Reducing web latency through TCP IW: be smart

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

Reducing web latency through TCP IW: be smart

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

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

Abstract— **Depending on the congestion level and the network characteristics (*e.g.*, buffer sizes, capacity of the bottleneck, deployment scenario, etc.) a fixed Initial Window (IW) would be either too conservative or too aggressive. This results in low usage of the network resource or damaging high congestion level. This paper presents a sender-side only modification to the slow-start of TCP, SmartIW, that bypasses the limitations and potential issues of a fixed IW. The Round Trip Time (RTT) is estimated during the establishment of the connection and further exploited by SmartIW to pace the transmission of an adequate number of packets during the first RTT. Our simulation results show that, since the IW has been set in adequacy with the available network information, larger IW can be transmitted without increasing the congestion level of the network. SmartIW eventually reduces the RTT dependence of the slow start stage to fairly provide significant performance improvements whatever the network characteristics (RTT and congestion level).**

***Keywords*—TCP; burst; IW; RTT; pacing; QoE; fairness**

I. Introduction

The Internet community is faced with the crucial challenge of how to meet ever-growing consumer expectations in terms of web performance, and Quality of Experience (QoE). It has therefore been observed that while the average web page size increases continuously, users are no longer tolerating more than two seconds of page load delay [1], [2]. Today's bit rate supplied by infrastructures can easily satisfy such requirements. But, the network protocols, responsible of the effective end-to-end throughput of a connection, continue to make this objective hard to achieve. As an illustration, the transport layer protocols that manage end-to-end communication services and support most web applications, including HTTP1.1 and HTTP2.0, are not geared to the new web traffic patterns.

Lately, the increase in the Initial Window (IW) size has been proposed by Google and standardized in the RFC 6928 [3]. It counters both the lack of aggressiveness and the Round Trip Time (RTT) dependence of the "slow-start" phase of the Transmission Control Protocol (TCP). Indeed, the increase in the IW from three to ten segments enables to transmit at once, *i.e.* without having to wait for acknowledgements, 90% of the Web objects [4], and thus, most of the Internet connections when HTTP1.1 is used.

However, despite its large deployment [5], the consequences [6] of this appealing solution are questionable regarding two major TCP evaluation criteria:

- **the end-to-end performance.** Indeed, it has been demonstrated that the transmission of a large batch of segments in one single burst, seriously deteriorates the connection performance when some congestion occurs in the network [7].
- **the fairness.** TCP's slow start has been designed to probe the available bandwidth of an unknown network in order to not overflow it by the transmission of an inappropriate number of segments. Thus, releasing a large batch of segments with no prior information on the network may trigger some unfairness.

Initial Spreading has been proposed to address the first point and support the use of a large IW [8]. It is a TCP server only mechanism that uses pacing to manage the transmission of the IW segments and prevent the burst repercussions. It achieves 30% of performance improvement in the transmission of short-lived flows in a congested environment [9], [10] when implemented with RFC6928, and eventually legitimizes an IW of 10 segments.

Nevertheless, the continuing growth of the average web objects size but also the significant changes induced by the fast deployment of HTTP2.0 are going to drop the number of connections shorter than 10 segments. In order to face it, and notably, to keep the RTT independence, the IW size should therefore be increased again, gainsaying the slow start precautionary principle. Focusing only on the performance, Initial Spreading is not adapted to support such an increase, and guarantee the fairness of the new connection.

In this paper, we propose a new fast start-up TCP mechanism, that enables to efficiently and fairly increase the amount of data sent during the first RTT. SmartIW, inheriting from our previous work on Initial Spreading, therefore focuses on the transmission rhythm of the first TCP segments of a connection to dynamically set the appropriate IW size in function of the RTT duration.

The rest of the paper is organized as follows. Section II, based on literature and previous works, analyzes major TCP weaknesses and introduces some limitations of the Initial Spreading mechanism. Section III presents our sender-side only modification, SmartIW. Section IV compares, by means of ns-2 simulations the performance of Initial Spreading, SmartIW and standard TCP variants. Section V concludes and discusses future work.

