Data Link Layer, Part 3

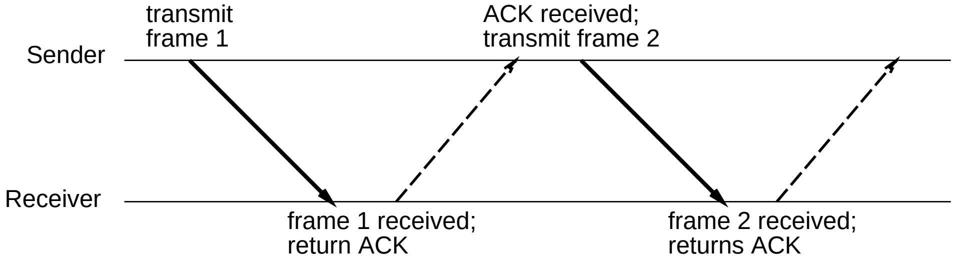
Sliding Window Protocols

Preface

- We are about to discuss a series of protocols used between a pair of sender and receiver.
- These protocols serve three purposes
  1. to guarantee delivery reliability,
  2. to enforce correct ordering of frames delivered to the network layer at the receiving end, and
  3. to provide flow control.
- Although this discussion is in the context of the DLL, we will see later in the semester that the same issues also arise at the transport layer.

Stop And Wait

- The sender transmits a frame.
- If the receiver successfully receives the frame, it delivers the frame to the network layer and returns an **acknowledgment** (ACK) to the sender.
- Only after receiving the ACK will the sender be allowed to transmit the next frame.



transmit
frame 1

ACK received;
transmit frame 2

Sender

Receiver

frame 1 received;
return ACK

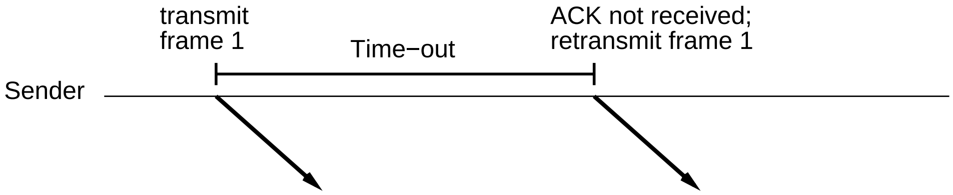
frame 2 received;
returns ACK

Discussion

- What could cause the receiver not to receive the frame successfully ?
  - ☞ the frame is so corrupted by noises that the receiver does not even recognize it
  - ☞ when the frame arrives at the receiver, there is not enough memory space to accommodate it
  - ☞ transmission errors reported by the error detection mechanism
- This simple protocol provides flow control but does not enforce reliability — actually, it fails in noisy links.
  - ☞ What would happen when any transmission (of the frame or the ACK) is corrupted ?

The Time-out Mechanism

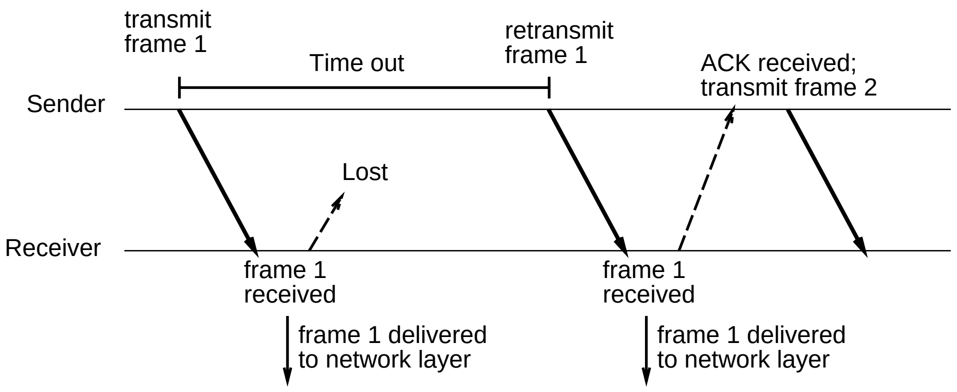
- The sender re-transmits a frame if the ACK does not come back within some predetermined length of time.



- This implies that the DLL of the sender must keep the frame until it receives the ACK.

