Packet Loss Burstiness: Measurements and Implications for Distributed Applications

David X. Wei[1], Pei Cao[2], Steven H. Low[1]

[1]Division of Engineering and Applied Science
California Institute of Technology
Pasadena, CA 91125 USA
{weixl,slow}@cs.caltech.edu

[2]Department of Computer Science
Stanford University
Stanford, CA 94305 USA
cao@cs.stanford.edu

Abstract

Many modern massively distributed systems deploy thousands of nodes to cooperate on a computation task. Network congestions occur in these systems. Most applications rely on congestion control protocols such as TCP to protect the systems from congestion collapse. Most TCP congestion control algorithms use packet loss as signal to detect congestion.

In this paper, we study the packet loss process in sub-round-trip-time (sub-RTT) timescale and its impact on the loss-based congestion control algorithms. Our study suggests that the packet loss in sub-RTT timescale is very bursty. This burstiness leads to two effects. First, the sub-RTT burstiness in packet loss process leads to complicated interactions between different loss-based algorithms. Second, the sub-RTT burstiness in packet loss process makes the latency of data transfers under TCP hard to predict.

Our results suggest that the design of a distributed system has to seriously consider the nature of packet loss process and carefully select the congestion control algorithms best suited for the distributed computation environments.

1 Introduction

Many modern massively distributed systems deploy thousands of nodes to cooperate on a computation task. Huge amount of data is transferred by the underlying network. Often, the network becomes a bottleneck for these data transfers and congestions occur.

Network congestion requires distributed system designers to deploy transfer control protocols that can prevent congestion collapse and achieve fair share of the network resources. The most commonly used control protocol for reliable data transmission is Transmission Control Protocol (TCP) [18], with a variety of congestion control algorithms (Reno [5], NewReno [11], etc.), and implementations (e.g. TCP Pacing [16, 14]). The most commonly used control protocol for unreliable data transmission is TCP Friendly Rate Control (TFRC) [10]. These algorithms all use packet loss as the congestion signal. In designs of these protocols, it is assumed that the packet loss process provides signals that can be detected by every flow sharing the network and hence the protocols can achieve fair share of the network resource and, at the same time, avoid congestion collapse.

However, our preliminary study shows that the packet loss process is very bursty in sub-RTT timescale. Such sub-RTT burstiness leads to different efficiencies of loss detection in different protocols. More specifically, the window-based protocols, which also have sub-RTT burstiness in their data transmission processes, tend to underestimate the packet loss rates. This leads to unfairly large network resource utilization for the window-based protocols when they share the network with rate-based protocols such as TFRC and TCP-Pacing. Also, the sub-RTT burstiness in packet loss process and in window-based protocols' data transmission processes lead to a poor predictability of TCP's performance. These results are based on detailed measurements of packet loss processes in simulation, emulation and the Internet.

Based on these results, we suggest that the design of a distributed system needs to seriously consider the effects of sub-RTT burstiness in packet loss process and select the appropriate congestion control algorithms and implementations for the specific application environment.

1-4244-0910-1/07/$20.00

2 Related work

There are two areas of related work to this paper: the interaction among different loss-based protocols and the measurement of packet loss processes.

In the area of interaction among different loss-based protocols, there are several studies on the friendliness between a pair of specific loss-based congestion control algorithms. For example, Aggarwal, Savage and Anderson observe that TCP Pacing usually loses to TCP NewReno in competition [4]. Rhee and Xu point out that TFRC can have lower throughput than TCP NewReno when they share the same bottlenecks [19]. These studies focus on a specific pair of algorithms. Our study addresses the underlying source for these complicated interactions between different loss-based algorithms. Tang et al. discuss the interaction among different protocols with different congestion signals in the congestion window control level [21]. Our study focuses on the relation between sub-RTT level burstiness in packet loss process and the interactions of different loss-based congestion control algorithms.