II. Impact of the Internet traffic evolution on TCP start-up mechanisms

A. Up to now

The steady and sustained increase in the average web objects size over time has motivated several updates of the TCP's IW size. Thus, when 3 years ago, Google pushed for rising this number up to 10 segments (RFC6928 [3]), they argued that the previous limit of 3 segments (RFC3390 [11]) was not anymore suitable to the Internet traffic evolution. On the other hand, the IW of 10 segments they recommend, enables to send more than 90% of the web objects [4] and thus, most Internet connections when HTTP1.1 is used, in only one RTT after the SYN-SYN/ACK exchange.

While RFC6928 enables to spare a couple of RTTs in an uncongested network, the transmission of a single large initial burst of segments increases the segment loss probability and reduces the benefits of TCP's recovery mechanisms when some congestion occurs [7], [10]. Initial Spreading has therefore been proposed to modify the sending rhythm of the Initial Window and lower the consequences of the congestion on the segments transmission. Linux experiments and analytic model showed that the combination of both Initial Spreading and RFC6928 outperforms shorter IW and other TCP optimization mechanisms, regardless of the load of the network [9].

B. By now

The rapid evolution of HTTP1.1 in HTTP2.0, is going to significantly change the average Internet connection size by modifying how the web objects of a same web page are transmitted. Thus, instead of opening multiple TCP connections to transmit the different objects of a page, those objects are multiplexed into a single, but larger connection. Without prejudging of the impact on the global end-to-end performance, we can, however, expect serious repercussions on previous start-up TCP mechanisms efficiency, as they focus on the transmission of short-lived TCP flows.

Indeed, as soon as the connection size exceeds the IW size, the RFC 6928, with and without Initial Spreading, ceases to transmit the segments at an RTT-independent rhythm, as the $IW + 1st$ segment will be transmitted one RTT after the acknowledgment reception of the first sent segment. Thus, average performance will be deteriorated, and some known drawbacks of the slow-start such as the unfairness between the TCP connections in function of their RTT will be exacerbated.

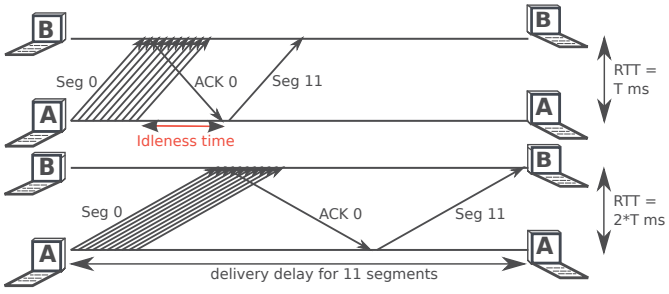

Figure 1. Transmission of 11 segments with an IW of 10

Figure 1 illustrates the two main consequences related to the loss of the RTT-independence through the transmission of 11 segments with an IW of 10 segments:

- exceeding the IW size, a connection will rise its idleness period ratio, i.e. the percentage of time only spent waiting for an acknowledgment and
- the unfairness due to the dependence on the RTT duration is increased. Thus, in a case without any congestion, the delivery delay, i.e. the average time it takes a source to successfully deliver i segments with an IW of size n, lasts $\frac{1}{2}RTT + (n-1) * T_B$ for $i = n$ segments, and $\frac{3}{2}RTT + n * T_B$ for $i = n+1$ segments, with T_B the time between two segments transmission.

Table I. Idleness and delivery time in function of the connection size

	10 segments	11 segments
Terrestrial case RTT = 100ms	Idleness: 88% Delivery: 60,8ms	Idleness: 94% Delivery: 162ms
Satellite case RTT = 500ms	Idleness: 98% Delivery: 310,8ms	Idleness: 99% Delivery: 912ms

Table I illustrates the inefficiency of above start-up mechanisms when the connection size is larger than the IW size and highlights the unfairness introduced by the RTT duration. In this case, a bottleneck bitrate of 10Mb/s has been considered, which implies a T_B of 1,2ms for 1.5KB segments.