Discussion

- Causes for not receiving the ACK:
  - ☞ the frame is corrupted by noises
  - ☞ the frame is successfully delivered but the ACK is corrupted by noises
- This protocol is still flawed: although the delivery of the frame is guaranteed, the network layer at the receiving end may see multiple copies of the frame.



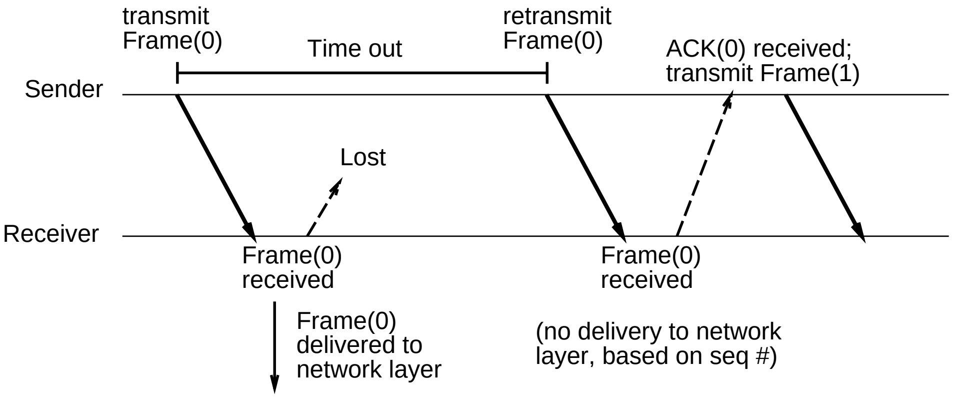
Sequence Numbers

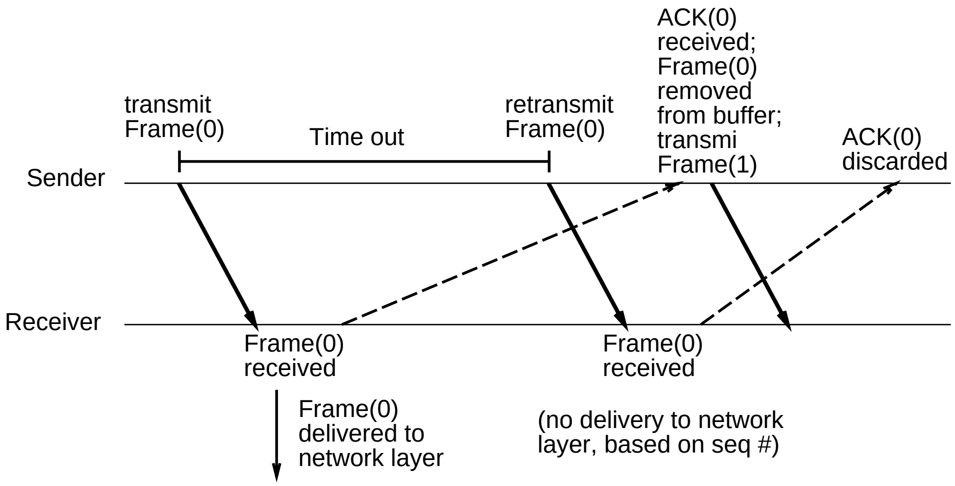
- The sender numbers frames.
- Using these numbers, called sequence numbers, the receiver keeps track of history:
  - ☞ Method 1: it remembers exactly which frames have been received
  - ☞ Method 2: it remembers only that all the frames up to a certain sequence number have been received
- How many bits in a sequence number ?
  - ☞ In a stop-and-wait protocol, only one bit !

Revising Stop-And-Wait Protocol

- Frames are numbered alternately: 0, 1, 0, 1, . . .
- Receiver acknowledges the correct frames with an ACK that has the same number as the frame.
- The sender transmits a frame after a fixed time-out period.
- This version of stop-and-wait works but is still inefficient.
  - ☞ link bandwidth is wasted when the sender "stops and waits"
  - ☞ for better performance, we must allow multiple outstanding frames

Example


transmit Frame(0)
Time out
retransmit Frame(0)
ACK(0) received; transmit Frame(1)
Sender
Lost
Receiver
Frame(0) received
Frame(0) received
Frame(0) delivered to network layer
(no delivery to network layer, based on seq #)



Note that when Frame(0) is received the second time, it is acknowledged but not delivered to the network layer.

