The Effect of Round Trip Time on Competing TCP Flows

Chris Holman*, Jason But, Phillip Branch
Centre for Advanced Internet Architectures, Technical Report 120405B
Swinburne University of Technology
Melbourne, Australia
6963420@student.swin.edu.au, jbut@swin.edu.au, pbranch@swin.edu.au

***Abstract*—In this paper we describe experiments to investigate the interactions of competing TCP flows typical of a home network. Using a standard FreeBSD 8.2-RELEASE installation, we explore the relationship between queue sizes, congestion windows, latency, and bandwidth and how they relate to each other in such an environment. We find that flows with a higher RTT are at a disadvantage, especially over long running competition.**

I. Introduction

Home Internet connections are often shared concurrently between multiple users with little or no Quality of Service (QoS) utilised. Therefore, Transmission Control Protocol (TCP) [1] streams will overlap and compete regularly. TCP, by design, attempts to use as much of the available bandwidth as possible. When two flows start competing for 100% of the available bandwidth, interactions occur to determine how much bandwidth each stream receives.

We investigate these interactions between two concurrent TCP connections with different endpoints. In particular we are interested in comparing interactions between low latency (near) flows and high latency (far) flows.

We find that high latency connections suffer the most when sharing a bottleneck link and that the bandwidht share of streams with equal Round Trip Time (RTT) will alternate which has more bandwidth but essentially will be be equal over a longer period of time.

The remainder of this paper is structured as follows. Section II details the experimental method used for this report. Section III looks at the data gathered and analyses the significance of it.

*The author was on an internship at CAIA while writing this report

II. Methods

Three PCs were configured with *FreeBSD 8.2 RELEASE* running the default NewReno congestion control algorithm (see Figure 1). Two of these were configured as endpoints while the third was configured as an Ethernet bridge. We implemented traffic shaping and random packet loss using *ipfw* and *dummynet* on the bridge device. Figure 2 describes how dummynet was configured to simulate varying data sources traversing the same bottleneck link.

The benchmark program *nttcp* [2] was used to generate traffic between endpoints. *nttcp* can create a TCP/UDP stream consisting of an arbitrary amount of test data, and then report basic statistics on the genreated flow.

tcpdump [3] was used to capture traffic from both the inbound and the outbound interface of the bridge, on either side of the delays and queues.

In this experiment we captured a series of data points for each flow:

- One-way latency: latency is calculated from the tcpdump files using the *SPP* [4] tool
- TCP congestion window size *(cwnd)*: *SIFTR* [5] was deployed to log cwnd data from within the kernel on server host.
- Instantaneous queue utilisation: modifications were made to dummynet to retrieve and log the number of packets in a queue as each packet enters that queue
- Estimate of bandwidth utilisation: all data transferred in the 300ms preceding a packet was summed to determine a moving average for bandwidth utilisation

We configured ”far” streams as those experiencing a 100ms RTT as a reasonable approximation for an international data source. An RTT of 100ms is typically

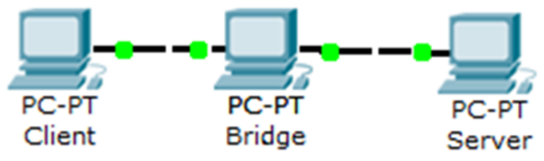

Fig. 1. The configuration of hardware

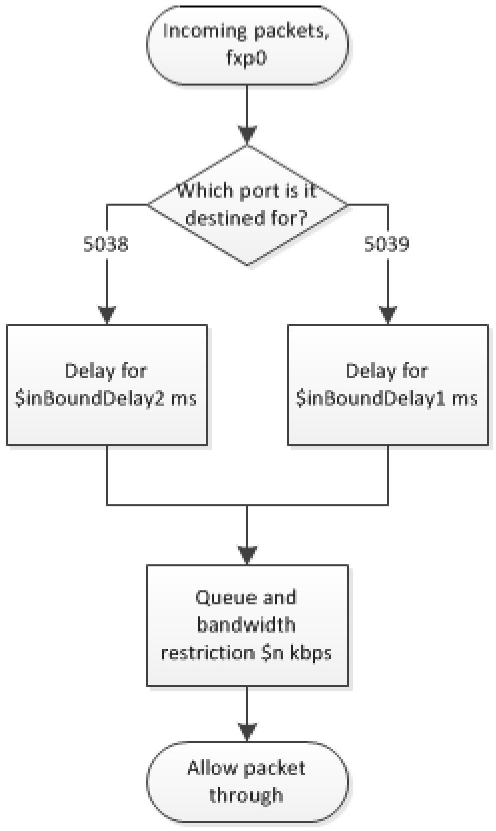
Incoming packets, fxp0
Which port is it destined for?
5038
5039
Delay for $inBoundDelay2 ms
Delay for $inBoundDelay1 ms
Queue and bandwidth restriction $n kbps
Allow packet through

