Comments on Bufferbloat

Mark Allman
International Computer Science Institute
mallman@icir.org

ABSTRACT

While there has been much buzz in the community about the large depth of queues throughout the Internet—the so-called "bufferbloat" problem—there has been little empirical understanding of the scope of the phenomenon. Yet, the supposed problem is being used as input to engineering decisions about the evolution of protocols. While we know from wide scale measurements that bufferbloat *can* happen, we have no empirically-based understanding of how often bufferbloat *does* happen. In this paper we use passive measurements to assess the bufferbloat phenomena.

Categories and Subject Descriptors

C.2.5 [**Computer Communication Networks**]: Internet (e.g., TCP/IP)

Keywords

Bufferbloat, delay, TCP, queues

1. INTRODUCTION

The past couple of years have seen a buzz in the networking community about *bufferbloat* [10]. This phenomenon is caused by a general over-buffering in router queues that hold traffic that cannot be immediately forwarded. This is especially problematic in small-scale gear for residential use [16]. The queues are deep, but also not actively managed. TCP's congestion control algorithms [12, 3] interact with the drop-tail queue management to fill the queues since TCP only detects congestion when noticing packet loss[1] and drop-tail queues only drop packets when they are full (and in many cases deep). As queues grow so does delay through the network. While delay does not have an appreciable impact on bulk TCP transfers it can degrade delay-sensitive traffic such as Internet telephony conversations. While problems with large possible queues are not new and in fact have received much study by the community over the years,[2] they have recently received new attention (perhaps as the Internet is being increasingly used for interactive services).

While some see bufferbloat as "dark storm clouds surround[ing] us" [8] we are struck by two things. First, while there is wide-scale evidence that suggests that large amounts of buffering exist in the network and hence buffers *can* be bloated [16] there is—to our knowledge—little empirically-based information about whether these deep potential buffers are naturally realized in practice. Rather, most of the debate revolves around anecdotes and not systematic study. Second, even in the absence of empirical grounding bufferbloat is used to argue against particular engineering changes. For instance, [9] invokes bufferbloat to argue against a proposal to increase TCP's initial congestion window from roughly 4KB to roughly 14KB [5]. Another example uses the supposed problem of bufferbloat in mobile networks to motivate a modification to TCP receivers to dynamically adjust the advertised window during data reception [14].

We address this dearth of empirically-based understanding by evaluating (*i*) bufferbloat's prevalence and magnitude and (*ii*) the impact of a modest change to TCP's initial parameters on the overall state of buffers in the Internet. While this is an initial study we believe it is the first systematic, non-anecdotal study of bufferbloat and its implications. We encourage additional large scale measurement-based studies of bufferbloat.

2. BUFFERBLOAT PREVALENCE

Our first analysis aims to study the prevalence of long queues. We use naturally occurring TCP traffic to assess the delay across a large number of network paths and gain an initial insight into the bufferbloat phenomenon.

2.1 Data

We use a packet trace collection taken from the Case Connection Zone (CCZ) [1] experimental fiber-to-the-home network which connects roughly 90 homes adjacent to Case Western Reserve University's campus with bi-directional 1 Gbps links. While the CCZ is a University project, the CCZ enjoys a direct connection with a regional ISP and is not shared with campus traffic. We record packet-level traces between the 11^{th} and the 18^{th} of each month. Each day during this week is broken into six hour blocks and we record a one-hour trace starting at a random time within each block. Before archiving the traces we remove the application payload in all but one of the traces taken each day.[3] We include all traffic in our traces. Our trace collection spans from February 2011 through March 2012 (14 months), leaving us with 392 distinct trace files.

We use Bro's [20] "capture-loss" script to assess how much measurement-based packet loss we experienced during our

[1]TCP can support Explicit Congestion Notification (ECN) [22], but to date this is not prevalent in the Internet [4].

[2]One of the goals of active queue management techniques such as RED [7] is to enable both high performance and low delays by carefully managing router buffers.

[3]Our tracing strategy was mainly designed around the logistical constraints we face when storing the collected data.

trace collection. This process is part of Bro's content reassembly procedure and identifies bytes in TCP traffic that are not present in the trace, but are none-the-less acknowledged by the recipient. While this does not account for all possible measurement loss (e.g., in non-TCP traffic) it gives a general indication about the level of measurement loss present in a given trace. Given this process only runs when reassembling content from a packet stream, we can only use it to assess the 25% of our traces that retain application payload. Therefore, our analysis should be viewed as a general calibration of measurement apparatus. Across the 98 traces in our dataset with content payload we find that measurement-based loss is (*i*) non-existent in 79 traces, (*ii*) less than 0.01% in 18 traces and (*iii*) 0.013% in 1 trace. Our conclusion is that while our monitoring apparatus does in fact drop traffic in some cases this is not a large issue.

