What Use is Top Speed without Acceleration?

Bob Briscoe
Simula Research Laboratory
Oslo, Norway
bob@simula.no

Mohammad Rajiullah
Karlstad University
Karlstad, Sweden
moharaji@kau.se

Anna Brunstrom
Karlstad University
Karlstad, Sweden
annabrun@kau.se

Andreas Petlund
Simula Research Laboratory
Oslo, Norway
apetlund@simula.no

Bengt Ahlgren
Swedish Institute of Computer Science
Stockholm, Sweden
bengta@sics.se

ABSTRACT

This paper is a call to action. The aim of the paper is to fully characterise the flow-start problem. This involves quantifying the latency penalty of the start-up phase of typical Internet flows through evaluating a model of the state-of-the-art flow-start-up dynamics and characterising typical flow lengths. We have analysed a decade of traces from an Internet backbone as well as traces collected from a point much closer to users to account for CDN traffic. This analysis shows that an ever-growing proportion of typical user's sessions is becoming limited by flow-start limitations, not capacity. Moreover, according to our trace analysis, often in most cases, access links are totally empty when a new flow starts, hinting at the potential of a better flow-start to exploit the idle capacity. The paper further discusses the lack of scalability of the known solutions that would be easiest to deploy—those that only depend on unilateral deployment by one party. This leaves only solutions that require coordinated deployment. The paper avoids siding with any particular solution, but identifies the critical deployment problems that any solution will have to overcome.

1. INTRODUCTION

Internet transport protocols use congestion control [1] to adapt to the available capacity in the path. A congestion control has to be designed on the basis that each time a flow starts the available capacity is unknown. Even a re-start after idling cannot assume that the capacity or other traffic has remained unchanged [2]. Path capacity changes rapidly when the physical capacity changes or as other flows arrive and depart, at least in scenarios with low numbers of flows at the bottleneck. TCP senders universally use some variant of the TCP slow-start algorithm [3,4], which sends an initial handful of packets, waits for feedback, then doubles how much it sends in each subsequent round as long as it has sensed no losses in the feedback. The question of how aggressively to probe for available capacity is not unique to TCP, but TCP's exponential slow-start (with various optimizations) has become widely considered as an acceptable compromise between acceleration and overshoot.

However, slow-start does not scale, because every doubling of target flow rate takes one more round trip to reach. Investment in bottleneck capacity can be used either to carry more flows, bigger flows, or to make each flow faster. Most individuals access the Internet through link capacity dedicated to their device or to their household, which is often idle when a new flow arrives (this paper quantifies this assertion, and finds it is the majority case). Over the past 2–3 decades, Internet access link capacities have consistently doubled every 18–21 months according to four different sources [5]. But such investment in access capacity can only be fully effective if individual flows can accelerate up to the increased top speed and still have enough data left to take advantage of it. Put the other way round, investment in access capacity gives no performance improvement at all for flows smaller than a certain threshold.

Unscalability of slow-start would not be a problem if all transfer sizes were getting larger, but they are not. As we continue to invest in greater link capacity, larger transfers become feasible. But it does not follow that all the demand for the smaller objects disappears—the range of feasible transfer sizes merely widens. We will show that, as link capacities increase, an ever-growing proportion of a typical user's sessions become limited by slow-start; "protocol-limited" rather than "capacity-limited". And, even for flows that are large enough to exploit faster links, the amount of loss experienced during overshoot increases. In the absence of any explicit signalling, all data flows face the same dilemma, whether using TCP or other transports. In the rest of this paper, we shall call this the 'flow-start' problem. [6] identified this as the main remaining open issue of Internet congestion control. The longer we fail to solve this flow-start problem, investing in capacity will make less and less difference to more and more people.

This paper is a call to action. The aim is to characterise the flow-start problem; to quantify how serious it is, to show that it has recently started to become particularly serious, that it is getting worse, that it will always get worse as long as stop-gaps are favoured over concerted agreement on an enduring solution, and finally to articulate the precise deployment problem that any enduring solution has to solve.

We start in sections 2 and 3.1 with two analytical models: i) a model of traffic flow arrivals; and ii) a model of the state-of-the-art acceleration mechanism these flows use today (TCP slow-start with various optimisations). Combining the two models gives a metric for the under-utilisation of capacity, and the effectiveness of upgrading this capacity. The eventual aim is to be able to distil Internet traffic flows into a simple model of all the flow-starts, and to further distil that down to one metric measuring the effectiveness of capacity investment.

The rest of Sec. 3 quantifies how bad the flow-start problem has become by using the models to assess packet trace data collected between 2002 and 2015. The paper then shifts from characterising the problem to assessing solutions. Sec. 4 divides proposed solutions into those designed for ease of deployment (host only) and more ambitious solutions that alter the interface between host and network. Host-only solutions have made slight improvements, but not sufficient to keep up with the growing scale of the problem. On the other hand, those solutions that face up to the scale of the problem have not been deployed—because of the chicken and egg problem of staging host and network deployments when there is no benefit until both parts are in place.

The paper ends with a summary of more generally related research in Sec. 5, followed by concluding remarks and an outline of potential future work in Sec. 6.

2. SCALING MODEL

The following scaling model shows that capacity increases can be used to absorb larger flows or more simultaneous flows, but there is also an expectation that increases in capacity should make individual flows faster. After Kelly [7] we model the change in scale of traffic using three independent scaling factors for:

a**,** the size (i.e. volume) of each flow;

b**,** the rate of a flow of constant size;

c**,** the number of simultaneous flows.

If the capacity growth factor is refereed to as x, then in order to absorb the above three dimensions of traffic scaling, x needs to be equal to $a.b.c$. Note that element a of capacity growth absorbs any scaled-up volume of each flow without changing its completion time while b is an *additional* speed-up, or equivalently the speed-up that a flow of constant size would experience. Increase factor c absorbs more flows while keeping the link utilisation unchanged. Of course, any of the factors a, b, c may take values less than one to represent decreases.

For a given capacity growth factor x, the ideal change in rate (let's call it b^*) is $\frac{x}{a.c}$. However, this is only achievable when flows are capacity-limited. In practice the change in rate is often smaller if flows are protocol-limited. So, given the rate actually grows by factor b, then $\frac{b}{b^*}$ measures the effectiveness of a capacity upgrade.

Each factor in the scaling model is a statistical quantity, with a mean and a distribution. In Sec. 3 we will use packet trace data over the last twelve years to put numbers to these factors. Nonetheless, the past is not always a good prediction of the future. So it is useful to have a generic model to be able to explore other possible ways traffic may scale in future.

3. QUANTIFYING THE PROBLEM

Sec. 3.1 presents a model of TCP slow-start that quantifies how the benefit from link capacity upgrades reduces to nothing for smaller flows. Sec. 3.2 illustrates how an ever increasing proportion of Internet flows are limited by flow-start, not by capacity, based on a longitudinal study of packet traces from an Internet core and traces from an Internet access link. Sec. 3.3 shows how different value-judgements can influence the significance of the flow-start problem, addressing the question of whether we should count benefit per flow, per byte or per something else.