Fig. 2. Delay and queue path. This process is repeated in the same order for packets returning through the bridge. The destination port number is used to determine which delay is applied

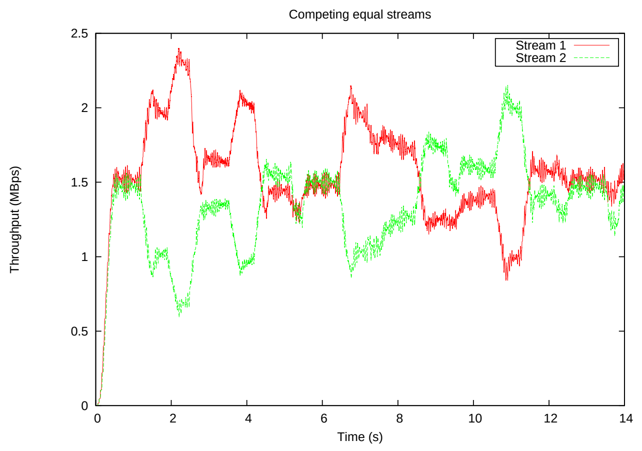

Fig. 3. Two identical TCP streams, started at the same time.

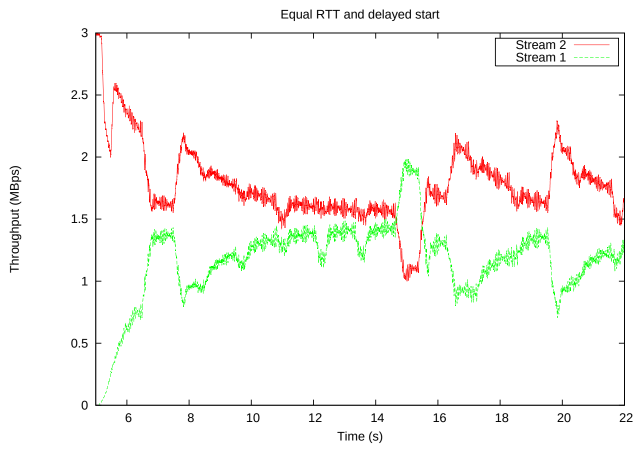

Fig. 4. Two identical TCP streams, with staggered starts

witnessed on communications between the East Coast of Australia and Japan. "Near" streams were configured with an RTT of 30ms as a typical value for local data sources.

We tested a variety of scenarios involving two TCP streams competing for bandwidth. We add delays, staggered starts and bursty traffic.

III. Analysis

A. Basic competition

When two TCP streams have no differentiating factors, the result is roughly 50/50 shared bandwidth. Figure 3 demonstrates that while instantaneous throughput fluctuates, it averages out over a longer period of time to roughly an equal split. When a packet loss event occurs, it typically only happens to one stream - that stream will reduce its *cwnd* and consequently reduce its transmission rate. The other stream sees that extra bandwidth is available and continues with normal growth until the first stream begins to recover, at which point they return to roughly equal throughputs again and the cycle continues.

Figure 4 shows throughput when one stream has started early enough to grow its window to fully utilise the available bandwidth prior to the second stream starting. While there is an initial transition period as this stream increases its throughput from 0, once the streams begin to share bandwidth equally, they continue interacting as in the previous scenario.

