Congestion Control Without a Startup Phase

Dan Liu[†], *Mark Allman*[‡], *Shudong Jin*[†], *Limin Wang*[¶]
[†]Case Western Reserve University,
[‡]International Computer Science Institute,
[¶]Bell Labs

Abstract— **The traditional TCP congestion control scheme often yields limited performance due to its conservative Slow Start algorithm, which can require many round-trip times to reach an appropriate sending rate that well utilizes the available capacity. A number of techniques in the literature address this issue, but we offer a different approach that is simple yet blunt: allow TCPs to begin transmission at whatever rate they deem appropriate. We term this technique "Jump Start". While Jump Start simply removes the traditional startup phase, the remainder of TCP's congestion control algorithms (additive-increase/multiplicative-decrease and timeout mechanism) remain intact. Our approach in this paper has two components. First, we attempt to understand the potential implications of removing TCP's startup phase. Second, we step back and attempt to identify network characteristics and techniques that may make Jump Start amenable to real networks. This paper represents an initial exploration of these ideas.**

I. Introduction

Traditionally, TCP congestion control begins transmission by probing for an appropriate sending rate using the Slow Start algorithm [1]. The general notion of Slow Start is to begin transmission with a conservative sending rate that is assumed to be appropriate for the vast majority of network paths (a handful of segments per round-trip time). From this modest starting point, Slow Start increases the number of segments transmitted, which is controlled by a *congestion window (cwnd)*, during each subsequent round-trip time (RTT) by a factor of two[1]. Therefore, if a connection needs a *cwnd* of N segments to fully utilize the network capacity, Slow Start will take $log_2 N$ round-trip times to build an appropriate *cwnd*. Finally, an application must send $2N$ segments worth of data to fully open *cwnd*, meaning that transfers of less than $2N$ will be doomed to suboptimal performance.

High-bandwidth and long-delay both increase the size of N required to fully utilize the available capacity across a path. As N increases, Slow Start becomes a time consuming and data intensive process to determine the appropriate sending rate. For instance, a TCP connection using 1500 byte packets over a 10 Gbps network path with a RTT of 100 msec will require a *cwnd* of over 83,000 segments. Even with Slow Start's exponential growth such a connection will require over 1.5 seconds to start fully using the available capacity. In addition, over 100 MB of data will be required to be transmitted before the *cwnd* is fully opened.

[1]Depending on whether the receiver uses delayed acknowledgments [2] and particulars of how the sender increases the *cwnd*, the rate the *cwnd* increases may not be two. For the clarity of our high-level discussion here we ignore this possible difference and assume the increase factor is two.

When a connection has little data to send, such that the data fits in the standard initial *cwnd*, Slow Start is not a hindrance to performance. Likewise, when a connection transmits a massive amount of data over the course of a long period of time, Slow Start becomes a small transient at the beginning of the connection and does not have a large impact on overall performance. Therefore, faster startup mainly aids medium-sized connections that take normal TCP longer than one RTT, but not so long as to relegate Slow Start to a small transient of the overall connection.

Three broad categories of mechanisms have been introduced to mitigate the performance problems caused by TCP's slow startup phase. The first category includes schemes that attempt to use bandwidth estimation techniques to assess the available capacity of the network path without consuming the capacity. For instance, in the *Swift-Start* algorithm [3] a volley of 4 segments is initially transmitted into the network and the packet-pair approach is used to estimate the available bottleneck capacity, B. TCP then uses B and the measured RTT to form an estimate for the appropriate *cwnd* size. [4] shows that network dynamics can often skew these kinds of bandwidth estimates.

The second category of mechanisms shares path capacity information between connections such that all connections do not have to probe the network path to independently determine an appropriate sending rate. A connection that starts between two peers that have not communicated recently and therefore have no recent path state history will use the standard Slow Start mechanism. However, parallel connections or connections that start soon after another connection to the peer has ended can leverage the previously learned *cwnd* to short-circuit Slow Start. The Congestion Manager [5] is an example of such a mechanism.

The last category consists of mechanisms that use help from each network element along the path to derive a sending rate that is explicitly allowed by the entire path. Quick-Start [6] calls for the advertisement of a desired sending rate in the SYN segment. Each hop in the path then must either agree to the desired rate or reduce the rate to an agreeable value. When each router checks off on a rate then the TCP sender can immediately increase *cwnd* to the given level. When one or more routers on the path do not agree to a rate the TCP sender must use standard Slow Start. (A router that does not understand the option is assumed to veto the use of a larger-than-standard initial *cwnd*.)