3.1 Model of Slow-Start

To analyse the flow-start problem we model the average transfer rate of a single TCP flow with a dedicated bottleneck capacity. We assume that access links are more often empty than full when a new flow starts, which we will later show to often be true based on our measurements of real traffic. The general approach is to find the number of round trips in which slow-start only partially fills the bottleneck, *before* the round in which either slow-start ends or the flow ends, if sooner. Then any remaining packets will arrive at the bottleneck rate, whether they are all sent in the next round while still in slow-start phase or there are enough packets to progress into the fast retransmit and congestion avoidance phases. Average transfer rate rather than completion time is chosen as the 'figure of merit'. Then, as explained above, for a capacity increase x, the factor b by which the average rate grows can be used to determine the effectiveness of the upgrade. An alternative could have been to use completion time, but average rate also has the advantage that it is meaningful for different flow volumes where a bigger number represents better performance. Besides, anyone can compare an average flow rate directly with well-known bottleneck access link rates. We define average rate as

$$\text{Average rate} = \text{flow volume}/\text{completion time}.$$

In order to quantify how inefficient slow-start is, we want to ensure that we criticise the most aggressive variant of slow-start. Therefore we use state of the art parameters for slow-start, that is:

- Initial window of ten Ethernet-size segments [8], as well as three for comparison;
- Base two exponential increase (i.e. doubling per round);

- A range of round trip times that are realistic for the public Internet, with and without caching;
- No use of hybrid slow-start [4] (because 'HyStart' softens the aggressiveness of slow start, which led to it becoming the default in Linux because it is believed to help general performance, but it does not suit our purposes here).
- No use of multiple parallel flows (because a larger initial window is equivalent), see Appendix B;

The model for the slow-start is derived in Appendix A. For a wide range of sizes of each transfer (1 B to 1 GB), Fig. 1 shows the average rate achieved over a whole transfer, including the slow-start phase[1], on a log-log scale for a few scenarios.

Taking the second plot in the figure as an example (IW=10 segments, R=20 ms, X=80 Mb/s), it shows that unless the transfer size is larger than around 1 MB, it hardly even starts to exploit a dedicated 80 Mb/s link. A 1 MB transfer can only average about 40 Mb/s, while a smaller 15 kB transfer can only average about 3 Mb/s. For transfer sizes less than 500 kB, any capacity increase beyond about 80 Mb/s would make no noticeable difference to average rate (illustrated by comparing the X=80 Mb/s and the X=1 Gb/s plot in the figure).

Furthermore, the last plot of Fig. 1 shows that the problem is significantly worse if the round trip time is longer. For an inter-continental round trip time that is ten times longer (R=200 ms) only flows ten times larger (more than 10 MB) can start to make use of 80Mb/s capacity.

One might think that the dependency between the capacity and traffic volume is the other way round: as capacity grows new applications emerge that exploit it by transferring larger objects—"build it and they will come". In the next section we will see that the distribution of transfer sizes has included larger flows as capacity has grown, but only a very slightly larger proportion of all flows.

3.2 Longitudinal Study of Packet Traces

The scaling behaviour illustrated in Fig. 1 would not be a problem if all Internet flows were getting larger (parameter a in the scaling model in Sec. 2). Next, we show that while increased capacity provides the possibility of larger transfers, the demand for small transfers continues pretty much unchanged.

To establish the proportion of traffic that has been limited by flow-start rather than capacity we have performed a longitudinal study of Internet flow sizes based on Cooperative Association for Internet Data Analysis (CAIDA) [10] packet traces. We also process the traces to characterise the number

[1]But excluding the initial round of handshaking, i.e. optimistically assuming the flow is re-starting without having to hand-shake, e.g. a re-start after a long idle period or using the experimental Fast Open enhancement to TCP [9] after an earlier connection between the same client-server pair.

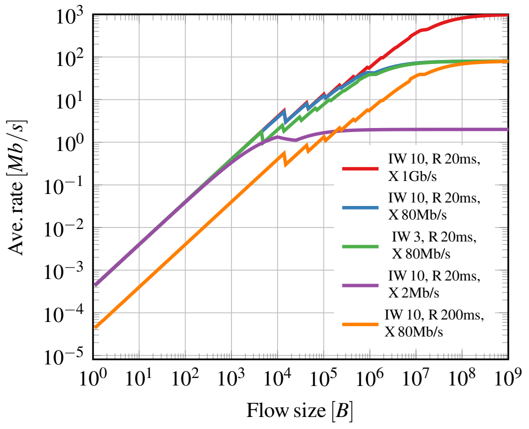
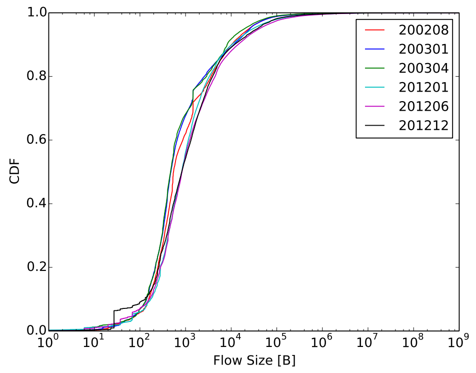
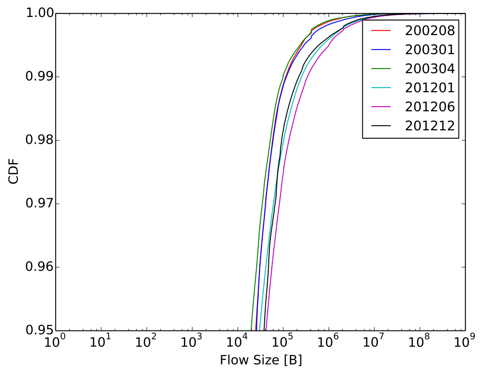


Figure 1: The problem: TCP has to limit the average transfer rate of different size flows despite dedicated capacity X. IW=initial window; R=round trip time.



(a) Whole CDF



(b) Tail of the CDF

Figure 2: Prevalence of flows of different sizes on the Internet, based on Tier 1 traces.

of active flows that are present when a new flow starts in a user's access link. Finally, since content distribution network (CDN) traffic is likely to be heavily under-represented in the Tier 1 ISP (CAIDA) traces, we do the same analysis on a trace collected at an access aggregation link through which all the Internet traffic of each connected user has to pass.

3.2.1 *Tier 1 Packet Traces*

Tier 1 ISP packet traces were recorded by CAIDA's data collection monitors at two Equinix datacenters. Each data set in 2012 was collected at these two monitors connected to bi-directional backbone links of a Tier 1 ISP between Chicago and Seattle, and San Jose and Los Angeles, respectively. Each data set generally consists of twelve monthly samples, taken at the same time of the day. Each monthly sample includes separate traces for each backbone direction. The average size of a monthly dataset varies between 100 and 350 GB. Earlier data sets (2002–2003) were obtained from the ampath-oc12 passive monitor, which tapped two directions of an OC12c link located in Miami, Florida and is no longer in use.

3.2.2 *Flow Size Distribution*

Fig. 2 shows the prevalence of flow sizes in the CAIDA data sets over the decade, 2002-2012 on the same log-scale as used in Fig. 1. As shown in Fig. 2a, the majority of the flows are short[2] and the distribution has hardly changed over time. The prevalence of short flows is also consistent with earlier studies [11–14]. Although large flows are rare in Fig. 2a, if we zoom in on the tail of the flow size distribution, as shown in Fig. 2b; the tail gets slightly longer over time, showing the presence of even larger flows in recent years. Nevertheless, the distribution of typical transfer sizes has hardly changed over the decade. From the CAIDA data, over this decade the annual growth of various statistical parameters of the flow size distribution, a (defined in Sec. 2) is tabulated below, showing that what little growth there is is mainly in the tail.

Flow size growth	mean	50%-ile	90%-ile	99%-ile
factor, a [/yr]	1.13	1.06	1.02	1.10

Over the same period, according to the sources in [5], bottleneck link capacities have grown by factor $x \approx 1.54$ pa or $\times 50$–$\times 100$ over the decade[3], compared to just $\times 3.4$ growth in mean flow size over the decade. This implies that it is becoming critical to improve the effectiveness of flow-start for small/medium flows because they clearly hold an enduring place in the the way users and applications use the Internet.

This can be shown by comparing the distribution of flow sizes in Fig. 2 with the slow-start model in Fig. 1. A 2 Mb/s link-rate plot is included in Fig. 1 to show that capacity increases up to 2 Mb/s (when broadband was first introduced) was effective for a large range of the flow size distribution. Increasing further from 2 Mb/s to 80 Mb/s is still effective, but only for flow sizes above the 75th percentile (any larger than about 3 kB). Further upgrades beyond about 80 Mb/s only have any affect on flow sizes greater than 1 MB, which is where the 1 Gb/s plot can be seen to start rising above the 80 Mb/s plot. That implies no benefit for about 96.6% of flows (using the Dec-2012 flow size distribution).

