Operating Systems and Networks

Networks Part: Project 1

Network Security Group
ETH Zürich

Reliable Transport

Implement a reliable packet stream (not byte stream!)

- Packet drops
- Packet corruption
- Flow control
- Packet reordering

stdin → [Client] —— UDP —— [Server] → stdout

Fundamental Mechanisms

- Error Detection
    - Corrupt packets must be discarded
    - Implemented via Checksum

- Acknowledgements (ACK)
    - Small control packet to confirm the reception of a packet
    - When sender gets an ACK, sender learns that recipient has successfully gotten a packet

- Timeouts
    - If sender doesn't get an ACK after "reasonable" time, it retransmits the original packet

Naive Approach: Stop-and-Wait

- Algorithm
  - After transmitting one packet, sender waits for an ACK
  - If the ACK doesn't arrive in time, sender retransmits

- Disadvantage
  - Inefficient use of link's capacity

Sliding Window Protocol

- Objective: Better utilization of link bandwidth
  $\Rightarrow$ Sender is allowed to send multiple
  unacknowledged packets (how many?)

- Windows
  - Number of Unacknowledged packets are
    determined by Windows
    - Sender Window (SW)
    - Receiver Window (RW)
  - Requirement: Need to keep sender's and
    receiver's windows synchronized (how?)

Sliding Window: Sender

- Assigns sequence number to each frame (seqno)
- Maintain three state variables:
  - Send Window Size (SWS): max # of unacknowledged frames that sender can transmit
  - Last Acknowledgement Received (LAR): seqno of last ACK
 - Last Frame Sent (LFS)
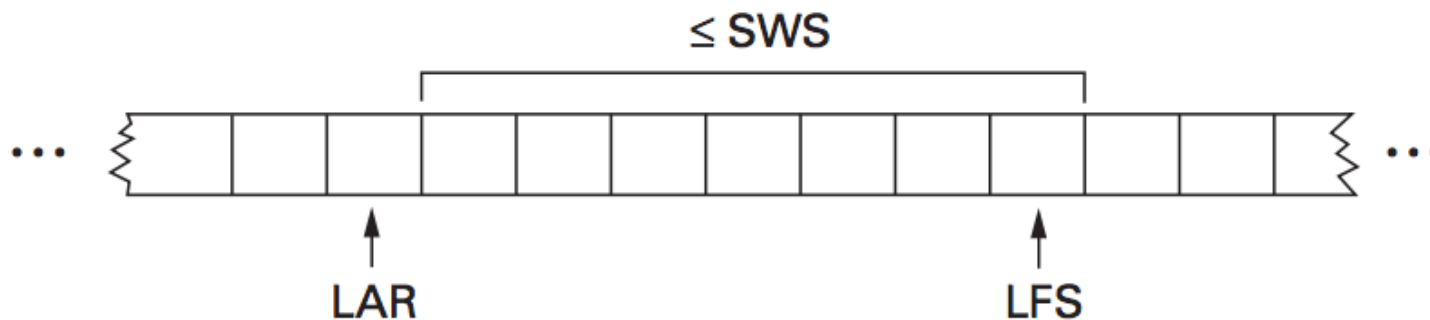
Sliding Window: Sender Invariant
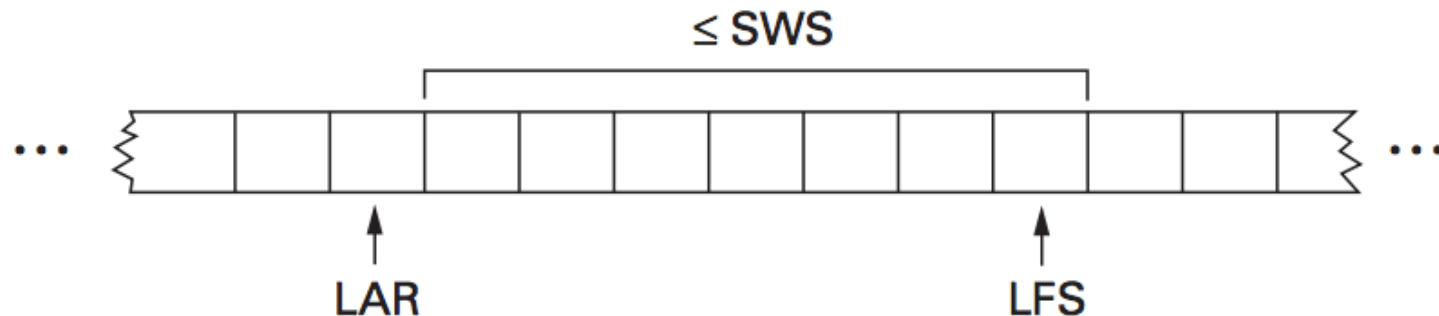
- Maintain invariant: LFS – LAR ≤ SWS
- Buffer up to SWS unacknowledged packets
- Associates timeout with each frame sent
    - Retransmits if no ACK received before timeout
- Advance LAR when ACK arrives
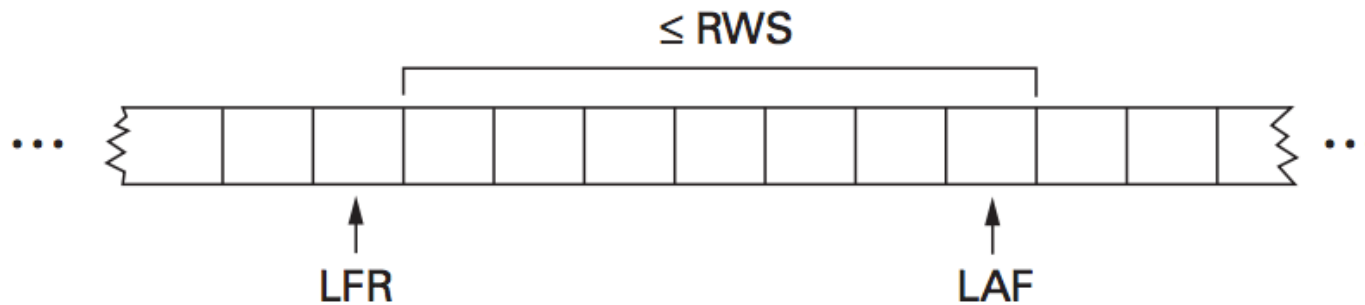   - Another frame can be sent

