DCTCP:
Congestion Control In a Datacenter Network Structure

Jakob Horner and Mohammad Hadi

(Dated: May 12, 2020)

Abstract: In data center network topologies, there are often strict expectations on throughput and round trip time even when dealing with large amounts of data. With normal network message passing approaches, often the large and varied flows of a data center can break the implemented congestion control systems. We attempt to implement a TCP over UDP protocol that offers congestion control explicitly tailored to data center topologies based on previous data center TCP research. This protocol is tested on a virtual data center network made with Mininet.
The git repo can be accessed here: `https://github.com/jakobh7/CSCI_5550_DCTCP`

I. INTRODUCTION

In the past decade, with the increase of social networks, cloud computing, and consolidation of other data into single data centers, it has become extremely important to provide consistent and improved performance. While there have been different strategies employed to address this performance, the implementation of a specific data center TCP protocol is a lucrative prospect that can offer cost savings for data center owners by offering greater performance without upgrading hardware.

The topology of data centers is distinct from common network topologies. Often data centers are created with a "fat-tree" network topology to separate and balance workloads. This "fat-tree" topology has a top layer switch connected to distinct lower-level switches which connect to server racks to process data. These network topologies often use a Partition/Aggregate workflow pattern to split work across switches in a layer and receive information in a real-time fashion. Because of the layered structure of the tree, and the workflow dependency on the workers below low latency is required for every level of the tree so that the overall response can be in real-time. This difference in topology and performance expectation is the driving motivation for implementing a specialized method of network communication.

In TCP network communication, when a packet is dropped - commonly due to overflowed queues in a switch - TCP's reliable data transfer requires the packet to be resent which is extremely costly in terms of latency. Usually TCP's congestion control algorithm adequately limits dropped packets by adjusting how many packets are sent at a time, reducing that number after a packet is dropped. This method is reactive, however, and is not ideal for a system that requires low latency. With knowledge of the common flow patterns of the network you're designing, it is possible to create a more proactive method of congestion control to stop packet loss before a packet gets dropped. This idea of proactive congestion control is central to the previous research on data center TCP.

Most TCP implementations are build into modern operating systems and network ports. This often obfuscates and protects properties and functions of the TCP implementation, making them harder to access and work with. Our implementation of this previous work on Data Center TCP is based on a TCP over UDP model. This not only allows us for easy access to TCP members such as the cwnd structure and incoming and outgoing queues, it also allows us to implement a version of the TCP header that is pared down to the information needed by our specific protocol. Some of these implementation details might add overhead compared to a lower level implementation of the protocols, but having a basic TCP over UDP written in the same format gives us a good benchmark to test against.

Our tests rely on a Mininet virtual network to run benchmark testing. This allows us to create our own fat-tree topology and pass bursts of high message flows through the switches. This way we can replicate the message flow patterns present in real world data centers and provide benchmarks for this implementation without worrying about harming real world data.

II. CONGESTION CONTROL

To understand how the data center TCP improves upon the performance of a typical TCP implementation, we need to understand the importance of congestion control in a data center network and the differences in congestion control strategies.

a. Effects of Congestion Control In a data center network, messages sent to server racks get sent through layers of switches with limited queues to hold messages to forward on. In all cases, switches have a limited memory such that when there are too many messages being sent across the network to a particular switch filling the queue, which, when full, drops any new messages sent to it.

The message flows in a data center analyzed by the previous DCTCP research were shown to be a mixture of consistent smaller data flows with inconsistent bursts of large sized data. These varied data flow allow for a considerable amount of dropped packets from the bursts of large data. When these large flows are streamed out to the switches in the data center, the queues quickly fill and lead to dropped packets.

The common strategy of congestion control is limiting the amount of packets sent across a network by implementing a "cwnd" value as a limit to how many packets can be sent without receiving acknowledgment. Once packets are received and subsequently removed from switch cues, the communication protocol sends an acknowledgment message to alert the sender that it is safe for more packets to be sent into the network. From this common framework, there are multiple strategies to address

b. TCP Congestion Control Strategies TCP's typical congestion control strategy has three main parts: Slow Start, Congestion Avoidance, and Fast Recovery. In Slow Start, cwnd is doubled every time an acknowledgment is received until a variable congestion threshold is reached. After this, cwnd is increased linearly with every ack. With its Congestion Avoidance, TCP waits until it receives an indication that one of its packets has been dropped: either the sender receives 3 duplicate acks or there is a complete timeout since no acks have been received in a reasonable amount of time $given the known RTT of packets for the connection.$ If the sender has received 3 duplicate acknowledgments, this indicates that just one package has been lost - a less severe issue - and the congestion control algorithm enters a fast recovery mode. However, if there is a timeout, the congestion control algorithm reverts back to the slow start and reduces the congestion threshold by half so that the slow start is even slower. Fast Recovery allows the cwnd to expand even when it waits on acknowledgment. Until a timeout occurs, the cwnd increases every time a duplicate ack is received.

c. Data Center TCP Congestion Control Strategies Where Data Center TCP's congestion control strategy is significantly different is its active management to determine if a switch is congested before a packet is dropped by marking the TCP header both on the level of the switch and the receiver.

III. DATA CENTER TCP OVER UDP IMPLEMENTATION

IV. BENCHMARK TESTING

A. Mininet Setup

B. Benchmark Results

V. CONCLUSION