Initial Spreading: a Fast Start-Up TCP Mechanism

Renaud Sallantin, Cédric Baudoin, Emmanuel Chaput, Fabrice Arnal,
Emmanuel Dubois, André-Luc Beylot

Initial Spreading: a Fast Start–Up TCP Mechanism

Renaud Sallantin*, Cédric Baudoin‡, Emmanuel Chaput*, Fabrice Arnal‡,
Emmanuel Dubois† and André-Luc Beylot*

* Université de Toulouse - IRIT Email: {sallantin, chaput, beylot}@enseeiht.fr
† CNES Email: {emmanuel.dubois}@cnes.fr
‡ Thales Alenia Space Email: {cedric.baudoin, fabrice.arnal}@thalesaleniaspace.com

***Abstract*— With most internet connections being short–lived (i.e. 10 segments), it is very tempting to enlarge the TCP Initial Window (IW). This would save two of the three RTTs needed to transfer most of the web pages through a legacy slow start. However it has been demonstrated that the bursts created by a larger IW greatly impair global performance. An intuitive solution is the TCP Pacing. By spreading the transmission over the whole RTT, Pacing smoothes the bursts and delays the congestion. While postponing congestion provides good performance for short-lived connections, it could significantly deteriorate global network performance insofar as the reaction to congestion is also delayed.**

This paper analyzes the weaknesses of large IW and TCP Pacing and proposes a fast Start-Up mechanism to speed up short-lived connections while preserving long-term connections. Extensive simulations and analysis demonstrate that our solution is as efficient as a larger IW would be in an uncongested network and better than current mechanisms in congested environments.

***Index Terms*—TCP; Fast Startup; Initial Window; Pacing; burst; RTT; congestion**

I. Introduction

It is well-known in the research community that the transport layer is responsible for a sizable part of network performance. It has also been observed that the Transmission Control Protocol (TCP) – the most widely-used transport protocol – is not very efficient in every network configuration, especially for Long Fat Networks (LFN). Several researchers have therefore attempted to improve this protocol, and optimized TCP releases [11] [16] have been rolled out in recent years. Their congestion algorithm has been improved to overcome some of the weaknesses of previous versions such Reno or new Reno protocols. The long latency needed to recover from a loss mainly due to a very conservative and linear congestion algorithm has thus been reduced using a dynamic and scalable algorithm in the Compound and Cubic protocols. Instead of reacting only to the detection of losses, they also exploit the delay between two losses or two successful acknowledgments to be more receptive to the network evolution. It has been demonstrated that Cubic and Compound protocols offer very high performance for long-lived connections even with LFNs [14].

Nevertheless, neither Cubic nor Compound protocols have changed their start up phases: both continue to use a regular slow start, while current traffic is mainly based on short-lived connections (90% of HTTP Web objects fit within 10 segments) [10] . Only a small percentage of connections then enter congestion control mode, while the majority are still constrained by slow start. It is thus necessary to improve the start up phase.

Intuitively, the easiest way to satisfy this constraint is to increase the Initial Window (IW). Beginning the connection with an emission of ten segments instead of one or three, as is currently the case, would allow nearly 90 % of web requests to be sent in one Round Trip Time (RTT) and then save at least three RTTs. For example, this would reduce by 1s the total transmission time for a ten-segments flow in satellite communication. Many papers [2] [5] [10] address this appealing solution but provide diverging conclusions as to its global results. While there are major advantages in most cases, it has been determined that this mechanism significantly impairs performance in certain scenarios.

This paper proposes a fast Start-Up solution – Initial Spreading – to solve the problems arising from a larger IW. Based on a combination of large IW and Pacing concept, our proposal offers at least the same benefits as a larger IW, while assuring significant improvements in current performance at problem points.

The rest of the paper is organized as follows. Section II, based on related works, provides an overview of the impact of an increase in IW. We then focus on the Pacing protocol as a possible way of efficiently sending more data within the same RTT. Both ideas are discussed separately, with an emphasis not only on the pros and cons of each, but also on the possibility of combining them. Section III presents our mechanism along with an evaluation based on NS2 simulations versus existing solutions. Section IV concludes our paper on the necessity of using Initial Spreading to safely implement a large IW.

II. Related works

TCP aims to share network resources fairly and efficiently. Its current operating procedure is quite simple: a TCP connection first uses a slow start to probe the network and

then increases its bit rate exponentially. Whenever the sender receives an acknowledgment, its congestion window (CWND) is incremented by one. The CWND corresponds to the number of segments which can be sent within the same RTT. Both CWND and bit rate are very low at first but double with each RTT until the first loss occurs. Considering each drop as a marker of possible congestion, TCP enters a congestion avoidance control stage.