At this point, it must be clarified that the number of smaller flows is extremely large, so benefit for the remaining 0.4% of flows greater than 1 MB still helps a very large number of flows (and the largest proportion of bytes—see later).

Also, we must not forget that we have assumed the flow is arriving at an empty link, which we shall see later is usually but not always true. In the significant minority of cases when a new flow joins $n-1$ others already using bottleneck capacity X, it roughly[4] translates to a similar scenario, but with one flow arriving at empty capacity of X/n. For instance, when a TCP flow joins 3 others sharing an 80 Mb/s bottleneck, it is roughly equivalent to arriving at an empty 20 Mb/s bottleneck. If a 20 Mb/s plot were shown in Fig. 1, it would peel away from the 1 Gb/s plot for flows larger than roughly 100 kB. Thus, multi-flow scenarios increase the range of capacity upgrades that will be effective. However, continued capacity upgrades still ultimately only benefit in fewer and fewer realistic scenarios.

In general, the flow-start problem does not play a role for flows that can fit in the initial congestion window [3] and therefore complete[1] in one RTT. The problem also hardly affects very large connections, where the initial delay due to an inefficient flow-start becomes insignificant compared to the overall completion time.

However, the flow-start problem hits the range of popular flow sizes between these two extremes. For example, b^* is 40 for a capacity increase from 2 to 80 Mb/s. However, for a flow size of 202 kB, R=20 ms and IW=10, based on the corresponding average rates in the figure, the actual rate increase, b is only $\frac{18Mb/s}{2Mb/s}$ or 9. So the effectiveness of this capacity upgrade for this case is only $\frac{9}{40}$ or 22.5%. The effectiveness reduces even further for larger upgrades. For example, a capacity upgrade from 2 Mb/s to 1 Gb/s is only 2% effective for this flow size. The effectiveness for larger flows would be greater, but according to the Dec-2012 CAIDA data, that was only about 1.5% of flows.

Comparing the second and third plots shows that increasing the initial window (IW) from 3 to 10 segments (as recently proposed and deployed by Google) makes a reasonable difference for flows between 4.5 kB and 100 kB (85 to 97.5 %-ile according to the Dec-2012 CAIDA data). The difference looks small because of the log-scale, but making a range of popular flow sizes twice as fast is a significant contribution. Whatever, for larger flows the difference becomes increasingly less significant. The merit of further increases to IW is discussed later.

3.2.3 *Competing Flows*

To circumvent the flow-start problem, applications sometimes open multiple flows together [15]. These flows are correlated and with respect to their capacity demand they

[2]The step artefact at about 20 B in one plot is due to how the original data was binned.

[3]Although we do not have data specifically for the US that would be more comparable to the CAIDA data

[4]The precise model is different; existing TCP flows make about half the capacity available, then the new flow competes with the others to fill it, usually remaining in slow-start for a few rounds while they increase more slowly in congestion avoidance.

should be considered as a single flow. However, it is difficult to find the correlation of a group of packets that are seen temporally together in the trace. These packets cannot be considered as a single flow if the relevant flows were not started together. Nevertheless, the number of active flows gives a notion of link occupancy or utilization. For example, if there are n active flows in a link, n gives an indication of the link occupancy for a new flow coming to this link.

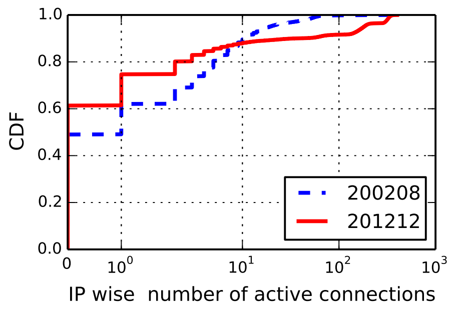
To this end, we identify the number of active flows from each IP address from the CAIDA datasets. We look per IP in order to approximate each bottleneck access link. This is not always true, but it is a useful working assumption because access capacity is often partitioned per IP, and we are solely interested in how many flows are competing, irrespective of how many users there might be sharing the bottleneck (e.g. behind a NAT).

In our analysis, an active flow means that at least a single packet[5] has been exchanged in the last q milliseconds which we call the 'max quiet time'. Our choice of q comes from rather coarse assumptions of the maximum RTT (we tried with 100 and 300 ms). Fig. 3 shows the CDF of the estimated number of active flows in an access bottleneck when a new flow starts, and how it differs between the years. The difference between Fig's 3a & 3b is due to the fact that when we assume a higher value of the max quiet time, $q = 300$ ms, sparse connections with packet inter-arrival times between 100–300 ms become considered as active connections.

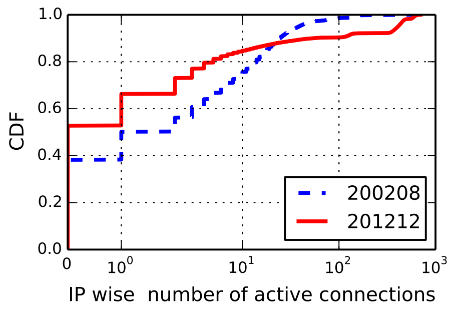
According to the graphs in Fig. 3, most of the time when a flow starts the access links is empty. This is not surprising given that, for capacity planning purposes, ISPs expect an *average* access link to be utilised *even during the peak hour* for only about 1-5% of the time.

Compared to 2002, in the 2012 data it is a little more likely that a flow arrives at an empty link, but also the likelihood of meeting hundreds of flows is greater. When interpreting data on competing flows, it should be borne in mind that most flows consist of a few or just one packet (see Sec. 3.2.2). So, when there appear to be hundreds of active flows it is likely that few (if any) are elephants. The mean number of flows encountered by a new flow in the 2012 data is either of (10.5, 30.4) for the two cases $q = (100, 300)$ ms, with respective annual growth factors, $c \approx (1.09, 1.12)$, where c is defined in Sec. 2. A possible explanation is that use of parallel flows by apps has become more common but, whenever a user is active, any larger flows complete sooner (because the link rate is so much faster), which means the link will be more likely to be free again before the next burst of activity.

This analysis counts flows by their port identifiers. Therefore it cannot tell the difference between a) flows opened by the same application in parallel (to circumvent the flow-start problem) and b) flows that happen to be active at the same time. Nonetheless, the analysis has some value as a comparison over the years. However, this analysis has a more serious limitation, because we cannot assume that for any user all the flows go through the CAIDA monitors. In the next section, we use traces that were collected from a point much closer to users.



(a) Max quiet time, $q = 100$ ms



(b) Max quiet time, $q = 300$ ms

Figure 3: Number of active flows seen when a new flow starts based on Tier 1 data sets.

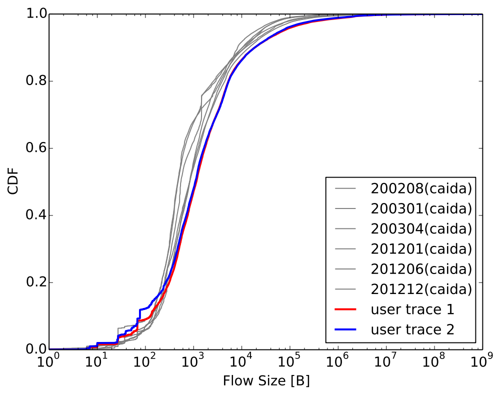
3.2.4 Access-link Packet Traces

In today's Internet, when apps request resources, they are often redirected to the set of CDN caches closest to them instead of the requested servers. CDNs were largely deployed during the decade of our longitudinal study. The involvement of CDNs makes a longitudinal study difficult, because we only have historic data under repeatable conditions from the Internet exchange point that CAIDA uses, which may no longer show traffic representative of that seen by end-users.

