TCP Sliding Windows, Flow Control, and Congestion Control

Sliding Windows

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

- **SWS**: the upper bound on the number of 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

\[ \text{LFS} - \text{LAR} \leq \text{SWS} \]
Sender Window

- An arriving ACK \( \rightarrow \) LAR moves right 1 \( \rightarrow \) sender can send one more frame.
- Associate a *timer* with each frame the 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 <= RWS
Receiver Window

When a frame arrives with \text{SeqNum}: 

If \text{(SeqNum} \leq \text{LFR or SeqNum} > \text{LAF)} 
\text{the frame is discarded because it is outside the window.}

If \text{(LFR < SeqNum} \leq \text{LAF)}  
\text{the frame is accepted.}
Receiver ACK Decisions

```
SeqNumToAck :: largest sequence number not yet ACKed such that all frames <= SeqNumToAck have been received.

• Receiver ACKs receipt of SeqNumToAck and sets

  LFR = SeqNumToAck
  LAF = LFR + RWS

SeqNumToAck is adjusted appropriately!
```
TCP Sliding Windows

* In practice, the TCP implementation switches 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.
Figure 5.8. Relationship between TCP send buffer (a) and receive buffer (b).
Receiver’s Advertised Window

- The big difference in TCP is that the size of the sliding window size at the TCP 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 we introduce the complexity that buffers are of finite size without worrying 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 receiver advertises:

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

i.e., the amount of free space available in the receiver’s buffer.
TCP Flow Control

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

\[ \text{LastByteSent} - \text{LastByteAacked} \leq \text{AdvertisedWindow} \]

or use EffectiveWindow:

\[ \text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAacked}) \]
TCP Flow Control

Sender Flow Control Rules:
1. EffectiveWindow > 0 for sender to send more data.
2. LastByteWritten – LastByteAcked <= MaxSendBuffer
   equality here ➔ send buffer is full!! ➔ TCP sender process must block the sender application.
TCP Congestion Control

• CongestionWindow :: a variable held by the TCP 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.

MaxWindow :: min (CongestionWindow, AdvertisedWindow)
TCP Congestion Control

Finally, we have that

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

The idea :: the source’s effective window can be no faster than the slowest of the network (i.e., its core routers) or the destination Host.

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