During slow start, TCP results in bursty traffic. Segment transmission is determined by the reception of acknowledgments. With the exception of data transmitted during the first RTT, the sending rate is limited by the slowest subsystem of the network, which is the bottleneck link. The sender receives the acknowledgments of the previously transmitted window of segments at the bottleneck rate and then transmits a micro-burst of two segments for each expected acknowledgments. The bottleneck router can only process one segment at a time, so it has to store the other one in its buffer. For each acknowledged window, the sender is therefore going to send double what the bottleneck router can handle. Finally, a CWND of W segments builds up a queue of size $\frac{W}{2}$ and this bursty traffic is responsible for a rapid increase in the bottleneck queue length.

In recent years, the development of LFNs and the race to higher performance has led to a combat against the sluggishness at the beginning of TCP connections. Research has therefore focused on Fast start-up mechanisms [3]. [15] underlines two solutions in particular – Jump Start & Quick Start – that can achieve good results. These solutions are not discussed further in this paper because both suffer from major drawbacks. Quick Start uses router explicit notification and then breaks the TCP end–to–end semantics, while Jump Start is far too aggressive in a congested environment.

The following sections focus on both IW increase as one of the most appealing start up mechanisms and solutions to deal with the natural burstiness of TCP traffic.

A. Increase in the TCP IW

In 2002, an RFC [2] legalized the use of a three-segments IW. This change allowed up to 4 kB to be send in the first RTT and then the elimination of up to three RTTs. At this time, three major arguments justified this choice. Firstly, an IW of one can imply waiting for an unnecessary Retransmission TimeOut (RTO) because of the delayed ack implementation. Secondly, the average size of web objects was below 4 kB so, in an uncongested environment, all the data to be transmitted could be sent in the same RTT. Lastly, it took into account the fact that increasing the IW was not transparent for either the global behavior of the network or for individual connections.

In a congested network, the blind transmission of many segments can generate multiple retransmissions and significantly impact network performance. Moreover, because the segments of the first window are not sent at the bottleneck rate but at the sender rate, sending an IW of n segments does not increase the queue by $\frac{n}{2}$ but may add up to $n-1$ segments in the queue. A large IW then creates a burst that may result into unnecessary drops in congested buffers, leading to noticeable reductions in the bit rate of individual TCP connections and notably short-lived connections.

In conclusion, an IW of three segments enables important time savings in most cases and remains conservative enough to be efficient in a congested environment.

Ten years later, several studies have emphasized that web standards have evolved. The average size of web objects has in particular risen from 4 kB to 15 kB. Depending on the Maximum Segment Size (MSS), these sizes can be roughly translated into three and ten segments respectively. In order to transmit most of the data in one RTT, [10] proposed to set the IW to ten segments without any other changes to cover 90 % of the HTTP web requests. The authors showed that this higher IW was neither responsible for lower global network performance [9] nor for the deterioration of TCP fairness. However, as the burst phenomenon remains, individual performance in the event of congestion is still affected, probably more severely. This solution can be seen as the outcome of a trade–off between the substantial improvement in performance in an uncongested network and the deterioration in individual performance that may occur in the event of severe congestion.

B. Pacing principle

The original Pacing idea is to prevent the generation of bursts insofar as possible. The principle is to spread window transmission over the RTT whenever possible. Each segment thus arrives separately at the bottleneck router. This differs from the legacy slow start, where segments arrive in pairs. If the time required to deal with the segment is shorter than the space between two successive transmissions, then the buffer queue size is not increased.

This appealing solution has been used and discussed in several studies as a very efficient way to reach the maximum bit rate for each connection fairly. However, [1] has pointed out some flaws caused by Pacing, especially a synchronization effect and a tendency to overload the network that seriously damage both individual and global performance.

As outlined previously, bursts are responsible for isolated congestion and for the transition from the slow start to congestion avoidance. They shape the bit rate of each connection. An early burst limits the sending rate and its growth, whereas a late burst allows a fast increase using slow start. Eliminating bursts may allow higher bit rates, but network resources and particularly the bottleneck buffer queue size are limited. By smoothing the traffic, Pacing avoids early and isolated congestion, but saturates the network. At some point, all connections suffer from multiple drops, even those in the congestion avoidance phase. Pacing can therefore result in worse performance.

C. Pacing used with large IW

As seen above, the burstiness of TCP traffic significantly affects connection performance. Both previous concepts suffer from bursts: the increase in IW from their occurrence and

Pacing from their absence. Intuitively, a combination of both may lead to beneficial results.

Figure 1 illustrates queue size evolution of the bottleneck router for a single connection. This Matlab calculation uses a simple three-link topology. The bottleneck link bit rate is 10 Mbps, while the other two are 40 Mbps. The RTT is 100ms. Three cases are studied: large IW, regular Pacing and Pacing with a large IW.

