Data Center TCP - Reproducing Key Findings

CS 656 Research Paper

Colin Howes
University of Waterloo
200 University Ave. W
Waterloo, ON N2L 3G1
chowes@uwaterloo.ca

ABSTRACT

Data centers supporting modern large-scale applications must host a diverse range of heterogenous traffic with varying throughput and delay requirements. Transport layer protocols powering data center traffic must be able to tolerate intense bursts of traffic, provide low latencies for time-sensitive traffic, all while maintaining high throughput for large data flows. TCP makes up the majority of traffic in modern data centers, though a number of performance impairments suggest that traditional TCP congestion control mechanisms are ill-suited to the demands of a data center environment. Several modifications to the TCP protocol have been proposed to address shortcomings in conventional TCP congestion control algorithms, which relies on packet loss as a metric of network congestion. This paper presents an overview of traffic problems in modern data centers, as well as a number of proposed modifications to TCP's congestion control mechanisms aimed at improving the performance of TCP for typical data center communication patterns. Finally, key results related to the performance of Data Center TCP are replicated and discussed.

1 INTRODUCTION

Data centers are responsible for driving large-scale web applications such as web search, storage, advertising, and social network composition [1, 3]. These applications generate diverse communcation patterns with strict throughput and latency requirements. In particular, many distributed web applications rely on a workflow patterns in which requests are broken down and distributed to multiple workers, which perform tasks simultaneously and return responses to an aggregator [1, 3]. As a result, data centers need to support bursts of many-to-one traffic traversing shared bottlenecks without affecting the throughput of long-lived flows needed to update and maintain application data [1].

In order to accomodate the workload generated by these applications while maintaining cost efficiency, modern data centers typically feature high speed links with very low propogation delays, connecting nodes via low-cost switches with shallow buffers [1, 3, 5]. The majority of communication between nodes is over TCP, which was designed based on the characteristics of wide area networks, where roundtrip times (RTTs) are orders of magnitude greater than in data centers [3]. As a result, while traditional TCP congestion control mechanisms which are usually considered “good enough” in the context of the internet, TCP does not perform well when faced with traffic patterns typical of data center networks [3, 6].

In particular, the traffic patterns common to data center networks suffer a number of impairments due to problems with traditional TCP congestion control mechanisms. In the synchronized many-to-one scenarios common to distributed applications such as MapReduce, TCP connections experience throughput collapse, or incast congestion. In this scenario, goodput at the receiver drops dramatically as simultaneous TCP flows from synchronized senders floods the buffer of a bottleneck switch, causing packet loss and large queueing delays [3, 4, 6].

In addition, long lived background flows tend to keep switch buffers full, leading to increased queueing delays for latency sensitive traffic. Moreover, improving resource utilization through dynamic buffer allocation means that long lived flows can even cause buffer pressure that impacts flows traversing other ports on a switch [1]. More generally, TCP's loss based congestion control mechanism means that large switch buffers are kept full, which means that traffic is subject to large queueing delays even in the absence of packet loss, while packet loss at switches with small buffers is misinterpreted as network congestion leading to underutilization of network resources [2].

This paper discusses a number of approaches to TCP congestion control aimed at improving the performance of TCP in data centers. In particular, Data Center TCP (DCTCP) is discussed in depth, and a number of key findings are reproduced using the Mininet network emulator.

1.1 Data Center Traffic Patterns

The applications running in modern data centers rely heavily on the *partition/aggregate* design pattern, in which an application is broken in into hierarchical layers and time-sensitive requests at higher layers are divided and delegated to workers in the lower layers. Workers perform some component of a task and return a result to an aggregator, which is combined with results from other workers and passed back up through the hierarchy.

1.2 Impairments

1.2.1 Incast. TCP throughput collapse, or *Incast*,

1.2.2 Queue Build-Up.

1.2.3 Buffer Pressure.

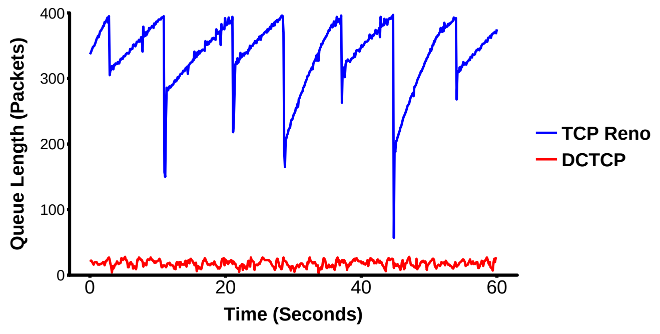

Figure 1: Comparison of queue length over time between DCTCP and TCP Reno with 2 flows

