ECE/COMPSCI 356 Computer Network Architecture

Lecture 6: Link layer: Error Detection and Reliable Transmission

Neil Gong

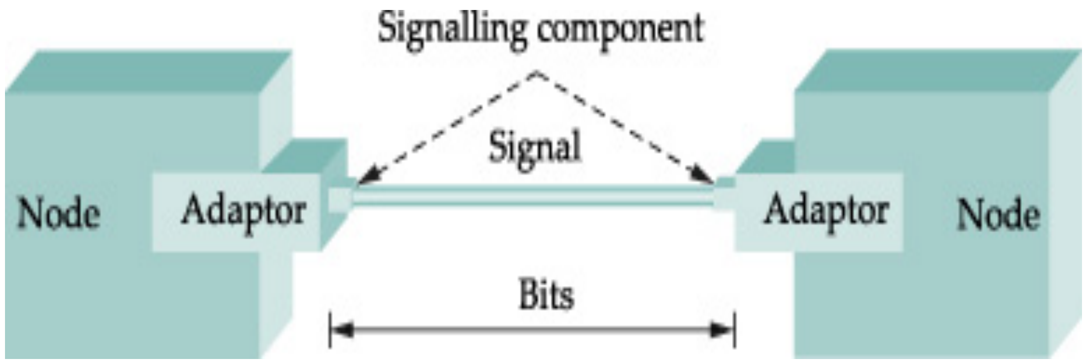
neil.gong@duke.edu

Slides credit: Xiaowei Yang, PD

Overview

- Link layer functions
  - Encoding
    - NRZ, NRZI, Manchester, 4B/5B
  - Framing
    - Byte-oriented, bit-oriented, clock-based
  - Error detection
    - Parity, checksum, CRC
  - Reliable transmission
    - Error correction, stop-and-wait, sliding window

Link-layer functions



- Completed by adapters
  - Encoding
  - Framing
  - **Error detection**
  - Reliable transmission

Error Detection

- Bit errors are introduced into frames
  - Because of electrical interference and thermal noises
- Common technique for detecting transmission error
  - CRC (Cyclic Redundancy Check)
    - Used in HDLC, DDCMP, CSMA/CD, Token Ring
  - Other approaches
    - Two Dimensional Parity (BISYNC)
    - Checksum (IP)

Error Detection

- Basic Idea of Error Detection
  - To add redundant information to a frame that can be used to determine if errors have been introduced
  - Imagine (Extreme Case)
    - Transmitting two complete copies of data
      - Identical → No error
      - Differ → Error
      - Poor Scheme ???
        - n bit message, n bit redundant information
        - Error can go undetected
    - In general, we can provide strong error detection technique
      - k redundant bits, n bits message, k << n
      - In Ethernet, a frame carrying up to 12,000 bits of data requires only 32-bit CRC

Cyclic Redundancy Check (CRC)

- High-level idea:
  - Represent an n+1-bit message with an n degree polynomial M(x)
    - Given a bit string 110001 we can associate a polynomial on a single variable *x* for it.

  $1.x^5+1.x^4+0.x^3+0.x^2+0.x^1+1.x^0 = x^5+x^4+1$ and the degree is 5.

  - Divide the polynomial by a degree-k divisor polynomial C(x)
  - k-bit CRC: remainder
  - Send Message + CRC that is dividable by C(x)