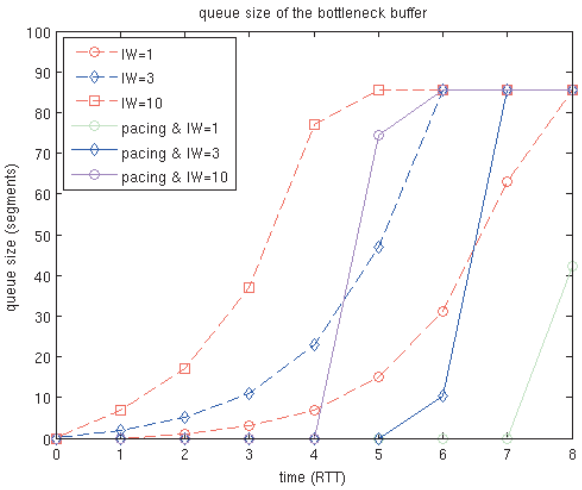

Figure 1. Queue Size evolution with and without Pacing

This figure highlights the consequences of each option for different IW sizes. A large IW almost fills the buffer right from the first RTTs. On the other hand, Pacing does not increase the queue size during the first RTTs even when combined with a large IW.

In a congested network, the bottleneck router can be very sensitive to a small increase in its buffer size. As a consequence, an IW of ten is likely to create more drops than an IW of three, impairing performance. Scenarios with Pacing do not suffer from the early bursts. The sender can transmit many segments before there is any disturbance in network behavior. However, Pacing allows a remarkable increase in the CWND and may lead to the network overload for long-lived connections. In the best case, there are multiple drops for each connection and flow synchronization. In the worst case, the network collapses.

Finally, using Pacing to increase the IW overcomes the flaws of a large IW. This combination is very competitive for short-lived connections but appears to exaggerate the Pacing drawbacks for long-lived connections.

In the following section, we use this result to propose a solution which efficiently combines Pacing and a large IW.

III. Initial Spreading: a start–up mechanism based on Pacing and a large IW

The previous section not only outlines the pros and cons of each concept but foreshadows the shape of a new protocol. This new protocol should be able to deal with the different consequences of bursts in a congested environment in order to be efficient when transmitting few data and at least be transparent for large data transmission.

The following sections introduce our mechanism and present its performance through our implementation in NS 2.

A. Our proposal

The basic idea is to space out an IW of n segments across the first RTT before letting the TCP algorithm continue conventionally. Until the first loss, this equates to sending n parallel connections starting with a regular slow start and a unit IW. When a loss occurs, recovery mechanisms quickly mitigate the impact of our proposal, known as Initial Spreading to easily differentiate the different mechanisms.

Figure 2 presents the behavior of the three mechanisms when transmitting 12 segments. T_2 and T_1 respectively denote the time to process a packet at the bottleneck rate and at the rate of the other links.

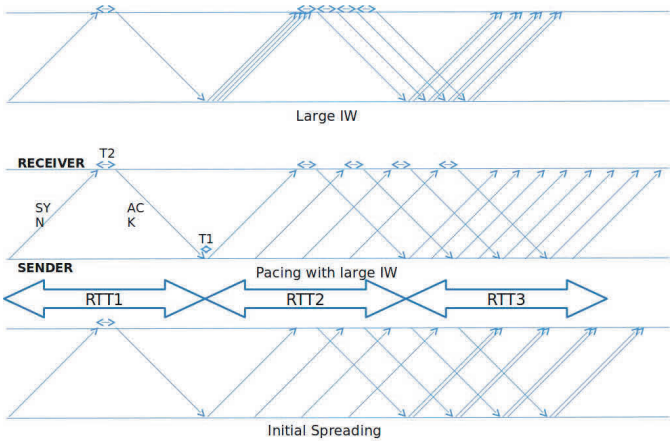

Figure 2. Time diagram picturing the transmission of 12 segments with the three different mechanisms using an IW of four

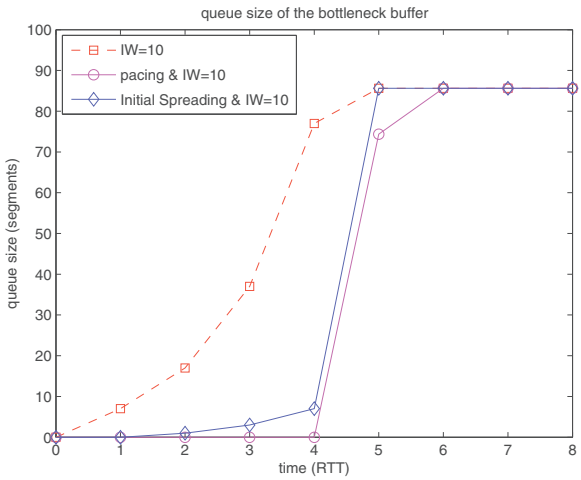

Figure 3. Queue size evolution with Pacing and Initial Spreading

Figure 3 depicts how the different fast start-up protocols fill the bottleneck router queue for an IW of ten segments during the slow start. The Syn/Ack exchange is used to estimate the RTT and thus the space between two successive segments transmitted with Initial Spreading. One noticeable difference between Pacing and Initial Spreading is the creation of bursts as early as the second RTT. We assume that these small bursts would prevent congestion to occur too late but still grant good performance with short-lived connection.

In the following sections, we verify our hypothesis and justify our protocol using NS2 simulations. Special attention is paid to the performance of the protocol in sensitive scenarios like short- or large-flow transmissions. Finally the consequences of a set IW size are studied and a suitable size proposed.

