Lab 3 Report

Congestion Control (pt. 1)

Kyle Pontius

# 1 Why Congestion Control

In our previous lab we implemented the TCP reliability feature. This enabled data to be from a source with the guarantee of having the data correctly and completely reassembled at the receiver, even with intermittent loss on the link. This implementation is extremely effective on a connection where there is only one sender and one receiver. Data can be freely sent back and forth without interference, or congestion, from other machines. However, when scaling TCP it quickly becomes obvious that network congestion is a serious concern that must be resolved if the network is going to continue to grow. One of the most serious symptoms of network congestions is data loss, which increases exponentially as queue size on routers in the network approach 100%. As routers approach this level of congestion, almost all data becomes deadlocked and the network can no longer function. To address this problem we use TCP’s congestion control, which was added after TCP was formally deployed to the Internet.

# 2 How Congestion Control Works

Congestion control functions on the assumption that all machines on the network are running this same protocol. If this is the case, the machines will approach an optimal state where bandwidth is shared fairly. Obviously there are certain conditions (such as round-trip time, packet size, and link speed) that make a difference in the real world. Congestion control relies on a few different features to be effective in controlling traffic on the network: (1) Dynamic congestion window, (2) Additive Increase with Multiplicative Decrease, (3) Slow Start, and (4) Fast Retransmit.

# 2.1 Dynamic Congestion Window

The congestion window used in TCP is central to controlling the rate at which packets are sent. Most, if not all of TCP’s congestion control tools are rooted in adjusting the congestion window to maximize throughput and minimize congestion on the network. The window size itself represents the number of packets that can be outstanding (packets sent, but not yet acknowledged). On connections where bandwidth is readily available a large window is desirable, enabling the user to send more data at a time. However, on connections where bandwidth is at a premium, smaller windows help to minimize congestion in the network by preventing a large number of packets from being outstanding, for each machine.

# 2.2 AIMD

In TCP Tahoe, after any loss event (either due to timeout or 3 duplicate ACKs) the congestion window size is dropped to one maximum packet size (MSS). This the multiplicative decrease portion of AIMD. The threshold, which represents the point at which additive increase should begin, is dropped to half of the window size when the loss occurred. Once window size reaches that threshold again, additive increase takes over and slowly increases the size of the window until loss occurs again. In our implementation of additive increase, we increment the size of the window by ***MSS \* new\_bytes\_received / window\_size***. On lossy connections, this allows the user to get very close to the point where data is lost before the loss actually occurs, maximizing the time spent sending data at the highest rate possible.

# 2.3 Slow Start

The goal of slow start is to quickly ramp up transmission speed, enabling TCP to quickly recover window size following a loss event. Due to the dynamic nature of connections, congestion, and frequent packet loss, it’s necessary to have this flexible and fast-recovering tool in order to get back up to speed quickly. This is especially helpful on systems with high bandwidth as slow start can promptly return them to maximum available speed. Slow start, in our implementation, works by incrementing the window size by the number of new bytes received in an ACK. For example, if my program sends sequence number 1000 and an ACK comes back for sequence number 8000, the congestion window is incremented by 7000. When graphing your solution, you should expect to see an exponential increase in the number of packets being sent. Slow start is ended, and additive increase begins, when the size of the window is greater than or equal to the threshold.

# 2.4 Fast Retransmit

The speed of modern connections is outstanding. However, as a side effect of this, when