In the area of packet loss measurement, Paxson has an extensive study on the Internet packet loss patterns [17]. His study uses TCP traces to reproduce loss events and study the timing and distribution of packet loss in different Internet paths. The study points out that packet loss is bursty in large time scales. However, since TCP traffic itself is very bursty in sub-RTT timescale, the measurement results from TCP traces are not able to differentiate the burstiness of TCP packets from the burstiness of packet loss in sub-RTT timescale. In our approach, we use Constant Bit Rate (CBR) traffic to measure the Internet loss patterns. With our measurement, it is convincing that packet loss process is bursty in sub-RTT level as our methodology excludes the TCP burstiness in our measurement setup. Borella et al. use User Datagram Protocol (UDP) packets to measure packet loss along one path [6]. The study focuses in the loss pattern and its impact to VoIP traffic. In our study, we use PlanetLab nodes to measure more than 500 paths in the Internet.

3 Observations: Sub-RTT Level Burstiness in Packet Loss Processes

We study sub-RTT level burstiness in packet loss processes in three different environments: simulation network (via NS-2 [3]), emulation network (via Dummynet [20]) and the Internet (via PlanetLab [2]).

From these three measurement sources, we find significant burstiness in sub-RTT time scales.

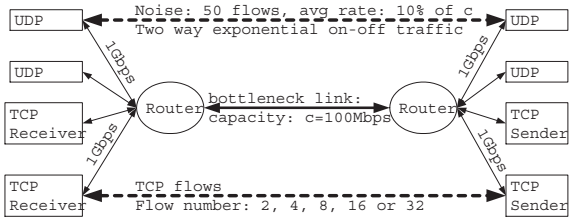

Figure 1. Setup of NS-2 simulations and Dummynet Emulations. c: capacity of the bottleneck router

3.1 Measurements

We measure the timing of each packet loss in three different environments: simulation network, emulation network and the Internet.

For each loss trace, we calculate the time interval between each two consecutive lost packets and analyze the packet loss process by plotting the probability density function (PDF) of the loss intervals. The resolution of the PDF (bin-size) is 0.02 RTT. We compare the PDF of the packet loss processes to the corresponding Poisson process with the same average arrival rate. We observe that the packet loss process is much more bursty than the Poisson process.

We conduct measurements in three environments: NS-2 simulation, Dummynet emulation and PlanetLab. NS-2 simulation is used to simulate a single ideal bottleneck shared by extremely heterogeneous sources. Dummynet is used to emulate a single non-ideal bottleneck (with noise in packet processing time) shared by heterogeneous sources. PlanetLab is used to measure the realist situations in the Internet.

For NS-2 simulation, we use a dumbbell topology with a set of senders and a set of receivers sharing a bottleneck, as shown in Figure 1. The receivers and senders are connected to the bottleneck with access links. The latencies of the access links are randomly distributed from 2ms to 200ms. We simulate with different buffer sizes, from 1/8 of the bandwidth-delay-product (BDP) to 2 times of the BDP. We record traces from the simulated routers for each event in which a packet is dropped.

Our Dummynet emulation follows the same topology setup as our NS-2 simulations. However, the traffic pattern consists of only 4 different latencies: 2ms, 10ms, 50ms and 200ms.[1] To collect the timing of each packet loss, we instrument the Dummynet router to record the time when each packet is dropped. The clock resolution in the Dummynet machine (FreeBSD operating system) is 1ms. Hence all Dummynet records have a resolution of 1ms.

[1] This limitation is due to the ability of the emulation testbed.