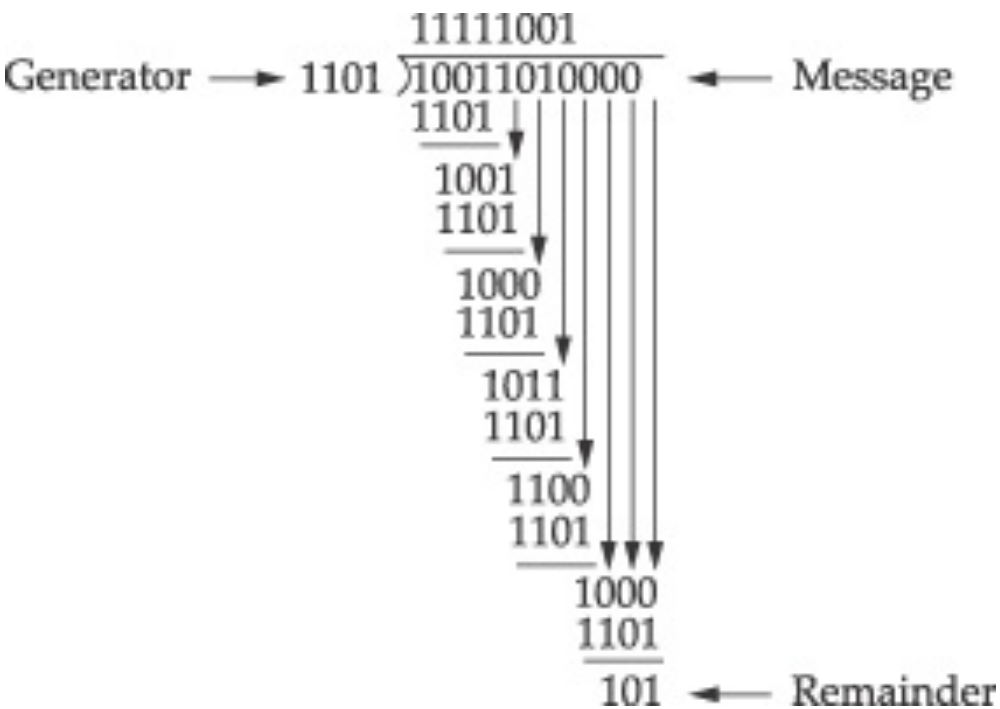
Polynomial arithmetic modulo 2

- $B(x)$ can be divided by $C(x)$ if $B(x)$ has higher degree
- $B(x)$ can be divided once by $C(x)$ if of same degree
  - $x^3 + 1$ can be divided by $x^3 + x^2 + 1$
  - The remainder would be $0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0$ (obtained by XORing the coefficients of each term)
- Substraction $B(x) – C(x)$ is done by XOR each pair of matching coefficients
  - $(x^4 + x^3 + 1) – (x^3 + x^2 + 1) = x^4 + x^2$

CRC algorithm - sender

- Select a divisor polynomial C(x)
- Represent message as polynomial M(x)
- Multiply M(x) by x^k
  - Add k zeros to message.
  - Call it T(x)
- Divide T(x) by C(x) and find the remainder
- Calculate P(x) = T(x) – remainder
  - P(x) dividable by C(x)
- Send P(x)

An example



8-bit msg

- 10011010

Divisor (3bit CRC)

- 1101

Msg sent: 10011010101

CRC algorithm - receiver

- Receive P(x) + E(x)
- Calculate (P(x) + E(x)) / C(x) and find the remainder
- If remainder is non-zero, error is detected

How to choose a divisor

- Intuition: unlikely to be divided evenly by an error
- Corrupted msg is $P(x) + E(x)$
- If $E(x)$ is single bit, then $E(x) = x^i$
- If $C(x)$ has the first and last term nonzero, then detects all single bit errors
- Find $C(x)$ by looking it up in a book

Divisor

- Six divisor polynomials that have become international standards are:
  - CRC-8 = x^8+x^2+x+1
  - CRC-10 = $x^{10}+x^9+x^5+x^4+x+1$
  - CRC-12 = $x^{12}+x^{11}+x^3+x^2+x+1$
  - CRC-16 = $x^{16}+x^{15}+x^2+1$
  - CRC-CCITT = $x^{16}+x^{12}+x^5+1$
  - CRC-32 = $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Overview

- Link layer functions
  - Encoding
    - NRZ, NRZI, Manchester, 4B/5B
  - Framing
    - Byte-oriented, bit-oriented, clock-based
  - Error detection
    - Parity, checksum, CRC
  - **Reliable transmission**
    - Error correction, stop-and-wait, sliding window

Reliable transmission

- What to do if a receiver detects bit errors?
- Two high-level approaches
  - Error correction
    - Some error codes are strong enough to correct errors.
    - The overhead is typically too high.
    - Corrupt frames must be discarded.
  - Retransmission
    - Acknowledgements and Timeouts

Acknowledgements

- An *acknowledgement* (ACK for short) is a small control frame that a protocol sends back to its peer saying that it has received the earlier frame.
  - A control frame is a frame with header only (no data).
- The receipt of an *acknowledgement* indicates to the sender of the original frame that its frame was successfully delivered.

