On the Impact of Bursting on TCP Performance

Ethan Blanton[1] and Mark Allman[2]

[1] Purdue University, eblanton@cs.purdue.edu
[2] ICSI/ICIR, mallman@icir.org

Abstract. Periodically in the transport protocol research community, the idea of introducing a *burst mitigation strategy* is voiced. In this paper we assess the prevalence and implications of bursts in the context of real TCP traffic in order to better inform a decision on whether TCP's congestion control algorithms need to incorporate some form of burst suppression. After analyzing traffic from three networks, we find that bursts are fairly rare and only large bursts (of hundreds of segments) cause loss in practice.

1 Introduction

Within the transport protocol research community, an idea that crops up periodically is that of introducing *bursting mitigation* into the standard congestion control algorithms. Transport protocols can naturally send line-rate bursts of segments for a number of reasons (as sketched below). Studies have shown that these line-rate bursts can cause performance problems by inducing a rapid build-up of queued segments at a bottleneck link, and ultimately dropped segments and a reduced transmission rate when the queue is exhausted.

In this paper we focus on the study of *micro-bursts* and exclude *macro-bursts* from consideration. A micro-burst is a group of segments transmitted at line rate in response to a single event (usually the receipt of an acknowledgment). A macro-burst, on the other hand, can stretch across larger time scales. For instance, while using the slow start algorithm [APS99], TCP[3] increases the congestion window (and therefore the transmission rate) exponentially from one round-trip to the next. This is an increase in the macro-burstiness of the connection. The micro-burstiness, however, is unaffected as TCP sends approximately 2–3 segments per received acknowledgment (ACK) throughout slow start (depending on whether the receiver employs delayed ACKs [Bra89, APS99]).

An example of naturally occurring bursting behavior is given in [Hay97], which shows that TCP connections over long-delay satellite links with advertised windows precisely tuned to the appropriate size for the delay and bandwidth of the network path suffer from burst-induced congestion when loss occurs. Ideally, a TCP connection is able to send both retransmissions and new data

[3] The measurements and discussions presented in this paper are in terms of TCP, but also apply to SCTP [SXM+00] and DCCP's [KHF04] CCID 2 [FK04], since they use similar congestion control techniques.

segments during a loss recovery phase [FF96]. However, if there is no room in the advertised window, new segments cannot be sent during loss recovery. Upon exiting loss recovery (via a large cumulative ACK), TCP's window will slide and a line-rate burst of segments will be transmitted. [Hay97] shows that this burst — which is roughly half the size of the congestion window before loss — can result in an overwhelmed queue at the bottleneck link, causing further loss and additional performance sacrifice. [Hay97] also shows this bursting situation to apply to a number of TCP variants (Reno [APS99], NewReno [Hoe96, FF96], FACK [MM96], etc.). Finally, [Hay97] shows that bursts *can impact TCP performance*, but the experiments outlined are lab-based and offer no insight into how often the given situation arises in the Internet. In this paper we assess the degree to which these micro-bursting situations arise in the wild in an attempt to inform a decision as to whether TCP should mitigate micro-bursting. While out of scope for this paper we note that [AB04] compares a number of burst mitigation techniques. In addition to advertised window constraints discussed above, micro-bursts can be caused by several other conditions, including (but not limited to):

- **ACK loss.** TCP uses a cumulative acknowledgment mechanism that is robust to ACK loss. However, ACK loss causes TCP's window to slide by a greater amount with less frequency, potentially triggering longer than desired bursts in the process.
- **Application Layer Dynamics.** Ultimately, the application provides TCP with a stream of data to transmit. If the application (for whatever reason) provides the data to TCP in a bursty fashion then TCP may well transmit micro-bursts into the network. Note: an operating system's socket buffer provides a mechanism that can absorb and smooth out some amount of application burstiness, especially in bulk transfer applications. However, the socket buffer does not always help in applications that asynchronously obtain data to send over the network.
- **ACK Reordering.** Reordered ACKs[4] cause an ACK stream that appears similar to a stream containing ACK loss. If a cumulative ACK "passes" ACKs transmitted earlier by the endpoint, then the later ACK (which now arrives earlier) triggers the transmission of a micro-burst, while the earlier ACKs (arriving later) will be thrown away as "stale".

The causes of bursting discussed above are outlined in more detail in [JD03] and [AB04]. Also note that the causes of bursts are not TCP variant specific, but rather apply to all common TCP versions (Reno, NewReno, SACK, etc.).

