ITS413 – Bandwidth Delay Product and TCP

Internet Technologies and Applications, Semester 2, 2010

Prepared by Steven Gordon on 7 March 2011
ITS413Y10S2H11, Steve/Courses/ITS413/Examples/bandwidth-delay.tex, r1732

1 Bandwidth Delay Product and TCP

Recall the sliding window mechanism in flow control (which is used in TCP). The basic mechanism can be illustrated as in Figure 1.

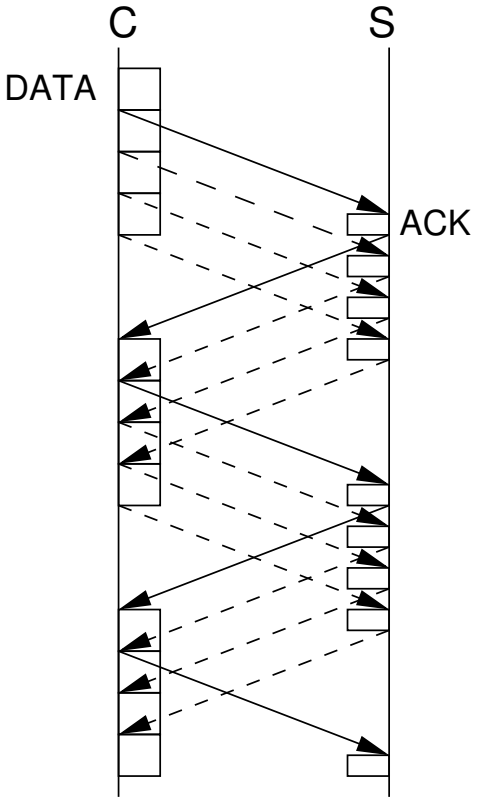

Figure 1: Sliding Window Mechanism

The client can send a window of packets, and then it has to wait for an ACK before it can send more. In the above example, the client sends 4 packets (the window size), and when receiving the ACK for the 1st packet the client can send another packet. When receiving the ACK for the 2nd packet, the client can send another, and so on. After sending the 8th packet, the client has to wait again for the ACK of the 5th packet.

Considering a single link, the time to receive an ACK, i.e. the round-trip-time, depends on the packet sizes (DATA and ACK), data rate ($rate$) and propagation time ($prop$):

$$RTT = \frac{DATA}{rate} + prop + \frac{ACK}{rate} + prop \quad (1)$$

In ideal conditions (no packet losses or retransmissions), the receiving rate is identical to the sending rate. If the client sends 4 packets per round trip time, then the server will receive 4 packets per round trip time. If we ignore packet headers, then the receive rate is identical to the throughput, η. If each packet was 1000 bytes and the RTT was 1ms, then $\eta = 32Mb/s$.

Note that the more time spent waiting, the less efficient the protocol is (i.e. lower throughput). If we spend no time waiting (and all the time transmitting) then the throughput will be at its maximum. From the above we want to determine when is throughput at its maximum?

In TCP the window size depends on the flow control and congestion control algorithms. Ignoring congestion control, then the window size is denoted as *awnd*, the advertised window set by the receiver. This is based on the available buffer space at the receiver. If the receiver has 10KB available, it will set $awnd = 10,000$.

To simplify the analysis, the size of the ACKs can be ignored. Figure 2 shows the same sliding window mechanism as Figure 1 but hides some details.

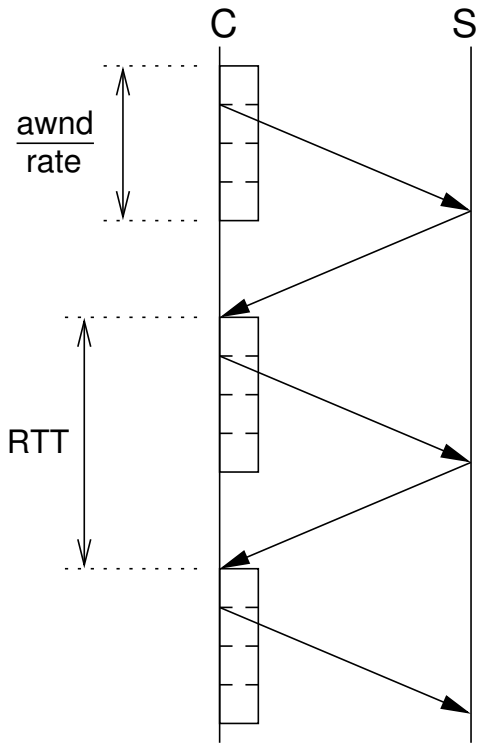

Figure 2: Simplified Sliding Window Mechanism

If the ACKs are very small, Equation 1 becomes:

$$RTT = \frac{DATA}{rate} + 2 \times prop \tag{2}$$

The time to transmit a window size of packets is $awnd/rate$.

Now consider two different cases where the window size *awnd* varies (but the other parameters, such as *rate* and packet sizes remains the same). Figure 3 shows two cases: on the left the window is 4 packets, on the right the window is 8 packets. With the smaller window, the client transmits all packets and then must wait before it can transmit more.

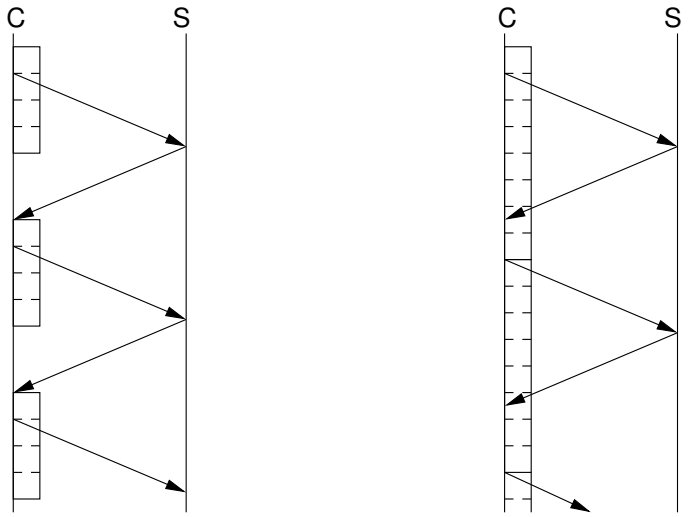

Figure 3: Sliding Window Mechanism with two different window sizes

This is because the time to transmit the window is less than the time to receive the first ACK. But in the second case on the right with a larger window, the time to transmit the window is larger than the time to receive the first ACK. Therefore as soon as the client has finished transmitting the window of packets, the client has already received an ACK and can immediately send the next packet. The client doesn't wait—it spends all the time transmitting, producing the maximum throughput.

So finally, under what conditions does the client achieve maximum throughput? When the time to transmit the window, *awnd*, of packets is equal to or greater than the time to receive the first ACK, i.e. the RTT:

$$\frac{awnd}{rate} \geq RTT$$

Re-arranging:

$$awnd \geq rate \times RTT \tag{3}$$

Note the right hand size is the Bandwidth Delay Product. If the advertised window (receive buffer) is equal to or greater than the BDP, then the client can achieve maximum throughput. If the advertised window is less than the BDP, then the throughput will be $awnd/RTT$.