C. Illustration in a satellite context

Satellite systems are an attractive solution for communicating with large and unreachable areas using a minimum of infrastructures and deployment. Major Internet players such as Google and Facebook are actively considering them as the solution for offering Internet access to as many people as possible, particularly in Africa. However, the long RTT (around 600ms in the geostationary case) inherent in this kind of communication greatly affects regular Internet protocol stacks and especially the TCP/IP model to the points where the satellite community had no other choice but to develop a satellite specific solution.

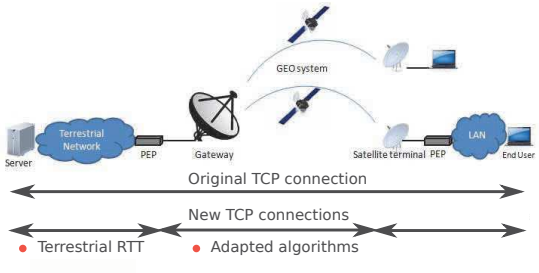

Figure 2. T-PEP in a satellite environment

As illustrated by Figure 2, this dedicated solution consists in introducing middleboxes known as Transport Performance-Enhancing Proxies (T-PEP), around the satellite segment. These T-PEPs intercept the original connection and split it into several new connections in order to hide the satellite segment, and notably its delay, to the end-users. This solution offers

very good performance but at the cost of the break of the end-to-end TCP paradigm, which has for direct consequences to prevent the integration of the satellite technologies in hybrid contexts, such as the next generation of wireless systems (5G).

In [9], emulation results showed that Initial Spreading competes with T-PEPs for short- and long-lived connections, (that means most connections), in congested and uncongested networks. Thus, Initial Spreading is eventually presented as an end-to-end alternative to the T-PEPs. However, the traffic evolution described in II-A mitigates this analysis and suggests again that the use of disruptive middleboxes is unavoidable.

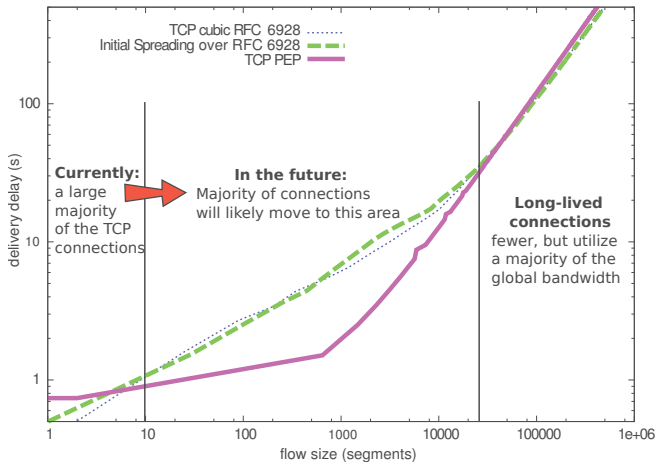

Figure 3. Performance comparison in an uncongested network

Figure 3 has been obtained using our implementation of Initial Spreading on Linux TCP hosts, an emulator satellite platform [12] as bottleneck link, and some commercial T-PEPs. The average delivery duration of a connection is plotted as a function of the flow size. It confirms that RFC6928, with and without Initial Spreading, is inefficient in an uncongested network when the connection size is larger than the IW. In the same time, the T-PEP solution keeps it remarkable level of performance.

Due to their complexity, satellite communications are ideal to highlight the weaknesses of a TCP solution. In the present case, they also inspired SmartIW design. Indeed, SmartIW profits from T-PEPs analysis, but offers a much more general mechanism, that does not break the end-to-end paradigm.