[JD03] also illustrates the impact of micro- and macro-bursts on aggregate network traffic. In particular, [JD03] finds that these source-level bursts create

[4] Reordered data segments can also cause small amounts of bursting, if the reordering is modest. However, if the reordering is too egregious then false loss recovery will be induced, which is a different problem from bursting. For a discussion of the issues caused by data segment reordering, see [BA02, ZKFP03].

scaling in short timescales and can cause increased queuing delays in intermediate nodes along a network path. In contrast, in this paper we concentrate on characterizing bursts and determine the frequency of bursts. We then use the analyzed data to inform a discussion on whether it behooves TCP to prevent bursts from a performance standpoint, as proposed in the literature (e.g., in [FF96, HTH01]).

We offer several contributions in this paper after outlining our measurement methodology in § 2. First, we characterize observed, naturally occurring micro-bursts from three networks in § 3. Next, we investigate the implications of the observed micro-bursts in § 4. Finally, in § 5 we conclude with some preliminary discussion into the meaning of the results from § 3 and § 4 as they relate to the question of whether a burst mitigation mechanism should be introduced into TCP.

2 Measurement Methodology

First, we define a "burst" for the remainder of the paper as a sequence of at least 4 segments sent between two successive ACKs (i.e., a "micro-burst"). The "magic number" of 4 segments comes from the specification of TCP's congestion control algorithms [APS99]. On each ACK during slow start, and roughly once every 1–2 round-trip times (RTT) in congestion avoidance, TCP's algorithms call for the transmission of 2–3 segments at line-rate. Therefore, micro-bursts of 3 or fewer segments have been deemed reasonable in ideal TCP and are common in the network. We consider bursts of more than 3 segments to be "unexpected", in that they are caused by network and application dynamics rather than the specification of the congestion control algorithms. That is not to say that TCP is in violation of the congestion control specification in these circumstances — just that outside dynamics have caused TCP to deviate from the envisioned sending pattern. These unexpected bursts are the impetus for the various proposals to mitigate TCP's burstiness, and therefore they are the focus of our study. We do note that [AFP02] allows TCP to transmit an initial 4 segment burst when the maximum segment size is less than 1096 bytes. However, this is not taken into account in our definition of a burst since it is a one-time only allowance.

In principle, the above definition of a burst is sound. However, in analyzing the data we found a significant vantage point problem in simply using the number of segments that arrive between two successive ACKs. As shown in [Pax97] there is a general problem in TCP analysis with matching particular ACKs with the packets they liberate — even when the monitoring point is the end host involved in the TCP connection. However, when the monitoring point is not the end host the problem is exacerbated. Table 1 illustrates the problem by showing the different order of events observed inside the TCP stack of the end host and at our monitor point. In the second column of this example, two ACKs arrive at the end host and each trigger the transmission of two data segments, as dictated by TCP's sliding window mechanism. However, the third column shows a different (and frequently observed) story from the monitor's vantage

Event Number	End Host	Monitor
0	ACK_{n+1}	ACK_{n+1}
1	$DATA_{m+1}$	ACK_{n+2}
2	$DATA_{m+2}$	$DATA_{m+1}$
3	ACK_{n+2}	$DATA_{m+2}$
4	$DATA_{m+3}$	$DATA_{m+3}$
5	$DATA_{m+4}$	$DATA_{m+4}$

Table 1. Example of vantage point problem present in our datasets.

point. In this case, the monitor observes both ACKs before observing any of the data segments and subsequently notes all four data segments transmitted. Using the notion sketched above, these four data segments would be recorded as a burst when in fact they were not. This scenario can, for example, be caused by ACK compression [Mog92]. If the ACKs are compressed before the monitor such that they arrive within t seconds, where t is less than the round-trip time (RTT) between the monitor and the end host, r, then the situation illustrated in table 1 will be found in the traces. An even thornier problem occurs when a group of compressed ACKs arrives over an interval a bit longer than r. In this case, the overlap between noting ACKs and data packets makes it nearly impossible to untangle the characteristics of the bursts (or, even their presence).