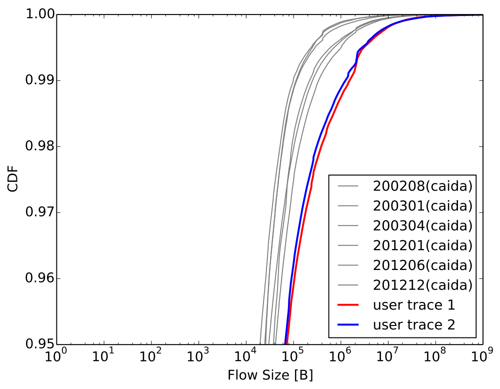
In this section, we use traces collected from a vantage point where we can see all the Internet traffic that users see—on the user-side of the CDNs. The trace was collected from a 1 Gb/s link between two local aggregation switches in an ISP network in Sweden. The traffic was collected from mid-December 2014 to early January 2015. The link aggregates more than 400 ADSL or VDSL households, with 60 Mbit/s as the highest subscription speed. In addition there are some tens of FTTX business accesses the fastest being 100 Mb/s and one 3G base station. Due to space limitation we only show results from traces taken on two of the days.

Fig. 4 shows the CDF of flow sizes for the user traces superimposed over the (greyed out) CAIDA traces from Sec. 3.2.2. It can be seen that the distribution of flow sizes is very similar to the CAIDA traces but, now that we have included all

[5]Excluding FIN and RST packets.

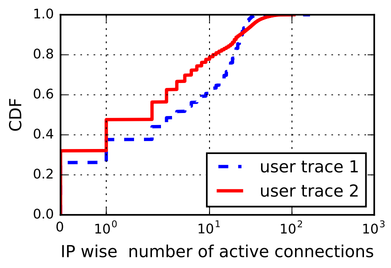


(a) Whole CDF

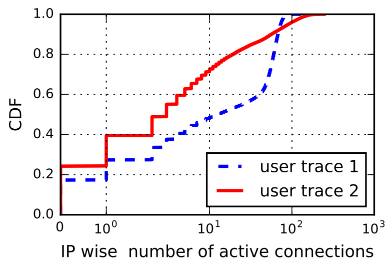


(b) Tail of the CDF

Figure 4: CDF of flow sizes for both Tier 1 and access link traces



(a) Distribution of active flows in user's bottleneck ($q = 100$ ms)



(b) Distribution of active flows in user's bottleneck ($q = 300$ ms)

Figure 5: Number of active flows seen when a new flow starts based on access link data sets.

the traffic that users see, there is a discernible shift towards larger flows, although it is still dominated by small flows. Similar to our observations for CAIDA traffic in the CDFs of Fig. 2a, the large majority of flows they experience the flow-start problem most acutely, i.e. they would accrue little or no benefit from further capacity investment.

Fig. 5a and 5b show the number of flows that are active when a new flow starts. The same approach as in Section 3.2.3 is adopted, where a flow is defined as active until none of its packets have been seen for max quiet time, q. Traces 1 & 2 show a typical weekday and weekend day during the collection period, each representing all IPs on that day. It can be seen that about 20-35% of flows arrive at an empty link—no longer a majority of flows now that we can see all the traffic a user sees. Interestinglythe tail is shorter than the CAIDA data sets. That is, in the access link data, it is nearly unheard of for a flow to arrive when there are over 100 flows in the bottleneck.

As we have already mentioned, some apps open multiple TCP connections in parallel in an attempt to mitigate the flow-start problem. For instance, at the time our access link traces were collected, the most popular browsers opened 6-13 connections per hostname and 10–17 max total.[6]

When parallel flows arrive at an idle link, our flow analysis records the first as competing with 0 flows, the second with 1, the third with 2, and so on. However, we should be characterising the underlying flow-start behaviour of applications and users. So, in terms of *behaviour*, parallel flows should all really be considered as one single larger aggregate, which we shall term a train. This would shift our flow-size distribution in Fig. 4 to the right (towards larger trains), and it would shift the CDFs in Fig. 5 to the left, reducing the number of active trains present when each new train arrives.

SPDY, which has been standardized as HTTP/2 aims to use a single TCP connection rather than multiple parallel ones. We will discuss the implications for flow-start behaviour, when we survey solutions in Sec. 4.

3.3 The Significance of User Value

In the following we will argue that the significance of the flow-start problem largely depends on how users value flows of various sizes, and illustrate the problem of judging value with an example concerning the latency of video interaction.

3.3.1 Weighting by Value

The analysis in Sec. 3.1 tells us that the limitation from flow-start mostly affects short flows and subsequently Sec. 3.2

[6]Using the Wayback Machine to view browserscope at the time: `http://web.archive.org/web/20141214012212/http://www.browserscope.org/?category=network`

suggests that most of the flows are short. Without saying explicitly, so far we have deferred the question of whether all flows are of equal importance. However, the significance of the flow-start limitation depends on how users value flows of different sizes.

Some might consider that, rather than each flow being of equal importance, each byte in each flow is of equal importance. This thinking comes from capacity planning and capacity scheduling, where network engineers and researchers are mainly interested in large flows representing applications like file downloading or video streaming. However value should not be confused with capacity usage.

Wischik argues that users particularly value the bytes in short flows that represent interactive applications like short messaging service (SMS), web browsing etc. [16]. A market study (unfortunately not published) has noted that, when users are expected to pay for certain applications, willingness to pay per byte ranges over more than six orders of magnitude, from messaging (SMS, IM) at the high end to software & video downloads at the low end, with interactive voice (telephony), Web, email, interactive video and music downloads in between. Very roughly, the more bytes there are in a session, the less per byte users are willing to pay. For instance, (some!) users are willing to pay about \$0.20 for an SMS (rough average size: 100 B). At that price, a 5 GB HD movie download would cost \$10M! Whereas, realistically, some people might pay about \$5. It must be noted that the value of transferring bytes is not the same as the value of the application as a whole, which is what the above numbers represent. However, the two are closely related.

These examples show that all bytes cannot be of equal importance (a video download is not worth \$10M). But also all flows are not of equal importance (a 20c SMS is not as valuable as a \$5 video). However, the latter is only 1 order of magnitude out, whereas the former is out by more than six orders. A more generic approach would be to weight each flow as a function of the size, *n*, of the flow. In this paper, we do not presume to determine the 'best' function. We merely propose the weighting technique as a way of precisely formulating the question. A market analysis could then answer the question.

Of course, size is not the only factor that determines the value of a flow (the type of application is obviously important), however we will progress on the assumption that there is some approximate relationship between flow size and flow value.

Ideally, each flow could be weighted by the likely value users would gain from improving the completion time of flows of that size. The weighting could be expressed as a function of flow size, so that the CDF of flow sizes could be recast as a CDF of weighted flow sizes. This recast CDF would then be compared with the slow-start model in Fig. 1 in the same way as we did in Sec. 3.2.2 with the original flow size CDF .

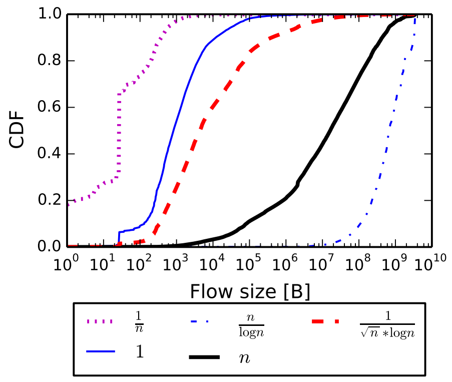


Figure 6: Value per flow analysis based on different weighting approaches.

In the following we will see, given different weighting approaches, how the significance of the flow-start problem changes depending on the chosen value-judgement.