Timeouts

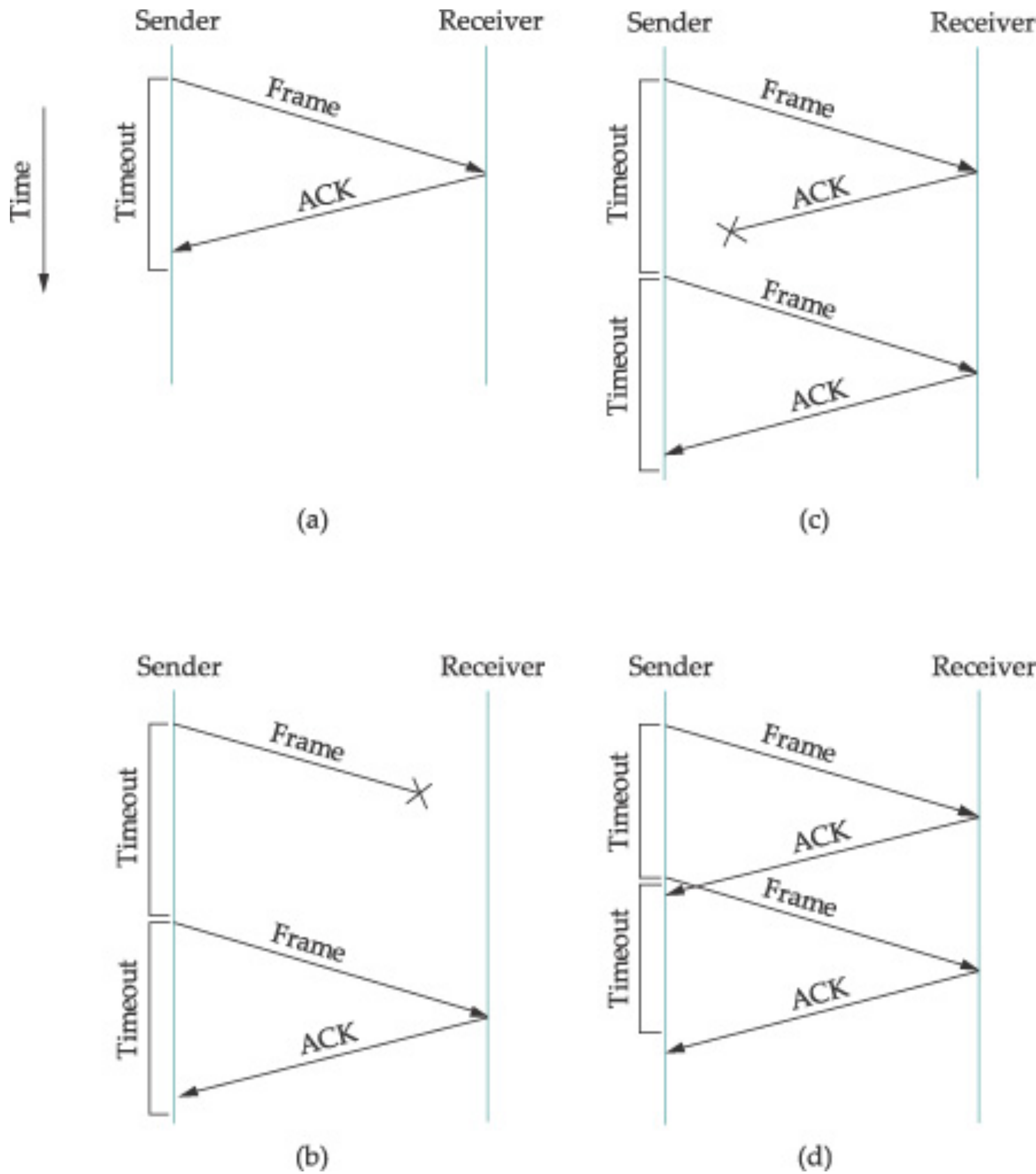
- If the sender does not receive an *acknowledgment* after a reasonable amount of time, then it retransmits the original frame.
- The action of waiting a reasonable amount of time is called a *timeout*.
- The general strategy of using *acknowledgements* and *timeouts* to implement reliable delivery is sometimes called Automatic Repeat reQuest (ARQ).

