TCP Sliding Windows, Flow Control, and Congestion Control

Sliding Windows Outline

• Generic Sliding Windows
• Receiver Response Choices
• Introduction to TCP Sliding Windows
  – Flow control and buffers
  – Advertised window
  – 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: 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 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 → LAR moves right 1 → 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

\[ \text{LAF} - \text{LFR} \leq \text{RWS} \]
Receiver Window

When a frame arrives with SeqNum:

If \((\text{SeqNum} \leq \text{LFR} \text{ or } \text{SeqNum} > \text{LAF})\)

the frame is **discarded** because it is outside the window.

If \((\text{LFR} < \text{SeqNum} \leq \text{LAF})\)

the frame is **accepted**.
Receiver ACK Decisions

**SeqNumToAck** :: largest sequence number *not yet ACKed* such that all frames \( \leq \) **SeqNumToAck** have been received.

- Receiver ACKs receipt of **SeqNumToAck** and sets
  - \( \text{LFR} = \text{SeqNumToAck} \)
  - \( \text{LAF} = \text{LFR} + \text{RWS} \)

**SeqNumToAck** is adjusted appropriately!
Generic ACK Choices

1. ACK sequence number indicates the *last frame successfully received.*
   - OR -

2. ACK sequence number indicates the *next frame the receiver expects to receive.*

   Both of these can be strictly *individual* ACKs or represent *cumulative* ACKing.

Cumulative ACKs is the most common technique.
Generic Responses to a Lost Packet or Frame

1. Use a duplicate ACK.

2. Use a selective ACK [SACK].

3. Use a negative ACK [NACK].
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 5.3
TCP Managing a Byte Stream

Application process
Write bytes
TCP
Send buffer
Segment Segment Segment
Transmit segments
TCP
Receive buffer
Application process
Read bytes
P&D slide
Figure 5.8 Relationship between TCP Send Buffer and TCP Receive Buffer

(a) Sending application
- LastByteWritten
- TCP
- LastByteSent
- LastByteAcked

(b) Receiving application
- LastByteRead
- TCP
- LastByteRcvd
- NextByteExpected
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.
Figure 5.4 TCP Header Format

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>SrcPort</td>
</tr>
<tr>
<td>4-7</td>
<td>DstPort</td>
</tr>
<tr>
<td>8</td>
<td>SequenceNum</td>
</tr>
<tr>
<td>16</td>
<td>Acknowledgment</td>
</tr>
<tr>
<td>17-20</td>
<td>Flags</td>
</tr>
<tr>
<td>21-24</td>
<td>AdvertisedWindow</td>
</tr>
<tr>
<td>25-28</td>
<td>Checksum</td>
</tr>
<tr>
<td>29-31</td>
<td>UrgPtr</td>
</tr>
<tr>
<td>32-31</td>
<td>Options (variable)</td>
</tr>
<tr>
<td>32</td>
<td>Data</td>
</tr>
</tbody>
</table>

P&D slide
Figure 5.5 Simplified TCP

Sender

Data (SequenceNum)

Receiver

Acknowledgment + AdvertisedWindow
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

- The receiver **throttles** the 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 receiver’s buffer.
The TCP sender must adhere to the AdvertisedWindow from the receiver such that

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

or use EffectiveWindow

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

Sender Flow Control Rules:

1. **EffectiveWindow > 0** for sender to send more data.

2. **LastByteWritten – LastByteAcked ≤ MaxSendBuffer**

   equality here \( \Rightarrow \) send buffer is full!!

   \( \Rightarrow \) 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 :: \( \min (\text{CongestionWindow}, \text{AdvertisedWindow}) \)
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 CongestionWindow.
Sliding Windows Summary

• Generic Sliding Windows
• Receiver Response Choices
• Introduction to TCP Sliding Windows
  – Flow control and buffers
  – Advertised window
  – Congestion control