Fig. 6 shows how different weighting functions affect the significance of flows of different sizes. For illustration, the 201212 CAIDA dataset has been used. For example, if the value per flow is proportional to the flow size, *n*, then the CDF of the proportion of value in flows up to each size would be the same as a CDF of the number of bytes in flows up to each size (see the graph labelled '*n*' in the figure). Under this assumption, the CDF is still rising healthily for flow sizes up to 10^9 B so, referring back to Fig. 1, capacity upgrades beyond 1 Gb/s would become valuable, because most of the value is in the relatively few large flows. All the smaller flows would not see any performance gain from these capacity upgrades. However, the chosen value judgement dictates that the value of doing anything about this is as small as the flows are, i.e. insignificant.

In contrast, if the value per flow is independent of flow size, then the CDF of value (see graph labelled '1') would be no different to the CDF of number of flows (as originally shown in Figure 2a). In this case, users value small flows as much as large, so most of the value is concentrated where large numbers of small flows cause the CDF to rise steeply (below about 10^5 B). Then, referring again to Fig. 1, capacity upgrades beyond about 80 Mb/s are of little value, because only a few large flows benefit. This is the same scenario as already described in Sec. 3.2.2.

The figure also shows how other weighting functions or value-judgements can change the distribution of value between flows of different sizes. There is no implication that any of these functions are particularly appropriate. They are merely intended to show a range of ways in which value can be focused more or less strongly on either end of the flow size distribution.

Nonetheless, in the SMS vs. video discussion above, it was suggested that a function somewhere between that labelled '1' and that labelled 'n' might be appropriate, with '1' seeming

Format	Resolution	Bytes	Packets	Download time	Minimum bandwidth required to transfer the I-frame in slow-start
DVD	720 × 576	62,208	43	150 ms	3 Mb/s
720p	1280 × 720	138,240	95	200 ms	4 Mb/s
1080p	1920 × 1080	311,040	212	250 ms	10 Mb/s
2160p	3840 × 2160	1,244,160	851	350 ms	38 Mb/s
4320p	7680 × 4320	4,976,640	3404	450 ms	150 Mb/s

Table 1: Download time for the 1st I-frame of streamed video using traditional slow-start on a link with unlimited bandwidth and RTT of 50 ms. The example RTT was chosen based on common user-CDN response times [17].

the better approximation. Therefore the function labelled '$1/(\sqrt{n}\log n)$' may represent a more typical value judgement than the others.

3.3.2 *Video Interaction Latency*

The detrimental effects of the slow-start problem do not only affect short flows. There are cases where massively greedy flows also depend on a fast speed-up to provide the QoE that users require. One example is HTTP segment streaming that is responsible for an increasing amount of the traffic on the Internet today [18]. When a control action is performed for the video, like start, pause or forward, a full frame of video must be sent to restart the video playout. The delay experienced by the user is negligible for low-resolution video, but as the resolution increases, the flow-start limitation comes into play whenever most of the I-frame transfer happens during slow-start. It rises to a level where the delay will easily be detected and experienced as annoying, even with a perfect link.

Table 1 shows the time it will take to download the first frame of video as the resolution increases[7]. The link speed is defined as unlimited with no packet loss, so the slow-start/RTT relation is the only limitation to the start-up latency. We can see that for recent Ultra High Definition TV (UHDTV) resolutions, the delay for starting the playout will be 450*ms*, a clearly noticeable delay for users. There are other emerging technologies that require much more data to be streamed over the Internet, like high-quality 3D mesh streaming. In such scenarios, the start-up delay will be even bigger.

The bandwidth column in Table 1 shows the minimum bandwidth required to transfer the whole I-frame in slow-start. Many of today's connections already provide such bandwidths. For example, the obtainable bandwidth when using very high rate digital subscriber line (VDSL2) in Fiber to the node (FTTN) is well over 100 Mbps [19]. LTE-advanced (LTE-A) developed by 3GPP provides a speed of 300 Mbps for cellular connection [20]. Besides, the flow sizes in Table 1 fall over the 90th percentiles in the CDFs of both Figure 2 and Figure 4. However, since most of the video streaming are CDN traffic today [21], they are probably missing in core Internet traces like the traces from CAIDA, but could potentially exist in the access link traces described in section 3.2.4.

[7]Since the I-frame needs to be transmitted in its entirety, the size is calculated based on the compression rate of 1:20 that is found in MPEG video. We have defined each pixel to, uncompressed, be represented by 24 bits. Thus, we use 1.2 bits per pixel to calculate the I-frame size in the table. 1500 B packet size is assumed.

4. SOLUTIONS AND THEIR DEPLOYMENT PROBLEMS

The aim of this paper is not to present a solution to the flow-start problem; it is to thoroughly investigate the nature of the problem. That includes understanding why there appear to be no solutions that are scalable and feasible to deploy. Scalable means ideally no additional start-up latency as flow-rates increase. Feasible to deploy means a solution should ideally give immediate benefit to a 'first mover', which we shall call a "unilaterally deployable solution". For example, no benefit until both host and network have been upgraded is not unilaterally deployable; rather it is a chicken-and-egg stalemate.

Sec. 4.1 outlines the various types of unilaterally deployable solutions. Although they all aim to make today's slow-start incrementally better, none address the enduring scalability challenge posed by the flow-start problem. Then Sec. 4.2 introduces proposals that do address the scalability challenge, but are not unilaterally deployable.

4.1 Limited Scope for Unilaterally Deployable Solutions

4.1.1 *Caching*

The idea behind all sorts of caching technology available in the Internet is to reduces the RTT by holding resources closer to clients. Caching trades off the performance benefit against both the size and the placement of the cache [22, 23]. For instance Kroeger *et al* [24] found an average of only 22%–26% reduction in latency with web caching, even for cache hit ratios of 47%–52%. Moreover, caching is not particularly suitable for a range of services dealing with real time/ dynamic content (e.g. gaming, interactive video), remote control operations or financial updates. Any legacy content caching or CDN is not suitable to cache dynamic contents [25]. This, caching gives a useful once-off reduction in RTT for some scenarios, but every doubling of link rate still needs one more RTT, albeit a reduced RTT.

4.1.2 *Larger Initial Window (IW)*

In 1998, Katz *et al* [26] pointed out that, with a small IW, the long probing period not only causes poor bandwidth utilisation but also increases latency. Balakrishnan *et al.* [15] further mentioned that as a consequence of slow probing in slow-start, several applications tend to start multiple connections to probe faster (see the next section for why parallel flows and increased IW are equivalent). In 2010, Dukkipati *et al* [8] conducted a large-scale experiment to show that it would be safe to increase TCP's initial window from its previous typical value of 3 to 10. Subsequently, there were a number of experiments finding cases where this would be detrimental, e.g. to heavily loaded low rate links. And there have been some experiments proposing combining IW10 with pacing would mitigate these problems [27]. Nonetheless, IW10 has now been documented by the IETF as an experimental RFC [28]

Clearly, given TCP slow-start doubles the window every round, starting from a higher number removes the need for some of the early rounds. For example, starting from 3 will double to 6 after 1 round, having sent 3+6=9 segments. So starting from 12 (which would have been the next step after 6) is the same as cutting out the first round (also adding 3 segments). We have already characterised the practical benefit that increased IW offers (at the end of Sec.3.2.2 using Fig. 1), because it allows a larger proportion of small flows to fit into the initial window.

Nonetheless, as each doubling of flow-rate with capacity adds a round to TCP slow-start, we cannot keep increasing IW to remove this round again, because this would lead to larger and larger sequences of segments being injected into the Internet before any feedback control had been established. Therefore, although increasing IW makes a useful gain, it is not an enduring solution to the slow-start scaling problem.