Retransmission protocols

- Stop-and-wait
- Sliding window

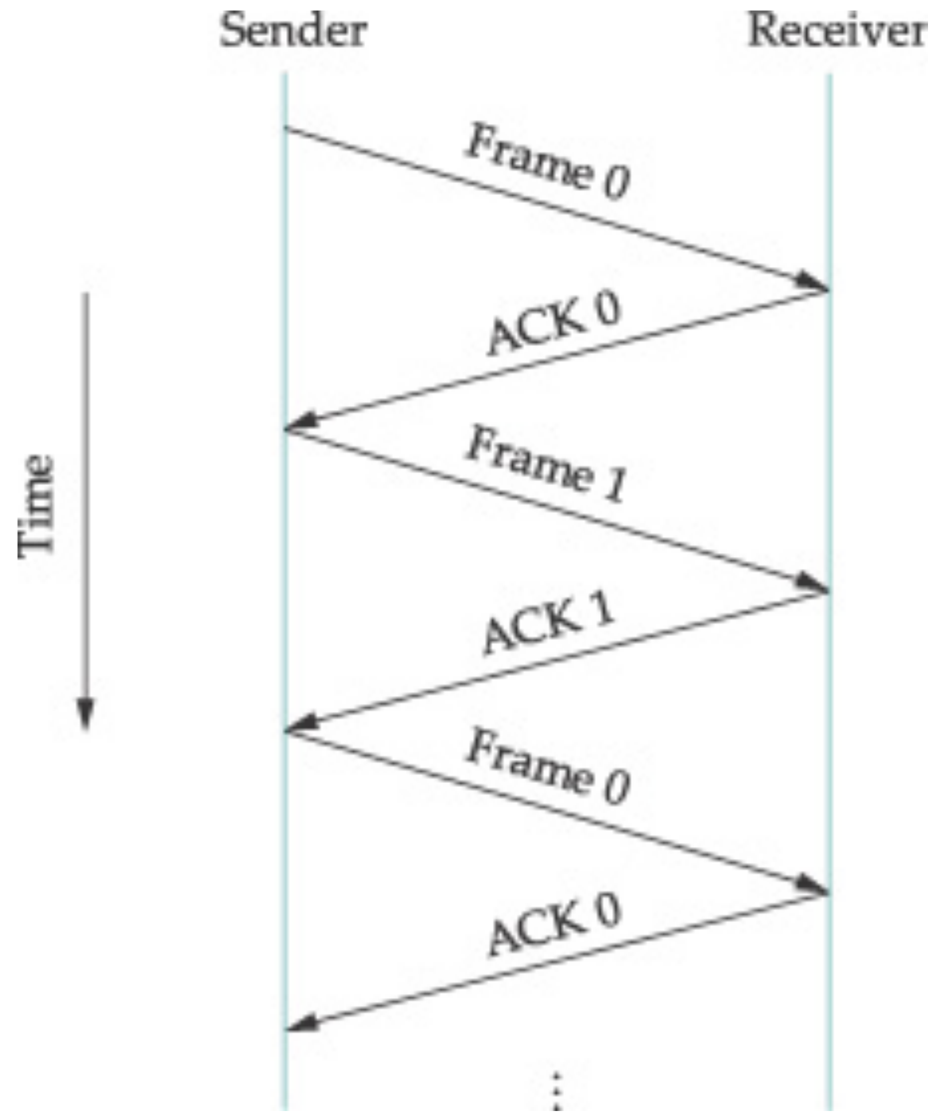
Stop-and-wait

- Send one frame, wait for an ack, and send the next
- Retransmit if times out
- Note in the last figure (d), there might be confusion: a new frame, or a duplicate?

