throughput.

The importance in understanding and then in avoiding this phenomenon is in its negative impact on the performance of the TCP connection. Indeed, since W is divided by two at the end of each of the slow-start phases, the congestion avoidance starts at a very small window, and it takes a long time for the connection to recover from the last loss (i.e. to increase the window size and thus the throughput to the available bandwidth). This means that the average throughput that is obtained is quite low.

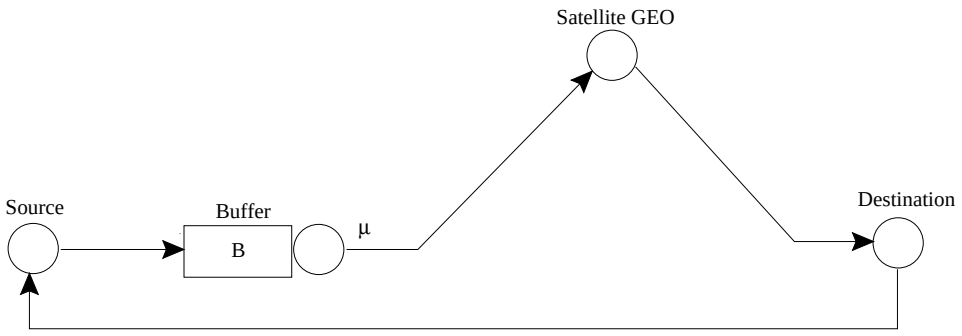

Figure 1: The Network Model

In this work, we use an approach similar to [1] to analyze this phenomenon. Their model is refined here in order to allow us to interpret the buffer overflow in the second slow-start phase. The effect of this behavior on TCP performance and its occurrence are studied. Then, the results are validated by a set of simulations using **ns**, the Network Simulator [7].

In the next section we outline the network model used in our analysis. We show how the fluid approach used in [1] to model a TCP connection isn't able to interpret the problem. In section 3, we present the approach we used to analyze the phenomenon. Section 4 contains the mathematical analysis. In section 5 and 6, we study the occurrence of the triple slow-start. In Section 7 the effect on TCP performance is studied. Section 8 presents the simulation and the results and finally we conclude the work in section 9.

2 The network model

The network is modeled as a single bottleneck node of bandwidth μ and of buffer capacity B having a FIFO service discipline. A TCP source running the Tahoe version is connected to the bottleneck via a high speed link. The bottleneck node routes TCP packets via a satellite link to the destination where they are acknowledged. We assume that the source has always packets to send and that the receiver acknowledges every new packet. The ACKs return to the source across a non congested path. The round trip time RTT is modeled as the sum of the propagation delay τ, the service time at the bottleneck $1/\mu$ and the waiting time in the buffer. The different parameters of the model are shown in Figure 1.

We suppose that additional mechanisms have been added to the satellite link to make it reliable (High transmission power, FEC, Link Level retransmission). Therefore, the window size at the end of congestion avoidance W_{max} represents the maximum number of TCP packets that can be fit into the pipe and the buffer. We take for this window the value

found in [1]

$$W_{max} = B + \mu\tau + 1 \tag{1}$$

After the detection of the loss due to a window size W_{max} exceeding the network capacity, a new slow-start is initiated with a threshold $W_{th} = W_{max}/2$. What happens when the buffer is not large enough is that it overflows before reaching W_{th} leading to a multiple losses in a window. When these losses are detected, a new slow-start begins with a threshold less than W_{th}.

To understand this premature overflow, the slow-start is divided into mini-cycles of duration RTT [5, 1]. At the beginning of mini-cycle n, a burst of 2^{n-1} ACKs arrives at the source at rate μ and triggers the transmission of a burst of 2^n TCP packets at rate 2μ. Thus, the queue at the bottleneck builds up at rate μ (the difference between the input rate 2μ and the output rate μ). The buffer overflows when the burst of TCP packets sent in a mini-cycle is sufficiently large to make the queue at the bottleneck exceed B. This happens at a window W_B given in [1] by

$$W_B = 2B \tag{2}$$

Hence, to get a loss in slow-start, we must have $W_B \leq W_{th}$. Let β be the normalized buffer capacity, therefore, this condition can be written as [5, 1]