4.1.3 Multiple Parallel Flows or a Large Initial Window

Developers of applications have tended to patch over the flow-start problem by opening multiple parallel flows. For instance, the Firefox Web browser opens 6 parallel connections for each domain[8] and up to 17 for Web pages that contain components from several domains (a common practice). Perhaps surprisingly, parallel flows to the same host provide exactly the same speed-up as starting with a larger IW, but with the overhead of opening all the extra flows. The total window across all flows still only doubles in the rounds after the first. For example, if four parallel flows each increase their windows in the sequence $3, 6, 12, \ldots$, the total window across them all will start higher, but still double each round: $4 \times 3, 4 \times 6, 4 \times 12 \ldots = 12, 24, 48 \ldots$.

Allman [29] made this point by grouping multiple parallel flows into a single flow to investigate the cumulative effect of a larger initial window. Like our work, he had to use an arbitrary resolution time to group parallel start times (1 s in his case).

Using parallel flows is feasible for modest speed-up of small flows, however, to keep up with capacity increases, the number of parallel flows would have to grow more than exponentially over time—the number of flows has to scale faster than capacity growth, which itself increases exponentially over the years [5]. This will quickly become infeasible, particularly as larger and larger flows become constrained by flow-start rather than capacity. For instance, over a 20 ms path, one 1MB flow can achieve an average rate of 70 Mb/s. To double its average rate requires 15 flows, but to exploit four times the capacity would require 58 parallel flows[9].

Interestingly, attempts are in progress to reduce sharding of Web sessions into multiple parallel flows through standards/protocols. Some efforts are being made to move in that direction, for instance through standardizing SPDY as HTTP/2 [**?**, **?**]. Most attention has been on being able to coordinate all the flow control within one TCP connection. However, it will be interesting to see whether the popularity of this initiative is harmed by the performance of just a single TCP slow-start.

In summary, increasing beyond the numbers of multiple flows already in use will have limited additional benefit, unless stupidly large numbers of flows are used, which would introduce considerable other problems (e.g. flow-state memory exhaustion in servers and NATs, as well as the multiple initial windows or equivalently a single larger initial window overflowing buffers on slower lines).

4.1.4 Faster Exponential

It may seem that a flow could get up to speed faster by more than doubling its rate in each round. However, in the last decade researchers at the leading edge of high-speed networking (e.g. for transferring large data sets in astronomy or physics) have found that even rate doubling leads to tremendous overshoot problems as link capacities grow [4, 30]. So far, attempts to detect overshoot early using delay measurements (see below) have produced mixed results, with worse performance as well as better even when just doubling.

More than doubling during flow-start would create the need to size buffers larger than one bandwidth-delay product, which already introduces delay problems for shorter flows and real-time applications.

4.1.5 Using Queuing Delay Measurements

Paced Start [31] monitors the queuing delay from the incoming ACKs and paces the packets sent in subsequent rounds. This avoids TCP's overshoot, but it takes even longer than TCP's slow-start to reach the available capacity. Swift-start [32] uses the packet-pair technique [33] to determine the initial estimate of available bandwidth , however, the paper does not discuss how good the estimation needs to be to be useful. The packet-pair technique for accurate bandwidth estimation at the beginning of a TCP connection has been widely criticised for giving unreliable results [34]. Later a more sophisticated bandwidth estimation algorithm called RAPID

[8]http://www.browserscope.org/?category=network

[9]The formula for how many flows would be needed to achieve different speed-ups is derived in Appendix B.

was proposed [35]. RAPID is based on PathChirp [36]. In RAPID, each packet transmission is precisely timed, which produces multiple sending rates in a single RTT. Under ideal conditions, RAPID can probe for or adapt to a large change in available bandwidth within one to four RTTs. However, precise timing of packet transmission has always been a great challenge for implementers [37].

In general, the precision of queuing delay measurements reduces as link rates increase, because the time in the queue decreases relative to the round trip time, making measurements more susceptible to noise. Therefore it is not believed that these techniques alone are a good basis for solving the long-term scaling problem for flow-starts.

4.1.6 *Other Solutions*

Liu *et al.* [38] investigated what the impact would be if every flow simply tried to send all its data paced out over the first round trip time (termed Jump Start). The authors monitored current Internet flows and found that only about 7.4% of them comprise more than the three packets that a sender would send immediately anyway under the then standard behaviour. The paper is inconclusive on whether the edges of the Internet would cope with the very high loss rates that this 7.4% of flows would cause (because they represent a very much larger proportion of the bytes on the Internet).

The congestion manager framework [39,40] use state sharing at the sender to know the available capacity in the network. Although joint congestion management can allow a new flow to quickly find a suitable sending rate, it is only applicable when multiple flows share a common bottleneck, so it is not a general solution to problems of scale.

4.2 Promising Solutions with Deployment Challenges

This section briefly surveys the landscape of solutions where deployment would be more involved. Solutions fall into the following categories:

- new in-band signalling for packets that choose to use it, but without changing the format of packet headers, e.g: Anti-ECN [41], VCP [42];
- new in-band per-packet signalling for all traffic using a new shim header, e.g. RCP [43], XCP [44];
- new out-of-band per-flow signalling for flows that choose to use it, e.g. Quick-Start [45];
- Solutions that rely on the presence of preferential discard or priority queuing in the network, in order to send at a high initial rate into a low priority class, e.g. RC3 [46].

Proposals such as [41,42,47] share the idea of starting to ECN-mark at a threshold set at some fraction of the link rate. They all require a decision on what fraction to use, but there is no natural 'best' setting, so it is unlikely that one standard will prevail. Therefore these approaches don't really solve the problem, they just move it.

Approaches like RCP [43], XCP [44], Quick-Start [45] and RC3 [46], in one way or another, involve end-systems asking network components to tell them whether it is safe to be more aggressive than slow-start. The main problem this raises is how the end-system can know whether sufficient network components along the path speak the new protocol, particularly whether the bottleneck does. This would not be a problem in green-field networks or isolated private networks under the control of a single administrator, e.g. a data center or an enterprise LAN surrounded by proxies. However, otherwise, this partial network deployment problem is the critical barrier to deployment.

This survey of existing solutions forces us to confront the conclusions that we seem to have reached the scaling limit of unilaterally deployable solutions, so we need to face the prospect of changing the host-network interface of the Internet. However that will be an industry co-ordination problem of epic proportions [6].

5. WORK RELATED TO CHARACTERIZING THE PROBLEM

Several existing works characterized the limitations of TCP's slow-start algorithm, including many of the works covered already in Sec. 4 that also propose solutions.

A large body of work exists that characterizes Internet traffic [11–14]. These works mainly investigated several characteristics of a flow such as size, rate, duration and burstiness. Similar to our work, protocol efficiency of TCP has been investigated in [48,49] for mobile networks. Their TCP modeling is mainly based on their measurements of loss and throughput in different mobile networks.

As well as studies on scalability of TCP's slow-start phase, the standard TCP congestion avoidance phase has also been shown to be unscalable [50–52], especially in high speed wide area networks. In steady state, with a loss rate p, standard TCP's average congestion window is proportional to $\frac{1}{\sqrt{p}}$, which makes low loss rates very low and the loss recovery time extremely slow with a high bandwidth-delay product. Some of the scalable algorithms proposed in the literature are Scalable TCP [51] and Relentless Congestion Control [50] that try to give constant recovery time between losses by providing an average congestion window proportional to $\frac{1}{p}$.

6. CONCLUSIONS

This paper has attempted to quantify the size of the problem where acceleration is not keeping pace with increasing link-speeds, which we term the flow-start problem. We have shown that flow sizes are growing slowly relative to link capacities. Because each doubling of capacity requires an additional round trip delay to reach top speed, more and more flows are finishing before they reach full speed. In other words average flow-rates are becoming protocol-limited, not capacity-limited.

We have also shown that when an application starts a new data transfer, the bottleneck link is as likely as not to be empty. This implies the flow-start problem is particularly acute, because capacity increases are often not being used for multiple flows simultaneously.