B. Implementation

We implemented our fast start–up algorithm in the NS2 environment. NS2 is the most widely deployed and shared simulator for network research. Most of the latest releases and TCP developments are already being implemented. Our simulations mainly used TCP Cubic, but we ensured that our conclusions also apply to other TCP flavors. In order to ensure the reliability of our results, we used the NS seeds to accurately repeat identical scenarios for each mechanism over numerous iterations and random emission times [4] [6]. A confidence interval of 95 % is indicated for each point.

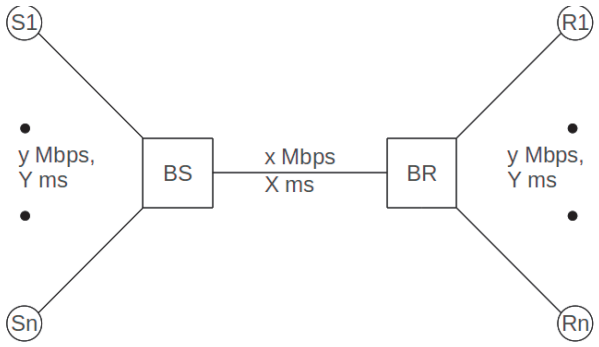

Figure 4. The network topology used for the simulation experiments

Figure 4 depicts the basic network topology used during our simulations. Multiple values were considered for delays and link bit rates to evaluate the behavior of Initial Spreading under different traffic conditions.

In order to observe the different behaviors in congested but realistic environments, short–lived connections between S_i and R_i with i in $[1:12]$ were initiated at a random time after unlimited TCP connections between S_i and R_i with i in $[13:15]$ were established.

Most of the following results were obtained in accordance with the testbed of [1] in order to be able to compare results.

Simulations were conducted with and without Delayed Acknowledgements (Del Ack), but the following sections only illustrate the case with the Del Ack TCP option turned off. Comparisons of the different IW management approaches did not reveal any significant differences in TCP behavior with or without Del Ack. Regarding long-lived connections and notably the steady state, the effects of Del Ack are lessened by TCP Cubic which tends to adapt its congestion algorithm to take into account whether the receiver uses the Del Ack option or not. In so doing, it can prevent the connection from being too slow, and still continue to reduce acknowledgments traffic. In the event of short-lived connections, a slow start is less aggressive with Del Ack than without, but there is no change in the burst propagation model. As soon as the first ack arrives, Del Ack reduces CWND growth but new segments continue to be transmitted with micro bursts of two-segments. The only noticeable impact of the Del Ack option is the decrease in the number of potential Dup Acks that trigger a fast losses recovery. This point works in favor of Initial Spreading, which alleviates bursty behavior and then reduces the probability of losses.

The exact configuration is given in each of the following subsections.

C. Performance of short-lived connections

1) Uncongested network: In an uncongested environment, the best performance is achieved using a large IW without any other fast start–up protocol. Pacing delays transmission and thus the acknowledgments throughout the entire RTT. With T_2 the time to process a packet at the bottleneck rate and n the IW size, acknowledgement of the whole IW lasts $RTT + (n-1) * T_2$ without Pacing, and $RTT + (n-1) * \frac{RTT}{n}$ with Pacing, so the difference is equal to $(n-1) * (\frac{RTT}{n} - T_2)$ with $(\frac{RTT}{n} - T_2) > 0$.

If the number of data to be transmitted does not fill up the IW, good performance continues to be mostly related to the increase in IW rather than the spacing between segments. As shown in Figure 2, not spreading the initial batch of packets over time is still faster when no drop occurs. However, penalties for both other approaches are limited to few milliseconds.

2) Congested network: In a congested environment, the burst phenomenon has a substantial impact on performance. In section II, we hypothesized that a large IW without Pacing can lead to poor efficiency because of the bursts it causes and the resulting congestion that may occur. On the contrary, Pacing smoothes the bursts and so may allow transmission of a few data efficiently.

The following figures show our extended simulation for verification purposes. The bottleneck link bit rate is set to 10 Mbps and the delay to 50 ms whereas the other links have a bit rate of 40 Mbps and a delay of 5 ms. The buffer size is set to half the Bandwidth-delay product. We chose to illustrate our proposal with a typical network instead of an LFN topology for example. The tests conducted with satellite scenarios show even better results in favor of Initial Spreading.