$$\beta = \frac{B}{\mu\tau + 1} \leq \frac{1}{3} \tag{3}$$

It has been supposed in [1] that these losses are immediately detected at a window W_B. Thus, the threshold of the next slow-start is taken equal to $W'_{th} = W_B/2$. In fact, we must wait a complete RTT before the receipt of three Duplicate ACKs announcing the overflow. During this time, the source receives W_B new ACKs. The window grows then from W_B to a value W_D depending on the position of W_{th} with respect to W_B and $2W_B$. The next slow-start gets $W'_{th} = W_D/2$ as a threshold. 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

$$W_D = min(W_{th}, 2W_B) \tag{4}$$

This fluid approach isn't able to interpret the three consecutive slow-start that may appear in a TCP cycle. Indeed, as we have said, a loss occurs in slow-start if W_B is less then W_{th}. According to [1], the second slow-start threshold $W'_{th} = W_B/2$ is so low so that the source gets in congestion avoidance before filling the buffer, therefore, a third slow-start isn't possible. Even if we adopt our more detailed description of the window evolution after the first loss (i.e. the introduction of W_D), this phenomenon also cannot be interpreted. Equation (4) shows that $W'_{th} = W_D/2$ cannot exceed W_B. Thus, it is supposed that a congestion

avoidance follows always the second slow-start.

To analyze these losses leading to a third slow-start, we descended at the packet level and we tried to understand the behavior of the source, the receiver and the queue. We found that the problem with the previous models is that they analyze the second slow-start as the first one. They suppose that the source sends at twice the ACKs rate and that the window doubles at the end of each mini-cycle. However, some of the packets sent between the first loss and its detection were correctly received by the destination and are stored in the receiver buffer. While acknowledging the packets sent during the second slow-start, the content of this buffer is taken into consideration; a packet already received in the first slow-start is directly acknowledged without asking the source to retransmit it. This leads to a different window evolution as described below. Also, these models estimate the window overflow W_B as the size of the burst needed to fill the buffer during slow-start. This can be true if the buffer capacity is a power of two. For the other B, a more precise value of W_B must be considered. In the following section, we describe our observations and present the refinements we added to this fluid approach.

3 The refined fluid approach and conditions for three consecutive slow-start phases

Suppose that the buffer capacity is so small so that a loss occurs during slow-start. In a normal slow-start, the window doubles at the end of each mini-cycle. Let n_B be the number of the mini-cycle in which the first loss occurs. At the beginning of n_B, the sender receives a stream of 2^{n_B-1} ACKs at rate μ. The congestion window is then equal to 2^{n_B-1}. Each incoming ACK triggers the sending of a burst of two packets. Due to the queue building rate μ, the buffer overflows when a burst of TCP packets of size $2B$ is sent. The window size when this overflow happens is equal to 2^{n_B-1} plus the number of ACKs needed to trigger the burst. Therefore,

$$W_B = 2^{n_B-1} + B \tag{5}$$

It is equal to that found in [1, 5] if we approximate 2^{n_B-1} by B. For a given B, n_B is chosen so that

$$2^{n_B-2} < B \leq 2^{n_B-1} \tag{6}$$

Because the buffer gets full, the packets sent between the overflow and the end of mini-cycle n_B are partially dropped. These packets are of total number $2^{n_B} - 2(B-1)$. Those who succeed to find a place in the bottleneck buffer are stored at the receiver waiting for the lost packets to be retransmitted. The first loss is detected in mini-cycle $n_B + 1$. This happens after the receipt of the ACKs of packets sent in cycle n_B before the overflow. If we suppose that the threshold W_{th} is greater than $2W_B$, the number of packets sent in cycle $n_B + 1$, before the loss detection is $4(B-1)$. The first half of these packets fills the buffer and is correctly received and the second half is again partially dropped.