We also run measurement over the Internet with the facilities of PlanetLab. We select 26 sites from PlanetLab. The experiment sites are listed in Table 1. The sites are geographically located in different states in the United States and also in different continents. Among them, 6 are in California, 11 are in other parts of United States, 3 are in Canada and the rest are in Asia, Europe and Southern America. The complete graph formed by these 26 sites has 650 directional edges. Each edge corresponds to a path. The RTTs of these paths have a range from 2ms to more than 200ms between each other.[2]

From October 2006 to December 2006, we periodically initiate constant bit rate (CBR) flows between two randomly picked sites. For each experiment, two runs of measurements are conducted: One with a packet size of 48 bytes and the other with a packet size of 400 bytes. We compare these two results and validate the measurement only if the two traces exhibit similar loss patterns. Hence, the effect of traffic load from our own measurement CBR flows is negligible in our measurement results.[3] Each run of measurement lasts for 5 minutes. In analysis, we normalize the loss interval by the RTT of the path.

3.2 Observations

The measurements from NS-2, Dummynet and the Internet all suggest that the sub-RTT packet loss process is very bursty.

3.2.1 Results in NS-2 Simulation

Figure 2 shows the PDF of the loss interval in NS-2 simulations. The RTTs of the flows in simulation are random between 2ms to 200ms. We observe that more than 95% of the packet losses cluster within short time periods smaller than 0.01 RTT. In the figure, we also present the PDF of a Poisson process which has the same average arrival rate as the measured packet loss process. We zoom in to a small time scale of 0 to 2 RTT and use log-scale in the Y-axle so that the Poisson process has a straight line in its PDF. Comparing to the Poisson process, the packet loss process is much more bursty as almost all the packet losses in the packet loss process happen in the smallest interval.

3.2.2 Results in Emulation Network

Figure 3 is the PDF of the loss interval in Dummynet emulations. The RTTs of the flows are fixed to 4 classes: 2ms, 10ms, 50ms, and 200ms. We observe that about 80% of the

[2]The highest measured RTT is more than 300ms, depending on the time of the day.

[3]To use a packet size of 400 bytes, we are sure that our measurement packets are not fragmented by old routers which only pass 500-byte packets.

Node	Location
planetlab2.cs.ucla.edu	Los Angeles, CA
planetlab2.postel.org	Marina Del Rey, CA
planet2.cs.ucsb.edu	Santa Barbara, CA
planetlab11.millennium.berkeley.edu	Berkeley, CA
planetlab1.nycm.internet2.planet-lab.org	Marina del Rey,CA
planetlab2.kscy.internet2.planet-lab.org	Marina del Rey,CA
planetlab3.cs.uoregon.edu	Eugene, OR
planetlab1.cs.ubc.ca	Vancouver, Canada
kupl1.ittc.ku.edu	Lawrence, KS
planetlab2.cs.uiuc.edu	Urbana, IL
planetlab2.tamu.edu	College Station, TX
planet.cc.gt.atl.ga.us	Atlanta, GA
planetlab2.uc.edu	Cincinnati, Ohio
planetlab-2.eecs.cwru.edu	Cleveland, OH
planetlab1.cs.duke.edu	Durham, NC
planetlab-10.cs.princeton.edu	Princeton, NJ
planetlab1.cs.cornell.edu	Ithaca, NY
planetlab2.isi.jhu.edu	Baltimore, MD
crt3.planetlab.umontreal.ca	Montereal, Canada
planet2.toronto.canet4.nodes.planet-lab.org	Toronto, Canada
planet1.cs.huji.ac.il	Jerusalem, Israel
thu1.6planetlab.edu.cn	Beijing, China
lzu1.6planetlab.edu.cn	Lanzhou, China
planetlab2.iis.sinica.edu.tw	Taipei, China
planetlab1.cesnet.cz	Czech
planetlab1.larc.usp.br	Brazil

Table 1. PlanetLab sites in measurement

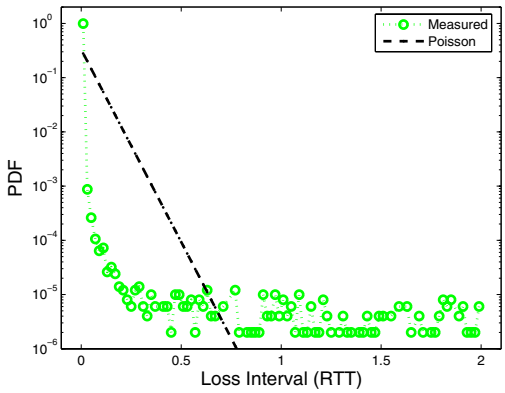

Figure 2. PDF of inter-loss time (NS-2 measurements).
Note that all the PDF figures in this paper have Y-axis in log-scale.