We cope with this problem by *accumulating* the ACK information. For instance, in table 1 since two ACKs without any subsequent data segments are recorded our analysis allows up to 6 data segments to be sent before determining a burst occurred. This heuristic does not always work. For instance, if 6 data segments were observed between two subsequent ACKs in a trace file there is no way to conclusively determine that 3 data segments were sent per ACK. The case when the first ACK triggered 2 data segments and the second ACK triggered 4 data segments (a burst) is completely obfuscated by this heuristic. Another problem is that ACKs could conceivably be lost between the monitor and the end host which would likewise cause the analysis to mis-estimate the bursting characteristic present on the network. In our analysis, if we note more than $3N$ segments sent in response to N ACKs we determine a burst has been transmitted. The length of this burst is simply recorded as the number of segments noted. In other words, we do not attempt to ascribe some of the data segments to each ACK. This is surely an overestimate of the size of the burst. However, as will be shown in the following sections, this small vantage point problem is unlikely to greatly impact the results because the results show that the difference between bursts of size M and bursts of size $M \pm x$ for some small value of x (e.g., 1–10) is negligible. Therefore, while the numbers reported in this paper are slight mis-estimates of the true picture of bursting in the network we believe the *insights* are solid.

To assess the prevalence and impact of micro-bursting, we gathered four sets of packet traces from three different networks. We analyze those connections

Dataset	Start	Duration	Servers	Clients (/24s)	Conns.	Bogus
Anon	7/24/03	≈26 hours	1,202	5,319 (4,541)	295,019	5,955 (2.0%)
LBNL	10/22/03	≈11 hours	947	22,788 (19,689)	196,085	2,362 (1.2%)
$ICSI_1$	1/4/04	≈14 days	1	24,752 (21,571)	223,906	221 (0.1%)
$ICSI_2$	9/18/04	≈14 days	1	23,956 (20,874)	198,935	114 (0.1%)

Table 2. Dataset characteristics.

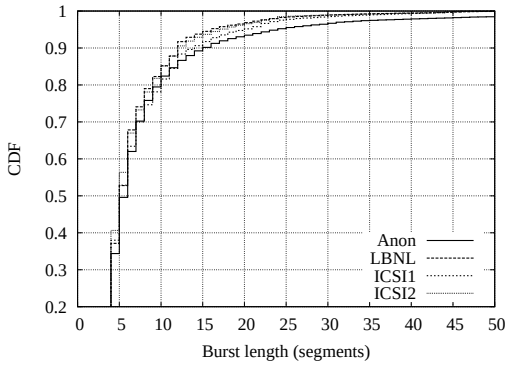
involving web servers on the enterprise network. That is, we focus on local web servers' sending patterns, rather than the sending patterns of remote servers that are responding to local web clients' requests. The characteristics of the four trace files used in our study are given in table 2. The first trace, denoted Anon, consists of roughly 26 hours of web traffic recorded near web servers at a biology-related research facility that asked not to be identified. The tracing architecture is, however, outlined in [MHK+03]. The second trace represents roughly 11 hours of web server traffic at the Lawrence Berkeley National Laboratory (LBNL) in Berkeley, CA, USA. The final two datasets represent requests to a single web server at the International Computer Science Institute (ICSI), also in Berkeley, during two different two week periods in 2004.

These packet traces are analyzed with a custom-written tool called *conninfo*, which analyzes the data-carrying segments sent by web servers on the enterprise network. *Conninfo* mainly tracks the number of data segments sent between two subsequent pure acknowledgment (ACK) segments as sketched above. In addition to recording micro-burst sizes, *conninfo* also records which (if any) segments within a burst are retransmitted. Finally, *conninfo* records several ancillary metrics such as the total data transfer size, the duration of the connection, etc.

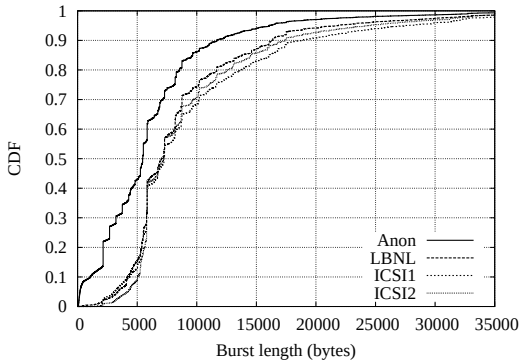
Conninfo attempts to process each connection in the dataset. However, as indicated in the last column of table 2, a small fraction of connections were removed from each dataset. These connections exhibit strange behavior that *conninfo* either does not or cannot understand; for example, several "connections" (which are perhaps some sort of attack or network probe) consist of a few random data segments with noncontiguous sequence numbers. As the table shows, the fraction of connections removed from further analysis is relatively small and, therefore, we do not believe this winnowing of the datasets biases the overall results presented in this paper.