III. SmartIW: a new way to think the IW

A. What is behind T-PEP efficiency?

Figure 4 compares the transmission of segments during the first RTT with and without T-PEPs. However, as most manufacturers use their own proprietary protocols between their T-PEPs, we will only draw what appears to be common benefits. Observations let therefore see that T-PEPs manage the TCP sender behavior, and notably influence both, the number of segments it will send during the first RTT, and their transmission rhythm. Indeed, based on feedback from the satellite Gateway (buffer occupancy, available bandwidth, congestion ...), the T-PEP before the satellite link controls the generation of new TCP segments by sending acknowledgments

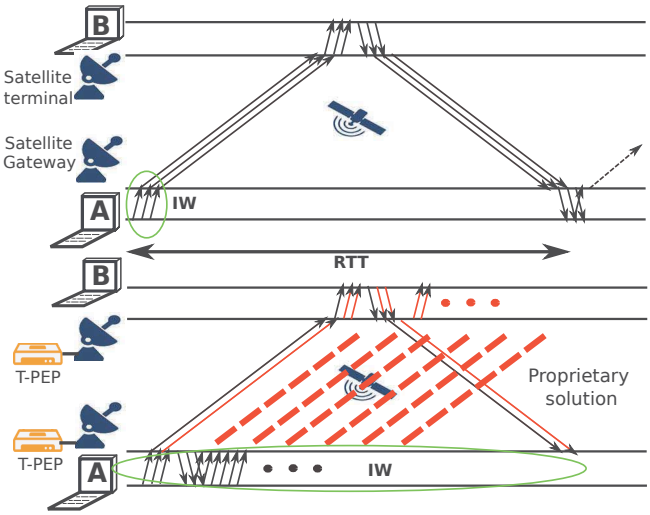

Figure 4. Behavior comparison in uncongested network with and without T-PEPs

at an appropriate rhythm. This way, it can control the end-to-end throughput. From the receiver side, T-PEPs are transparent, and finally, TCP end-user solely notices that an uncommonly large IW has been sent with a particular mode of transmission.

An end-to-end solution can not count on the knowledge of the network state to manage the transmission rhythm of its IW. But, our previous work showed that there exist some end-to-end mechanisms, such as Initial Spreading, that improve the transmission of the large initial batch of segments. Thus, in the following paragraphs, we propose a TCP mechanism, that enables the smart transmission of a larger IW, without breaking the TCP end-to-end paradigm.

B. Definition of SmartIW

In our previous works [7], [9], [10], we have demonstrated that the transmission rhythm and not the size of the initial batch of data sent, is responsible for the end-to-end throughput of a short-lived connection. Thus, we have first observed the limit of the increase in the IW size for transmitting flows shorter than 10 segments in a congested network. Then, we have shown, in the exact same conditions, that managing the large burst transmission, the addition of Initial Spreading lets the IW size be a benefit again.

So, based on this observation, we proposed to relegate the size parameter of the IW to a question of a secondary importance and focused on its transmission rate parameter. Our objective is that the IW size therefore ceases to be the result of an arbitrary trade-off, but becomes the maximal amount of data that can be sent in an efficient way, independently of the congestion level.

We therefore introduced a new spreading function that deals with the time to wait between two segments transmission according to the number of segments already sent. This last function, called $T_{Spreading}(n)$, based on the burst model and analysis presented in [7], aims to adequately spread the segments sent in the first RTT in order to reduce their drop probability and ease their recovery when necessary.

$T_{Spreading}(n)$ is a discrete function that sets the time to wait between the transmission of the segments n and $n-1$. Figure 5 illustrates $T_{Spreading}(n)$ trough 3 different examples. In the first one, all the segments are transmitted at a constant rhythm, while the second function differentiates the transmission between the first 10 segments and the followings. The last one generates an incremental spreading time between two successive transmissions ($T_{Incremental}(n) = T_{Incremental}(n-1) + 1ms$).