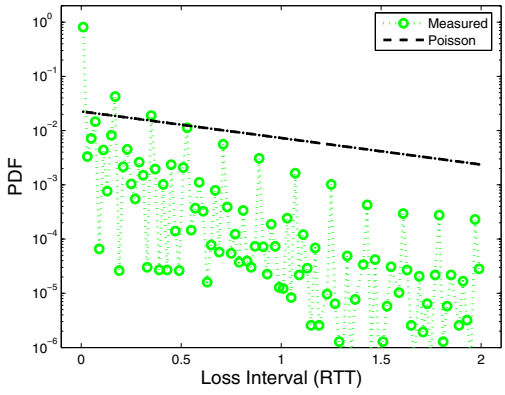

Figure 3. PDF of inter-loss time (Dummynet measurements).

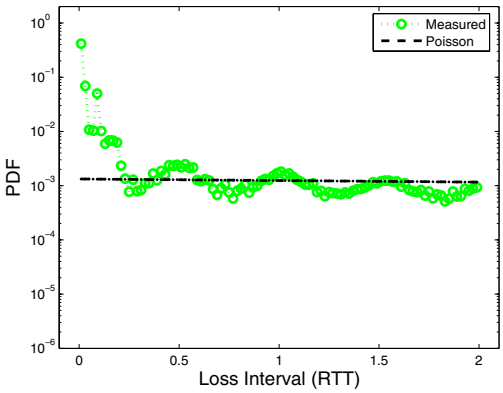

Figure 4. PDF of inter-loss time (PlanetLab measurements).

packet losses cluster within short time periods smaller than 0.01 RTT. As shown in the figure, the packet loss process is much more bursty than the corresponding Poisson process.

3.2.3 Results in the Internet

Figure 4 presents the PDF of loss intervals in Internet measurement via PlanetLab. The Internet measurement shows less burstiness in packet loss processes than we observe in simulation and emulation. This is due to the heterogeneity of the Internet, in terms of application types, traffic patterns and queuing delay. In such extremely heterogeneous environment, we observe that 40% of the packet losses cluster within short time periods of 0.01 RTT and 60% of the packet losses cluster within time periods of 1 RTT. This evidence is still very strong for sub-RTT burstiness in packet loss processes.

Comparing to the Poisson process with the same arrival rate, we observe that the loss process is much more bursty than the Poisson process in sub-RTT timescale (within 0 to 0.25 RTT).

3.3 Possible Sources of sub-RTT Burstiness

As shown by the results of NS-2 simulations, Dummynet emulations and the Internet, packet loss is highly bursty in sub-RTT timescale. There are several possible sources that lead to such burstiness.

DropTail router is considered the major source of packet loss burstiness. DropTail router serves as a FIFO queue, accepting incoming packets until the buffer is full. Working with DropTail router, loss-based congestion control algorithms keep increasing the data rate when the router's buffer is not full. When the router's buffer is full and packets are

dropped, the aggregate data rate is higher than the router's capacity and packet drops persist until the loss-based congestion control algorithms detect the loss of packets and reduce the data rate, usually half an RTT later. In between the first packet loss and the reduction of data rate, there is a peak of packet losses in the DropTail router. There are some proposals to introduce randomness in the router. For example, Random Early Dropping (RED) [12] is proposed to drop the packets earlier before the buffer is overflowed. However, these proposals suffer from difficult parameter settings problems.