These schemes all have their pros and cons. In this paper, we present a different approach to the problem. We introduce

Jump Start in which a startup phase is not required and a host can begin transmission at an arbitrary rate, with the major caveat that the remainder of TCP's congestion control algorithms remain in place. In some sense, Jump Start and Quick Start are at opposite ends of a spectrum. On the one end, Jump Start is a simple scheme that makes an assumption that the network can handle traffic without carefully probing for an appropriate sending rate. On the other end of the spectrum, Quick Start carefully checks with all elements along the network path for explicit permission to begin transmission with a large sending rate.

In § II we outline Jump Start in a bit more detail. Obviously, Jump Start takes the risk of using a large initial sending rate which may cause problems for the connection itself and for traffic competing for resources at congested routers. As we show in § III there are clearly scenarios whereby Jump Start aids performance and others whereby Jump Start can aggravate an already congested network by causing more harm than good. In § IV we take a step back and look at several factors that may mitigate some of the problems that are possible when using Jump Start. This paper presents an initial discussion of Jump Start. Our goal is to gain community feedback while undertaking a more rigorous and well-rounded evaluation of the proposal. Therefore, the careful reader will find many lacking aspects of the evaluation, as noted in § V.

II. Jump Start

Jump Start is a sender-side change to TCP's congestion control algorithms[2]. After the three-way handshake the TCP determines how many data packets, D, can be transmitted as the minimum of the receiver's advertised window and the amount of data queued locally for transmission. As a matter of course, the sender takes an RTT sample from the three-way handshake. The sender then paces the D packets over the first RTT. Jump Start terminates when all D packets are transmitted or when an acknowledgment (ACK) arrives (indicating an RTT has passed). At this point the TCP switches to TCP's normal congestion control algorithms (if more data is available to send), including normal timeout handling mechanisms.

In the case of loss from the first RTT of data transmission, standard loss recovery algorithms are used. In our simulations, we use a scheme based on selective acknowledgments (SACKs) [7]. Upon loss detection the congestion window (*cwnd*) is halved, per normal TCP [1]. However, at the end of loss recovery, we further reduce the *cwnd* to reflect the possible over-aggressiveness of Jump Start. During loss recovery we count the number of retransmissions, R. At the end of loss recovery we set $cwnd = \frac{D-R}{2}$, which is roughly half of the load the network was able to support. This is akin to the TCP's normal response during congestion avoidance when the load imposed on the network is slightly too much (due to linear *cwnd* growth) at which point the *cwnd* is halved.

[2]We discuss Jump Start in terms of TCP, but since it is an algorithmic change the results should be applicable to other transports that use TCP-like algorithms.

III. A First Look

We take a first look at Jump Start in this section to show both that it can be a benefit to performance and then that it can be a detriment. The simulation results presented in this section are meant to be *illustrative* but not comprehensive. We are currently working on a well-rounded study of Jump Start that involves a variety of networks and traffic models. However, within the space constraints of this paper our goal is to simply illustrate some of the pros and cons to start a discussion on TCP startup. Our intention is not to over-claim these results as being the final word on the topic.

A. Simulation Setup

We constructed a simulation using *ns-2* that uses a dumbbell topology with an RTT of 64 msec and a bottleneck capacity of 5 Mbps. The routers at the bottleneck employ drop-tail queuing with a queue size of 60 packets (a common default, confirmed by [8]). The simulations consist of either all Slow Start or all Jump Start traffic. The Slow Start connections use an initial window of 3 packets, per [9]. The Jump Start connections send their entire transfer (number of packets configured by the experiments) in their first RTT. An advertised window of 10,000 packets is employed and never comes into play in our simulations. The *sack1* TCP variant is used for these simulations. We use 1500 byte packets, so each packet can hold up to 1460 bytes of data.

B. Benefits

Our first simulations involve a single flow to illustrate that Jump Start indeed can offer performance improvements. Figure 1 shows the percent improvement (in terms of connection duration) for Jump Start as a function of the transfer size. When the transfer size is 1 packet both schemes perform identically. When the transfer size is 2 packets Slow Start performs better than Jump Start simply because Jump Start paces out the two packets while Slow Start simply transmits them as soon as possible[3]. For transfer sizes over 2 packets we see 10–40+% performance improvement from using Jump Start[4]. No drops occurred in any of these simulations.