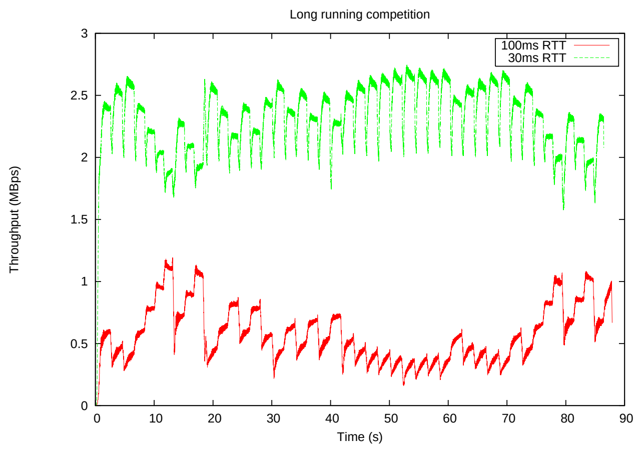

Fig. 5. Delay in competing streams, over a long time duration

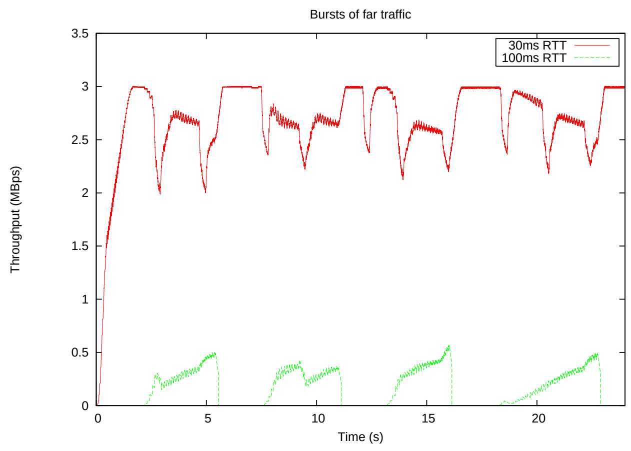

Fig. 7. A long running near stream coexists with bursts of far traffic

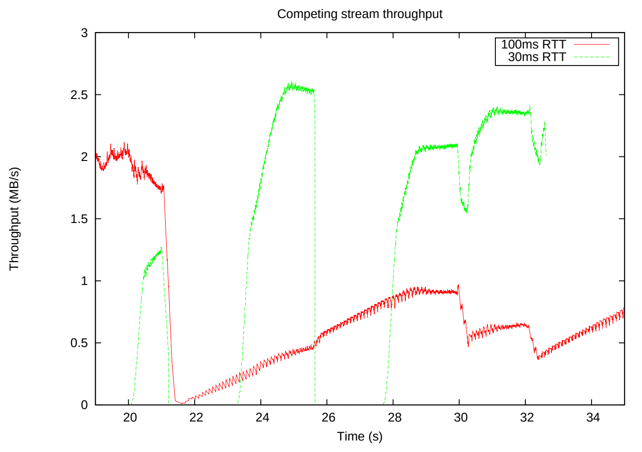

Fig. 6. A far TCP stream allowed to reach its full throughput before near streams are introduced

B. *The effect of latency on competition*

We then experimented with combinations of near and far away data sources, each transferring both large and small files. An example scenario would be akin to streaming a video from the USA while updating a Linux distro from a local ISP mirror - would the successive fast but small flows negatively impact on the long running video flow? Or likewise, would streaming an on demand TV show from the ISP service be affected at all by a large, far away stream?

When different latencies are configured for the two flows, a significant discrepancy in the throughput obtained by the competing streams is apparent; this can be seen in Figure 5. As the near stream has a lower RTT, it is able to complete more round trips in the same time period than the far stream. As such, its inflight window increases at a faster rate, subsequently allowing it to transmit at a higher bitrate. This results in the queue at the bottleneck filling and causing congestion events. The far stream experiences the same congestion events and reduces its own *cwnd*[1], effectively putting a bandwidth cap on it. Given its shorter RTT, the near stream is better able to recover, and therefore take more of the available bandwidth.