Figure 5 plots the average completion time, i.e., time it takes a source to successfully transfer a given number of data, for a large IW of ten segments with and without Initial Spreading as a function of flow size. We noticed non–intuitive results that

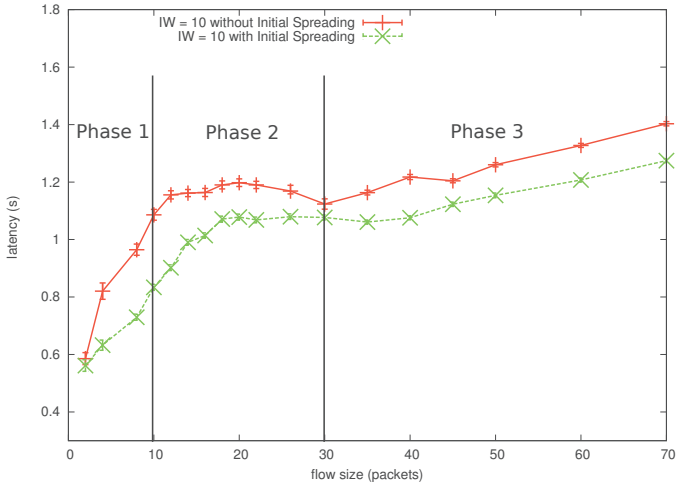

Figure 5. Average latency for a large IW with and without Initial Spreading

require further explanation. The graph may be divided into three distinct phases:

1) Phase 1: flow size varies from one to ten. Both mechanisms lead to very different performance.
   Without Initial Spreading, the flow is sent in one burst. In a bursty loss model, it is ordinary to consider that the loss of one segment implies the loss of all the following segments in the burst [7] [17]. The probability of packet drop greatly increases with flow size and the packets sent last are then more likely to be lost than the first ones. During this phase, each loss leads to a timeout because no Duplicated Acknowledgements (Dup Ack) are generated. Indeed, TCP has two ways of recovering from a loss: either waiting for a timeout before transmitting the segments again or receiving three Dup Ack that enable Fast Retransmit and Fast Recovery. Each additional segment in the flow raises then the drop probability and so the average estimation of the completion time
   With Initial Spreading, bursts are smoothed and only one packet is sent every $\frac{RTT}{10}$. Initial Spreading tends to make losses independent, and with a stationary background traffic, the probability for a spread segment to be dropped becomes constant. The probability of loss increases then with flow size, but this is offset by the concomitant increase in the probability of entering in Fast Retransmit. Results are thus better with Initial Spreading.

2) Phase 2: flow size varies from 11 to 30, which is the maximum number of packets that can be transmitted in the first two RTTs. Once again, both solutions have different impacts on the average transmission time.
   Without Initial Spreading, all the acknowledgments can only result from the first segments sent in the IW, i.e., at that moment, the receiver receives only expected acknowledgments and is not aware of any congestion. The connection continues therefore in slow start mode and each acknowledgment received triggers the transmission of micro-bursts of two new packets at bottleneck arrival rate (see Figure 2). These micro–bursts have a higher probability of successful transmission over the network than segments from the initial burst. So enlarging the flow size raises the probability of receiving Dup Acks and then fast-retransmitting the lost segments and entering a Fast Recovery stage. When in Fast Recovery, each Dup Ack received enables the transmission of a new packet and a high bit rate is conserved. Average completion time tends to decrease due to those recovery mechanisms.
   With Initial Spreading, each expected acknowledgment leads to the transmission of micro–bursts of two segments. But unlike the case where there is no Initial Spreading, these bursts have a higher probability of being lost than the spread IW segments. However the higher probability of losing these segments is counterbalanced by a higher probability of entering in fast recovery due to Dup Acks. When enough segments have been transmitted, the probability of a fast recovery is significant and average latency evolves very slowly.

3) Phase 3: flows over 30 packets. The sender receives non-duplicate acknowledgments from the segments sent when in Fast Recovery. Reception of the first one leads to a transition from Fast Recovery to Congestion Avoidance and CWND is reduced. Initial Spreading and Large IW mechanisms behave similarly and the slope of their curves is the same.

Figure 6 and Figure 7 depict the effects of different IWs for TCP connection with and without Initial Spreading. For the flow sizes of interest (around ten segments), larger IWs without Initial Spreading are responsible for worse performance than shorter ones because of the burst phenomenon described in phases 1 & 2. For a flow size of ten segments, the most efficient IW without Initial Spreading is then of three segments.

On the other hand, our proposal leads to very good results whatever the IW, and notably IWs larger or equal to ten. A further study in the last section analyzed the impact of the IW size on Initial Spreading in a more accurate way.

Focusing on short-lived flows, and particularly on flows shorter than the IW, both Pacing and Initial Spreading give similar results. Both use the spacing of the first segments to significantly decrease the number of losses due to congestion.

Figure 8 compares the best results obtained with and without Initial Spreading. Initial Spreading reduces the completion time of short-lived connections by roughly 30 %.

In conclusion, using a combination of Initial Spreading and a large IW does not deteriorate performance in uncongested scenarios and is at least 30 % more efficient than a large IW alone in a congested environment.