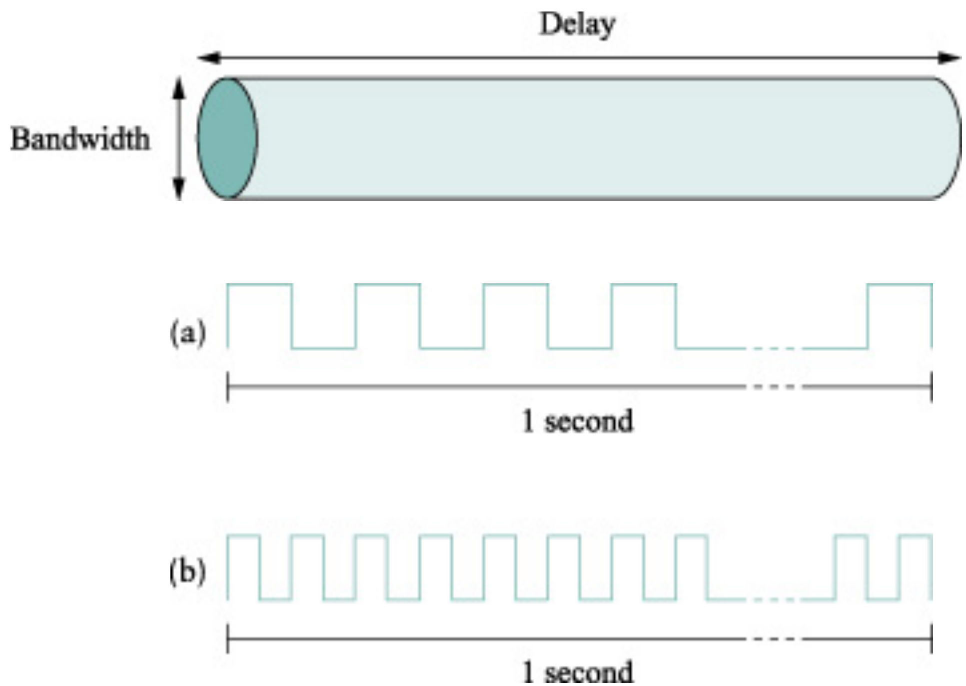


Sequence number

- Add a sequence number to each frame to avoid the ambiguity

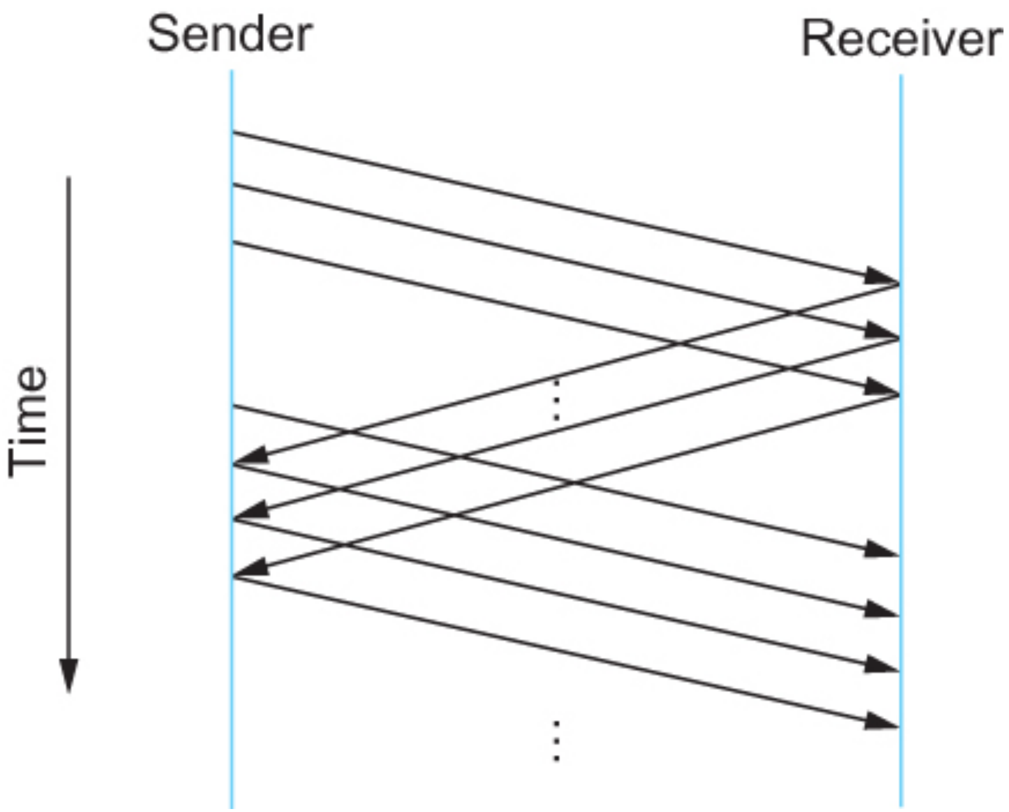


Stop-and-wait drawback



- For a 1Mbps pipe, it takes 8 seconds to transmit 1MB. If the link latency is less than 8 seconds, the pipe is full before all data are pumped into the pipe
- For a 1Gbps pipe, it takes 8 ms to transmit 1MB.
- Inefficient

Sliding Window Protocol



Timeline for Sliding Window Protocol

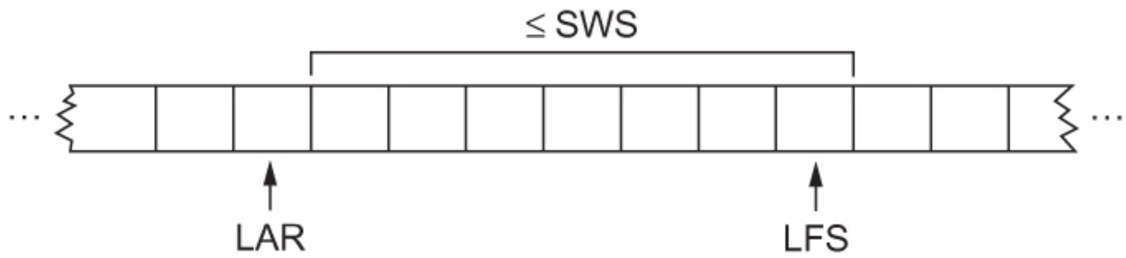
Sliding Window Protocol - Sender

- Sender assigns a sequence number denoted as SeqNum to each frame.
  - Assume it can grow infinitely large
- Sender maintains three variables
  - Sending Window Size (SWS)
    - Upper bound on the number of outstanding (unacknowledged) frames that the sender can transmit
  - Last Acknowledgement Received (LAR)
    - Sequence number of the last acknowledgement received
  - Last Frame Sent (LFS)