C. Drawbacks

We next turn our attention to a congested situation where Jump Start reduces the performance of the traffic on the network. We base another set of simulations on the framework outlined above with several changes. We connect 100 nodes to each side of the 5 Mbps bottleneck with 50 Mbps links. Each node acts both as a source and as a sink, and one TCP connection is set up between a source and sink pair from alternating sides of the bottleneck link. Each of the 200 connections sends 200 packets. TCP connections are started

[3]This suggests a straightforward change to Jump Start by not using pacing when the initial burst of traffic is allowed by TCP's standard initial window size of 4380 bytes [9]. Given the space constraints of this paper we do not further consider this small refinement.

[4]Increasing the transfer size to the next power-of-2 causes Slow Start to outperform Jump Start. We purposely leave discussions of the drawbacks of Jump Start to the next subsection.

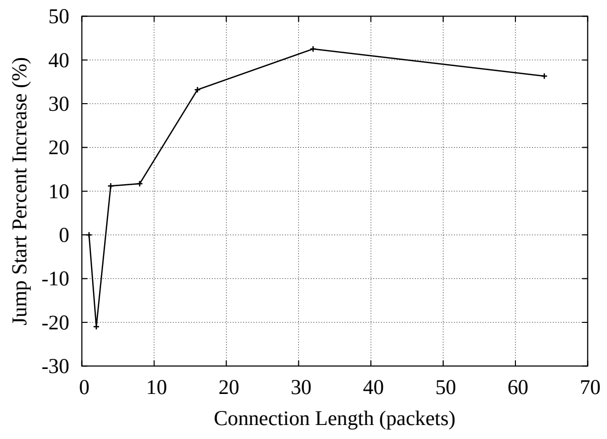


Fig. 1. Percent improvement for Jump Start when compared to Slow Start across a variety of transfer sizes.

on each node at varying periodic intervals (as shown in the plots below). The interval between connections controls the amount of contention for the bottleneck capacity.

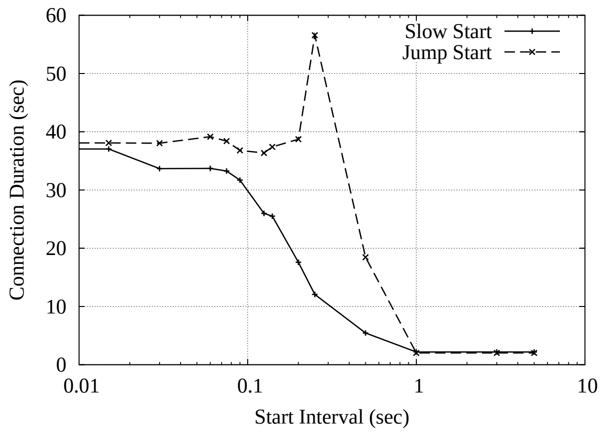
Figure 2 shows the median connection duration and drop rates as a function of the interval between connection startup. The plot shows that once the interval between connections is at least 1 second, Jump Start and Slow Start perform similarly (with the scale of the plot obscuring Jump Start's ≈7% performance boost in these no-contention cases). The results show that standard Slow Start enjoys better performance in cases where there are multiple connections contending for resources. The plot of drop rates shows the reason. Even in the case where there is no contention from other connections, Jump Start's loss rate is over four times that of Slow Start. We observe that both Slow Start and Jump Start can burst enough to cause overflow even in the absence of contention between connections and that Jump Start aggravates this condition. When there is contention between connections Jump Start's loss rate is roughly twice that experienced by Slow Start, again explaining the performance disparity. These simulations confirm our intuition that Jump Start *can* degrade individual connection's performance and also increase the overall congestion level on the network.