Slow start of short flows is another source of packet loss burstiness, which is even harder to be eliminated. A TCP flow starts with a very small rate (sending two packets every round trip), and doubles its data rate if no loss is observed. This process can quickly fill up the bottleneck buffer in a few round trips and produce a large number of continuous packet losses in the router. Some new congestion control algorithms such as Rate Control Protocol (RCP) [8] and QuickStart [9] have been proposed to avoid such aggressive detection. These algorithms require changes in both the IP routers and data senders, which are expensive for the existing infrastructure.

Hence, the sources of sub-RTT burstiness in packet loss processes will exist in the foreseeable future and it is important to understand the impact of burstiness in packet loss processes to the data transfer in distributed systems.

4 Impact of sub-RTT Packet Loss Burstiness on Data Transfer

Burstiness in packet loss affects the fairness and predictability of TCP congestion control algorithms. When rate-based flows compete with applications using window-based flows for bandwidth, packet loss burstiness results in rate-based flows receiving less than their fair-share. Packet loss burstiness also leads to concurrent flows obtaining different transfer rate, making it very difficult to predict latency of data transfers in distributed applications.

4.1 Interaction of loss-based congestion control protocols

Different loss-based congestion control protocols can be categorized into two classes, rate-based implementations and window-based implementations.

A rate-based implementation controls the data transmission rate $x(t)$ (bytes/second) on each data source, based on the loss measurement from the network. The data packets are sent into the network with a constant interval of $\frac{M}{x(t)}$ seconds, where M is the packet sizes in bytes. Examples of such controls are TFRC and TCP Pacing.[4]

A window-based implementation does not directly control the transmission rate $x(t)$. Instead, $x(t)$ is implicitly controlled by a window-based implementation with two important variables:

- $pif(t) = \int_{t-RTT}^{t} x(u)du$: the *number of packets in flight*. It is the number of packets that are transmitted in the previous RTT;
- $w(t)$: the *congestion window*, the upper bound of $pif(t)$.

The window-based implementation only controls $w\,(t)$ on RTT time scale. $w\,(t)$ determines the average rate of a TCP flow in each RTT. Whenever $pif\,(t)$ is smaller than $w\,(t)$ (for example, in the occasion of acknowledgment arrival, or congestion window increment), $w\,(t) - pif\,(t)$ packets are sent in a burst to fill the gap.

Window-based implementations are used in most implementations of TCP. One important issue introduced by the window-based implementation is the sub-RTT burstiness in data packet arrival processes. Since the data packets are sent at almost the same time whenever there is a gap between $pif(t)$ and $w(t)$, the data packets arrive at the bottleneck routers back-to-back. Once sub-RTT level burstiness is formed in the data packet arrival processes, the burstiness is maintained for the life of the connection and its effect cannot be eliminated by a large buffer size or high multiplexing level. Jiang and Dovrolis show that sub-RTT level burstiness persists in scenarios with a single TCP flow as well as in daily Internet traffic (from router trace) where the number of flows is very large [15].

Hence, there is a significant difference in sub-RTT packet arrival patterns between rate-based implementations and window-based implementations. The packets controlled by a rate-based implementation arrive at the bottleneck evenly spaced. The packets controlled by a window-based implementation exhibit an on-off pattern in sub-RTT timescale.

When interact with the packet loss dynamic, complicated interaction happens between these two different classes of algorithms.

With rate-based implementations, since data packets are evenly distributed in the bottleneck, the majority of the flows will lose packets when the bottleneck drops packets in bursty pattern. Figure 5 illustrates this situation.

With window-based implementations, since data packets are in on-off pattern, some flows will lose multiple packets and some flows will not lose any packets when the bottle-

[4] In sub-RTT timescale, TCP Pacing is a rate-based implementation since the transmission rate $x(t)$ is directly controlled by the congestion control algorithm. However, in a larger timescale, TCP Pacing also has the concept of window to achieve reliable communication. This paper focuses in sub-RTT timescale dynamics so TCP Pacing is categorized as a rate-based implementation.