Pipelining

- Waiting for ACKs can be very inefficient, especially if the propagation delay is long.
- Solution is to pipeline frames, that is, to send the next (several) frame before the current one is acknowledged.
- Two approaches:

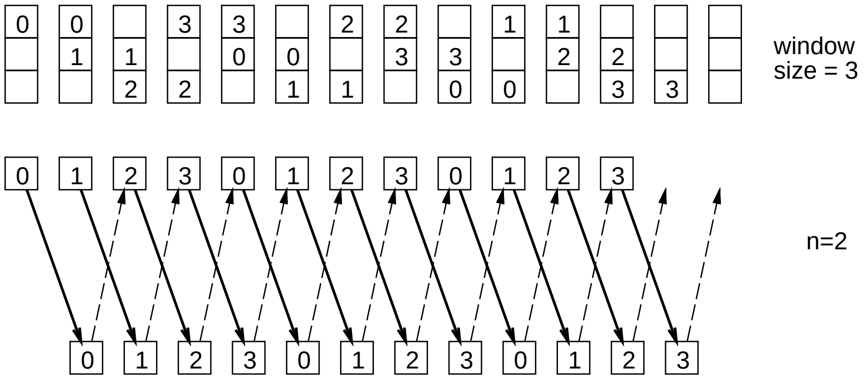
  1. **go-back-n**: receiver discards all subsequent frames following an error, forcing the sender to “go back” to the damaged/lost frame, that is, to retransmit the bad and all subsequent frames
  2. **selective repeat**: receiver stores correct frames following the bad one(s); sender retransmits only bad frames

Go-Back-N

- Even though we have 2^n distinct sequence numbers, only $2^n - 1$ frames may be outstanding at any one time.
- The sender maintains a set of $2^n - 1$ buffers, called a window, to keep unacknowledged frames.
- Whenever the sender transmits a frame, the frame is copied to a slot in the window.
  - ☞ the copy is used in the retransmission of the frame if its ACK does not return before time-out
  - ☞ the slot is freed when the ACK of the frame arrives
- Receiver window size = 1

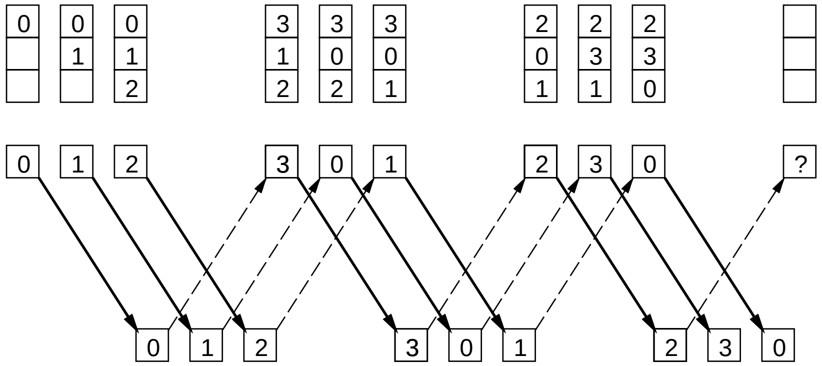
Example: Large n Values

When n is large enough relative to the round-trip time, we can take full advantage of the bandwidth of the link.