Our strategy is to use round-trip times observed in naturally occurring TCP traffic to assess the queuing imposed along the wide-area path between the CCZ and the remote host.[4] We use Bro to assess and log all possible RTT samples in our traces [17]. Since our monitoring point is near the CCZ hosts we take RTT samples using data packets sent from a CCZ host and the corresponding ACKs from the remote peer. This measures the wide-area and remote edge networks, but not the portion of the network path within the CCZ network. In other words, we are not assessing buffering within the FTTH network. Bro attempts to take an RTT sample for each data segment observed in the trace. Note, RTTs taken from retransmitted segments are fundamentally ambiguous and therefore we—like TCP—apply Karn's algorithm [15] and do not use these RTT samples.[5] We aggregate RTT samples by host pair, regardless of TCP connection. Over the course of our data collection we are able to leverage the 90 homes in the CCZ to assess bufferbloat at 118K remote peers on 95K distinct /24s and 16K distinct /16s. While a larger sample would be useful, assessing bufferbloat for 118K hosts is a reasonable initial starting point—and at least offers more data that any other study to date.

RTTs vary for many reasons. We aim to assess one of these causes: whether buffers are naturally filling. We therefore make two assumptions in this paper. First, we assume routing does not change during each trace (1 hour) and hence we only make direct comparisons between RTT samples within traces. At this time scale we know routes can change, but generally these changes are subtle shifting within ASes along the path and not wholesale changes of the entire path [19]. Further, we note that buffers close to the remote peer will likely be constant even if the intermediate route changes. Second, we assume that all other delay fluctuations besides queue occupancy are relatively small. As sketched above, we know deep buffers exist and can cause large delays [16]. In comparison, fluctuations due to things like router processing of packets or local link-layer retransmissions are fairly minor and hence we do not believe will skew our results. While assumptions are never ideal, we do not believe our assumptions are unreasonable.

[4]Assessing queues within the CCZ is future work, but we presume these will be mostly non-existent given the 1 Gbps capacity of the network links.

[5]Some TCP implementations use the timestamp option [13] to remove the ambiguity and obviate the need for Karn's algorithm. Bro's TCP analysis code does not currently leverage this option and therefore neither does our analysis.

Many remote peers will be in residential settings where we know that low-end networking gear which is often configured with deep buffers will be found [16]. We use the SpamHaus PBL [24] to determine whether the remote peers in our dataset are end-user systems.[6] While the PBL cannot give us perfect information about whether a host is based in a residence, in this paper we colloquially denote peers found on the PBL as "residential". We also present "non-residential" traffic, as well, and hence the distinction does not leave us susceptible to blind spots. Further, the data given below lends some credence to this approximation in that it generally shows RTTs to residential peers to be longer than those for non-residential peers which follows our general intuition. Across our dataset we study the bufferbloat phenomena for nearly 54K residential peers and over 64K non-residential peers. A study of CCZ's traffic [23] shows that the majority of CCZ's outgoing data volume is peer-to-peer traffic and hence it is not surprising to find such a high fraction of remote peers also in residential settings.

Finally, we take the minimum RTT sample observed between two hosts as a baseline that does not include any queuing delay. We have no way to know the queue depth at any given time except in relative terms. Therefore, we use this baseline to assess how much additional queuing we observe in the network. While a persistent queue across a trace file (1 hour) would obscure some of the buffering present in the network we will at least be able to assess some of the buffering. Even if this assumption does not generally hold our results are at least a lower bound on the amount of buffering that happens in the network.

We use only IP pairs for which we gather at least 30 samples such that quick transactions with only a few packets do not skew our statistical analysis. Using our procedure we find 79 million RTT samples across our 14 month dataset.

2.2 Analysis

Figure 1 shows the distribution of non-residential (left) and residential (right) RTTs in our dataset. Each plot contains two lines. The "all samples" line is the distribution of all the RTT samples taken over the entire 14 month dataset, as described above. The "peer minimum" line is constructed by replacing each sample in dataset with the minimum RTT across the host pair within the given trace. This naturally weights the resulting CDF by the number of samples to a given remote peer. These plots lead us to several observations, as follows.

First, the RTT to residential peers is generally longer than to non-residential peers. The median of the peer minimum distribution to non-residential peers is 25 msec, while the corresponding point for residential peers is 78 msec. We believe the differences in the minimums are at least partially explained by low bandwidth residential links and geography. We use the MaxMind city-level GeoIP database [18] to lookup each remote IP address.[7] We find roughly 40% of residential peers are more than 2000 miles from the CCZ. For non-residential hosts we find only about 20% of residential peers are more than 2000 miles away. Therefore, the geographic distance at least partially explains why the distribution of minimum RTT samples to residential peers

[6]We have historical snapshots of the PBL and use the first snapshot of the day a given trace is taken to categorize hosts.