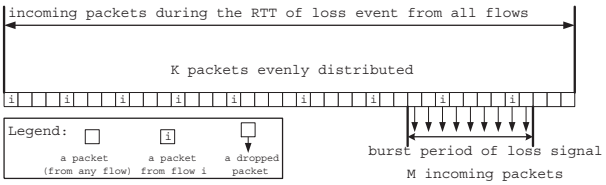

Figure 5. Packet loss with rate-based implementations: A flow (Flow i) has high probability to detect the loss event.

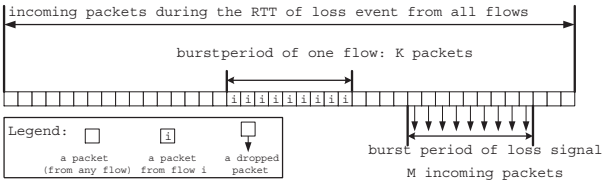

Figure 6. Packet loss with window-based implementations: A flow (Flow i) has low probability to detect the loss event.

neck drops packets in bursty pattern. This situation is illustrated in Figure 6.

If a window-based implementation and a rate-based implementation share the same bottleneck, the flows controlled by the window-based implementation will see less congestion events and get higher throughput than the flows controlled by the rate-based implementation.

In the ideal cases, assume the packets controlled by the rate-based implementation are perfectly evenly distributed in the bottleneck and assume the packets controlled by the window-based implementation are clustered in a continuous trunk of packets. The expected number of rate-based flows that see a packet loss event with M dropped packets is

$$L_{rate-based} = \min\{M, N\} \quad (1)$$

where N is the number of flows. And the expected number of window-based flows that see a packet loss in the same event is:

$$L_{win-based} = \max\left\{\frac{M}{K}, 1\right\} \quad (2)$$

where K is the number of packets sent by a flow in that RTT. Hence $L_{rate-based} >> L_{win-based}$.

In reality, if the rate-based flows and window-based flows are sharing the same bottleneck, the window-based flows will be slightly spread out by the rate-based flows. In this case, the difference in throughput is not as large as the one shown by the equations. But the effect is still significant. Figure 7 shows the aggregate throughput of 16

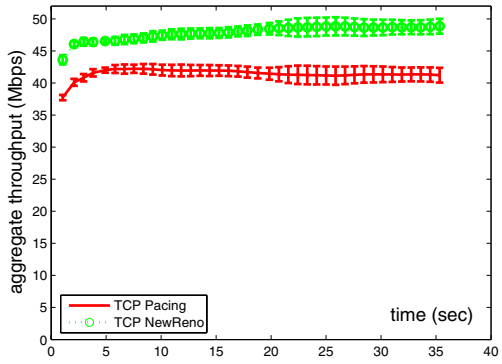

Figure 7. Aggregate throughput of TCP Pacing (16 flows) and TCP NewReno (16 flows) sharing a bottleneck of 100Mbps and a path of 50ms RTT: TCP Pacing flows lose to TCP NewReno flows in aggregate throughput.

TCP Pacing (rate-based) and the aggregate throughput of 16 TCP NewReno (window-based) flows, which share the same path with 50ms delay and 100Mbps capacity. TCP Pacing uses exactly the same loss detection and congestion reaction algorithms as TCP NewReno. However, since TCP Pacing is a rate-based control protocol and it is easier to see packet losses, it has a 17% lower throughput than TCP NewReno in the competitions. We observe the same behavior with different parameters (different RTTs and different number of flows).

Hence, the sub-RTT burstiness in packet loss process can lead to a lower throughput of rate-based protocols when they compete with window-based protocols.

4.2 Predictability of Data Transfer Latency

Another impact of packet loss burstiness is that the performance of window-based implementations becomes less predictable.