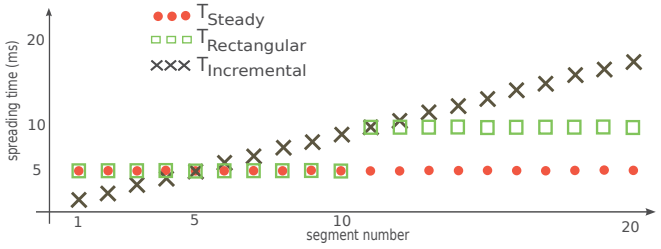

Figure 5. 3 different $T_{Spreading(n)}$

Finally, the combination of the spreading function with the RTT measurement done during the SYN/SYN-ACK exchange, enables to calculate the IW size. This last is therefore the solution of the following equation:

$$RTT(1-m) \geq \sum_{n=1}^{\mathbf{IW}} T_{Spreading}(n)$$
$$m \in [0;1]$$

A margin m has been introduced to deal with a potential inaccuracy of the RTT measurement. Indeed, we recommend to use the average RTT value, inherited from previous connections, when available. But our experiments showed that the introduction of a margin prevents the damage related to a wrong calculation of the IW size in other cases. In the following, we have used $m = 20\%$ as indicative value.

C. SmartIW algorithm

1) The RTT is measured during the SYN-SYN/ACK exchange, or inherited from previous connections
2) The IW size is calculated

Algorithm 1 Calculate IW_{size}

$T_{IW} \leftarrow 0$
$n \leftarrow 1$
while $T_{IW} < RTT\ (1-m)$ **do**
  $T_{IW} \leftarrow T_{IW} + T_{Spreading}(n)$
  $n \leftarrow n+1$
end while
$IW_{size} \leftarrow n$

3) The n^{th} segment of the IW is sent $T_{Spreading}(n)$ seconds after the $n-1^{th}$, with n $\in [1; IW_{size}]$
4) The regular TCP behavior is used as soon as the first ACK is received

IV. Evaluation of SmartIW

In the following, we compare 2 different $T_{Spreading}(n)$ with current TCP mechanisms, such as the RFC6928 with and without Initial Spreading. The behavior of an IW of 50 segments without spreading (IW50) is also plotted to ease the understanding. Indeed, we pay a particular attention to the consequences of the burstiness and size parameters of the IW of each mechanism, on the individual and collective performance. The objective of this paper is not to discuss the optimal spreading function, but, to introduce the remarkable benefits of the global SmartIW concept.

Following results have been obtained using ns2 and the operating method described in [10]. A classic Dumbell topology is therefore used in various environments. The congestion is generated with the establishment of seven long-lived connections thorough the bottleneck, long before the transmission of the new connection of interest. Thousands of iterations are averaged to ensure a good reliability in congestion. This remains a simple but realistic testbed.

Mechanism's performances are analyzed using both the duration it takes to transmit a connection in function of its size, and the actual throughput of this connection. Two distinct scenarios are scrutinized:

- a one-way delay of 250ms, with a bottleneck bit rate of 10 Mb/s and other links bit rates of 100 Mb/s. In both congested or uncongested networks, the RTT, higher than 500ms, enables SmartIW to have an IW larger than 80 segments with T_{Steady} and 45 segments with $T_{Rectangular}$.
- a one-way delay of 25ms, with the same bit rates. Due to the congestion that triggers bufferbloat and increases the average RTT, smartIW respectively enables IW sizes of 21 and 16 segments.

A. Consequences on the individual performance

1) In uncongested network: As expected, the IW size rules the performance (Figure 6). Thus, SmartIW offering a larger IW, is far more efficient than regular RFC6928. While the benefits related to the IW raise are quasi similar with IW50 in this uncongested environment, both SmartIW variants happen to smooth the transmission rhythm. In the following, we study if this distinguishing feature is sufficient to reduce the burst damages that occur in a congested environment and are responsible for preventing further increase in the IW size.

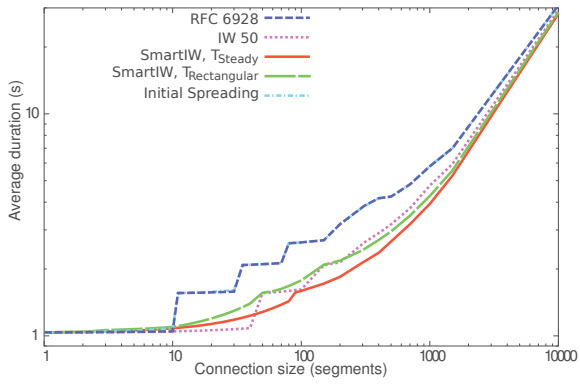

Figure 6. Latency without congestion, one way delay = 250ms

2) In congested networks: Figure 7 & 8 confirm that the aggressiveness of IW50 exacerbates the known drawbacks [7] of IW10 in a congested environment.