3 Characterizing Bursts

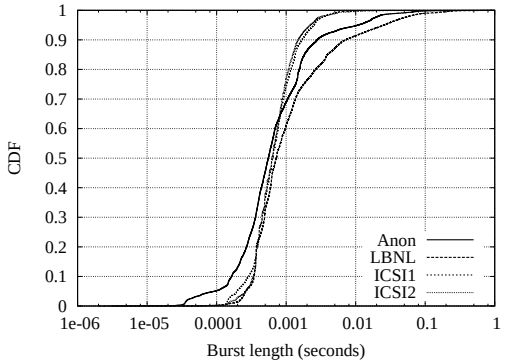
In this section we provide a characterization of the bursts observed in the traces we studied. Figure 1 shows the distributions of burst sizes in each of the datasets in terms of both segments, bytes and time. Figure 1(a) shows that the distribution of burst sizes when measured in terms of segments is similar across all datasets. In addition, the figure shows that over 90% of the bursts are less than 15 segments in length. Figure 1(b) shows the burst size in terms of bytes per



(a) Segments



(b) Bytes



(c) Time

Fig. 1. Distribution of burst sizes.

burst. This distribution generally follows from the segment-based distribution if 1500 byte segments are assumed. While the LBNL and ICSI datasets are similar in terms of byte-based burst size, the Anon distribution indicates smaller bursts. Since we did not observe smaller bursts in the Anon dataset when measuring in terms of segments, it appears that the segment sizes used in the Anon network are generally smaller than at LBNL and ICSI. We generated packet size distributions for all networks and there is a clear mode of 20% in the Anon dataset at 576 bytes that is not present in either the ICSI or LBNL datasets. Otherwise, the distributions of packet sizes are roughly the same, thus explaining the discrepancy in figure 1.

Figure 1(c) shows the distribution of the amount of elapsed time covered by each burst. This plot confirms that the vast majority of bursts happen within a short (less than 10 msec) window of time. This is essentially a double-check that our methodology of checking data between subsequent ACKs and our analysis tool are both working as envisioned. While we found bursts that encompass a fairly long period of time (over a second) these are the exception rather than the rule and upon close examination of the time-sequence plots these look to be artifacts of network dynamics mixing with application sending patterns that are difficult to systematically detect. Therefore, we believe that our analysis techniques are overall sound.

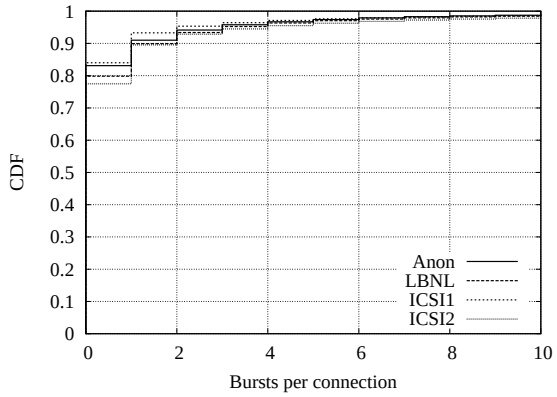


Fig. 2. Distribution of bursts per connection.

Next we turn our attention to the prevalence of bursts. Figure 2 shows the distribution of the number of bursts per connection in our four datasets. The figure shows that the burst prevalence is roughly the same across datasets. As shown, over 75% of the connections across all the datasets experienced no bursts of 4 or more segments. However, note that many of the connections that did not burst could not because of the limited amount of data sent or because TCP's

Dataset	Bursts	Initial Window	Exit Loss Recovery	Stretch ACKs	Window Opening	App. Pattern	Unknown
Anon	274,880	1.8	0.2	26.3	5.0	17.0	49.6
LBNL	187,176	0.9	0.3	22.9	3.1	32.8	40.0
$ICSI_1$	165,023	6.4	0.7	23.5	4.8	24.0	40.6
$ICSI_2$	228,063	4.2	5.1	22.4	4.5	23.3	45.1

Table 3. Percentage of bursts triggered by the given root cause.

congestion window never opened far enough to allow 4 or more segments to be transmitted.

