TCP Sliding Windows, with Flow Control, and Congestion Control

Based on Peterson and Davie Textbook
Sliding Windows

- Normally a data link layer concept
- Interest is understanding TCP mechanism at the transport layer.
- Each frame is assigned a sequence number - SeqNum
- The sender maintains three variables: send window size (SWS), last ACK received (LAR), and last Frame sent (LFS)
Sender variables

- **SWS** :: the upper bound on the number outstanding frames (not ACKed) the sender can transmit
- **LAR** :: the sequence number of the last ACK received
- **LFS** :: the sequence number of the last frame sent
Sender Invariant

LFS – LAR $\leq$ SWS
Sender Window

- An arriving ACK ➔ LAR moves right 1 ➔ sender can send one more frame
- Associate a *timer* with each frame sender transmits
- Sender retransmits the frame if the timer *times out*
- Sender buffer :: up to *SWS* frames
Receiver variables

• **Receiver window size (RWS)** :: the upper bound on the number of out-of-order frames the receiver is willing to accept

• **Largest acceptable frame (LAF)** :: the sequence number of the largest acceptable frame

• **Last frame received (LFR)** :: the sequence number of the last frame received
Receiver Invariant

$LAF - LFR \leq RWS$
Receiver Window

• When a frame arrives with \texttt{SeqNum}:
  
  If \((\texttt{SeqNum} \leq \texttt{LFR} \text{ or } \texttt{SeqNum} > \texttt{LAF})\),
  \begin{quote}
  \textit{the frame is discarded because it is outside the window.}
  \end{quote}

  If \((\texttt{LFR} < \texttt{SeqNum} \leq \texttt{LAF})\),
  \begin{quote}
  \textit{the frame is accepted.}
  \end{quote}
Receiver ACK Decisions

\textbf{SeqNumToAck} :: largest sequence number not yet ACKed such that all frames $\leq$ SeqNumToAck have been received.

- Receiver ACKs receipt of \textbf{SeqNumToAck} set

\quad \text{LFR} = \text{SeqNumToAck}

\quad \text{LAF} = \text{LFR} + \text{RWS}
TCP Sliding Windows

* switch from packet pointers to byte pointers
  • Guarantees reliable delivery of data.
  • Ensures data delivered in order.
  • Enforces flow control between sender and receiver.
  • The idea is: the sender does not overrun the receiver’s buffer
Receiver’s Advertised Window

- The big difference is the size of the sliding window size at the receiver is **not fixed**.
- The receiver *advertises* an adjustable window size (**AdvertisedWindow** field in TCP header).
- Sender is limited to having **no more than AdvertisedWindow** bytes of unACKed data at any time.
TCP Flow Control

• The discussion is similar to the previous sliding window mechanism except we add the complexity of sending and receiving *application processes* that are filling and emptying their local buffers.

• Also introduce complexity that buffers are of finite size, but not worried about where the buffers are stored.

  MaxSendBuffer
  MaxRcvBuffer
TCP Flow Control

• Receiver throttles sender by advertising a window size no larger than the amount it can buffer.

On TCP receiver side:

\[ \text{LastByteRcvd} - \text{LastByteRead} \leq \text{MaxRcvBuffer} \]

to avoid buffer overflow!
TCP Flow Control

TCP receiver advertises:

\[
\text{AdvertisedWindow} = \text{MaxRcvBuffer} - \\
(\text{LastByteRcvd} - \text{LastByteRead})
\]

i.e., the amount of free space available in the receive buffer.
TCP Flow Control

TCP sender must adhere to $AdvertisedWindow$ from the receiver such that

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{AdvertisedWindow}$$

or use $EffectiveWindow$:

$$EffectiveWindow = AdvertisedWindow - (\text{LastByteSent} - \text{LastByteAcked})$$
TCP Flow Control

Sender Flow Control Rules:

1. **EffectiveWindow** > 0 for sender to send more data
2. **LastByteWritten** – **LastByteAcked** <= **MaxSendBuffer**
   
   *equality ➔ send buffer is full!!*
   
   ➔ *TCP sender must block sender application.*
TCP Congestion Control

- **CongestionWindow**: a variable held by source for each connection.

TCP is modified such that the maximum number of bytes of unacknowledged data allowed is the *minimum of* CongestionWindow and AdvertisedWindow.

\[
\text{MaxWindow} :: \min (\text{CongestionWindow}, \text{AdvertisedWindow})
\]
TCP Congestion Control

And finally, we have:

\[
\text{EffectiveWindow} = \text{MaxWindow} - (\text{LastByteSent} - \text{LastByteAcked})
\]

The idea :: the source effective window can be no faster than the slowest of the network (routers) or the destination Host.

* The TCP source receives implicit and/or explicit indications of congestion by which to reduce the size of CongestionWindow.