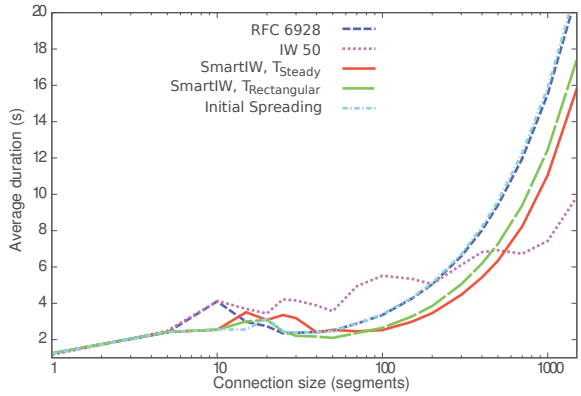

Figure 7. Latency with congestion, one way delay = 250ms

Thus, without SmartIW, the increase in the burst size raises the individual segment loss probability, correlates the losses and decreases the TCP recovery mechanism efficiency. Finally, this leads to a dramatic increase in the average duration, which speaks in favour of a hard limitation of the IW size.

On the other hand, the management of the burstiness of the initial batch of segments, enables both smartIW flavours to weaken the congestion impact on each segment transmission, by notably reducing the losses correlation. Thus, a large number of segments can be sent efficiently during the first RTT, whatever the congestion level. Finally, SmartIW supports the increase of the IW size far beyond current standards.

SmartIW effects are emphasized by the incidence of the different spreading functions on the average duration. Thus, while a more conservative burst management, such as the one induced by $T_{Rectangular}$, appears to be more efficient for transmitting shorter connections, the larger IW size offered by T_{Steady} benefits the longer ones. In fact, while the better achievement rate of the IW transmission is preponderant in the performance of short-lived flows, the larger size of the IW seems to be more profitable for longer connections.

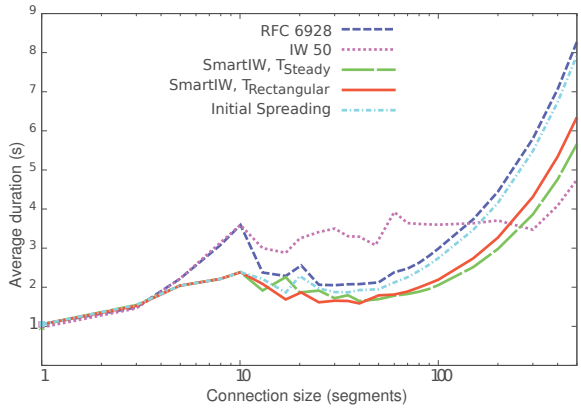

Figure 8. Latency with congestion, one way delay = 25ms

Figure 8 also shows that subjecting the IW size to the spreading function and the RTT measurement notably enables SmartIW to be efficient with shorter RTTs, and to not suffer from an inadequate and unproductive aggressiveness.

Finally, using SmartIW enables to smartly raise the IW size when it is suitable, and so, to spare some RTTs without suffering from the bursts inconveniences. In the following, we study, whether the important benefits in terms of individual performance, offer by SmartIW in both congested and uncongested environments, are not at the detriment of the other connections.

B. SmartIW fairness

We now evaluate the fairness of SmartIW, by studying the consequences of the different fast start-up mechanisms on the other connections performance. Thus, we plot, on the same figure, the throughput of both the new connection and the concurrent connections (per connection), in function of the size of the new connection and the mechanism it used. For each size of the new connection, the throughput of the concurrent connections is measured between the establishment and the end of the new connection.

