Announcements

• Reminder: No class on Monday, Sept. 20

• Project 1 will be handed out next Weds.
  – Form groups of two or three
  – Mail group members to Baohua Wu.
    baohua@seas.upenn.edu
  – If you can’t find a partner, mail Baohua.

  – Groups should be formed before project is handed out
Recap

• Link-layer protocols
  – Frame transmission
  – Error detection (CRC)
Today: Reliable Transmission

• Now we can detect errors…
• What do we do when we find one?

• Corrupt frames must be discarded.
• Need to recover these lost frames.
Fundamental mechanisms

• **Acknowledgments (ACK)**
  – Small control frame (little data)
  – When sender gets an ACK, recipient has successfully gotten a frame

• **Timeouts**
  – If sender doesn’t get an ACK after “reasonable” time it retransmits the original

• General strategy called Automatic Repeat Request (ARQ)
Stop-and-Wait

• Simplest scheme
  – After transmitting one frame, sender waits for an ACK
  – If the ACK doesn’t arrive, sender retransmits
Stop-and-Wait scenario 1

Everything works
Stop-and-Wait scenario 2

<table>
<thead>
<tr>
<th>Frame</th>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original frame</td>
<td>Lost &amp; Resent</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stop-and-Wait scenario 3

Sender

Timeout

frame

Timeout

ACK

timeout

Receiver

frame

ACK

ACK Lost

Time
Stop-and-Wait scenario 4

Sender

Receiver

Timeout

Timeout

Time

Timeout

frame

ACK

Timeout too short

ACK

Timeout
Add a 1-bit sequence number so receiver can detect repeated frames.
Stop-and-Wait

• Inefficient use of link’s capacity
• Sends 1 frame per RTT

• Example:
  – 10Mbs Link
  – 16ms RTT
  – Delay x Bandwidth product is about 20KB
  – Frame size of 1K yields about 5% link capacity
More efficient solution

Sender ready to transmit N+1\textsuperscript{st} frame just as 1\textsuperscript{st} ACK arrives.

\[ N = \frac{\text{delay} \times \text{bandwidth}}{\text{frame size}} \]
Sliding Window Algorithm

• Sender assigns a *sequence number* to each frame: SeqNum
  – For now, assume SeqNum can grow infinitely

• Send Window Size (SWS)
  – Upper bound on # of unacknowledged frames sender can transmit

• Last ACK Received (LAR)
  – Sequence number of last ACK

• Last Frame Sent (LFS)
Sender Invariant

- LFS – LAR ≤ SWS
- Associates timeout with each frame sent
  - Retransmits if no ACK received before timeout
- When ACK arrives, increase LAR
  - Means another frame can be sent
Receiver

- Receive Window Size (RWS)
  - Number of out-of-order frames it will accept
- Largest Acceptable Frame (LAF)
- Largest Frame Received (LFR)
- $LAF - LFR \leq RWS$
Receiver Algorithm

• When packet numbered SeqNum arrives
  – If (SeqNum ≤ LFR) or (SeqNum > LAF) discard
  – Else accept the packet

• Define: SeqNumToAck
  – Largest unACK’ed sequence # s.t. all earlier frames have been accepted

• Receiver sends ACK(SeqNumToAck)

• LFR = SeqNumToAck

• Laf = LFR + RWS
Example Sliding Window Protocol

Sender begins sending frames 1,2,3 with appropriate timers.

SWS = RWS = 3
Example Sliding Window Protocol

SWS = RWS = 3

Receiver gets frame 1
SeqNumToAck = 1
Receiver sends ACK(1)

LFR = 1
LAF = LFR + RWS
Example Sliding Window Protocol

SWS = RWS = 3

While ACK(1) is in transit, frame 2 is lost and frame 3 is accepted.
Example Sliding Window Protocol

SWS = RWS = 3

SeqNumToAck = 1
Receiver sends another Ack(1) message.
Example Sliding Window Protocol

LAR

SWS = RWS = 3

LFR

Sender gets ACK(1).
Sets LAR = 1
Increases LFS to 4
Example Sliding Window Protocol

Sender transmits frame 4 and then the timer for frame 2 expires, so it resends.
Example Sliding Window Protocol

SWS = RWS = 3

LAR

Sender gets ACK(1) again—ignores it.

Receiver gets frame 4
SeqNumToAck = 1
Receiver sends ACK(1)

LFR

CSE331 Fall 2004
Example Sliding Window Protocol

SWS = RWS = 3

Sender

Receiver

LAR

LFR

Receiver gets frame 2
SeqNumToAck = 4
Receiver sends ACK(4)

LFR = 4
LAF = LFR + RWS
Example Sliding Window Protocol

**Sender**

SWS = RWS = 3

```
1 2 3 4 5 6 7
```

LAR

1. Sender gets ACK(1) again—ignores it.
2. Sender gets ACK(4)
   - Sets LAR = 4
   - Increases LFS

**Receiver**

```
1 2 3 4 5 6 7
```

LFR
Variants on Sliding Window

- Receiver doesn’t transmit redundant ACKs
- Receiver transmits *selective ACKS*
  - ACK indicates exactly which frames have been accepted
Window Sizes

- If RTT x Bandwidth product is known then
  \[ SWS = \frac{RTT \times Bandwidth}{Framesize} \]

- Receive window size:
  - 1 = no buffering of out-of-order frames
  - RWS = SWS buffers as many as can be in flight
  - Note that RWS > SWS is not sensible
Finite Sequence Numbers

• Recall that for Stop-and-Wait we needed two sequence numbers.
• How many do we need for Sliding Window?

• Suppose SWS=RWS
  – How many sequence numbers should there be?

  – Is SWS + 1 sufficient?
Sufficient MaxSeqNum

• Frame i’s sequence num is $i \% \text{MaxSeqNum}$

• Assuming SWS = RWS
• SWS < $(\text{MaxSeqNum} + 1)/2$

• Why?
  – Consider case where all the ACKS are lost.
  – Suppose SWS = RWS = 3
  – MaxSeqNum = 5 (sequence numbers = 0,1,2,3,4) is insufficient
Roles of Sliding Window Algorithm

• Reliable delivery
  – It provides an efficient retransmission protocol for dealing with errors
• In-order delivery
  – The receiver buffers frames and delivers them in sequence number order
• Flow control
  – It sends ACKs back to give hints to sender
  – More sophisticated version could give # of frames the receiver has room for—throttles the sender.
Sliding window in practice

• TCP (Transmission Control Protocol)
  – Transportation layer protocol
  – Uses sliding window algorithm
  – More complex because it’s used in an Internetwork – not over a direct link
    – Bandwidth x delay not known
    – Dynamically changes timeouts
    – Larger buffers for in-order delivery