D. Summary

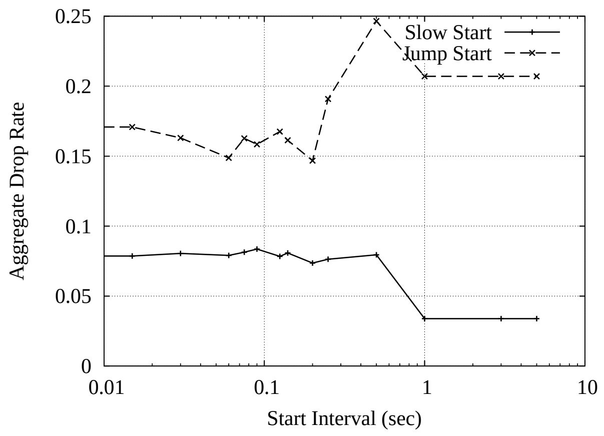
The simulations presented in this section are not meant to represent a complete and well-rounded set of experiments. Rather, these simulations serve to illustrate that Jump Start can both increase and decrease performance, as intuition would suggest. Our future work includes a more well-rounded evaluation of Jump Start with different network setups (bandwidths, RTTs, number of congested gateways, etc.), as well as more realistic traffic patterns.

IV. Coping

In this section, we discuss a number of ways in which the network already does or might cope with TCPs that use Jump Start. None of the items discussed in this section fully mitigate the potential impact of Jump Start, but taken together they may offer a glimmer of hope that more aggressive startup is feasible.



(a) Connection duration.



(b) Aggregate drop rate.

Fig. 2. Slow Start vs. Jump Start when the amount of contention for the bottleneck capacity varies.

A. Traffic is Heavy-Tailed

The first observation we make about coping with Jump Start is the well-founded notion that traffic is heavy tailed. That is, most of the connections cannot place a large burden on the network because they transfer only a small amount of data. As a data point we analyzed traffic from ICSI's border for one day (July 27, 2006). We observed over 3.1 million TCP connections over the course of the 24 hour period. Of these, roughly 1.2 million were valid connections (as opposed to scans, probes and backscatter for which no connection was actually established). We used the RTT from the three-way handshake and the total amount of data sent in each direction to form a Jump Start rate for each direction of the 1.2 million connections[5]. This is the rate required to transmit the entire transfer in one RTT. From the 1.2 million connections we

[5]This is the worst-case in that it assumes that when data transmission starts all the data that will be transferred is available to be sent.

obtained almost 2.3 million rates (meaning that roughly 100K connections did not send data in one of the directions). Of these, only 169K transfers required an initial congestion window of more than the 4380 bytes provided by [9]. In other words, roughly 7.4% of the connections at ICSI's border are currently controlled by the initial congestion window. This shows that the fraction of connections that would benefit from faster startup is small, but also that if Jump Start were used the fraction of connections imposing a higher load would also be small.

Figure 3 plots the rates of the 169K connections that could make use of Jump Start as a function of the amount of data transmitted on the given connection[6]. The bulk of the connections either (i) send at less than 10 Mbps or (ii) send less than 1 MB of data. As a point of comparison, consider that a normal connection using an initial window of 4,380 bytes over a path with an RTT of 50 msec would transmit at roughly 80 KB/second in the absence of *cwnd* growth. A sizable fraction of the points in the plot lie below 80 KB/second. Especially when the RTT is large, the initial sending rate of Jump Start can remain reasonable.

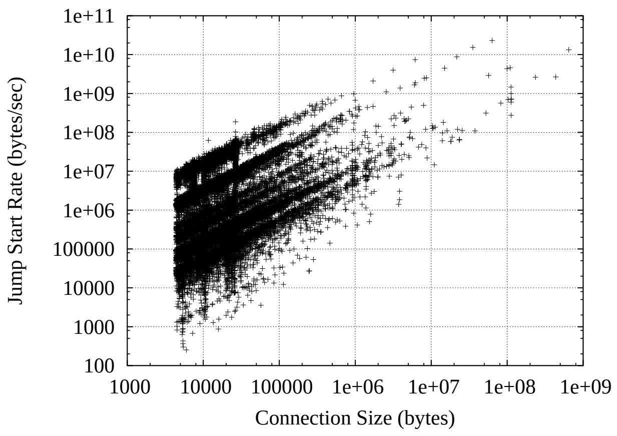


Fig. 3. Observed initial Jump Start rates as a function of transfer size at ICSI's border.

We next turn our attention to simulations that are meant to further explore Jump Start's behavior under a more realistic traffic model than used in § III. We simulate typical HTTP 1.0 transactions over a simple dumbbell topology that consists of one client and one server. We use the same settings outlined in § III except that we use a 10 Mbps bottleneck. We use *ns-2*'s web traffic generator with parameters set to approximate reality as reported in [10]. Specifically, our configuration uses 50 web sessions consisting of 5 web pages per session on average. The size of each web page follows the Pareto-II distribution with an average of 4 web objects and a shape parameter of 1.5. The inter-page time is exponentially distributed with an average of 500 msec. The size of each web object follows the Pareto-II distribution with an average of 4 packets and a shape parameter of 1.2. The inter-object time is exponentially distributed with an average size of 50 msec. The time interval between web sessions is set to 5 msec.