[7]We have monthly snapshots of the GeoIP database and use the corresponding snapshot for each of our traces.

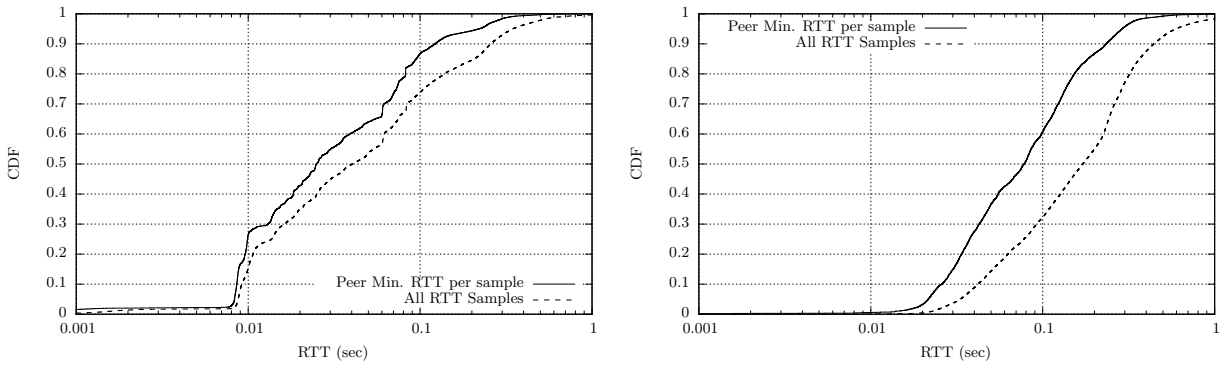

Figure 1: Distributions of minimum RTT sample to each remote peer for each RTT sample and all RTTs samples taken for non-residential (left) and residential (right) peers.

is generally larger than the corresponding distribution for non-residential peers.

The second observation from figure 1 is that 99.6% of residential RTT samples and 98.3% of non-residential RTT samples are less than one second. Previous measurements—both wide-scale [16] and anecdotal [10]—have illustrated that buffers are often deep enough to make RTTs of more than one second possible. However, we find that RTTs above one second are rare in normal traffic. A similar empirical result is shown in [21] as part of the case for setting TCP's initial retransmission timeout to 1 second. This result highlights the importance of (*i*) careful empirical study in identifying and understanding the scope of problems and (*ii*) carefully understanding the lessons from well-conducted previous experiments.

Even though we find RTTs generally less than one second in our dataset, our third observation is that buffering is happening to some extent. In both plots in figure 1 the "all samples" distribution shows longer RTTs than the peer minimum distribution. This indicates that in general TCP segments are encountering some queuing delay through the network and hence take longer than the baseline would suggest. An exception to this is non-residential hosts that are within 10 msec of the CCZ hosts. In this case our expectation is that any difference with the minimum does not necessarily reflect queuing, but at these time scales could be caused by myriad small issues. While our results show queuing does happen with in the network—as we would expect since queues are in place for a reason—Whether this constitutes "bloat" is a subjective judgment. All we conclude is that we do not find queues that impose the seconds of delay often touted as the hallmark of bufferbloat in our dataset.

Our final observation from figure 1 is that the amount of buffering is larger for residential peers than for non-residential peers. One reason for this could be that residential users generally have slower links than non-residential hosts and therefore draining a queue—which the FTTH-connected sender could readily fill—naturally takes longer. Additionally, residential users are both geographically and delay-wise further from the CCZ hosts. Therefore, traffic to residential peers may have a natural tendency to accumulate more delay as it passes through more routers. Finally, home networking gear may simply have a higher propensity to over-buffer as suggested in [16]. Without additional ancillary data we

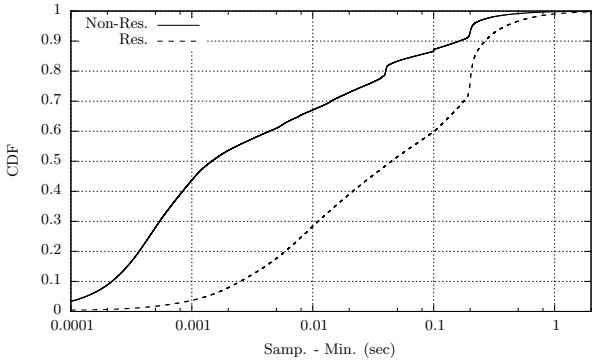

Figure 2: Distribution of each RTT sample minus minimum RTT to the given remote peer.

cannot ascribe a reason to the larger buffering to residential peers even though it is unmistakably present.