    - Sequence number of the last frame sent

Sliding Window Protocol - Sender

- Sender also maintains the following invariant

$$LFS - LAR \le SWS$$



Sliding Window on Sender

Sliding Window Protocol - Sender

- When an acknowledgement arrives
  - the sender moves LAR to right, thereby allowing the sender to transmit another frame
- Also the sender associates a timer with each frame it transmits
  - It retransmits the frame if the timer expires before the ACK is received
- Note that the sender has to be willing to buffer up to SWS frames

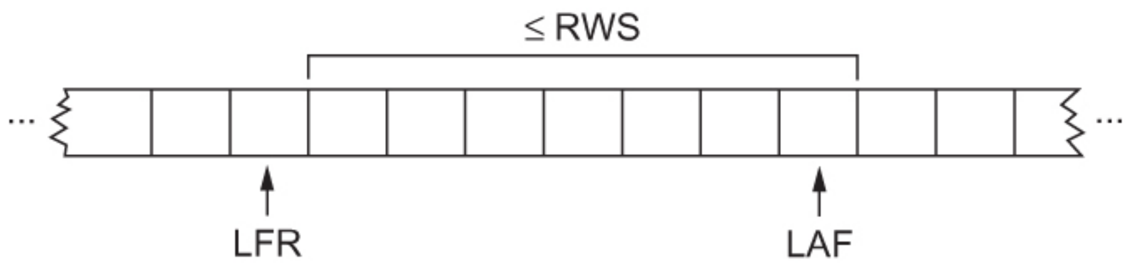
Sliding Window Protocol -- Receiver

- Receiver maintains three variables
  - Receiving Window Size (RWS)
    - Upper bound on the number of out-of-order frames that the receiver is willing to accept
  - Largest Acceptable Frame (LAF)
    - Sequence number of the largest acceptable frame
  - Last Frame Received (LFR)
    - Sequence number of the last frame received