As shown in Figure 6 and equation (2), only a few TCP flows experience packet loss during a congestion event. Other flows continue to increase their rates until another congestion event happens. This phenomenon introduces unfairness to different TCP flows and makes it hard to predict the performance when applications use many TCP flows between many hosts.

This is especially significant if the TCP flows are short and start-up phase (slow-start) is the main part of the lives of the flows. During start-up phase, each TCP flow doubles its data rate every round trip until it experiences a packet loss. After that, the flow enters congestion avoidance state

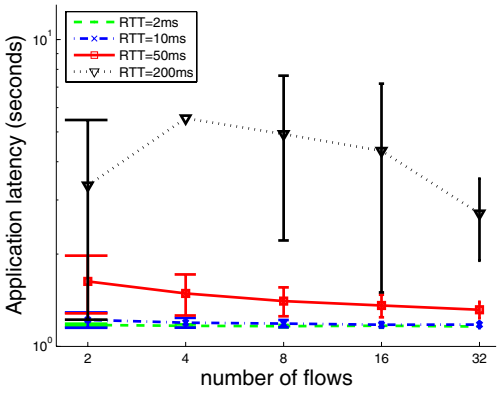

Figure 8. Data transfer latency (normalized by theoretic lower bound) with parallel flows sending a total of 64MB data.
Note that the standard variation at the point of RTT=200ms and flow number=4 is too large and cannot be displayed in the scale of the figure.

in which the data rate is increased very slowly. If a few flows enters congestion avoidance state earlier than other flows, these flows will be very slow (half of the rates of other flows, or even lower) and become the bottleneck.

Figure 8 presents the latency of parallel flows (as in GridFTP [1] or GFS [13]) that transfer a total of 64MB of data. The 64MB data is divided into chunks with the same size for each flow.

In the 100Mbps network, the theoretic lower bound of completion time of a 64MB transfer is 5.39 seconds. The bound is tight if the network is fully utilized in all times.

With TCP NewReno, the transfer latency with 200ms RTT varies from 11 seconds to 50 seconds, depending on how many flows enter the congestion avoidance phase prematurely.

5 Implications for Distributed Applications

Section 4 demonstrates that the sub-RTT burstiness in packet loss process has significant impact on the data transfer protocols in the distributed system. Thus, one has to carefully consider the selection of congestion control algorithms used in distributed systems.

One important lesson is that rate-based implementations and window-based implementations should not mix. If they do, the rate-based flows tend to suffer in throughput. For example, if a distributed application has to use both UDP (controlled by the rate-based TFRC), and TCP (controlled by window-based implementation) in the data communication, TFRC will have unexpectedly low throughput. In this case, TCP Pacing should be considered to replace the window-based TCP implementation.

Another important lesson is that if the computing environment is tightly controlled and the same TCP implementation can be enforced on every node, then a rate-based implementation has an advantage in that it makes TCP more fair, and leads to better predictability of throughput for concurrent flows.

There are proposals to reduce the burstiness of packet loss process in the DropTail bottleneck by introducing randomness in the packet loss process. RED [12] is the most famous one. Thus, perhaps RED should be deployed if one wants to eliminate loss burstiness. However, the parameter tunings of RED are difficult, and we suggest this approach be used only when the scenarios in the distributed system are simple and the RED's effect can be well understood in the scenarios.

Another suggestion is to use other congestion signals, instead of loss, to by-pass the burstiness problem in packet loss process. In [22], we suggest a simple Explicit Congestion Notification (ECN) algorithm which can provide persistent congestion signal for one RTT, covering most of the participating flows. This algorithm significantly improves the TCP performance and also solves the competition problem of rate-based implementation and window-based implementations. In [23], a delay-based algorithm is proposed and achieved better stability and fairness.

6 Conclusion and Future work

This paper is a report of our ongoing work on understanding the sub-RTT pattern of packet loss process and its impact to the performance of loss-based congestion control protocols. Our extensive measurements in simulation network, emulation network and the Internet all show significant sub-RTT burstiness in the packet loss process.