Figure 4 shows the distribution of connection duration and per-connection drop rates for the Web traffic experiments. Similar to the scoping employed above, these plots are winnowed to connections that carry more data than would fit in a standard TCP initial *cwnd*. The results show that connections with short durations are aided by Jump Start with the 25^{th} percentile duration showing that Slow Start takes more than twice as long as Jump Start. Once the connection duration reaches approximately 1 second Slow Start and Jump Start show similar performance. At this point a connection has been open for roughly 16 RTTs and so has hit steady state and the impact of the startup phase is reduced. The drop rate results show that 10% more Jump Start connections have no loss when compared with traditional Slow Start connections. We attribute this to more Jump Start connections "getting out of the way" quicker, as the distribution of connection durations shows. In other words, the data is transferred quickly, and while the data consumes network resources, it does not spread the resource usage out over a lengthy period of time. As a result, congestion does not last as long with Jump Start, where, in some cases, the queue can absorb Jump Start's burst. In general, however, Jump Start does show a higher loss rate than Slow Start, in many cases increasing the loss rate by approximately two-fold. This is expected since many studies have shown the loss rate to be correlated with the size of the initial window (e.g., [11]).

While more aggressiveness leads to a higher loss rate, it does not necessarily yield lower performance in these simulations. The reason for this lies in Jump Start's aggressiveness, which puts the connection "ahead" of Slow Start. For instance, consider Jump Start sending a stream of 100 segments in the first RTT of a connection into a network with a loss rate of 50%. Even though it will take TCP a bit of time to retransmit 50 segments, successfully transmitting 50 segments would require 4–5 RTTs in Slow Start and so Jump Start is well ahead.

B. *Active Queue Mamanegment*

Another way the network could cope with end hosts using more aggressive initial sending rates is to employ active queue management (AQM). [12] advocates the use of AQM, with one of the reasons being to better absorb bursts of traffic. Using AQM, a network that operates with small average queue sizes even when congested would be in a better position to absorb Jump Start induced bursts of traffic and protect competing connections, even if the burst was inappropriately large. AQM schemes would be especially useful if they calculate a flow's drop probability based on the flow's arrival rate (e.g., CHOKe [13]).

C. *Edge Bandwidth Limits*

Even though Jump Start can be overly aggressive, the network is set up in such a way that any congestion caused by Jump Start is likely to occur towards the edge of the network. For instance, the higher rates given in Figure 3 would not all be possible given that ICSI's link to the broader network is

[6]The plot actually shows a sample of the data which is obtained by plotting every fourth connection in the dataset. This is done to reduce the size of the plot and visually does not change the character of the plot.