After the retransmission of the first loss, the receiver asks for the gaps in his buffer. When a missing packet arrives, it sends an ACK covering the packets correctly received until the next gap. However, it ignores the incoming packets which already exist in its buffer. The sender, having no idea on the packets waiting at the receiver (those packets have been sent between the first loss and its detection), sends the gaps but also some unnecessary packets (This problem could be avoided if the TCP SACK option [6] is used). This behavior leads to an evolution of the window which is different than a normal slow-start operation. This new evolution is analyzed in the next section. We are now ready to state our results

Theorem 1 *If $W_{th} > 2W_B$ then*

Let n_1 be the smallest integer satisfying

$$4 + \frac{3}{2}n_1(n_1+1) > 2^{n_B} - 2(B-1). \tag{7}$$

Let n_2 be the smallest integer satisfying

$$S[(\frac{3}{2})^{n_2+1} - 1] > B - 1. \tag{8}$$

If either

$$2^{n_B} - \frac{3}{4}n_1(n_1-1) > 3B - 1 \tag{9}$$

or if

$$(2^{n_B-1} - B + 2)[2(\frac{3}{2})^{n_2} - 1] > 2B - 1 \tag{10}$$

hold then there are three consecutive slow-start phases in each cycle.

Theorem 2 *The triple slow-start problem doesn't exist in case of $W_{th} < 2W_B$.*

In the next section, we establish the proof of Theorem 1. The feasibility of the conditions is studied in section 5. Theorem 2 is proved in section 6 where we show that a necessary condition to get three consecutive slow-start phases is

$$\frac{B}{\mu\tau + 1} < \frac{1}{7}$$

4 The mathematical analysis: Proof of Theorem 1

As we have said, we study in this section and the next one the case $W_{th} > 2W_B$. Because the source sends at twice the bottleneck rate μ (during bursts in slow-start [5, 1]), half of the packets sent after the buffer overflow are lost. Let 0 be the sequence number of the first packet lost. Therefore, the packets sent during the first slow start between the loss of 0 and its detection are divided into three parts:

(i) The packets numbered $0, 2, \ldots, 2^{n_B} - 2(B-1) - 2$ are lost and those numbered $1, 3, \ldots, 2^{n_B} - 2(B-1) - 1$ are correctly received.

(ii) The packets from $2^{n_B} - 2(B-1)$ to $2^{n_B} - 1$ are correctly received.

(iii) The packets numbered $2^{n_B}, 2^{n_B}+2, \ldots, 2^{n_B}+2(B-1)-2$ are lost and those numbered $2^{n_B}+1, 2^{n_B}+3, \ldots, 2^{n_B}+2(B-1)-1$ are correctly received.

The source restarts by retransmitting packet 0. When 0 arrives at the receiver, an ACK is sent asking for packet 2. This ACK increases the window to 2 and the source sends two packets 2 and 3. When packet 2 is received, the receiver asks for packet 4. However, packet 3 isn't acknowledged (even if it is acknowledged, the corresponding ACK is considered as a Duplicate ACK at the source and, hence, it is ignored). When the ACK of 2 arrives at the source, the window becomes three and three packets are sent (4, 5 and 6). If we continue like this, we find that the transmission of packets half received is also accomplished in mini-cycles. Each incoming ACK triggers the transmission of three packets and the size of the burst of packets sent increases by three at the end of each mini-cycle. Except the first two cycles, this size is always a multiple of three. Also, all the sequence number are retransmitted (except 1). Thus, half the packets of a burst are acknowledged. TCP packets arrive at the destination at rate μ and ACKs return to the source at rate $\mu/2$. These ACKs trigger the transmission of packets at rate $3\mu/2$. To overflow the buffer at this rate, the source must send $3B$ consecutive packets. This requires B ACKs and then $2B$ packets in the previous mini-cycle. Because the total number of packets of the first part is less than $2B$ (using equation (6)), no buffer overflow occurs due to this new sending rate.

This cyclic behavior continues until the source sends a burst formed of packets of (i) and (ii). Let $N_1 = 3n_1$ be this burst size. The total number of packets sent until the end of the mini-cycle $3(n_1 - 1)$ is

$$4 + 3 + 6 + \cdots + 3(n_1 - 1) = 4 + \frac{3}{2} n_1 (n_1 - 1) \tag{11}$$

The 4 represents the first two cycles. Thus, n_1 is the smallest integer satisfying (7).

The number of packets in the burst N_1 belonging to (ii) (N_1^2) is equal to the left side of equation (7) minus its right side. The remaining packets come from (i) (N_1^1). Their number is equal to the right side of equation (7) minus that of equation (11).

When these N_1 packets reach the receiver, $N_1^1/2$ ACKs at rate $\mu/2$ are sent in response to packets from (i). The others don't generate any ACKs because they have been already received. The last ACK sent asks for the first packet of (iii).