Our simulation results show that such sub-RTT level burstiness have significant impacts on the performance of TCP and the interactions between window-based implementations and rate-based implementations. Hence, we suggest that the design of a distributed system should seriously consider the sub-RTT level burstiness in the selection of congestion control algorithms.

Our future work lies in the following three directions:

First, we plan to have more rigorous analysis on the burstiness of packet loss process. In this paper, we present preliminary analysis with PDF. We plan to analyze the loss trace with more rigorous model and provide more concrete evidence. We also plan to compare our results with the results obtained from TCP trace analysis to understand the extent of difference due to measurement methodology.

Second, we plan to enrich our solutions to sub-RTT level burstiness. Two promising sub-directions are to improve and finalize the ECN algorithm proposed in [22] and to study the possibility of using rate-based control and window-based control for differential services.

Finally, we plan to extend our simulation framework to include more detailed model for distributed applications. Currently, we only use parallel flows as the application. We plan to simulate more complicate scenarios such as a complete graph topology in MapReduce [7].

References

[1] GridFTP. URL: http://www.globus.org/toolkit/docs/4.0/data/gridftp/.

[2] PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. URL:http://www.planet-lab.org.

[3] The Network Simulator - NS-2. URL: http://www.isi.edu/nsnam/ns/index.html.

[4] A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of TCP pacing. In *Proceedings on INFOCOM 2000*, pages 1157–1165, 2000.

[5] M. Allman, V. Paxson, and W. Stevens. RFC 2581: TCP Congestion Control, April 1999.

[6] M. Borella, D. Swider, S. Uludag, and G. Brewster. Internet Packet Loss: Measurement and Implications for End-to-End QoS, 1998.

[7] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, December 2004.

[8] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev.*, 36(1):59–62, 2006.

[9] S. Floyd, M. Allman, A. Jain, and P. Sarolahti. Internet Draft: Quick-Start for TCP and IP, Oct 2006.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM*, pages 43–56, Stockholm, Sweden, 2000.

[11] S. Floyd and T. Henderson. RFC 2582: The New Reno Modification to TCP's Fast Recovery Algorithm, April 1999.

[12] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[13] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, New York, NY, USA, 2003. ACM Press.

[14] D. Hong. FTCP Fluid Congestion Control, 2000.

[15] H. Jiang and C. Dovrolis. Why is the Internet traffic bursty in short time scales? In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 241–252, New York, NY, USA, 2005. ACM Press.

[16] J. Kulik, R. Coutler, D. Rockwell, and C. Partridge. A simulation study of paced TCP. Technical Report BBN Technical Memorandum No. 1218, BBN Technologies, 1999.

[17] V. Paxson. End-to-end Internet packet dynamics. In *Proceedings of the ACM SIGCOMM '97 conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, volume 27,4 of *Computer Communication Review*, pages 139–154, Cannes, France, September 1997. ACM Press.

[18] J. Postel. RFC 793 - Transmission Control Protocol, Sep 1981.

[19] I. Rhee and L. Xu. Limitations of Equation-based Congestion Control. In *Proceedings of ACM Sigcomm 2005*, 2005.

[20] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, 1997.

[21] A. Tang, D. Wei, S. Low, and M. Chiang. Heterogeneous Congestion Control: Efficiency, Fairness and Design. In *Proceedings of ICNP 2006*, 2006.

[22] D. X. Wei, P. Cao, and S. H. Low. Fairness Convergence of Loss-based TCP. URL: http://www.cs.caltech.edu/~weixl/pacing/sync.pdf.

[23] D. X. Wei, C. Jin, S. H. Low, and S. Hedge. FAST TCP: Motivation, Architecture, Algorithms, Performance. *IEEE/ACM Transactions on Networking*, 14(6):1246–1259, 2006.