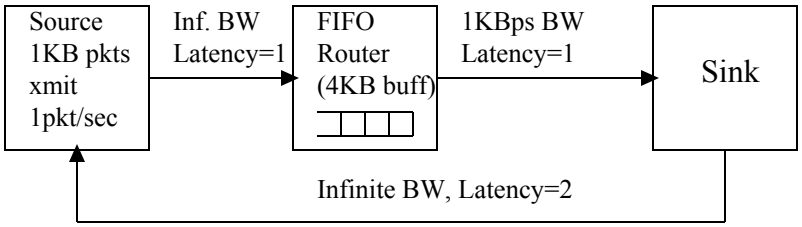
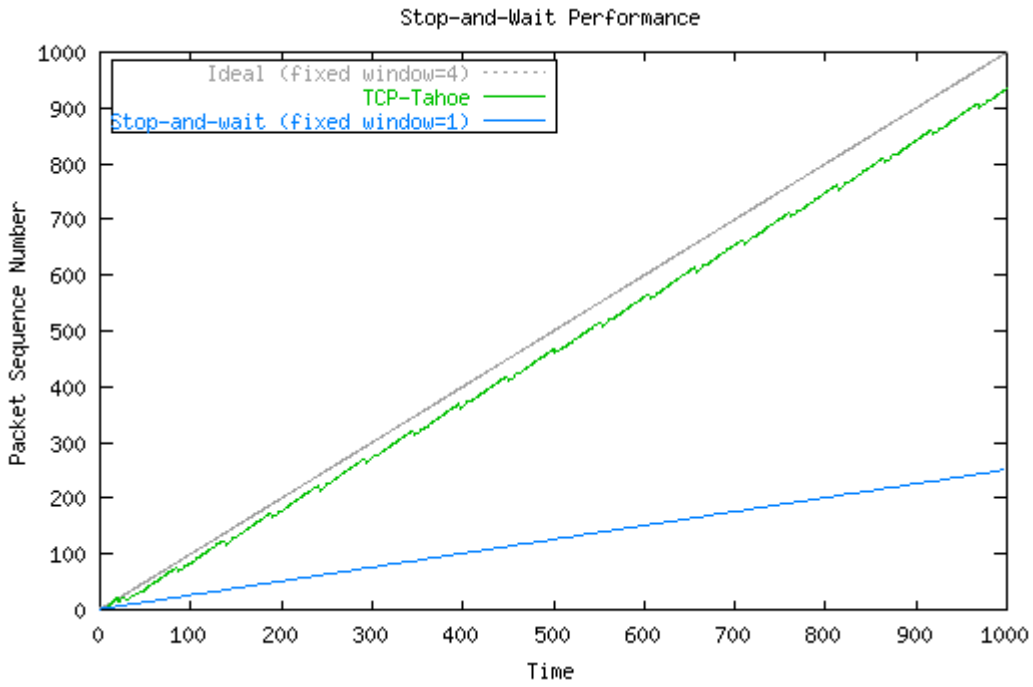
CSE 588 Scribe Notes, 5/7/02 (First half of class) **Kevin T. Manley**

Class started with the TCP Congestion game. The network we simulated was similar to the following diagram:



