Christian Rotondo

Computer Networking

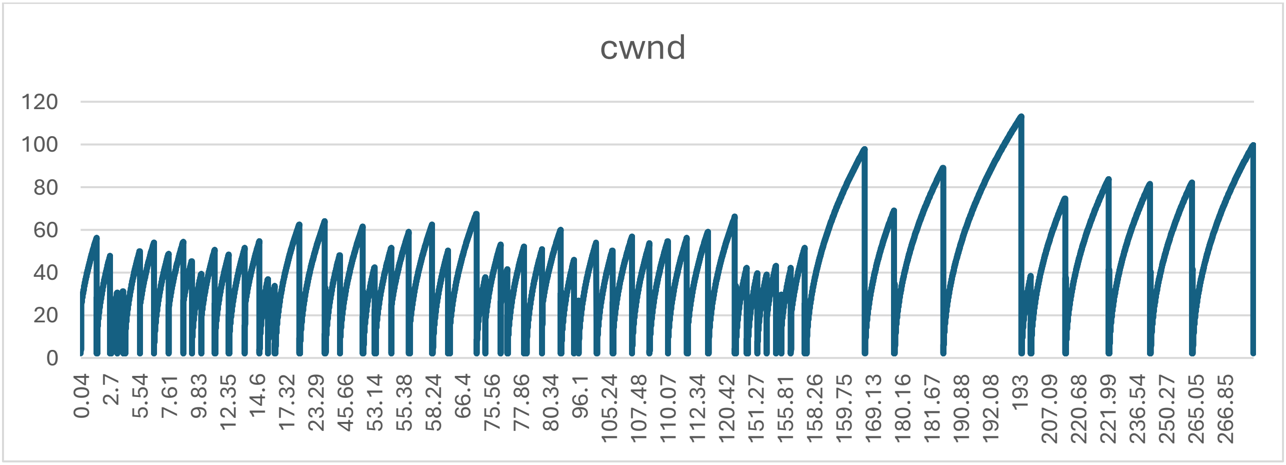
Professor Walsh

The saw client implementation does a great job of sending packets to the server in order. The downside is how slow it is. On the servers I ran it on, it ran at 0.1 frames per second. The other implementations can reach a higher fps, although they come with their own issues.

The burst client implementation tends to have issues with out-of-order packets. As N for the burst increases for the implementation, so does the frame rate, the out-of-order, and lost packets. The reason for the packet losses is due to the implementation not keeping track of ACKs to know what packets to resend. The unreliability of the client's delivery results in a choppy version of the video.

The pipelined version of the client features fewer out-of-order packets than the burst and no packet loss because it resends packets when ACKs of the corresponding packet aren't received within the given timeout. However, the implementation is inefficient, as it resends all packets contained in outstanding when a timeout occurs, which only worsens as N increases. The receiver may have received packets that were outstanding and sent back an ACK, meaning there was no need to resend the packets. This places an unnecessary load on the server and negatively impacts the frame rate, making it slower than the burst.

For the custom implementation, I implemented TCP Tahoe. It calculates the time-out with the estimated RTT and deviation RTT based on the times packets were sent and received. It uses a congestion window to adjust the number of outstanding packets and a threshold value provided by the user. Instead of sending all packets that are outstanding when a time-out occurs, like the pipelined version, it sends a single packet. Like Tahoe, when the congestion window increases, two packets are sent. Below is a graph of the congestion window at the recorded time when the ACK number matched the desired ACK number.



The graph is squished, but in the latter half, it's clear when the congestion window hits the threshold value and increases linearly, so it works according to TCP Tahoe. The highest frame rate achieved with the client was 4.5 fps, then dropped to around 2.0. It's more efficient and reliable than the previous protocols.