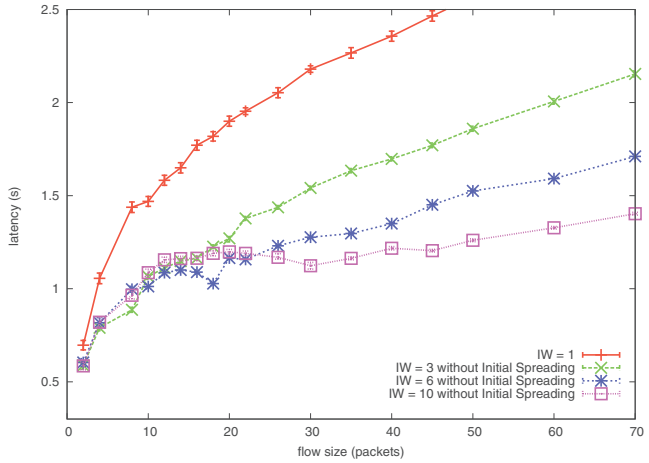

Figure 6. Impact of a large IW without Initial Spreading on average Latency

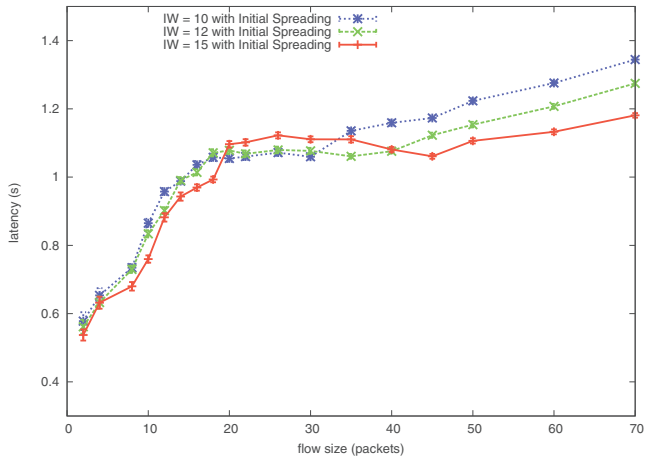

Figure 7. Impact of a large IW with Initial Spreading on average Latency

D. Performance for long-lived connections

Unlike the previous simulations where transmission durations were measured for different flow sizes in an already congested environment, we near simultaneously sent 15 connections of the same size over an empty network to create a congestion from scratch. This test realized by [1] revealed Pacing's very bad performance and underlined a synchronization effect.

Up to a certain number of segments per flow (around a few hundred), there is no significant congestion and performance of both the legacy mechanism and Pacing is similar (see III-C-1). Both are then affected by congestion due to the increase in data and some flows are likely forced to time out. After an RTO, connections restart with a slow start and an IW of one. The bursty traffic of the legacy mechanism slows down establishment of the connection and then deteriorates individual performance in favor of global performance. On the other hand, connection with Pacing overloads the network because of late congestion detection until all the connections suffer

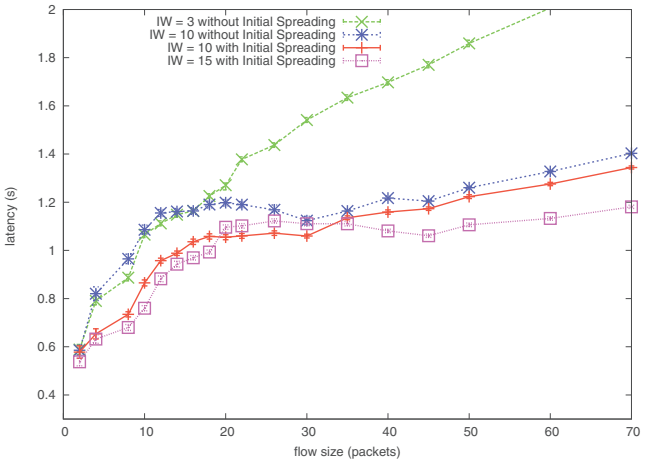

Figure 8. Comparison of the average latency for connections using large IW with and without Initial Spreading

from congestion and flows are synchronized. Performance with Pacing is then only half the performance without.

Figure 9 illustrates the behavior of our Initial Spreading proposal and a large IW without Pacing for long–lived connections. The results are normalized using an estimate of fair latency, which corresponds to the time required for a connection beginning with a slow start and then keeping a stable and fair bit rate: $\frac{1}{15}$ of the bottleneck bit rate.

The results obtained show similar performance for both the legacy mechanism and our proposal. As the Initial Spreading transmission segment behavior reverts back to regular bursty behavior after the first window, it is transparent after a timeout when the IW is set to one and does not affect long-lived connections. Regarding intermediary flow size when there is still little probability of a timeout, the bursts created as early as the second RTT avoid overloading the network.