The paper has been intended as a call to action. It has surveyed the status of solutions to the flow-start problem and found that they are either unambitious in order to be deployable, or ambitious but probably undeployable. We ask the research community to focus on what will be necessary to deploy an ambitious solution that will properly solve this scaling problem. Clearly incremental solutions have their place, but the research community needs to face the question of how to co-ordinate an industry into adopting an enduring solution to this scaling problem.

In future work, we intend to build consensus on extending the current network/host interface in order to establish the generic information flow (in a precise and timely manner) from the network to the transport layer. We hope this will become the foundation on which scalable solutions to the flow-start problem can be built. Given the constraints set by the existing IP protocol, we believe the updated interface may have to rely on some form of encoding of the ECN field as the information channel. This encoding will have to squeeze an extra signalling channel into the 2-bit ECN field, without interfering with the existing use of the ECN field by AQM algorithms.

There is also further work to do to fully quantify and characterize the problem. We intend to build a tool to extract information about trains of packets that applications send between times when they become application-limited, whether within one flow or across multiple parallel flows. We also hope that the access link data we have collected can form the start of a new longitudinal study.

7. ACKNOWLEDGMENTS

This work was funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the author(s). Raffaello Secchi reviewed the slow-start model. Koen De Schepper gave useful comments on a draft of the paper. Data was extracted from the CAIDA traces using tool-tracetool written by Olga Bondarenko.

APPENDIX

A. TCP AVERAGE RATE MODEL

This is a model of the average rate of a single TCP flow in dedicated bottleneck capacity, defined using the following independent variables:

Link capacity, X [b/s]

Round trip time, R [s]

Initial window, i [pkt]

Packet size, S [B] or $s = 8S$ [b]

Flow size, F [B].

Flow size, $f = F/S$ [pkt]

Bandwidth-delay product, $W = XR/s$ [pkt].

The general approach is to find the number n of round trips in which slow-start only partially fills the bottleneck, *before* the round in which either slow-start ends or the flow ends, if sooner. Then any remaining packets will arrive at the bottleneck rate, whether they are all sent in the next round while still in slow-start phase or there are enough to progress into the fast retransmit and congestion avoidance phases. In the latter case, the number of retransmissions is added to the number of packets to be forwarded before completion. One round must be added between the receiver's request and the start of the response.[10] Then **completion time** T between the client's request and it receiving the last data consists of:

$$T = (n+1)R + (m+M)s/X,$$

The analysis below finds the values of n, m & M,

Round index, n [integer]

Remainder, m is the remaining packets of the flow that all leave the bottleneck at rate X

Retransmissions, M due to losses during the overshoot at the end of slow-start.

We define the number of packets in slow-start as f_s. Index u [real number] and the final window w [pkt] are only used to derive formulae during slow-start, outside which they are invalid:

$$\begin{aligned} \sum_{j=0}^{n-1} i2^j < \quad & f_s \leq \sum_{j=0}^{n} i2^j \\ i(2^n - 1) < \quad & f_s \leq i(2^{(n+1)} - 1) \\ & n \geq \lg(f_s/i + 1) - 1 \\ & n = \lceil \lg(f_s/i + 1) \rceil - 1; \end{aligned}$$

The critical flow size, f_c is defined as the number of packets that can be sent until buffer overflow ends slow-start, and the critical round as the real number u_c. The buffer is assumed

[10]Any handshaking rounds are excluded by the assumption that either the flow is re-starting after idle or using TCP fast open after an earlier connection between the same hosts.

perfectly sized for the flow, at $1\text{BDP} = W$. Therefore at the critical flow size, the critical window,

$$\begin{aligned}
w_c &= 2W \\
&= i2^{u_c} && \text{if } 2W \geq i \\
\text{when } f_c &= i(2^{(u_c+1)} - 1) \\
&= 2i2^{u_c} - i \\
&= 4W - i && \text{if } 2W \geq i; \\
f_c &= 2W && \text{if } 2W < i; \\
f_s &= \min(f, \quad \max\big(2W,\ (4W - i)\big));
\end{aligned}$$

The number of packets, e, in partially empty rounds of slow-start is the number of packets in n rounds,

$$\begin{aligned}
e &= i(2^n - 1) \\
m &= f - e \\
M &= \min\big(\max(i, 2W),\ \max(0, (m - 2W))\big).
\end{aligned}$$

Explanation of the last formula for M: The buffer always has time to empty the packets sent in the the round before the round in which overflow occurs (if it occurs). The number of packets dropped is the number of packets not constrained by slow-start m less the 2 BDP of packets that the link can forward or buffer, but limited by the number that will ever be sent in one round during SS or CA, which is also $2W$ (or the initial window i if it is larger, so it can overflow the buffer on its own).

B. PARALLEL SLOW-STARTS

If a source opens m parallel flows all using the same slow-start with initial window i, it only gives equivalent performance to one flow in slow-start using a larger initial window mi (but without all the overhead of the extra flows). The total window across all flows still only doubles in the rounds after the first.

The increase in bit-rate from using m-times larger packets is also equivalent to using an m-times larger initial window or m parallel flows (but without so much packet overhead).

Opening parallel flows (or equivalently increasing IW or packet size) only offers a modest increase in average rate. Put the other way round, to achieve a decent change in rate or speed-up ratio b requires a stupidly large number of parallel flows m.

The following analysis proves this by approximating from the model of slow-start in A on condition that $f \gg i$, where f is the flow-size in packets. The notation $\bar{x}_m$ or T_m means the average rate or completion time for m parallel flows (or equivalently IW=mi or m times larger packets).

$$\begin{aligned}
T_m &\approx \lg(f/mi); \\
b &= \frac{\bar{x}_m}{\bar{x}_1} \\
&= \frac{T_1}{T_m} \\
&\approx \frac{\lg(f/i)}{\lg(f/mi)} \\
&\approx \frac{\lg(f/i)}{\lg(f/i) - \lg m} \\
\lg m &\approx (1 - 1/b)\lg(f/i) \\
m &\approx 2^{\big((1-1/b)\lg(f/i)\big)}.
\end{aligned}$$

8. REFERENCES

[1] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 157–187, Jan. 1995.

[2] J. Heidemann, "Performance Interactions Between P-HTTP and TCP Implementations," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 65–73, Apr. 1997. [Online]. Available: http://doi.acm.org/10.1145/263876.263886

[3] M. Allman, V. Paxson, and E. Blanton, "Tcp congestion control," IETF, RFC 5681, 2009.

[4] S. Ha and I. Rhee, "Hybrid slow start for high-bandwidth and long-distance networks," in *Proceedings of International Workshop on Protocols for Future, Large-scale & Diverse Network Transports*, Manchester, UK, Mar. 2008.

[5] G. Kim, "Bandwidth Growth: Nearly What One Would Expect from Moore's Law," Online: http://ipcarrier.blogspot.co.uk/2014/02/bandwidth-growth-nearly-what-one-would.html, Feb. 2014.

[6] D. Papadimitriou, M. Welzl, M. Scharf, and B. Briscoe, "Open research issues in Internet congestion control," IETF, RFC 6077, Feb. 2011.

[7] F. P. Kelly, "Models for a self-managed internet," *Philosophical Transactions of the Royal Society*, vol. 358, no. 1773, pp. 2335–2348, Aug. 2000.

[8] 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," *ACM SIGCOMM Computer Communication Review*, vol. 40, pp. 27–33, Jul. 2010.

[9] Y. Cheng, R. Sivasankar, and A. Jain, "TCP Fast Open," IETF, RFC 7413, Dec. 2014.

[10] C. for Applied Internet Data Analysis (CAIDA), "University of california, san diego supercomputer," http://www.caida.org, accessed: 2016-06-05.

[11] N. Brownlee and k. claffy, "Internet stream size distributions," in *Proceedings of the 2002 ACM SIGMETRICS International Conference on*

Measurement and Modeling of Computer Systems, ser. SIGMETRICS '02. New York, NY, USA: ACM, 2002, pp. 282–283. [Online]. Available: http://doi.acm.org/10.1145/511334.511381

[12] K. Lan and J. Heidemann, "A measurement study of correlations of Internet flow characteristics," *Computer Networks*, vol. 50, no. 1, pp. 46–62, Jan. 2006.