When arriving back, the $N_1^1/2$ ACKs make the sender inject $3N_1^1/2$ packets from which one third stays in the buffer. The last ACK arrival increases the congestion window to the number of packets of (i) which were not received during the first slow-start. The left edge

of the window advances quickly to the beginning of (iii) and a burst of packets is sent. This burst size is simply

$$S = 2^{n_B-1} - (B-1) + 1 = 2^{n_B-1} - B + 2 \tag{12}$$

Because S is smaller than $2(B-1)$, the burst contains only packets from (iii). These S packets can overflow the buffer if

$$2^{n_B-1} - B + 2 + N_1^1/2 > B$$

This yields the first condition (9) for a loss in the second slow-start.

Suppose now that this condition isn't satisfied. No Acks are sent until the reception of the first packet of burst S. Because half of the burst is in the receiver buffer (it belongs to (iii)), $S/2$ ACKs are sent back. These ACKs trigger the transmission of $3S/2$ TCP packets. Again, a cyclic behavior appears which ends when the source sends a burst containing packets transmitted for the first time. This burst size can be written as

$$N_2 = (\frac{3}{2})^{n_2} S \tag{13}$$

Losses cannot occur before this mini-cycle. Indeed, a burst of 3B packets is needed to fill the buffer when the source transmits at rate $3\mu/2$. But, the total number of packets in (iii) is less than 3B. Therefore, a burst formed only of packets from (iii) isn't so large to cause a buffer overflow. To calculate N_2, we proceed as follows. The total number of packets sent until the end of the previous mini-cycle $(n_2 - 1)$ is

$$S[1 + \frac{3}{2} + (\frac{3}{2})^2 + \cdots + (\frac{3}{2})^{n_2-1}] = 2S[(\frac{3}{2})^{n_2} - 1]$$

Therefore, n_2 (and then N_2) is the smallest integer that satisfies (8).

This burst of N_2 packets leaves the bottleneck at rate μ. It contains N_2^3 packets from (iii). The remaining packets are sent for the first time. N_2^3 is equal to

$$N_2^3 = 2(B-1) - 2S[(\frac{3}{2})^{n_2} - 1] \tag{14}$$

As N_1^1, the N_2^3 packets result in $N_2^3/2$ ACKs which trigger $3N_2^3/2$ packets at the source. One third of these packets stays in the buffer. However, each packet sent for the first time triggers one ACK. Here, we switch to a normal slow-start where ACKs are sent back at a rate μ. At the sender, when the ACK acknowledging all packets sent in the first slow start arrives, the congestion window is equal to 2^{n_B-1}. This window is below the slow-start threshold ($W'_{th} = W_B$) and then we are still in slow start. To get a triple slow-start, the ACK stream sent in response to the N_2 burst must overflow the buffer otherwise we enter the next mini-cycle with a window greater than 2^{n_B-1}. Because in a normal slow-start a burst of $2B$ packets is required to fill the buffer, such window makes the sender get in congestion avoidance in the next mini-cycle before a loss occurs.

As in a normal slow-start, half of the packets triggered by the ACKs arriving at rate μ stays in the buffer. These packets of number $N_2 - N_2^3$ wait after those of number $N_2^3/2$. To get a buffer overflow, the total number of those waiting packets must be greater than the buffer capacity B

$$N_2 - N_2^3/2 > B$$

This way, we get the second condition (10) to make a loss occur during the second slow-start.

5 Feasibility of the conditions for three slow-start phases

The only parameter figuring in the previous analysis is the buffer size B. Indeed, taking $W_{th} > 2W_B$ eliminates the effect of the network capacity μ and the propagation delay τ on the window variation during slow start. The first slow-start ends at $2W_B$ and the second one starts with a threshold $W'_{th} = W_B$ whatever are μ and τ. Thus, the understanding of the above conditions requires only a variation of the buffer capacity B. The effect of μ and τ, which figures when $W_B < W_{th} < 2W_B$, is introduced in the next section.

Using equation (6), the set of B can be divided into intervals having the same n_B. Moreover, using the definition of n_1 (resp. n_2), each n_B interval can be divided further into subintervals having the same n_1 (resp. n_2). The study of the above conditions is done by increasing B inside n_2 and n_1 intervals.