Sliding Window Protocol -- Receiver

- Receiver also maintains the following invariant

$$\text{LAF} - \text{LFR} \leq \text{RWS}$$



Sliding Window on Receiver

Sliding Window Protocol -- Receiver

- When a frame with sequence number SeqNum arrives, what does the receiver do?
  - If SeqNum ≤ LFR or SeqNum > LAF
    - Discard it (the frame is outside the receiver window)
  - If LFR < SeqNum ≤ LAF
    - Accept it
    - Now the receiver needs to decide whether or not to send an ACK

Sliding Window Protocol -- Receiver

- Let SeqNumToAck
  - Denote the largest sequence number not yet acknowledged, such that all frames with sequence number less than or equal to SeqNumToAck have been received
- The receiver acknowledges the receipt of SeqNumToAck even if high-numbered packets have been received
  - This acknowledgement is said to be cumulative.
- The receiver then sets
  - LFR = SeqNumToAck and adjusts
  - LAF = LFR + RWS

An example

For example, suppose LFR = 5 and RWS = 4

(i.e. the last ACK that the receiver sent was for seq. no. 5)

$\Rightarrow$ LAF = 9

If frames 7 and 8 arrive, they will be buffered because they are within the receiver window

But no ACK will be sent since frame 6 is yet to arrive

Frames 7 and 8 are out of order

Frame 6 arrives (it is late because it was lost first time and had to be retransmitted)

Now Receiver Acknowledges Frame 8

and bumps LFR to 8

and LAF to 12

Issues with Sliding Window Protocol

- When timeout occurs, the amount of data in transit decreases
  - Since the sender is unable to advance its window
- When the frame loss occurs, this scheme is no longer keeping the pipe full
  - The longer it takes to notice that a frame loss has occurred, the more severe the problem becomes
- How to improve this
  - Negative Acknowledgement (NAK)
  - Additional Acknowledgement
  - Selective Acknowledgement

Issues with Sliding Window Protocol

- Negative Acknowledgement (NAK)
  - Receiver sends NAK for frame 6 when frame 7 arrive (in the previous example)
- Additional Acknowledgement
  - Receiver sends additional ACK for frame 5 when frame 7 arrives
    - Sender uses duplicate ACK as a clue for frame loss
- Selective Acknowledgement
  - Receiver will acknowledge exactly those frames it has received, rather than the highest number frames
    - Receiver will acknowledge frames 7 and 8
    - Sender knows frame 6 is lost

How to select the window size

- SWS is easy to compute
  - Delay × Bandwidth
- RWS can be anything
  - Two common setting
    - RWS = 1

      No buffer at the receiver for frames that arrive out of order
    - RWS = SWS

      The receiver can buffer frames that the sender transmits
    - It does not make any sense to keep RWS > SWS

Finite Sequence Number

- Frame sequence number is specified in the header field
  - Finite size
    - 3 bit: eight possible sequence number: 0, 1, 2, 3, 4, 5, 6, 7
  - It is necessary to wrap around

Impact on window size

- How to distinguish between different incarnations of the same sequence number?
  - Let MaxSeqNum be the number of available sequence numbers
  - SWS + 1 $\leq$ MaxSeqNum
    - Is this sufficient?

Impact on window size

SWS + 1 ≤ MaxSeqNum

- Is this sufficient?

- Depends on RWS
- If RWS = 1, then sufficient
- If RWS = SWS, then not good enough