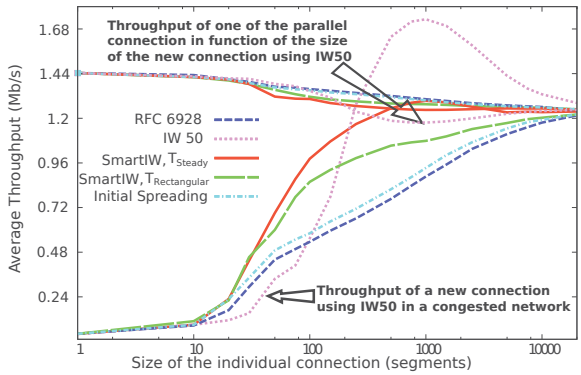

Figure 9. Throughput with congestion, one way delay = 25ms

We can observe on the figures 9 & 10, that the 10Mb/s of bit rate offered by the bottleneck are first fairly split between the seven concurrent connections. Then, the increase in the size of the new connection, but also the choice of the fast start-up mechanism, reduce the throughput of the other connections in different manners. Finally, each of the 8 connections approaches a fair share of the available bandwidth when the new connection is long enough, independently of the mechanism that has been used. We therefore focus on what happens in the intermediary phase, i.e., in between the two phases of fair share of the resources.

Figure 9 first confirms previous conclusions on the individual connection performance. Using IW50 therefore results in the lowest bit rate for short connection sizes, without any impact on the concurrent connections. But, regarding longer connections, the connection with IW50 appears to reach a very high bit rate by downgrading the performance of the concurrent connections. The TCP traces analysis gives an explanation to what occurs. First, the large initial burst floods the bottleneck buffer and causes the entry of most connections into Fast Retransmit and Fast Recovery mode. Then, its numerous in-flight segments monopolize an unfair portion of the available bandwidth.

On the other end, both SmartIW functions solely accelerate the fair share of the available bandwidth. The new connection does not downgrade the other connections performance more than necessary. Benefits are therefore up to 50% for intermediary size of connections, in comparison with RFC6928.

Regarding Figure 10, the IW size induced by T_{Steady} is too important, and an unfairness is also noticeable. $T_{Rectangular}$, which softens the aggressiveness of SmartIW in case of long RTTs, is therefore more appropriate. This last still enables to double the bit rate allowed by the current solution (RFC6928) for intermediary flow sizes, while being fair with the other connections.

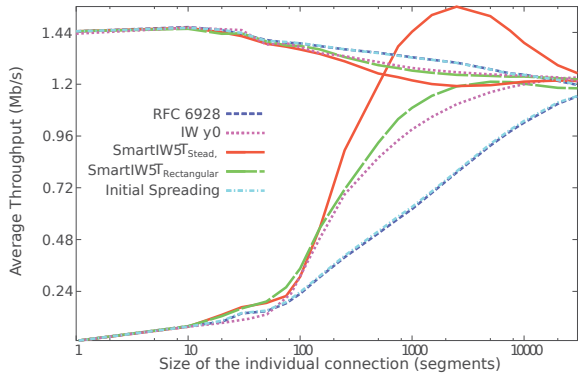

Figure 10. Throughput with congestion, one way delay = 250ms

In conclusion, on the contrary of IW50 or even the RFC6928, SmartIW does not suffer, in a congested environment, from a performance deterioration due to the transmission of the large initial burst. This therefore authorizes an increase in the IW size. But, an in-depth analysis has shown that, in certain scenarios, some SmartIW flavours are too aggressive, and may be unfair with the already established connections. Nevertheless, the model of $T_{Rectangular}$ shows that the $T_{Spreading}$ function can easily take into consideration the most constraining scenarios. Thus, $T_{Rectangular}$ softens the aggressiveness of T_{Steady} in case of long RTT, and enables significant improvements in every considered scenarios, with a maximal fairness. Previous figures eventually showed that SmartIW just accelerates the fair share of available resources, by reducing the impairments caused by the RTT on the slow start efficiency.