In Figure 6, a far TCP flow is running at its peak throughput. A near flow joins the bottleneck and fills the queue. This results in the far flow losing about 30 packets over the space of 30ms - one RTT of the near flow - and subsequently causes the TCP slow start algorithm to reset the *cwnd*. Slow start determines that this was two separate instances of packet loss and subsequently the Slow Start Threshold (SSTHRESH) is reduced twice. As the second reduction happens very soon after the first, the congestion window has not had the opportunity to recover, resulting in the SSTHRESH being set very low as well. Consequently, very little time is spent in the exponential growth phase, resulting in the majority of future window growth occurring under congestion avoidance.

C. *Bursty traffic*

Often, traffic flows are bursty in nature. Loading web pages, for instance, will result in a series of short TCP flows in a bursty sequence every time a new page is loaded.

In the scenario in Figure 7, a near stream is allowed to reach full bandwidth before short (1-3MB) bursts of far streams are started. As the round trip for the far streams

[1]The far stream is not always impacted by the congestion events, see Section III-E

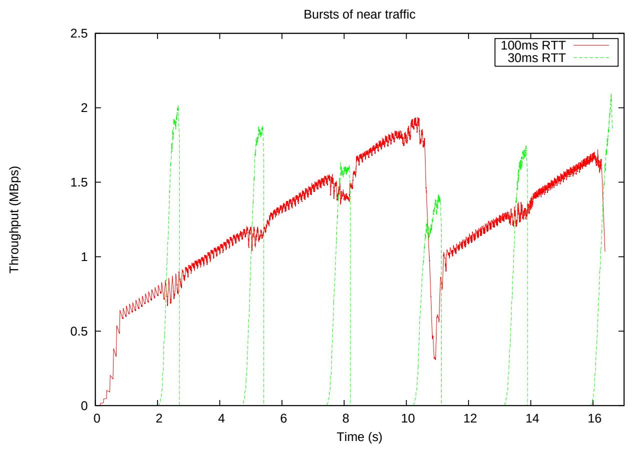

Fig. 8. A long running far stream is interrupted by bursts of near traffic

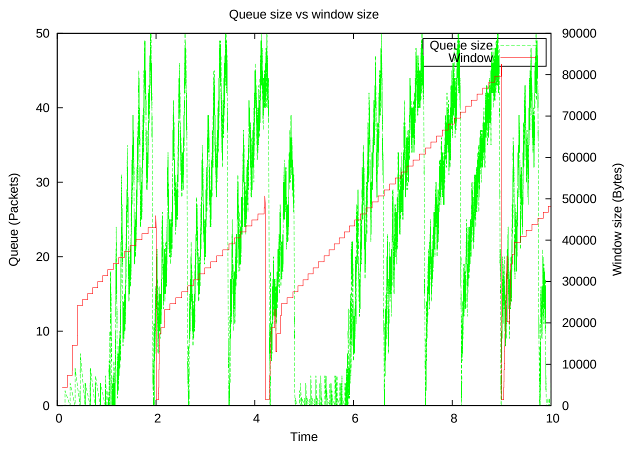

Fig. 9. *cwmd* size of a far TCP flow with a near TCP competitor (not shown) and the queue size of the bottleneck

is 100ms, the average inter-packet arrival time is initially quite high, and consequently takes some time to build up the throughput. The stream does not send enough packets per second to fill up any queues, so there is only a minor impact on the near stream.

Conversely, bursts of near streams are not as fair to a long running far stream. The RTT is short enough that even a small payload can occasionally cause the TCP slow start algorithm to engage and reset the cwnd of the far stream, causing significant disruption to the far stream, as shown in Figure 8.

D. Window size and queue utilisation

Figure 9 plots the queue utlisation, and the corresponding window size, of a large, far TCP flow. During the life of the far flow, two shortlived near TCP flows are initiated (at $time$=1s and again at $time$=6). The low RTT of the near flow causes the queue to fill more quickly. When the queue fills, one or more packets are dropped. This will be detected by the TCP sender which reduces the *cwnd* size and consequently transmits at a slower rate.

However, the far flow does not always suffer a *cwnd* reduction when the queue fills in this scenario; when the near flow suffers packet loss, it reacts quickly enough to allow the queue to drain, allowing the far flow to avoid any effect the congestion might have had on its transmission rate.