- For example, we have eight sequence numbers

  0, 1, 2, 3, 4, 5, 6, 7

  RWS = SWS = 7

  Sender sends 0, 1, …, 6

  Receiver receives 0, 1, … ,6

  Receiver acknowledges 0, 1, …, 6

  ACK (0, 1, …, 6) are lost

  Sender retransmits 0, 1, …, 6

  Receiver is expecting 7, 0, …., 5

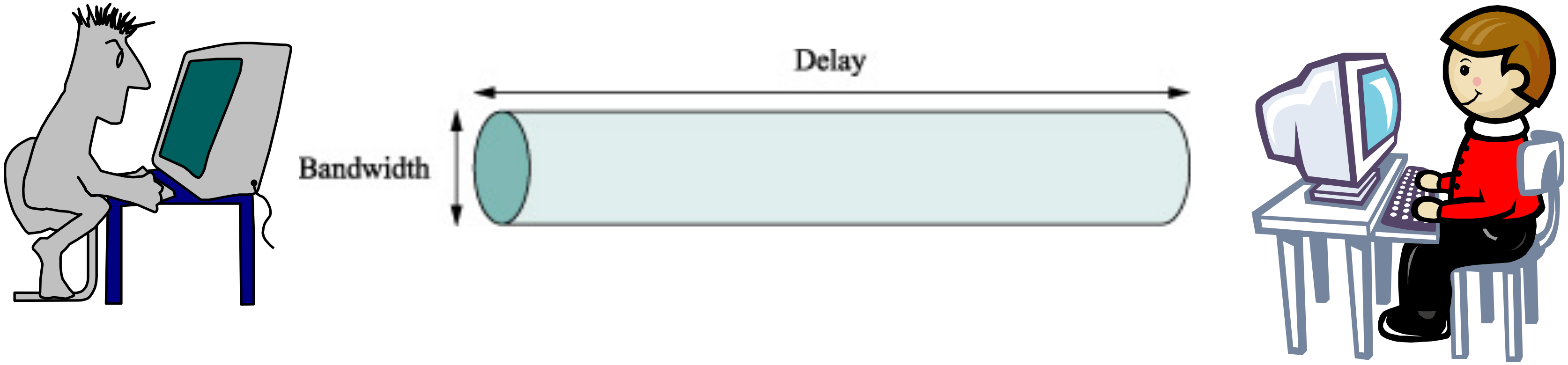
Impact on window size

To avoid this,

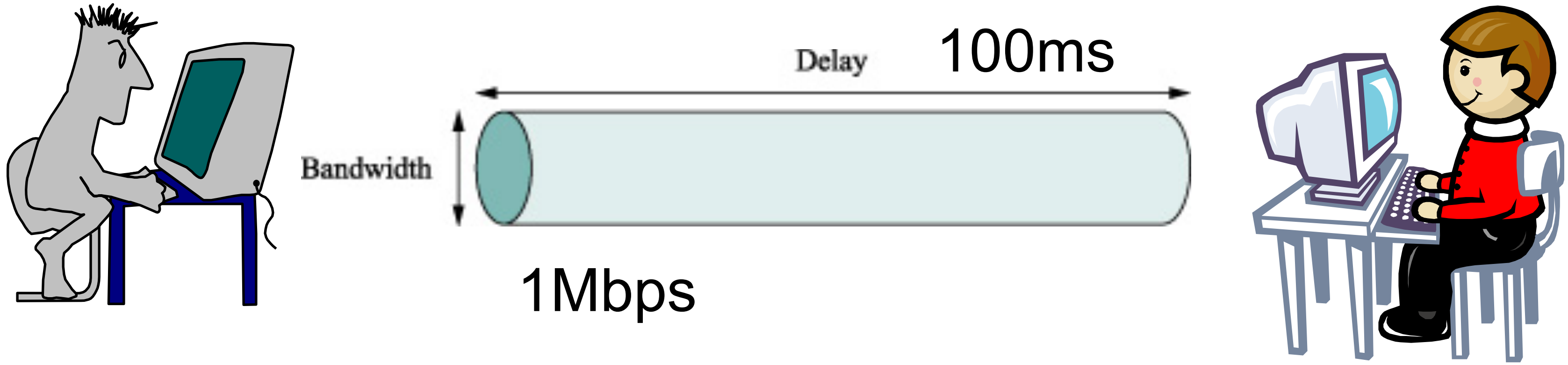
If RWS = SWS

$$SWS < (MaxSeqNum + 1)/2$$

Exercise



- Delay: 100ms; Bandwidth: 1Mbps; Frame Size: 1000 Bytes; Ack: 40 Bytes
- Q: the smallest window size to keep the pipe full?



- Window size = largest amount of unacked data
- How long does it take to ack a frame?
  - RTT = 100 ms * 2 + transmission delay of a frame (1000B) + transmission delay of an ack (40B) ~=208ms
- How many frames can the sender send in an RTT?
  - 1Mbps * 208ms / 8000 bits = 26
- Roughly 13 frames in the pipe from sender to receiver, and 13 acks from receiver to sender

Summary

- CRC
- Reliable transmission
  - Stop-and-wait
  - Sliding window