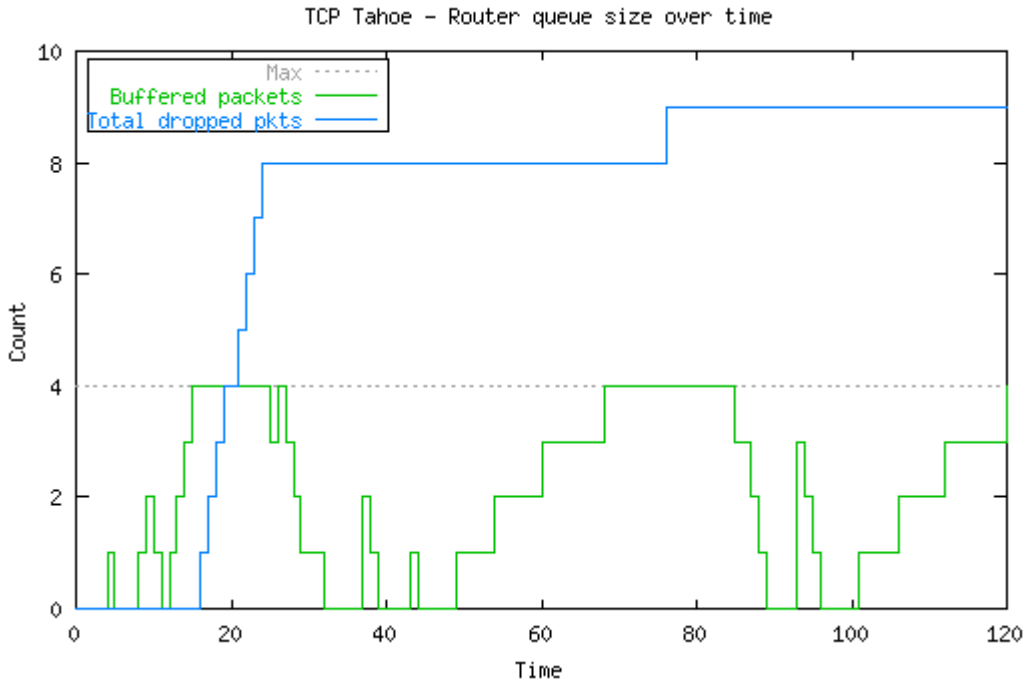
The total round trip time (RTT) for this network is 4. The bottleneck bandwidth is 1KBps. The ideal window size is the bottleneck bandwidth * RTT = 1KBps * 4 = 4KBps. Given 1KB packets, the ideal window size is 4 packets.

We first compare the performance of 3 kinds of sources using different sending policies. The first uses a simple stop and wait algorithm. This is the same as using a fixed window of size=1. The second uses a fixed window of ideal size=4. The third uses "TCP Tahoe" congestion avoidance/control (this includes RTT estimation, slow-start, and additive increase/multiplicative decrease but not fast retransmit). Note that flow control is not an issue here because the sink has an infinite window. We are concerned only with congestion avoidance/control.