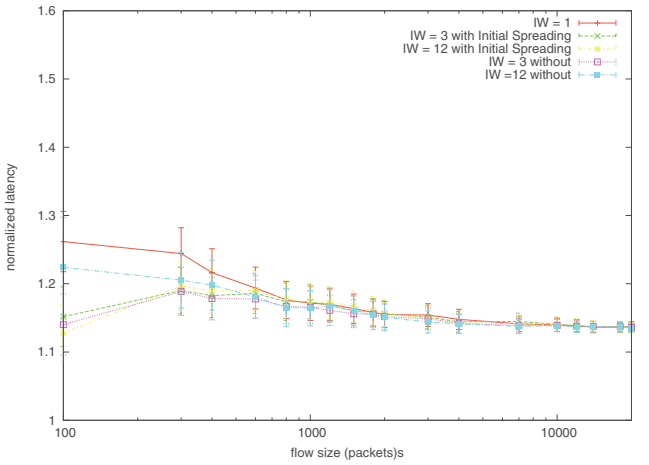

Figure 9. Impact of the synchronization effect on normalized latency

E. Interaction of Initial Spreading and non-Initial Spreading Flows

Scenarios considered so far consisted of flows with the same kind of CWND management evolving together. In this section, we present results related to the interaction of flows using a large IW with and without Initial Spreading.

[1] noted that Pacing performance decreased when facing flows using a regular algorithm. The authors assumed that beyond a certain number of in-flight segments, the probability of a paced segment encountering a burst, and then being dropped, is higher due to the uniform spreading of packets over the RTT. On the other hand, a bursty traffic has a smaller probability to meet an other burst and then to drop a packet. These interactions were responsible for better performance without Pacing.

Figure 10 below illustrates the consequences of mixing sources with different initial window management approaches.

In contrast to the previous observation, Initial Spreading performance is actually not mitigated by the concurrence of other flows, and continues to grant significant improvements. Moreover, results obtained without Initial Spreading tend to be better in a mixed environment. Initial Spreading helps then to improve TCP friendliness.

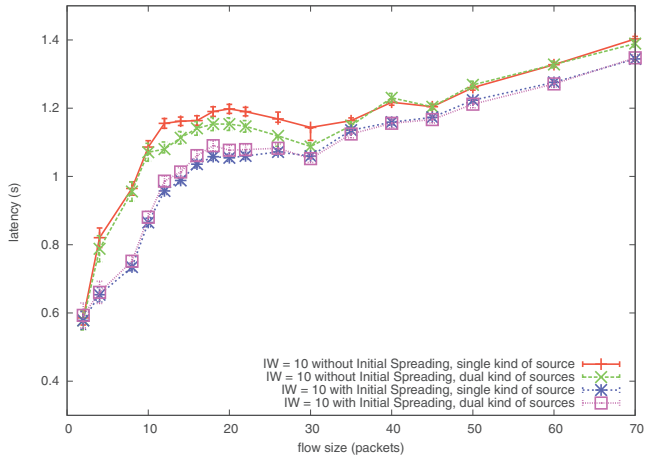

Figure 10. Effect of different sources sharing a bottleneck

To conclude, the previous sections support our hypotheses and confirm that Initial Spreading solve the problems caused by a large IW and Pacing .

F. Choice of IW Size

Having presented the global improvement offered by Initial Spreading, we now focus on the influence of the IW on Initial Spreading behavior. Different network configurations are introduced in the following simulations, designed to further evaluate the possibility of setting the IW with RTT as the only known parameter.

The purpose of this research was to provide excellent performance for short–lived flows (i.e. around 10 segments) in both uncongested and congested networks. Section III-C shows that Initial Spreading meets these requirements when combined with a large IW. As Initial Spreading with an IW larger than ten is more efficient than any large IW without Initial Spreading whatever the case, we then focused on IW equal to or greater than 10.

Table I
SAVINGS FOR 10 SEGMENTS USING INITIAL SPREADING

	IW = 10	**IW = 12**	**IW = 15**
Average Savings	35.2 %	41.5 %	37 %
Min Savings	30 %	38 %	30 %

Table I presents the average benefits obtained from using Initial Spreading with an IW of 10, 12 or 15 segments for flows shorter or equal to 10 segments in a congested environment. The time savings are relative to the best results obtained without Initial Spreading, i.e., what could be reached without Initial Spreading if we used the most well-adapted IW for each scenario (in terms of degree of congestion and flow size). Several configurations were tested and the following table highlights the average results. Regarding the ten-segment flows, a remarkable improvement of over 30 % is achieved for every scenario tested.

We noted that the best results are achieved with an IW of 12, though ten and 15-segments IWs offered similar results. A further study considering the influence of scenario metrics on the evolution of the performance explains these results and highlights two different behaviors:

- Regarding long delays, the space between two transmissions is large enough to send ten or 15 segments in the first RTT just as efficiently, i.e., segments continue to have independent loss probability. So when IW equals 15 segments instead of ten, the time required to transmit ten segments is lowered by $9 * (\frac{RTT}{10} - \frac{RTT}{15})$. This behavior was verified during our simulations of satellite network.
- Otherwise, as enlarging the IW will reduce the space between two initial transmissions, Initial Spreading is more likely to suffer from congestion with an IW of 15 than with an IW of 12 or ten. For example, a bottleneck of 1ms in the classic topology (see Figure 4) leads to a greater efficiency with an IW of ten or 12 than 15. Nevertheless, in the sample group of existing scenarios we considered, Initial Spreading always performed better with an IW of 12 than with ten.