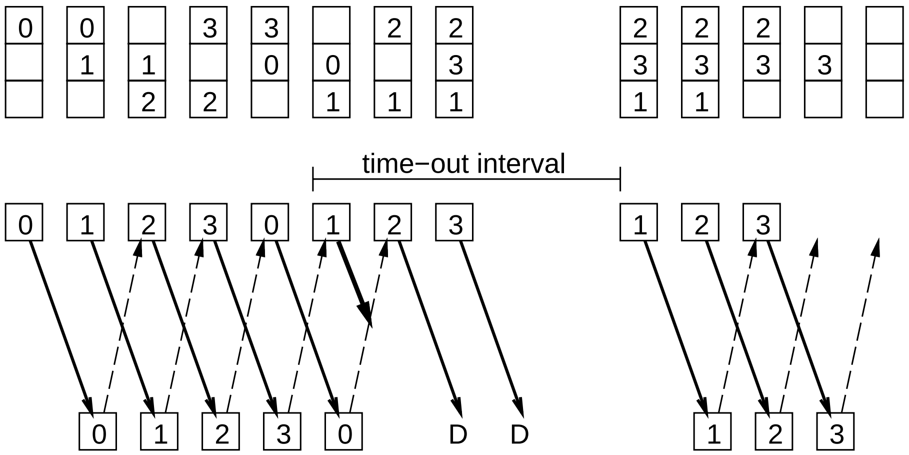
Example 2: Value of n Too Small

If the value of n is too small (again, relative to the round-trip time), then the "stop-and-wait" phenomenon still occurs.

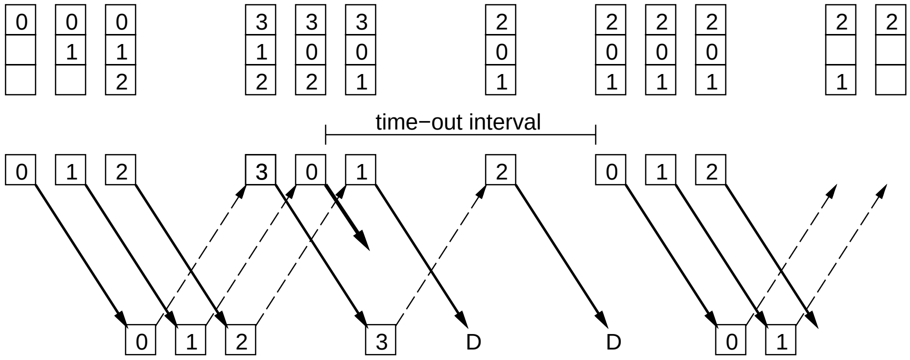


Go-Back-N with Transmission Errors

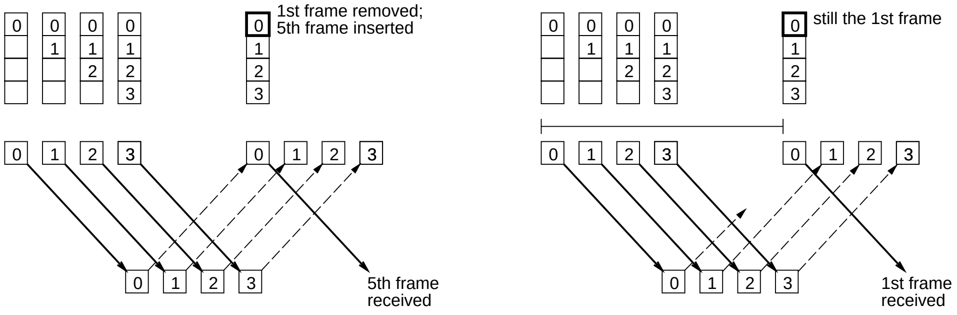
Case 1: n large enough



Case 2: n too small



What happens if window size = 2^n ?



A question asked by receiver : Is the second frame that has sequence number 0 the fifth frame or the retransmission of the first frame ?

Put in another way, should the receiver deliver this frame to its upper layer ?