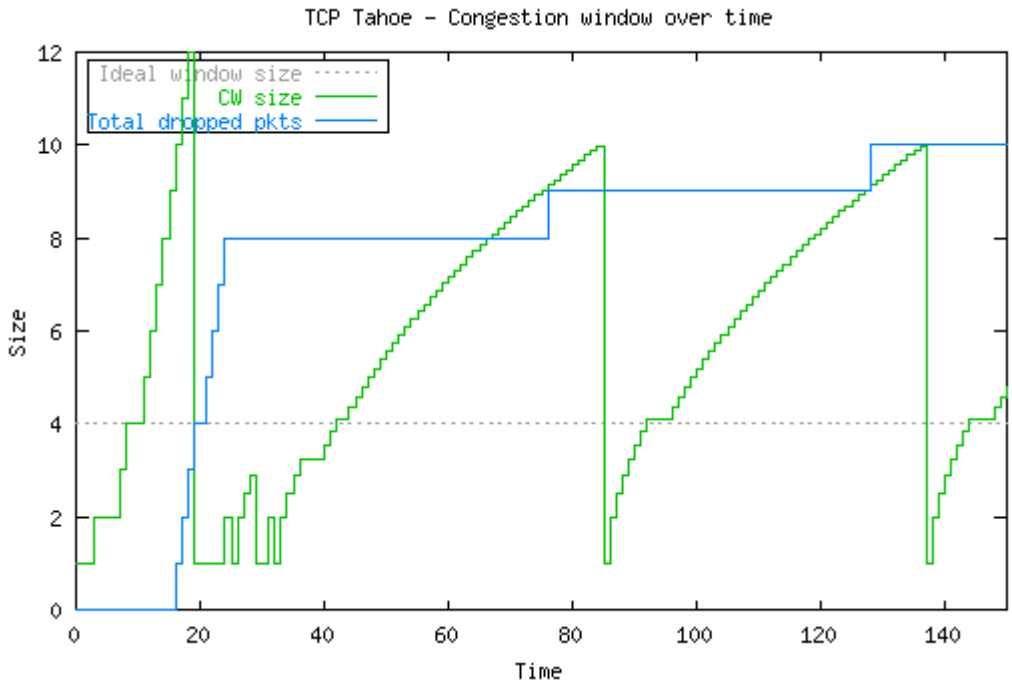


We simulate packet flow over 1000 time steps. Since the bottleneck link only supports transmitting one 1KB packet per time step, the max packet sequence number over 1000 time steps is 1000. Note that the source using a fixed window of ideal size=4 utilizes 100% of the available bandwidth. On the other hand, the fixed window source using stop-and-wait only transmits 250 packets in the same amount of time. Thus, the stop-and-wait source only utilizes 25% of the available bandwidth. Performance of the stop-and-wait source decreases as the bandwidth-delay product increases.

The TCP Tahoe source utilizes close to the maximum bandwidth of the link without any need for configuring an ideal window size. TCP Tahoe converges on a window size close to ideal by continuously probing for additional bandwidth and backing off when packet losses are detected. The jags in the chart line for TCP Tahoe shows the occasional retransmissions that occur as a result of the source periodically exceeding its ideal sending rate, which fills the buffer at the bottleneck router and forces the router to drop packets.



This chart shows the router queue size and cumulative number of dropped packets over time for the TCP Tahoe source. The router can buffer up to 4 packets. When the connection starts the source initiates slow-start. During this time, the source's congestion window is opened exponentially. This causes the bottleneck router's buffer to fill rapidly and forces the router to drop packets. The source detects the packet loss some time later (when expected ACKs do not arrive from the sink) and throttles back its sending rate. However, after this point the source continues to open its congestion window additively, probing for additional bandwidth. Again, this periodically causes the buffers at the bottleneck router to fill, resulting in further packets loss. This cycle repeats during the lifetime of the connection. Note that the source can only detect packet loss by assuming that an ACK that doesn't arrive in time indicates a sent packet was lost. The source needs accurate RTT estimation to set its packet retransmit timer to a value appropriate for the current connection.



This chart shows the TCP Tahoe source's congestion window size over time, as well as the cumulative number of packets dropped at the router. The exponential increase in CW size due to slow-start is clearly visible at the left side of the chart. Note that the source far overshoots the available bandwidth of the connection initially, resulting in multiple dropped packets. The source reacts by setting its CW threshold (ssthresh--described in Appendix B of "Congestion Avoidance and Control") to half its previous value, dropping its congestion window to one, and reinitiating slow start to the threshold level. After reaching the threshold level, additive increase takes over. Periodically even additive increase causes a packet loss. Note that after each loss it takes some amount of time for the source to detect the loss and drop its congestion window back to 1. This reaction time is ideally <= 1 RTT--again this emphasizes the importance of good RTT estimation at the source.

After playing the TCP congestion game, we discussed some other potential solutions to congestion avoidance/control:

- Equalize bandwidth (avoid bottleneck links). Not viable since you can't control end-host connectivity.
- Use more buffering. It turns out more buffering only hurts performance, since it delays packets and allows slow-start to increase the congestion window even further past the ideal size. Anyway "more" buffering can never be enough if the router's inbound and outbound links are unbalanced (you would need infinite buffering). The best size for the router buffer for a flow is the bandwidth-delay product for that flow.
- ICMP source quench. Proposed, but nobody uses it. The main problem is that it adds more packets to an already congested network.
- Router assist . Mark packets when average queue length > x. End host must include the mark bit in ACKs to the source. This strategy is used by DECbit and RED.
- Better end-host software. TCP Vegas is an example of this.