While figure 1 shows buffering does indeed happen, we now assess the magnitude of the phenomenon in our dataset. Figure 2 shows the distribution of difference between each sample in our dataset and the minimum RTT for the given host pair (within each trace). The median increase in RTT is just over 1 msec for non-residential peers and roughly 45 msec for residential peers. At the 99^{th} percentile the buffering represents 450 msec and 976 msec of added delay for non-residential and residential peers, respectively. In other words, fewer than 1% of the packets experience an RTT increase of more than 1 second.

Finally, we turn our attention to the buffering patterns we observe. First, we wish to gain an understanding of whether RTTs are generally increasing or decreasing across a pair of hosts in a given trace. We therefore count the number of times the RTT increases (decreases) across subsequent RTT samples for each host pair. In figure 3 (solid lines) we plot the ratio of the count of RTT increases to the count of RTT decreases for non-residential (top) and residential (bottom) peers. A ratio greater than one indicates the RTT increases more times than it decreases, with a ratio of less than one indicating the opposite. The plots show that in over 90% of the cases the host pair has either more RTT increases

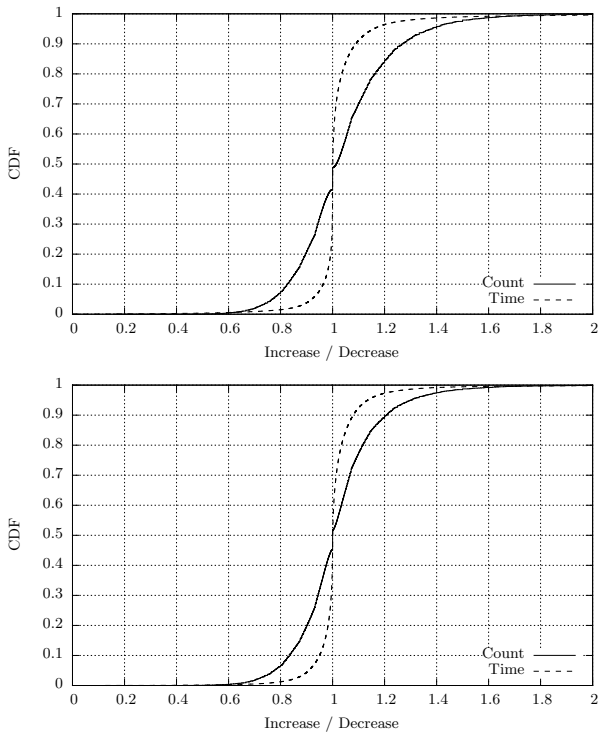

Figure 3: The distribution of the ratio of RTT increases to RTT decreases between subsequent samples in our traces. The top and bottom plots reflect non-residential and residential traffic, respectively.

or more RTT decreases. We only see rough parity between increases and decreases (i.e., a ratio of one) in about 10% of the cases. In general the split between an increasing trend and a decreasing trend is balanced—with non-residential remote hosts showing a bit more of an increasing trend than residential hosts.

In addition to a simple count, we measured the magnitude of the increase (decrease) between subsequent RTT samples and again plot the distribution of the ratio between the aggregate RTT increase and the aggregate RTT decrease. Figure 3 (dotted lines) shows that in terms of the magnitude of the RTT increase and decrease there is much more parity than in terms of the count. At least half the host pairs show a ratio of RTT increase to RTT decrease of roughly one. Again we find slightly more RTT increase than RTT decrease when taking into account magnitude instead of just a raw count—especially for non-residential traffic. Finally, the ratios are generally within 20% of parity and hence show that the aggregate increase or decrease over the active period of a host pair is modest. In other words, RTTs are not simply growing or declining, but are fluctuating up and down over time.

We also consider runs of RTT increases (decreases), both in terms of the number of samples in each run and the overall increase (decrease) in the RTT over the course of the run. We find that for all runs—increasing, decreasing, non-residential and residential—more than 97% consist of three or fewer samples. In terms of the magnitude of RTT increases across runs we find the 90^{th} percentile for increasing runs add 122 msec and 229 msec to the RTT for non-residential and residential traffic, respectively. This is consistent with the RTT increase illustrated above in figure 2. Further, we also find roughly the same results for decreasing runs, again indicating that RTTs are fluctuating and not simply rising or falling.[8]

Connections	CCZ	ICSI	LBL
Total	16.1M	20.5M	365M
Bogus	8.5M	18.0M	279M
Remaining	7.6M	2.5M	85M

Table 1: Connection log dataset description.