Next we look beyond the on-the-wire nature of bursts and attempt to determine the *root cause* of the bursts. Table 3 shows the determined causes of the bursts found in each dataset. First, the second column of the table shows that each dataset contains a wealth of bursts. Next, the third column of the table shows that 1–6% of the bursts are observed in the initial window of data transmission. The fourth column shows a similarly small amount of bursting caused by the sender being limited by the advertised window during loss recovery and then transmitting a burst upon leaving loss recovery (when a large amount of the advertised window is freed by an incoming ACK). These first two causes of loss account for a small fraction of the bursts, but the fraction does vary across datasets. We have been unable to derive a cause for this difference and so ascribe it to the heterogeneous nature of the hosts, operating systems, routers, etc. at the various locations in the network.

The fifth column in table 3 shows that roughly 20–25% of the bursts are caused by stretch ACKs (acknowledgments that newly ACK more than 2 times the number of bytes in the largest segment seen in the connection). Stretch ACKs arrive for a number of reasons. For instance, some operating systems generate stretch ACKs [Pax97] in the name of economy of processing and bandwidth. In addition, since TCP's ACKs are cumulative in nature simple ACK loss can cause stretch ACKs to arrive. Finally, ACK reordering can cause stretch ACKs due to an ACK generated later passing an earlier ACK. The origin of each stretch ACK is therefore ambiguous given our limited vantage point, and hence we did not try to untangle the possible root causes.

The sixth column represents a somewhat surprising bursting cause that we did not expect. From the server's vantage point we observe ACKs arriving from the web client that acknowledge the data transmitted as expected but that do not free space in the advertised window — and, hence, do not trigger further data transmission when the sender is constrained by the advertised window. When an ACK that opens advertised window space finally does arrive a burst of data is transmitted. This phenomenon happens in modest amounts (3–5% of bursts) in all the datasets we examined.

The seventh column in the table shows the percentage of bursts caused by the application's sending pattern. We expected this cause of bursts to be fairly

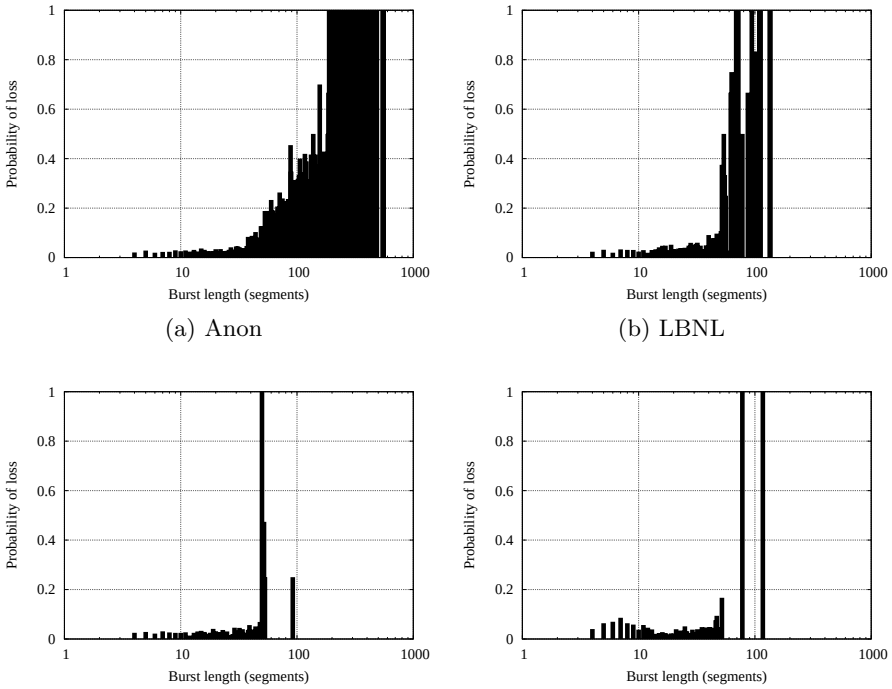
low since our mental model of web transfers is that objects are pushed onto the network as fast as possible. However, 17–33% of the bursts happened after all transmitted data was acknowledged and no other bursting scenario explained the burst, indicating that the data was triggered by the application rather than being clocked out by the incoming ACKs. This could be explained by a persistent HTTP connection that did not use pipelining [FGM+97] — or, which was kept open while the user was viewing a given web page and then re-used to fetch another item from the same server.