For the first condition (9), it is clear that any increase in B inside an n_1 interval makes us closer to the end of this condition if it is satisfied and farther from it otherwise (n_1 and n_B don't change). Also, we can easily show that when B is on the left edge of any n_1 interval (i.e. the smallest B), equation (9) cannot be true. Hence, the transmission of the burst S at the end of (i) doesn't cause a buffer overflow and any triple slow-start must be interpreted by the second condition.

As in the previous case, any increase in B inside an n_2 interval pushes us farther from condition (10). However, a study of this condition on the edges of the interval shows that it cannot be neither always satisfied nor always unsatisfied. Indeed, using equation (8), we can define an n_2 interval as the set of B satisfying

$$(\frac{3}{2})^{n_2} < \frac{B-1}{S} + 1 < (\frac{3}{2})^{n_2+1} \tag{15}$$

Because the middle term is an increasing function of B, we can approximate the two edges by the value of B solution of these equations:

$(\frac{3}{2})^{n_2} = \frac{B-1}{S} + 1$ for the left edge

$(\frac{3}{2})^{n_2+1} = \frac{B-1}{S} + 1$ for the right edge

If we substitute $(\frac{3}{2})^{n_2}$ by its value in equation (10), we find that the second condition is always satisfied on the left edge (we have always S>1). This means that the triple slow-start appears in each n_2 interval. For the right edge, substituting $(\frac{3}{2})^{n_2}$ by its value shows that the condition gets satisfied if $S > 2B + 1$. This isn't true because, using equations (6) and (12), we have always $S < B + 2$. Thus, the triple slow-start doesn't exist on the right edge and surely it disappears at some point inside interval n_2.

Given that the n_2 intervals are contiguous, we conclude that the triple slow-start appears in a cyclic manner when B increases from 1 to the value corresponding to $W_{th} = 2W_B$. The length of a cycle is calculated by equation (15). It appears at the beginning of the cycle and disappears at some point inside. This point is given by the equality of the two sides of equation (10).

6 The case of $W_{th} < 2W_B$: Proof of Theorem 2

First, a $W_{th} < W_B$ avoids a loss in the first slow start and, thus, it eliminates completely the triple slow-start problem. It remains to study the case of $W_B < W_{th} < 2W_B$ to complete the proof.
In this latter case, the loss in the first slow-start is detected at a window equal to W_{th} and, therefore, the threshold of the second slow-start is set to $W'_{th} = W_{th}/2$. This new threshold is less than W_B, then it is more likely that the TCP source gets in congestion avoidance in the second slow-start before overflowing the buffer even if the condition (10) is satisfied. In fact, in this case, the source stops to send bursts of packets at some point between the loss in the first slow-start and its detection. This is due to getting in congestion avoidance before detecting the loss ($W_{th} < 2W_B$). Thus, the three parts (i, ii and iii) describing the content of the receiver buffer at the beginning of the second slow-start change. They disappear from right to left (from packet $2^{n_B} + 2(B-1) - 1$ to 0) when W_{th} moves from $2W_B$ to W_B. The problem becomes simpler due to an earlier switch to a normal slow-start when (iii) ends. Because W'_{th} is less than W_B, a normal slow-start cannot cause a buffer overflow. To study the problem in its worst case, we keep the three parts unchanged and we simply add a third condition to those of Theorem 1.

The window size when the buffer overflows must be smaller than W'_{th}. The ACK of the last packet of (iii) makes the window equal to $2^{n_B-1} + 1$. When the next ACK arrives (it acknowledges a packet sent for the first time), $N_2^3/2$ packets are waiting in the buffer. Thus, a burst of $B - N_2^3/2$ ACKs arriving at rate μ is needed to produce a loss. This burst increases the window to $2^{n_B-1} + 1 + B - N_2^3/2$. The new condition is then

$$W'_{th} > 2^{n_B-1} + B - N_2^3/2 + 1$$

Substituting W'_{th} by its value in function of μ and τ ($W'_{th} = W_{max}/4 = (B + \mu\tau + 1)/4$), we get the following conditions to make the triple slow-start appear

$$\left.\begin{array}{l}(2^{n_B-1} - B + 2)[2(\frac{3}{2})^{n_2} - 1] > 2B - 1 \\ 4(2^{n_B-1} - B + 2)(\frac{3}{2})^{n_2} < \mu\tau - 3B + 1\end{array}\right\} \quad (16)$$