3. IMPACT OF THE INITIAL WINDOW

We next address the use of the suggested bufferbloat problem to influence Internet engineering decisions. As an exemplar, currently TCP's initial congestion window (IW)—the amount of data that can be transmitted before receiving an ACK—is 4,380 bytes [3].[9] A proposal is currently being considered to increase IW to 10 packets or 14,600 bytes [5]. A counter-argument to this proposal cites bufferbloat as one reason the proposed increase in the IW "must be considered deeply harmful" [9]. The major argument given in [9] against the larger IW revolves around the larger traffic impulse the proposal would create as the IW will more than triple. Additionally, [9] argues that browsers have a multiplicative effect by opening many parallel TCP connections. While [9] offers some brief analytical analysis there is no empirical evidence offered to support the claim that the impulse will be as problematic as stated. E.g., [9] claims in the abstract that the proposed increase in the IW "makes the problem of transient congestion at a user's broadband connection two and a half times worse". In this section we set out to evaluate the situation empirically.

3.1 Data

To analyze how bufferbloat interacts with a larger IW we use passively gathered traffic summaries to study the traffic impulses the proposed IW increase would trigger. We collect connection logs produced by Bro [20] for one week—March 20–26, 2012—from three vantage points in the network, as illustrated in table 1. The "CCZ" data comes from the CCZ project described in § 2.1. The "ICSI" data comes from the border between the International Computer Science Institute and the Internet and encompasses roughly 90 users' activity. Finally, the "LBL" data comes from the border between Lawrence Berkeley National Laboratory and the Internet and encompasses 3-4,000 users' activity. Note, these datasets are used simply to distill user activity which is then used to study the impact of a larger IW. Also note that while perhaps not reflective of precisely the traffic pattern at other points in the network, the crucial characteristic (as developed below) is that the distribution of transfer sizes is

[8]This fluctuation is expected to some degree due to TCP dynamics which multiplicatively decrease the sending rate when congestion is detected in an attempt to relieve the resource contention within the network. Subsequently, the congestion window is linearly increased until congestion is detected again.

[9]The actual IW specified in [3] is 2–4 packets depending on the MSS, but for purposes of this paper we consider the standard IW to be three packets of 1,460 bytes each.

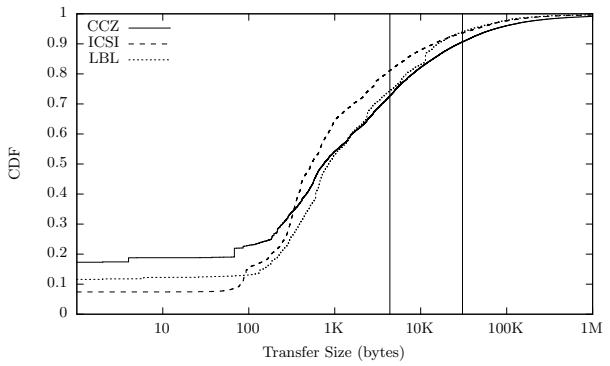

Figure 4: Distributions of transfer sizes to the monitored networks.

heavy-tailed, which has been well established in the literature (e.g., [6]). Therefore, we expect the insights found from our datasets to hold more generally.

The logs include all observed connections or partial connections (e.g., those connections started by a scanner but never fully established). We remove connections from further analysis for one of four reasons. First, a small number of records are simply malformed and hence unusable. Second, we removed non-TCP traffic. Third, some records are missing key information, such as byte counts in one or both direction or a connection duration, which indicates the connection accomplishes no useful work. Finally, we found a large *whois* harvest running at ICSI during our collection period. This activity skewed our data towards connections that transfer less than 100 bytes in both directions. Further, this traffic is (largely) caused by an experiment and not normal user activity. Therefore, we do not further consider *whois* connections. Table 1 summarizes our dataset. For our analysis we rely on each connection's timestamp, the involved hosts and the byte counts in each direction. We treat each connection as two uni-directional connections for traffic flowing from and to the monitored network.

3.2 Per-Connection Analysis

Our first analysis aims to understand the impact of allowing individual TCP connections to use an IW of up to 10 segments. Figure 4 shows the transfer size distribution for data sent to the monitored networks.[10] The data shows the expected heavy-tailed distributions.

First note the left-most vertical line on the plot is 4,380 bytes, the current standard IW. We find that across all distributions at least 73% of the transfers completely fit within the current IW and therefore increasing the IW will have no impact on any bufferbloat caused by these connections.