Finally, the last column of the table is the most troubling in that it indicates that we could not accurately determine the cause of 40–50% of the bursts across all the datasets. Part of the future work in this area will be to develop additional techniques to determine why this bursting is happening. However, the problem is daunting in that we examined a large number of time-sequence plots for connections containing the unknown burst causes and at times we could not figure out why the burst happened ourselves — let alone design a heuristic to detect it!

4 Implications of Bursts

In this section we explore the implications of the bursting uncovered in the last section on the TCP connections themselves. It is beyond our scope (and data) to evaluate the implications the bursting has on competing traffic and the network itself. Figure 3 shows the probability of losing at least one segment in a burst as a function of the burst size (in segments) for each of our datasets. The figure shows that for modest burst sizes (tens of segments or less) that the probability of losing a segment from the burst is fairly low (roughly less than 5%). As the burst size increases, the likelihood of experiencing a drop within a burst also increases. Bursts on the order of hundreds of segments in our datasets are clearly doomed to overwhelm intervening routers and experience congestion. One interesting note is in the shape of the plots. The Anon dataset shows a fairly smooth ramp-up in the probability of loss in a burst as the burst size increases. However, in both the LBNL and ICSI datasets there is a clear point at which the chances of losing at least one segment in a burst jumps from less than 5% to over 20% and often to 100%. In the LBNL dataset this happens when burst size reaches approximately 60 segments and in the ICSI dataset when the burst size reaches roughly 50 segments. These results may indicate a maximum queue size at or near LBNL and ICSI that ultimately limits the burst size that can be absorbed. The Anon network may be more congested than the ICSI or LBNL networks and therefore the chances of a non-empty queue vary with time and hence the ability to absorb bursts likewise varies over time. Alternatively, the Anon results could indicate the presence of active queue management, whose ability to absorb bursts depends on the traffic dynamics at any given point in time.

The analysis above assesses the question of whether bursts cause *some* loss. Next we focus our attention on the amount of loss caused by bursts. In other



(a) Anon (b) LBNL

(c) ICSI_1 (d) ICSI_2

Fig. 3. Probability of losing at least one segment in a burst as a function of burst size (in segments).

Dataset	Conns.	Bursts	Burst Loss Rate (%)	Non-Burst Loss Rate (%)
Anon	4,233	69,299	70.9	41.2
LBNL	5,685	45,282	23.0	22.9
$ICSI_1$	4,805	39,832	16.1	14.5
$ICSI_2$	8,201	72,069	26.6	20.5

Table 4. Retransmission rates observed inside and outside bursts.

words we address the question: is the loss rate higher when bursting than when not bursting? Given the information at hand we cannot determine precise loss rates and therefore use the retransmission rate as an indicator (understanding that the retransmission rate and the loss rate can be quite different depending on loss patterns, TCP variant used, etc. [AEO03]). We first winnow each dataset to connections that experience both bursting and retransmissions to allow for a valid comparison. This has the effect of making the reported rates *appear* to be much higher than loss rates measured in previous network studies (e.g., [AEO03]) because we are disregarding all connections that experienced no loss. Table 4 shows the results of our analysis. When comparing this table with tables 2 and 3 it is apparent that only a small minority of the connections from each dataset contain both bursting and retransmissions. The table shows the aggregate retransmission rate to be higher when connections are bursting than when connections are not bursting. The change in the retransmission rate ranges from nearly non-existent in the LBNL dataset to a roughly 75% increase in the Anon dataset. The large increase in the Anon network agrees with the results presented above that the network is generally more congested and the bottleneck queue closer to the drop point than the other networks we studied. Therefore, bursts cause a large increase in the loss rates experienced in this network while the other networks were better able to absorb the bursts.

5 Conclusions and Future Work

The work in this paper is focused on the impact of bursting on TCP connections themselves. From the above preliminary data analysis we note that micro-bursts are not frequent in TCP connections — with over 75% of the connections in the three networks studied showing no bursting. When bursting does occur, burst sizes are predominantly modest with over 90% of the bursts are less than 15 segments across the datasets we studied. Furthermore, in these modest bursts the probability of experiencing loss within the burst is small (generally less than 5% across datasets). However, bursts of hundreds of segments do occur and such large bursts nearly always experience some loss. We analyzed the cause of bursts and found the two predominant known causes of bursting to be the reception of stretch ACKs and application sending patterns. Unfortunately, our analysis techniques also failed to find the cause of 40–50% of the bursts we observed. An area for future work will be to further refine the analysis to gain further insights

into these unclassified bursts (however, as described in § 3 this is a challenging task). Finally, we find an increase in the loss rate experienced within bursts with the loss rate experienced outside of bursts. The increase ranged from slight to approximately 75% depending on the network in question.