To understand the importance of the problem in this case, a study of the new condition is required. As in the previous section, the set of B giving a W_{th} between W_B and $2W_B$ is divided into intervals and subintervals. In contrast to the first one, we get closer to the new condition when B increases in a n_2 interval (the left side of the condition decreases faster than the right one). A simple calculus shows that this new condition cannot be satisfied on the left edge of an n_2 interval (we use the fact that $W_{th} < 2W_B$ which leads to $\mu\tau+1 < 2^{n_B+1}+3B$). However, on the right edge, there is not an unique behavior but we can easily see that we stay close to the condition. Thus, if this new condition can be satisfied, this happens just before the end of the interval. Combining these two conditions with different behaviors, we conclude that the triple slow-start isn't very pronounced in this case and it can be ignored. The possible appearance at the beginning of an n_2 interval is discarded by the new condition.

As a result, the problem exists only when the buffer capacity is so small so that the slow-start threshold W_{th} is located above twice the overflow window W_B. If we use the approximation of 2^{n_B} by $2B$ adopted in [5], we get an upper bound on the required condition, and then a necessary one, to see a cyclic triple slow-start phenomenon

$$\mu\tau + 1 < 7B \Longrightarrow \beta = \frac{B}{\mu\tau + 1} < \frac{1}{7}$$

The lower bound on this condition is get when B is equal to the left side of equation (6). Substituting $B = 2^{n_B-2}$ in $W_{th} > 2W_B$, we get

$$\mu\tau + 1 < 11B \Longrightarrow \beta = \frac{B}{\mu\tau + 1} < \frac{1}{11}$$

7 Throughput approximation

Understanding how and where the triple slow-start occurs, we can now evaluate its effect on TCP throughput. Due the small size of network buffers, the waiting time can be neglected in front of the propagation delay. Thus, RTT can be considered as always constant which yields a linear growth of the window during congestion avoidance. Also, we ignore the packets sent during slow-start. Given the window size at the beginning (W_S) and at the end (W_E) of the congestion avoidance phase, the throughout can then be approximated by

$$Throuput = \frac{W_S + W_E}{2RTT}$$

We are interested in the maximum variation of the throughput due to the cyclic behavior of the triple slow-start problem. Of course, we are under the condition $W_{th} > 2W_B$. In case of a double slow-start, the congestion avoidance starts at $W_S = W'_{th} = W_B$ and ends at $W_E = W_{max}$. However, when a triple slow-start appears, the source gets shortly in congestion avoidance before detecting the loss occurring in the second slow-start. Therefore, the loss is detected at W_B and the threshold of the third slow-start is set to $W_B/2$. In this case, the congestion avoidance starts at $W_S = W_B/2$ and ends, as before, at $W_E = W_{max}$.

Let R be the ratio of the throughput in the two cases. We get

$$R = \frac{W_B/2 + W_{max}}{W_B + W_{max}} = \frac{\alpha + 2}{2\alpha + 2}$$

with $\alpha = W_B/W_{max} = W_B/2W_{th}$.

R is less than one which means that the triple slow-start degrades the performance. Also it is a decreasing function of α. Therefore, the maximum throughput deterioration is got for $\alpha = 1/4$ which corresponds to $W_{th} = 2W_B$. For this α, we find a ratio of 0.9. Thus, TCP can loss up to 10% of its performance when this problem appears.

8 Simulation results

The model of our simulation is represented in figure 1. A source running the Tahoe version of TCP transmits an unlimited amount of data to a TCP receiver which, in turn, acknowledges each incoming packet. The source is connected with a high speed link of rate 10 Mbps to a bottleneck node which routed the TCP packets via a T1 satellite link of rate 1,5 Mbps to the destination. The total propagation delay between the source and the receiver is taken equal to 280 ms, thus τ =560 ms. TCP packets are of total size 512 Bytes, therefore, the bottleneck service rate is μ = 366 Packets/s.