[13] S. Molnar and Z. Moczar, "Three-dimensional characterization of Internet flows," in *Proceedings of IEEE International Conference on Communications*. Kyoto, Japan: IEEE, Jun. 2011, pp. 1–6.

[14] F. Qian, A. Gerber, Z. M. Mao, S. Sen, O. Spatscheck, and W. Willinger, "TCP revisited: a fresh look at TCP in the wild," in *Proceedings of ACM SIGCOMM Conference on Internet Measurement*. Chicago, Illinois, USA: ACM, Nov. 2009, pp. 76–89.

[15] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for internet hosts," in *Proceedings of ACM SIGCOMM*, vol. 29, no. 4. ACM, 1999, pp. 175–187.

[16] D. Wischik, "Short messages," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1872, pp. 1941–1953, Jun. 2008.

[17] Cedexis, "Cedexis CDN response time report." http://www.cedexis.com/reports/#?report=cdn_response_time&country=US&date=2016-06-16, accessed: 2016-06-17.

[18] T. Kupka, C. Griwodz, P. Halvorsen, D. Johansen, and T. Hovden, "Analysis of a real-world http segment streaming case," in *Proceedings of the 11th European Conference on Interactive TV and Video*. New York, NY, USA: ACM, 2013, pp. 75–84.

[19] V. Oksman, H. Schenk, A. Clausen, J. M. Cioffi, M. Mohseni, G. Ginis, C. Nuzman, J. Maes, M. Peeters, K. Fisher, and P. e. Eriksson, "The ITU-T's new G.vector standard proliferates 100 mb/s DSL," *IEEE Communications Magazine*, vol. 48, no. 10, pp. 140–148, Oct. 2010.

[20] A. Technologies, "Introducing LTE-Advanced," http://cp.literature.agilent.com/litweb/pdf/5990-6706EN.pdf, accessed: 2016-06-05.

[21] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A case for a coordinated internet video control plane," in *Proceedings of ACM SIGCOMM*. New York, NY, USA: ACM, Aug. 2012, pp. 359–370.

[22] B. Bjurling and P. Kreuger, "Performance of cache placement policies," in *10th Swedish National Computer Networking Workshop, SNCNW*, Västerås, Sweden, Jun. 2014.

[23] I. Marsh, "Elastic network resources: controlled caching," in *10th Swedish National Computer Networking Workshop, SNCNW*, Västerås, Sweden, Jun. 2014.

[24] T. M. Kroeger and D. D. E. Long, "Exploring the bounds of web latency reduction from caching and prefetching," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*. Anaheim, CA, USA: USENIX Association, Jan. 1997, pp. 13–22.

[25] J. Ravi, Z. Yu, and W. Shi, "A survey on dynamic web content generation and delivery techniques," *Journal of Network and Computer Applications*, vol. 32, no. 5, pp. 943–960, Sep. 2009.

[26] R. H. Katz and V. N. Padmanabhan, "TCP fast start: A technique for speeding up web transfers," in *Proceedings of IEEE Globecom Internet Mini-Conference*. Sydney, Australia: IEEE, Nov. 1998.

[27] R. Sallantin, C. Baudoin, E. Chaput, F. Arnal, E. Dubois, and A.-L. Beylot, "Initial spreading: a fast start-up TCP mechanism," in *Proceedings of IEEE Conference on Local Computer Networks*. Sydney, Australia: IEEE, Oct. 2013, pp. 492–499.

[28] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing TCP's Initial Window," RFC Editor, RFC 6928, Apr. 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6928.txt

[29] M. Allman, "Comments on bufferbloat," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 1, pp. 30–37, Jan. 2012.

[30] A. Antony, J. Blom, C. Laat, and J. Lee, "Exploring practical limitations of TCP over transatlantic networks," *Future Generation Computing System*, vol. 21, no. 4, pp. 489–499, 2005.

[31] N. Hu and P. Steenkiste, "Improving TCP startup performance using active measurements: algorithm and evaluation," in *Proceedings of IEEE International Conference on Network Protocols*. Atlanta, Georgia, USA: IEEE, Nov. 2003, pp. 107–118.

[32] C. Partridge, D. Rockwell, M. Allman, R. Krishnan, and J. Sterbenz, "A swifter start for TCP," BBN Technologies, Cambridge, MA, BBN Technical Report, Tech. Rep., 2002.

[33] S. Keshav, "A control-theoretic approach to flow control," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 188–201, 1995.

[34] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, no. 6, pp. 27–35, 2003.

[35] V. Konda and J. Kaur, "RAPID: Shrinking the congestion-control timescale," in *Proceedings of IEEE Conference on Computer Communications*. Rio de Janeiro, Brazil: IEEE, Apr. 2009, pp. 1–9.

[36] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "Pathchirp: Efficient available bandwidth estimation for network paths," in *Proceedings of*

International Conference on Passive and Active Network Measurement. San Diego, CA, USA: Springer-verlag, Apr. 2003.

[37] M. Kühlewind and B. Briscoe, "Chirping for congestion control-implementation feasibility," in *Proceedings of International Workshop on Protocols for Future, Large-scale & Diverse Network Transports*, Lancaster, PA, USA, Nov. 2010.

[38] D. Liu, M. Allman, K. Fall, and L. Wang, "Congestion control without a startup phase," in *Proceedings of International Workshop on Protocols for Future, Large-scale & Diverse Network Transports*, Los-Angeles, CA, USA, Feb. 2007, pp. 61–66.

[39] J. Touch, "TCP control block interdependence," IETF, RFC 2140, Apr. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2140.txt

[40] H. Balakrishnan and S. Seshan, "The congestion manager," IETF, RFC 3124, Jun. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3124.txt

[41] S. S. Kunniyur, "AntiECN marking: A marking scheme for high bandwidth delay connections," in *Proceedings of IEEE International Conference on Communications*, vol. 1. Anchorage, Alaska, USA: IEEE, May 2003, pp. 647–651.

[42] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 37–48, 2005.

[43] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor sharing flows in the Internet," in *Proceedings of International Workshop on QoS*. Passau, Germany: Springer-verlag, Jun. 2005, pp. 271–285.

[44] D. Katabi, "Decoupling congestion control from the bandwidth allocation policy and its application to high bandwidth-delay product networks," Ph.D. dissertation, MIT, Mar. 2003.

[45] S. Floyd, M. Allman, A. Jain, and P. Sarolahti, "Quick-Start for TCP and IP," IETF, RFC 4782, Jan. 2007, (Status: Experimental).

[46] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Recursively cautious congestion control," in *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*. Philadelphia, PA, USA: USENIX, Jun. 2014, pp. 373–385.

[47] P. Eardley, "Metering and marking behaviour of PCN-nodes," IETF, RFC 5670, Nov. 2009.

[48] J. Garcia, S. Alfredsson, and A. Brunstrom, "A measurement based study of TCP protocol efficiency in cellular networks," in *Proceedings of International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*. Hammamet, Tunisia: IEEE, May 2014, pp. 131–136.

[49] K. Liu and J. Y. Lee, "Impact of TCP protocol efficiency on mobile network capacity loss," in *Proceedings of International Symposium on Modeling & Optimization in Mobile, Ad Hoc & Wireless Networks*. Tsukuba Science City, Japan: IEEE, 2013, pp. 1–6.

[50] M. Mathis, "Relentless congestion control," in *Proceedings of International Workshop on Protocols for Future, Large-scale & Diverse Network Transports*, Akihabara, Tokyo, Japan, May 2009.

[51] T. Kelly, "Scalable TCP: improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 83–91, 2003.

[52] S. Floyd, "HighSpeed TCP for large congestion windows," IETF, RFC 3649, 2003.