A key piece of future work is in understanding how the results given in [JD03] relate to those given in this paper. That is, the preliminary results of this paper indicate that micro-bursting is not likely to hurt performance, while [JD03] shows that the network impact of bursting is non-trivial. Before applying a mitigation to TCP to smooth or reduce bursts it would be useful to correlate the network issues found in [JD03] with specific bursting situations. For instance, if only particular kinds of bursting are yielding the scaling behavior noted in [JD03] then mitigating only those bursting situations may be a desirable path forward.

Acknowledgments

Andrew Moore and Vern Paxson provided the Anon and LBNL datasets, respectively. Sally Floyd, Vern Paxson and Scott Shenker provided discussions on this study. The anonymous reviewers provided good feedback on the submission of this paper and their comments improved the final product. This work was partially funded by the National Science Foundation under grant number ANI-0205519. Our thanks to all!

References

[AB04] Mark Allman and Ethan Blanton. Notes on Burst Mitigation for Transport Protocols. December 2004. Under submission.

[AEO03] Mark Allman, Wesley Eddy, and Shawn Ostermann. Estimating Loss Rates with TCP. *ACM Performance Evaluation Review*, 31(3), December 2003.

[AFP02] Mark Allman, Sally Floyd, and Craig Partridge. Increasing TCP's Initial Window, October 2002. RFC 3390.

[APS99] Mark Allman, Vern Paxson, and W. Richard Stevens. TCP Congestion Control, April 1999. RFC 2581.

[BA02] Ethan Blanton and Mark Allman. On Making TCP More Robust to Packet Reordering. *ACM Computer Communication Review*, 32(1):20–30, January 2002.

[Bra89] Robert Braden. Requirements for Internet Hosts – Communication Layers, October 1989. RFC 1122.

[FF96] Kevin Fall and Sally Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Computer Communications Review*, 26(3), July 1996.

[FGM+97] R. Fielding, Jim Gettys, Jeffrey C. Mogul, H. Frystyk, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, January 1997. RFC 2068.

[FK04] Sally Floyd and Eddie Kohler. Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control, November 2004. Internet-Draft draft-ietf-dccp-ccid2-08.txt (work in progress).

[Hay97] Chris Hayes. Analyzing the Performance of New TCP Extensions Over Satellite Links. Master's thesis, Ohio University, August 1997.

[Hoe96] Janey Hoe. Improving the Start-up Behavior of a Congestion Control Scheme for TCP. In *ACM SIGCOMM*, August 1996.

[HTH01] Amy Hughes, Joe Touch, and John Heidemann. Issues in TCP Slow-Start Restart After Idle, December 2001. Internet-Draft draft-hughes-restart-00.txt (work in progress).

[JD03] Hao Jiang and Constantinos Dovrolis. Source-Level IP Packet Bursts: Causes and Effects. In *ACM SIGCOMM/Usenix Internet Measurement Conference*, October 2003.

[KHF04] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Control Protocol (DCCP), November 2004. Internet-Draft draft-ietf-dccp-spec-09.txt (work in progress).

[MHK+03] Andrew Moore, James Hall, Christian Kreibich, Euan Harris, and Ian Pratt. Architecture of a Network Monitor. In *Passive & Active Measurement Workshop 2003 (PAM2003)*, April 2003.

[MM96] Matt Mathis and Jamshid Mahdavi. Forward Acknowledgment: Refining TCP Congestion Control. In *ACM SIGCOMM*, August 1996.

[Mog92] Jeffrey C. Mogul. Observing TCP Dynamics in Real Networks. In *ACM SIGCOMM*, pages 305–317, 1992.

[Pax97] Vern Paxson. Automated Packet Trace Analysis of TCP Implementations. In *ACM SIGCOMM*, September 1997.

[SXM+00] Randall Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Juergen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang, and Vern Paxson. Stream Control Transmission Protocol, October 2000. RFC 2960.

[ZKFP03] Ming Zhang, Brad Karp, Sally Floyd, and Larry Peterson. RR-TCP: A Reordering-Robust TCP with DSACK. In *Proceedings of the Eleventh IEEE International Conference on Networking Protocols (ICNP)*, November 2003.