According to these parameters, a buffer capacity smaller than 26 packets is needed to get $W_{th} > 2W_B$ and then to make the triple slow-start possible. For B between 27 packets and 64, W_{th} is located between W_B and $2W_B$ and a double slow-start only exists. The figure 2 shows the variation of TCP throughput when B increases. We notice the oscillations due to the cyclic appearance of the triple slow-start. Also, we notice that this phenomenon disappears completely when B exceeds 26. The linear increase in the throughput is logical due to the linear increase in W_I and W_S when B grows. The jump in the throughput (other than that due to the disappearance of the triple slow-start) is caused by an increase in n_B. Thus, W_B jumps leading to a sudden increase in W_S and then in the throughput. It is clear that the throughput deterioration is about 10% when we approach $B = 26$. These results confirm the exactitude of our analysis.

To visualize this phenomenon, we plot the congestion window evolution for three values

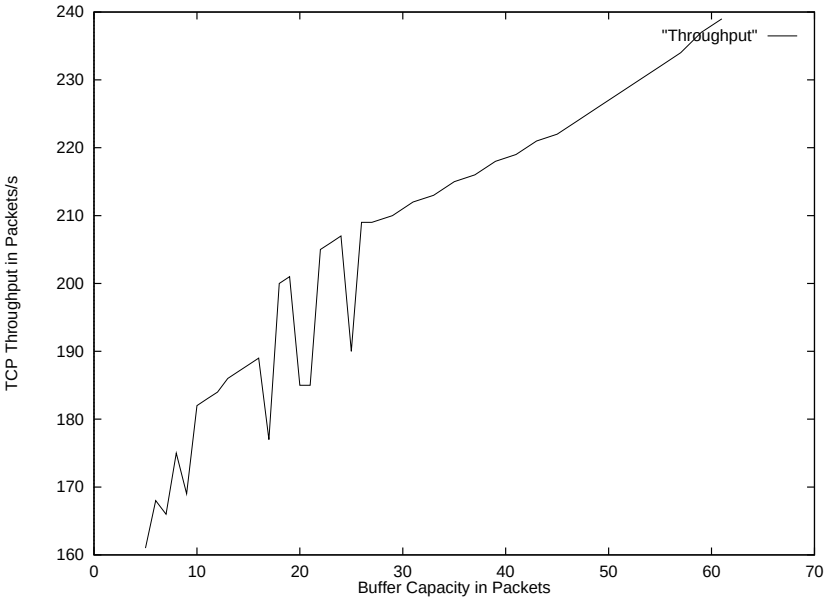

Figure 2: The cyclic occurrence of triple slow-start

of B (19, 20 and 22). Figure 3 shows that the buffer overflows only in the first slow-start. However when $B = 20$, although it is larger, the buffer overflows twice leading to a triple slow-start, therefore, to a lower throughput (Figure 4). Increasing B to 22 (Figure 5) eliminates the overflow in the second slow-start. This makes the congestion avoidance starts at a larger window resulting in a higher throughput.

9 Conclusion

In this paper, we studied by mathematical analysis and simulations the occurrence of three consecutive slow-starts in a TCP cycle. Due to the multiplicative decrease of the congestion window after a loss detection, these consecutive slow-starts results in a deterioration in TCP throughput. We showed that this phenomenon occurs in a cyclic manner in networks having a small buffering capacity compared to their bandwidth delay product. Satellite links, with their long propagation delay, are vulnerable to this problem. We derived conditions characterizing when these three slow-starts may occur and then we evaluate their effect on TCP throughput. We found that the appearance and the disappearance of this phenomenon result in a throughput degradation up to 10%. The interesting thing is that an increase in the buffer capacity can make the problem appear and therefore reduce TCP performance.