Figure 10 shows the throughput of the same streams over the same time period. The far stream's rate of growth decreases when the near stream starts, as the queue is being shared and packets take longer to make a round trip. The near stream is able to reach its peak

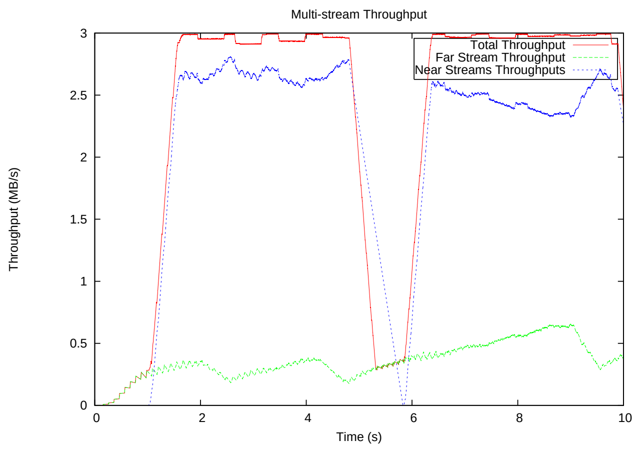

Fig. 10. The variation in throughput of two varied competing TCP streams, starting at the same time. Total throughput is also shown.

throughput within half a second, while the far stream is left running at a fraction over 1/10th of the available bandwidth.

E. Latency and queue sizes

While not specific to competing TCP flows, the latency induced by queue sizes has an impact on the competition between two flows. As the queue utilisation grows, the instantaneous RTT grows proportionally, as shown in Figure 11. In this example shown, a maximum of an extra 25ms is added to the RTT of a flow by the queue filling up. In the case of a growing far stream, this could be quite a hindrance in allowing *cwnd* to grow promptly.

IV. CONCLUSION

The aim of this report was to investigate the interactions of TCP streams with different RTTs over a

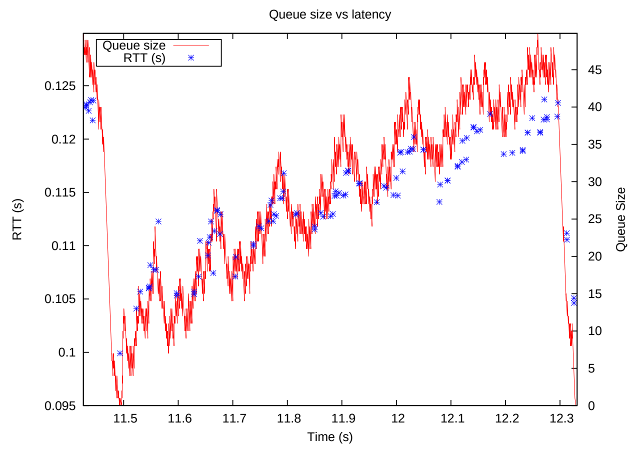

Fig. 11. Queue and latency relationship with competing TCP flows

bottleneck. A testbed was set up to limit bandwidth and add arbitrary delays across a link to collect data.

An unfair negative impact on a stream's throughput was seen when the competing stream had a smaller RTT. Under the latency conditions used for our "long running" experiments, the bias was found to be large, with the low latency stream obtaining about 80% of the available bandwidth. Additionally, short bursts of data from a lower RTT were shown to have a significant detrimental effect on existing, larger RTT, streams.

These results demonstrate a weakness in the NewReno congestion control algorithm (and perhaps loss-based TCP congestion control algorithms in general), in that low RTTs are favoured in bandwidth contention scenarios.

REFERENCES

[1] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Internet Engineering Task Force, Sep. 1981, updated by RFCs 1122, 3168, 6093, 6528. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[2] "NTTCP." [Online]. Available: http://www.freebsdsoftware.org/benchmarks/nttcp.html

[3] "tcpdump." [Online]. Available: http://www.tcpdump.org/

[4] CAIA, "SPP - Synthetic Packet Pairs." [Online]. Available: http://caia.swin.edu.au/tools/spp/

[5] CAIA, "SIFTR - Statistical Information For TCP Research." [Online]. Available: http://caia.swin.edu.au/urp/newtcp/tools.html