TCP uses the slow start algorithm [12, 3] to exponentially increase the *cwnd* to find an appropriate operating point for the observed conditions across the path. Therefore, regardless of IW used, sufficiently long transfers will increase the *cwnd* to the point allowed by the proposed IW. Precisely how long and how much data is required to increase the *cwnd* from 3 to 10 segments depends on (*i*) whether the TCP sender increases the *cwnd* according to the number of ACKs received or the number of bytes acknowledged [2] and (*ii*) whether the TCP receiver ACKs each segment or uses delayed ACKs [3] to ACK roughly every second segment. In theory the process requires 2–3 RTTs and 19–26 segments of data. The worst case[11] is a receiver that ACKs roughly every second segment (except when a second does not arrive within some timeout) and a sender that increases the *cwnd* based on the number of arriving ACKs. In this case for the current IW the first RTT has the sender transmitting 3 segments, which triggers 2 ACKs and hence the *cwnd* increases to 5 segments. The sender then transmits 5 segments in the second RTT. The process repeats and the sender transmits 8 segments in the third RTT and finally reaches a *cwnd* of 10 segments in the fourth RTT.[12] All told this process requires 26 data segments to build *cwnd* from 3 segment to 10 segments.

The right-hand vertical line in figure 4 is at 37,960 bytes—or 26 segments assuming an MSS of 1,460 bytes. All connections in the distributions to the right of this line will attain a *cwnd* of 10 segments even if the IW remains at 3 segments. The plot shows that 6–8% of the connections will utilize a *cwnd* of at least 10 segments even if the IW is 3 segments. While these connections will be more bursty at the beginning with the larger IW—and hence require buffering—the overall impact of the IW on bufferbloat is likely to be modest at best since these connections are long and their overall behavior not dictated by TCP's initial parameters.

We are then left with the connections that fall between the vertical lines in figure 4. These connections carry more data than will fit in the currently standard IW, but not enough to increase the *cwnd* to the proposed initial value of 10 segments. Therefore, these connections arguably have the potential to transiently increase the size of the buffers along the network path when using an IW of 10 segments as opposed to an IW of 3 segments. Depending on the vantage point we find 7–20% of the connections falling into this middle area. Note that while these connections will increase transient burstiness, they are relatively short and therefore cannot drive long-term buffer occupancy.

We use the distribution with the largest proportion of connections in the middle area—i.e., the worst case—to further assess the possible impact on intermediate buffers. The middle portion of the LBL distribution represents roughly 20% of the connections. For each of these connections we calculate the *additional impulse* that an IW of 10 segments would add to the sending pattern compared to using an initial *cwnd* of 3 segments. As developed above, assuming TCP is using delayed ACKs and increasing the *cwnd* based on the number of returning ACKs the largest additional impulse for these middle-sized connections is 17 segments (i.e., 7 and 10 additional segments in the first and second RTTs respectively).

We find that for the LBL data, 88% of the connections in the middle portion of the distribution would send one additional impulse of up to 10,220 bytes—or 7 segments—worth of data, with the size of the impulse spread fairly uniformly from 1– 10,220 bytes. The remaining 12% of the connections will add a full 7 segment first impulse and then

[10]The transfer sizes in the opposite direction are generally a bit shorter, but are not presented due to space constraints. We derive no additional insights from scoping our presentation to only incoming traffic.

[11]We assume no ACK loss, which would in some cases make for an even slower process.

[12]The sender would have a *cwnd* of 12 segments in the fourth RTT, but our analysis stops at the point of the proposed IW.

a second additional impulse of up to 14,600 bytes—or 10 segments.

One additional note is that while a larger IW will increase the burstiness of the traffic with these additional impulses, there will also be periods of less activity than the current case. In other words, because a larger initial *cwnd* makes TCP more aggressively transmit data, the duration of the transmission is likely to be reduced, hence leaving periods of less buffer utilization.

Our per-connection analysis suggests transfer sizes largely dictate that either a larger IW cannot cause additional buffering or that regardless of the IW the connection will utilize a *cwnd* of at least the proposed larger IW and hence the initial parameter has a minor effect. Further, in up to 20% of the cases we find the proposed IW will cause modest additional traffic impulses that would cause additional short-term buffering compared to the currently standard IW.