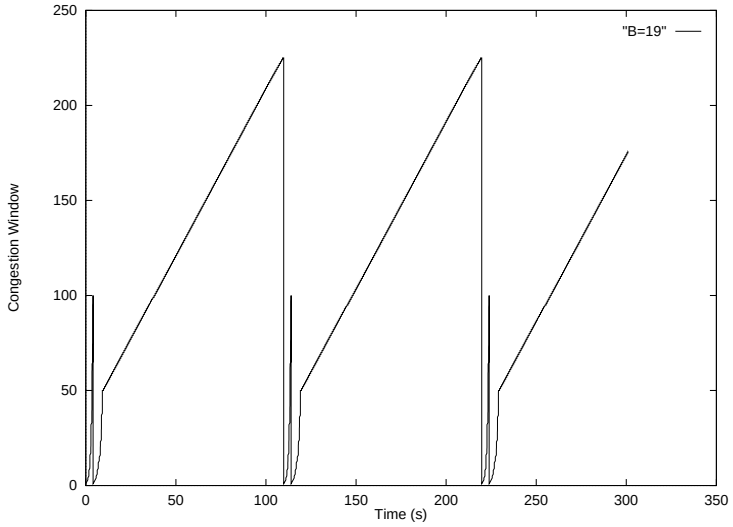

Figure 3: The case of B=19 Packets

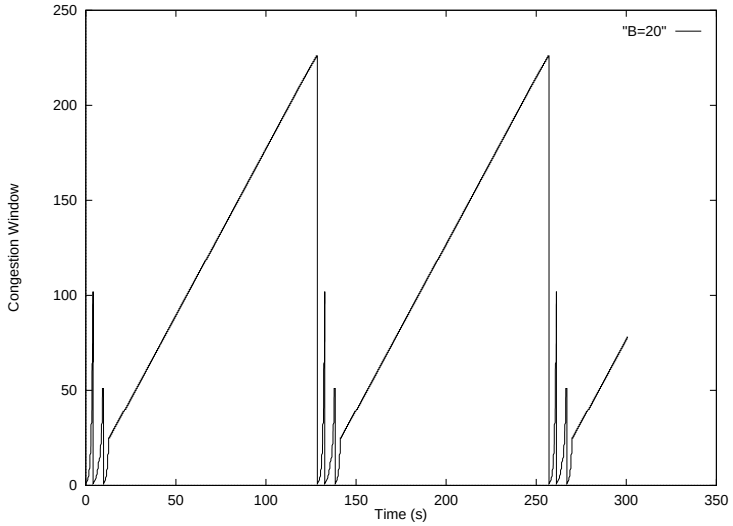

Figure 4: The case of B=20 Packets

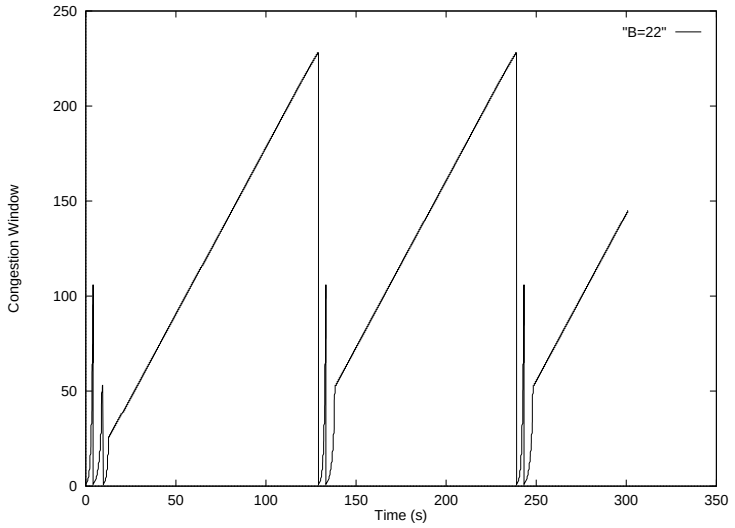

Figure 5: The case of B=22 Packets

However, our analysis of the problem considers only a buffer dedicated to the TCP connection. In reality, the bottleneck is shared with other sources which changes the behavior of TCP. The influence of this exogenous traffic must be considered. Also, we assumed that the satellite link is reliable which is not the case in reality. Random losses can cause a premature loss and then a lower estimation of the available capacity which may affect the phenomenon. Further study is required to evaluate the negative effect of these three slow-starts in real network conditions.

References

[1] 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.

[2] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", computer Communications Review, July 1996.

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

[4] V. Jacobson, "Congestion avoidance and control", Proc. ACM Sigcomm'88, Stanford, CA, USA, Aug. 1988.

[5] 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.

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

[7] S. McCanne and S. Floyd, "NS (Network Simulator)", 1995. URLs http://www-nrg.ee.lbl.gov/ns, http://www-mash.cs.berkeley.e du/ns.

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

[9] 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