An other possibility is to calculate an ideal IW. [13] proposes a variable IW based on the different network bandwidths, buffer sizes and RTT to improve the efficiency of fast Start–up TCP mechanisms. In the light of our results, such a proposal does not lead to sufficient progress to justify the extra and prohibitive complexity.

We conclude that time savings due to a shorter space between two transmissions is less important than a potential deterioration due to insufficient spacing. Given that Initial Spreading with an IW of ten always offers time savings of over 30 % compared to the standard mechanism, we

propose remaining conservative and setting the value at ten. We thus keep a reasonable margin in order to take into account potential developments in future networks.

IV. Conclusion

This paper evaluates the effect on TCP performance of an Initial Spreading mechanism to speed up the beginning of TCP connection by supporting safely large IW. Without Initial Spreading, the current proposal [9] to increase the IW from three to ten is extremely controversial because of the deterioration in performance in several sensitive cases, including congested networks.

Our hypothesis was that a large IW and Pacing were suffering from bursty TCP traffic and that our proposal should take this into account to optimise efficiency in most scenarios. Extensive simulations have shown that Initial Spreading enables us to keep the same high level of performance as the best start up proposals in an uncongested environment, while offering significant improvements in congested environment. More particularly, in congested network, Initial Spreading offers:

- time savings of over 30 % for connections with few data in comparison with a regular slow start whatever the IW size, including three- and ten-segments.
- a higher or equal bit rate to that obtained with any IW for bigger flow sizes
- a way of avoiding overload due to late congestion and synchronization

To conclude, the simplicity of implementation of Initial Spreading is an additional advantage. Initial Spreading may be used as a warranty to enlarge the IW from three to ten while avoiding any controversy.

In the future, we plan to work on two potential TCP start options in order to improve Initial Spreading: the use of the initial SYN/ACK exchange to begin data transmission [8], and the possibility of restarting with a large IW after a time–out [12]. As our simulations have shown that Initial Spreading enables greater improvements for LFNs and particularly satellite communication, we shall focus on this last area. We aim to determine whether Initial Spreading can remove the need for TCP accelerators such as Performance Enhancing Proxy (PEPs).

References

[1] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *INFOCOM*, vol. 3, mar 2000, pp. 1157 –1165.

[2] A. Allman and S. Floyd, "Increasing tcp's initial window," no. 3390, 2002.

[3] ——, "Quick-Start for TCP and IP," no. 4782, 2007.

[4] M. Allman and A. Falk, "On the Effective Evaluation of TCP," *ACM Computer Communication Review*, 1999.

[5] M. Allman, C. Hayes, and S. Ostermann, "An evaluation of TCP with Larger Initial Windows," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 3, pp. 41–52, Jul. 1998.

[6] A. Andrew, C. Marcondes, S. Floyd, L. Dunn, and T. e. a. Eggert, "Towards a Common TCP Evaluation Suite," in *Sixth International Workshop on Protocols for FAST Long-Distance Networks*, Mar. 2008.

[7] N. Cardwell, S. Savage, and T. Anderson, "Modeling tcp latency," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2000, pp. 1742–1751 vol.3.

[8] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain, "TCP Fast Open," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-tcpm-fastopen-02, Oct. 2012.

[9] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing tcp's initial window," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-tcpm-initcwnd-07, Jan. 2013.

[10] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, "An Argument for Increasing TCP's Initial Congestion Window," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 3, pp. 26–33, Jun. 2010.

[11] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1400097.1400105

[12] P. Hurtig, A. Petlund, and M. Welzl, "TCP and SCTP RTO Restart," Working Draft, IETF Secretariat, Internet-Draft draft-hurtig-tcpm-rtorestart-03, Oct. 2012.

[13] S. Kodama, M. Shimamura, and K. Iida, "Initial CWND Determination Method for Fast Startup TCP Algorithms," in *Proceedings of the Nineteenth International Workshop on Quality of Service*, ser. IWQoS. IEEE Press, 2011, pp. 11:1–11:3.

[14] R. Sallantin, E. Chaput, E. P. Dubois, C. Baudoin, F. Arnal, and A.-L. Beylot, "On the sustainability of PEPs for satellite Internet access," in *ICSSC*. AIAA, 2012.

[15] M. Scharf, "Performance Evaluation of Fast Startup Congestion Control Schemes," in *NETWORKING 2009*, ser. Lecture Notes in Computer Science, L. Fratta, H. Schulzrinne, Y. Takahashi, and O. Spaniol, Eds. Springer Berlin Heidelberg, 2009, vol. 5550, pp. 716–727.

[16] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A Scalable and TCP-friendly Congestion Control for High-speed Networks," in *in 4th International workshop on Protocols for Fast Long-Distance Networks (PFLDNet), 2006*, 2006.

[17] K. Zhou, K. Yeung, and V. Li, "Throughput modeling of TCP with slow-start and fast recovery," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 1, nov.-2 dec. 2005, pp. 261–265.