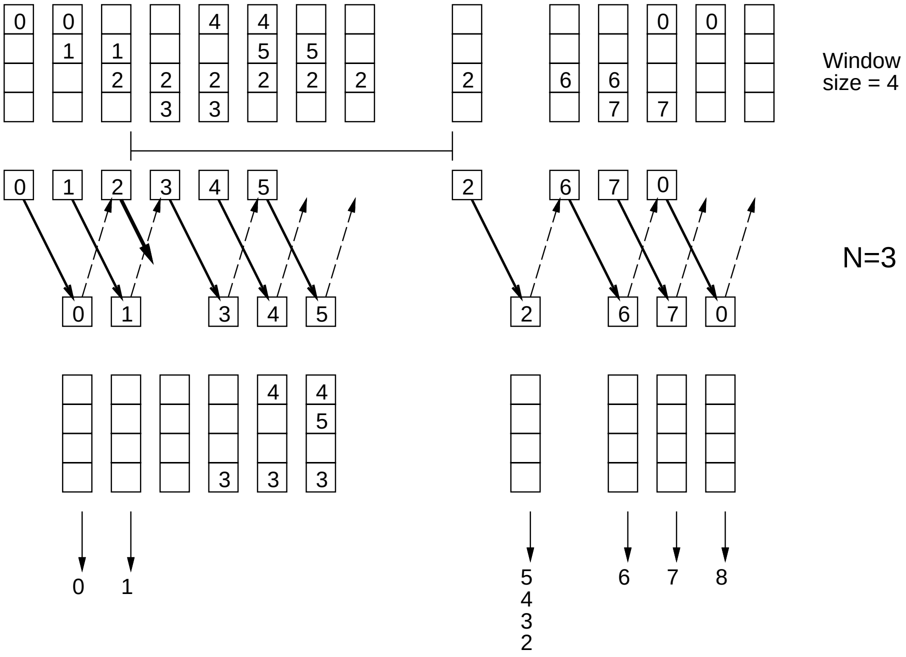
Optimizations

- The receiver attaches ACKs to outgoing frames destined to the sender; this technique is called **piggybacking**.
- Further, we can "aggregate ACKs," that is, the ACK of the i-th frame also acknowledges the receipt of all the frames up to i.
- One design issue is that how long the receiver should wait for an outgoing data frame — waiting too long may cause the sender to time out and retransmit.
- Another technique is to have the receiver, upon seeing a corrupted frame, immediately returns a **negative acknowledgment** (NACK), which forces the sender to retransmit before time-out.

  ☞ this is useful when

Selective Repeat

- Sender maintains a window of size 2^{n-1} to keep unacknowledged frames, where n is the number of bits in sequence numbers.
- Sender retransmits a frame that is not acknowledged after a time-out.
- Receiver acknowledges the receipt of a frame with an ACK that has the same sequence number as the frame.
- Receiver also maintains a window of size 2^{n-1}, which is used to keep frames following a damaged one.
- Receiver delivers frames to the network layer in order.



You are encouraged to work out the example using long round-trip times.

Why the Restriction of 2^{n-1}

Performance Analysis

- C: channel capacity in bps (the R on page 301)
- I: interrupt and service time + propagation delay (A combined result of D, S, and T on page 301)

  ☞ think of $2I$ as the round-trip time, excluding the transmission time of the frame itself
- F: number of bits per frame (the F on page 301)

Performance of Stop and Wait

- At time $(F/C + 2I)$, the sender has processed the ACK.
- Total bandwidth during this period is

$$C(F/C + 2I) = F + 2CI.$$

- So, the utilization is

$$U = \frac{F}{F + 2CI}.$$

  ☞ the longer the frame, the better the bandwidth utilization

  ☞ the larger the capacity, the lower the utilization

  ☞ the longer the propagation delay, the lower the utilization

Performance of Sliding Window

- Sender can send for $W \times F/C$ seconds before it must stop and wait.
- ACK of first frame arrives at time $F/C + 2I$

1. large window (sender may transmit continuously)
   - ☞ $WF/C \geq F/C + 2I$
   - ☞ $W \geq 1 + 2CI/F$
   - ☞ Of course, $U = 100$ %.
2. small window (sender must stop and wait)
   - ☞ $W < 1 + 2CI/F$
   - ☞ Sender can transmit W frames in time $F/C + 2I$.
   - ☞ Therefore, $U = WF/(F + 2CI)$.

The Meaning of CI

- Boundary between large and small windows has been $W = 1 + 2CI/F$
- Hence, what is CI ?
- Example of CI:
  - ☞ 10Mbps over 1km, $CI = 10 \times 10^6 \times (1000/c) \approx 50$ bits, where c is the speed of light, 2×10^8 meters per second.
  - ☞ 65kbps over 3000 km, $CI \approx 960$ bits
  - ☞ (satellite) 64 kbps with $I = 270$ msec, $CI \approx 17000$ bits.
- What is CI/F ?

Discussion

- To determine the window size over a given link, we must consider:

  1. the capacity/bandwidth of the link
  2. the propagation delay over the link
  3. the length of frames

- The second factor is especially important in WANs.

- For high-bandwidth and/or long-distance links, better performance (up to a point) can be achieved by larger window sizes, that is, the use of more memory space.