Sliding Window: Receiver

- Maintain three state variables:
  - Receive Window Size (RWS): max # of out-of-order frames it will accept
  - Last Acceptable Frame (LAF)
  - Last Frame Received (LFR)
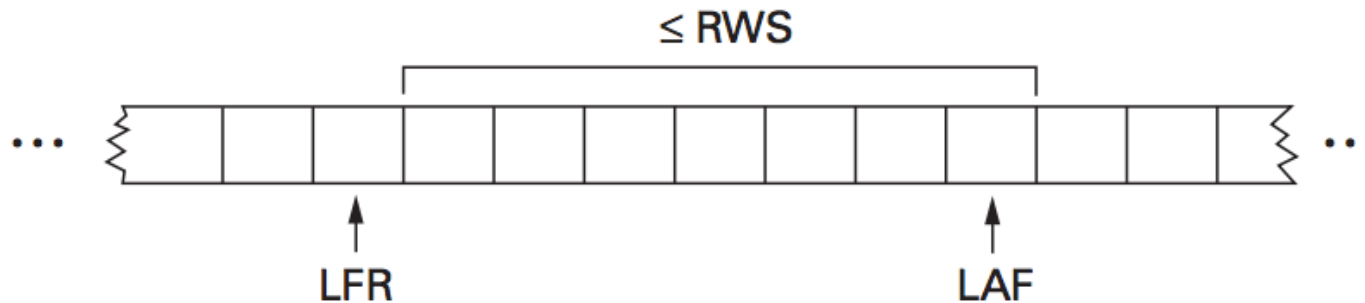
Sliding Window: Receiver Invariant

- Maintain invariant: LAF − LFR ≤ RWS
- When frame #seqno arrives:
    - if LFR < seqno ≤ LAF accept
    - if seqno ≤ LFR or seqno > LAF discard
- Receiver ACKs the next seqno it's expecting (CumACK)
    - LFR = CumACK − 1
    - LAF = CumACK + RWS-1