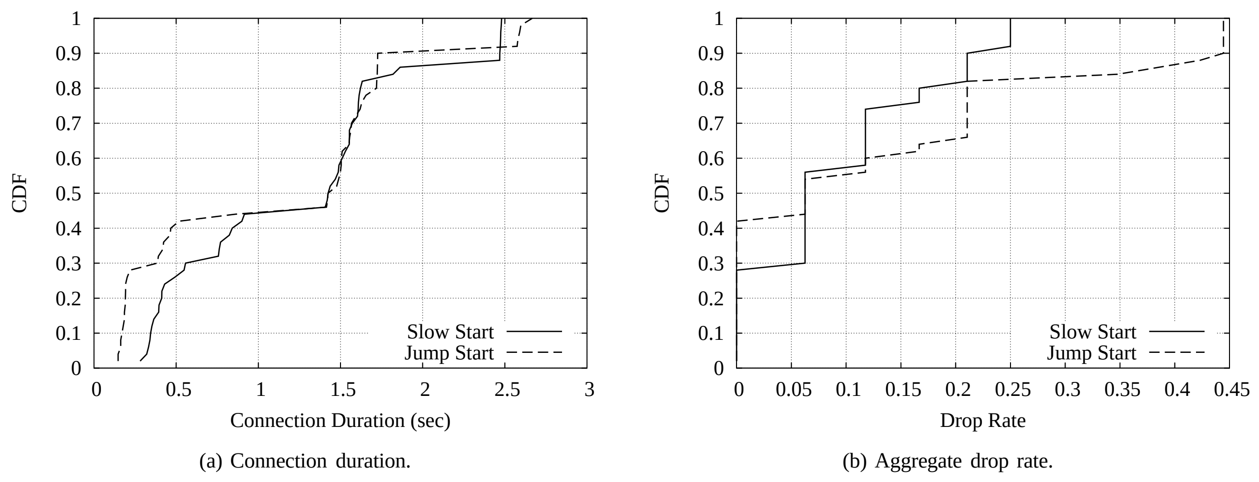


Fig. 4. Distributions of connection duration and drop rates for Slow Start vs. Jump Start using a web traffic load.

100 Mbps. Commonly held wisdom is that backbone networks are massively over-provisioned such that being overwhelmed by the edges would be difficult and would require many hosts from many points in the network to be sending at high-rates simultaneously. Jump Start is a one RTT per connection event the backbones would likely absorb. By way of evidence, [14] uses the sending pattern of the Witty worm (observed at two network telescopes) to show the effective bandwidths of infected hosts. The paper shows that 75% of the infected hosts have an effective bandwidth of at most 1 Mbps. So, it may be that Jump Start's impact will largely be felt locally—where it may be more reasonably dealt with—as opposed to globally.

In addition, a receiver can control the amount of data a sender can transmit in the initial window of data via the advertised window. Therefore, if the receiver has some notion of the maximum bandwidth on its edge of the network, the advertised window setting could be set to some reasonable initial value. As a data point, Figure 5 shows the distribution of the rate reduction that would be imposed by taking into account the advertised window in the 169K connections that could benefit from using Jump Start in the ICSI data described above. We note that almost 75% of the connections would not be hampered by the advertised window. Of the rate reductions that we observe, however, some are quite large—with over 13% of the reductions being over 1 megabyte/second.

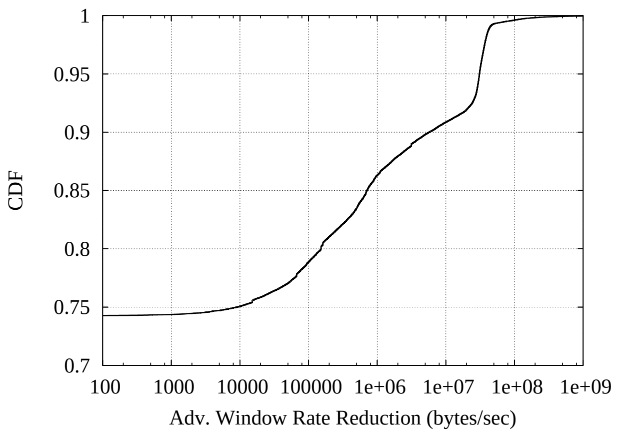


Fig. 5. Distribution of the rate reduction if the advertised window is taken into account.

D. Policy

Just because Jump Start frees an end system from the standard initial *cwnd* and Slow Start procedure, it does not mean end hosts *must* transmit data as fast as possible. In particular, busy servers and networks may want to cap the data rate they wish to initially allocate to any one particular flow, as a *policy decision*. For instance, for the ICSI case noted above a policy could be implemented that no connection could initially send at more than 500 Kbps (or 0.5% of the bandwidth from ICSI to its service provider).

In addition to basing policy decisions on sending rates, policy could be implemented in terms of a *target* transmission time. For instance, an application may request that data be transmitted within 500 msec. The TCP could then combine this goal with the RTT measurement taken while setting up the connection to choose an initial congestion window that would meet the given target.

E. End System Hints

An additional way of coping with Jump Start would be to have end systems using Jump Start (i.e., sending above the RFC 3390 [9] mandated initial *cwnd*) to flag the packets in the initial window as such. This would allow congested routers along the path to preferentially drop packets from large initial congestion windows. It seems counter-intuitive to advertise a feature that may ultimately lead to more packet drops for the given flow. However, maybe by highlighting this behavior and giving the network additional insight, Jump Start would be viable on a wide scale because competing flows could be protected.