V. Conclusion

Setting a limit to the TCP IW size is a complex problem. Indeed, a same value could be too conservative and vainly limit the throughput in an uncongested network, or, be too aggressive and have detrimental consequences on both individual and collective connection's performance in a congested environment. Based on this observation, this paper proposes a new way to think the IW that relegates the IW size to a secondary concern, and focus on how the TCP end-user is going to send the data.

Indeed, our previous studies on Initial Spreading revealed that it is not the amount of data sent during the first RTT that determines the performance of the IW, but the way they are transmitted. Thus, SmartIW proposes to jointly use the RTT measurement and a spreading function that sets the time to wait between the transmission of two successive segments, in order to send an initial batch of TCP segments without suffering from the congestion inconveniences. Finally, the IW size ceases to be the result of an arbitrary trade-off, but is the maximal amount of data that can be sent in an efficient way, independently of the congestion level.

Simulation results showed that SmartIW authorizes larger IW size than the standardized value (RFC6928) to not be similarly affected by the congestion, but to remain fair with the other connections. This notably provides significant performance improvement in both congested and uncongested networks. Furthermore, considering that a long RTT is an opportunity to safely send more segments in the first RTT, SmartIW reduces the unfairness introduced by the RTT duration during the slow start stage of TCP and is notably an appropriate solution for technologies suffering from a long RTT, such as the satellite communications.

Finally, this paper presents the remarkable benefits of using SmartIW instead of a constant IW size, but do not recommend a specific spreading function. Indeed, as for the congestion control algorithms, it seems to the authors that several $T_{Spreading}(n)$ can be proposed that fit the different web usages. Future works therefore aims at analyzing the consequences of the different spreading behaviors in more complex scenarios, in order to propose new refined spreading functions but also to ease the creation of new ones.

References

[1] F. F.-H. Nah, "A study on tolerable waiting time:how long are web users willing to wait?" *Behaviour & Info. Tech.*, vol. 23, no. 3, pp. 153–163, 2004.

[2] Y. Elkhatib, G. Tyson, and M. Welzl, "Can SPDY really make the web faster?" in *Networking Conference, 2014 IFIP*, June 2014, pp. 1–9.

[3] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing TCP's initial Window," RFC 6928, IETF, Experimental, Jan. 2013.

[4] 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.

[5] Initcwnd settings of major cdn providers. [Online]. Available: http://www.cdnplanet.com/blog/initcwnd-settings-major-cdn-providers

[6] J. Gettys, "IW10 considered harmful," Working Draft, IETF Secretariat, Internet-Draft draft-gettys-IW10-considered-harmful-00.txt, Aug. 2011.

[7] R. Sallantin, C. Baudoin, E. Chaput, F. Arnal, E. Dubois, and A.-L. Beylot, "A TCP model for short-lived flows to validate initial spreading," in *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*, Sept 2014, pp. 177–184.

[8] R. Sallantin, C. Baudoin, E. Chaput, F. Arnal, E. Dubois, and A. Beylot, "Safe increase of the TCP's Initial Window Using Initial Spreading," Working Draft, IETF Secretariat, Internet-Draft draft-sallantin-tcpm-initial-spreading-01, Oct. 2015.

[9] R. Sallantin, C. Baudoin, E. Chaput, E. Dubois, F. Arnal, and A. Beylot, "An end-to-end alternative to TCP PEPs: Initial Spreading, a TCP fast Start-Up mechanism," *International Journal of Satellite Communications and Networking*, p. to appear, 2015.

[10] R. Sallantin, C. Baudoin, E. Chaput, F. Arnal, E. Dubois, and A.-L. Beylot, "Initial spreading: A fast Start-Up TCP mechanism," in *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, Oct 2013, pp. 492–499.

[11] A. Allman and S. Floyd, "Increasing TCP's Initial Window," RFC 3390, IETF, Proposed Standard, 2002.

[12] Opensand: A satellite telecommunication system emulation platform. [Online]. Available: http://opensand.org/