3.3 Per-Aggregate Analysis

While our per connection analysis is a start, [9] correctly notes that parallel TCP connections can cause a multiplicative effect, whereby even if the increased IW is reasonable for a single connection it may be problematic across a set of connections that are all running simultaneously. To assess this we group connections into aggregates. Our connection logs are not amenable to drawing fine-grained dependence between connections (e.g., that two connections represent two images loading for single web page). Therefore, we adopt the following conservative approach. For each local host we monitor we consider two connections (regardless of peer) to be in the same aggregate if their start times differ by no more than some threshold t seconds. For the initial analysis in this paper we choose $t = 1$ *sec*, but we have tried other values and the insights presented below are not highly sensitive to this choice. This process is conservative in that the aggregated connections may not share a buffer, either because they are disparate enough in time or are traversing different network paths.

We find that using the this strategy leads to aggregates of a single connection in over 55% of the cases across the three datasets (and close to 70% in the ICSI and LBL datasets). Additionally, at least 92% of the aggregates are less than 10 connections and the 99^{th} percentile is 37 connections.

After constructing the aggregates we calculate the effective size of the initial *cwnd* across all connections within the aggregate. Note, we assume that (*i*) all data transmitted on each connection is immediately available when transmission begins and (*ii*) that all connections in an aggregate start at the same time rather than following their natural pattern. These assumptions will exacerbate the buffering requirement, but makes the analysis tractable and serves to provide a rough indication of the impact of aggregates on buffering. We then calculate the effective IW across the aggregate based on the data volume and individual connection IW values of 3 and 10 segments. We then compute the difference in the effective aggregate initial *cwnd* when using the two connection-based values.

Figure 5 shows the distributions of the additional aggregate IW that would occur if an IW of 10 segments were universally adopted as opposed to using an IW of 3 segments, for all connections sending traffic to the monitored network.[13] We find at least two-thirds of the aggregates

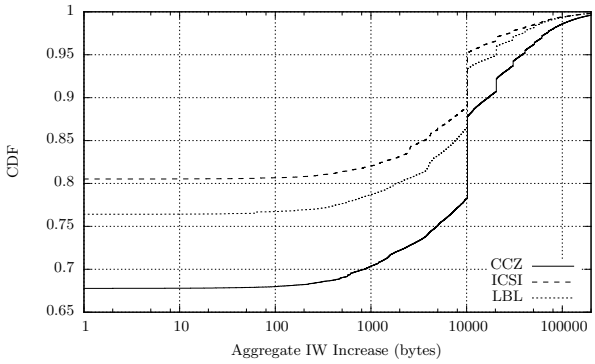

Figure 5: Distribution of effective IW increase with IW=10 vs. IW=3.

(over 80% for ICSI) do not present any increase in the aggregate IW when each individual connection is allowed to use a larger IW. This is expected from our connection-level analysis as most of the connections do not even fully utilize the current IW. Aggregating such connections therefore also has no impact on the buffering requirement.

We also find that the mode across datasets is 10,220 bytes—or 7 segments of additional initial *cwnd* usage. This nearly always corresponds to aggregates that contain only one connection and can use the full proposed IW. Finally, in 5–12% of the cases (depending on distribution) we find aggregate IW values in excess of 10,220 bytes. We find a few aggregates that increase the effective initial *cwnd* by 100+ KB or even megabytes in some cases. However, it is important to note that in such cases even using an IW of 3 segments represents a significant load on the network and the buffers.

Given the above, we conclude that much like our single connection analysis, aggregates would cause modest additional buffering demands. However, in at least two-thirds of the cases there will be no impact and the impact will be modest—up to 7 additional segments—in the majority of the remaining cases.

4. FUTURE WORK

As we note above, this short paper presents only an initial investigation of the bufferbloat phenomena. We encourage the community to continue to investigate bufferbloat. Several areas for future work include:

- Bringing datasets from additional vantage points to bear on the questions surrounding bufferbloat is unquestionably useful. While we study bufferbloat related to 118K peers for some modest period of time (up to one week), following more peers and over the course of a longer period would be useful.
- While we are able to assess 118K peers, we are only able to do so opportunistically when a host on the network we monitor communicates with those peers. A vantage point that provides a more comprehensive view of residential peers' behavior would be useful.
- While we still presume residential broadband equipment has too much buffering—even if we find that

[13]As in the last section we analyzed the traffic sent by the monitored network, as well, and found the same insights.

it is not often utilized—and we find that queues are built (to modest levels), and therefore a concrete understanding of where in the network queues actually form would be useful.

- In addition to passive observation, long term active measurements from many vantage points can offer insights into delay fluctuations throughout the network.
- Finally, while we concentrate on understanding how buffers are actually used in the network, we also encourage continuing studies such as [16] that illuminate the amount of buffering present—even if unused—across network paths.