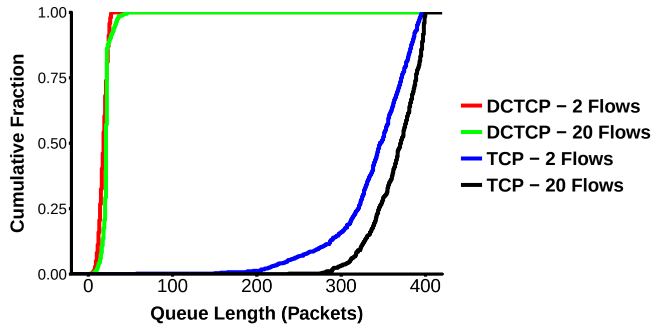

Figure 2: CDF of queue length for DCTCP and TCP Reno with 2 and 20 flows

1.2.4 *Buffer Bloat and Underflow.*

2 IMPROVING TCP FOR DATACENTERS

2.1 Incast TCP

2.2 Multipath TCP

2.3 TCP BBR

2.4 Data Center TCP

Data center TCP (DCTCP) attempts to address the problem of latency in partition/aggregate traffic by reducing queue length without affecting throughput for large TCP flows.

3 REPRODUCING KEY DCTCP RESULTS

3.0.1 Methods. Selected results from [1] were reproduced using the Mininet network emulator running on Ubuntu 12.04 with a patched version of the 3.2.18 Linux kernel patched to add in support for DCTCP. A custom utility was modified from the Mininet utilities repository to monitor queue length, and another utility was created to monitor bandwidth.

3.0.2 Results.

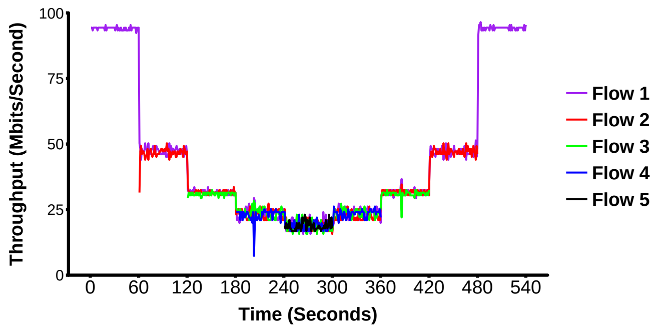

Figure 3: Convergence of 5 flows DCTCP

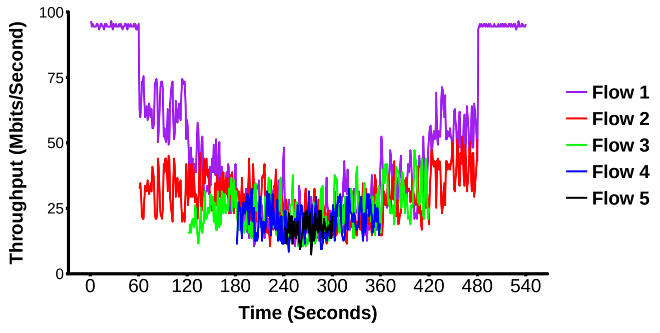

Figure 4: Convergence of 5 flows TCP Reno

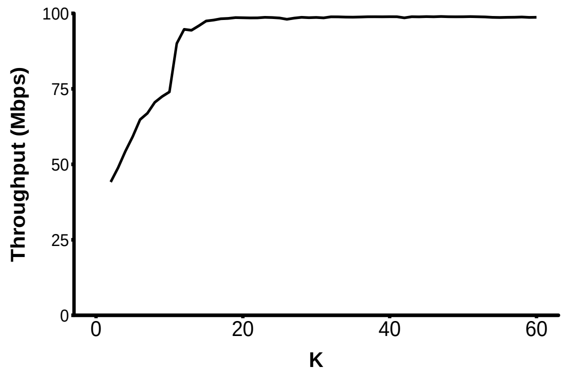

Figure 5: DCTCP throughput for varying values of K

3.0.3 Discussion.

3.0.4 Limitations.

4 CONCLUSIONS

REFERENCES

[1] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center tcp (dctcp). In *ACM SIGCOMM computer communication review*, Vol. 40. ACM, 63–74. http://dl.acm.org/citation.cfm?id=1851192

[2] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control. *Queue* 14, 5 (Oct. 2016), 50:20–50:53. https://doi.org/10.1145/3012426.3022184

[3] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, and Anthony D. Joseph. 2009. Understanding TCP Incast Throughput Collapse in Datacenter Networks. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking (WREN '09)*. ACM, New York, NY, USA, 73–82. https://doi.org/10.1145/1592681.1592693

[4] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: simplified data processing on large clusters. *Com-munications ofthe ACM* 51, 1 (2004), 107–l.

[5] James Hamilton. 2007. On Designing and Deploying Internet-Scale Services. In *Proceedings of the 21st Large Installation System Administration Conference*. Austin, TX, 231–242.

[6] Amar Phanishayee, Elie Krevat, Vijay Vasudevan, David G Andersen, Gregory R Ganger, Garth A Gibson, and Srinivasan Seshan. 2008. Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems.. In *FAST*, Vol. 8. 1–14.