F. Congestion Control

We also stress that Jump Start *does not* eliminate end host congestion control. While allowing for an arbitrary initial sending rate, the remainder of TCP's additive-increase multiplicative-decrease congestion control system remains in place to govern steady-state transmission behavior. In addition, we do not change the response to a retransmission timeout, which still collapses the congestion window to 1 segment and then uses the Slow Start algorithm. Even if a connection is overly aggressive in its first RTT of data transmission, this aggressiveness will not persist throughout the connection, but will represent a transient load imposed on the path.

G. Scoped Environments

Finally, we note that Jump Start may be especially useful in enterprise networks that have a low degree of congestion. [15] studies one particular enterprise network and shows the available capacity is generally significantly underutilized. While [15] is not a comprehensive study, we believe that enterprise-level networks are often over-provisioned for the relevant application demands. In such an environment an increase in the initial aggressiveness of connections that transmit appreciable amounts of data may well use *spare* capacity in the network rather than unfairly competing for congested resources.

V. Future Work

The general topic of quickly using the available capacity of a network path is getting more crucial as high delay-bandwidth product networks and data intensive applications become more prevalent. This paper presents an initial sketch of Jump Start, which eliminates TCP's startup phase. As shown, Jump Start has its good and bad points. We offer some evidence that indicates it may be viable. However, future work is required to better understand the how Jump Start's cost-benefit tradeoffs will ultimately manifest themselves in real networks. Specific future work includes: (i) more comprehensive experiments (more realistic traffic patterns, a variety of network bandwidths, various RTTs, etc.), and (ii) investigation of fairness issues with a mix of Jump Start and non-Jump Start traffic. Such experiments will provide a much broader and deeper view into the implications of using Jump Start. Running real-world experiments will also be key in gaining an understanding of Jump Start. We also note that future investigation should consider the security impacts of Jump Start, with a particular eye towards attackers using Jump Start to coax some server into sending very high-rate bursts of traffic to facilitate pulsing attacks. Finally, we stress that we are not claiming Jump Start to be *the* solution, but rather a possible solution. We encourage researchers to broadly explore alternate methods for aiding startup over high-bandwidth and long-delay networks.

Acknowledgments

This paper has benefited from discussions with many people, including Sally Floyd and Pasi Sarolahti. Mark Allman's work is funded in part by the National Science Foundation under grant ITR/ANI-0205519.

References

[1] V. Jacobson, "Congestion Avoidance and Control," in *ACM SIGCOMM*, 1988.

[2] R. Braden, "Requirements for Internet Hosts – Communication Layers," Oct. 1989, rFC 1122.

[3] C. Partridge, D. Rockwell, M. Allman, R. Krishnan, and J. P. Sterbenz, "A Swifter Start for TCP," BBN Technologies, Tech. Rep. TR-8339, Mar. 2002.

[4] M. Allman and V. Paxson, "On Estimating End-to-End Network Path Properties," in *ACM SIGCOMM*, Sept. 1999.

[5] H. Balakrishnan, H. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts," in *ACM SIGCOMM*, Sept. 1999.

[6] S. Floyd, M. Allman, A. Jain, and P. Sarolahti, "Quick-Start for TCP and IP," June 2006, internet-Draft draft-ietf-tsvwg-quickstart-04.txt (work in progress).

[7] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *Computer Communications Review*, vol. 26, no. 3, July 1996.

[8] E. Blanton and M. Allman, "On the Impact of Bursting on TCP Performance," in *Passive and Active Measurement Workshop*, Mar. 2005.

[9] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," Oct. 2002, rFC 3390.

[10] B. Krishnamurty, C. Wills, and Y. Zhang, "Preliminary Measurements on the Effect of Server Adaptation for Web Content Delivery," in *ACM/USENIX Internet Measurement Workshop*, Nov. 2002.

[11] M. Allman, C. Hayes, and S. Ostermann, "An Evaluation of TCP with Larger Initial Windows," *Computer Communication Review*, vol. 28, no. 3, July 1998.

[12] R. Braden and et.al., "Recommendations on Queue Management and Congestion Avoidance in the Internet," Apr. 1998, rFC 2309.

[13] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: A Stateless Scheme AQM for Approximating Fair Bandwidth Allocation," in *IEEE Infocom*, Mar. 2000.

[14] A. Kumar, V. Paxson, and N. Weaver, "Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event," in *ACM/USENIX Internet Measurement Conference*, Oct. 2005.

[15] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney, "A First Look at Modern Enterprise Traffic," in *SIGCOMM/USENIX Internet Measurement Conference*, Oct. 2005.