5. SUMMARY

Our contribution is an initial systematic, empirical study of the bufferbloat issue. Our investigation shows that bufferbloat does indeed happen—and more so in residential than non-residential networks. However, the magnitude of the problem as manifest in live traffic is modest. In particular, we find bloat bounded at (*i*) 100 msec for more than half the samples and (*ii*) 250 msec for at least 94% of the samples. We additionally find little evidence of large scale persistent queues, as opposed to fluctuating queues. Further we study the impact of increasing TCP's IW to 10 segments on bufferbloat. We find that in the majority of the cases an initial *cwnd* of 10 segments will have no impact on buffering. Further, in the remaining cases the larger initial *cwnd* will result in modestly larger bursts and hence a more aggressive use of buffering. These results hold when considering aggregates of multiple parallel connections, as well.

We stress that we are not concluding deep buffers are not possible nor arguing that filling deep buffers is not problematic. To the contrary, we believe the literature has well established that deep buffers can occur and further when they do they are suboptimal for some traffic. However, our results show a disconnect between whether long queues *can* occur and how often they *do* occur. We believe this is a key distinction in reasoning about the magnitude of the issue.

Finally, based on our study and the wider literature on bufferbloat, we make three meta-comments the we believe apply broadly to networking research and engineering.

- We must carefully understand what previous research shows, but also what it does not show. E.g., [16] shows that large delays due to buffering can happen, not that they do happen.
- While it is sometimes easy to dismiss some measurement papers as "just another data dump", these can have significant value in informing our mental models as to how the network operates. In the absence of empirical assessment of the Internet we are left with mental models that are potentially skewed by our own (limited) experiences.
- We should aim to understand problems before we set out to find solutions. Without understanding problems our "solutions" may well simply cause more problems. See [11] for a recent exploration of this pitfall in the context of video streaming over TCP.

Acknowledgments

This paper benefits from a great many conversations with a list of people too long to name. Katrina LaCurts wrote and debugged the Bro TCP analysis code we leverage. Vern Paxson provided the LBL data used in § 3. Rob Beverly, Oliver Hohlfeld, Vern Paxson and the anonymous reviewers provided useful comments on a draft of this paper. This work is funded in part by the National Science Foundation under grant CNS-1213157. My thanks to all!

6. REFERENCES

[1] Case Connection Zone. `http://caseconnectionzone.org/`.
[2] M. Allman. TCP Congestion Control with Appropriate Byte Counting (ABC), Feb. 2003. RFC 3465.
[3] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control, Sept. 2009. RFC 5681.
[4] S. Bauer, R. Beverly, and A. Berger. Measuring the state of ecn readiness in servers, clients,and routers. In *ACM Internet Measurement Conference*, 2011.
[5] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis. Increasing TCP's Initial Window, Feb. 2012. Internet-Draft draft-ietf-tcpm-initcwnd-03.txt (work in progress).
[6] M. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5(6), Dec. 1997.
[7] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4), Aug. 1993.
[8] J. Gettys. Bufferbloat: Dark buffers in the internet. *IEEE Computing*, May/June 2011.
[9] J. Gettys. IW10 Considered Harmful, Aug. 2011. Internet-Draft draft-gettys-iw10-considered-harmful-00.txt (work in progress).
[10] J. Gettys and K. Nichols. Bufferbloat: Dark Buffers in the Internet. *ACM Queue*, Nov. 2011.
[11] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *ACM Internet Measurement Conference*, Nov. 2012.
[12] V. Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM*, 1988.
[13] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance, May 1992. RFC 1323.
[14] H. Jiang, Y. Wang, K. Lee, and I. Rhee. Tackling Bufferbloat in 3G/4G Networks. In *ACM Internet Measurement Conference*, Nov. 2012.
[15] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. In *ACM SIGCOMM*, Aug. 1987.
[16] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the edge network. In *ACM Internet Measurement Conference*, Nov. 2010.
[17] K. LaCurts. Bro tcp anomaly script, 2012.
[18] L. MaxMind. City-level geoip database. `http://www.maxmind.com/`.
[19] V. Paxson. End-to-End Routing Behavior in the Internet. In *ACM SIGCOMM*, Aug. 1996.
[20] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, Dec. 1999.
[21] V. Paxson, M. Allman, H. J. Chu, and M. Sargent. Computing TCP's Retransmission Timeout, June 2011. RFC 6298.
[22] K. K. Ramakrishnan and S. Floyd. A Proposal to Add Explicit Congestion Notification (ECN) to IP, Jan. 1999. RFC 2481.
[23] M. Sargent, B. Stack, T. Dooner, and M. Allman. A First Look at 1 Gbps Fiber-To-The-Home Traffic. Technical Report 12-009, International Computer Science Institute, Aug. 2012.
[24] The Spamhaus Project - PBL. `http://www.spamhaus.org/pbl/`.