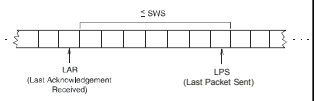
Flow Control

- Goal: control the flow of packets on the link so that receiver always has sufficient buffers to accept them until they can be processed
- Sliding Window:
  - Imposes a limit on the number of outstanding unACKed packets, i.e. length of retransmission list, called ***send window***
  - For stop-and-wait, send window = 1 → poor link utilization
  - The size of the send window is chosen to achieve ***both*** high link utilization and flow control
  - Send Window Size =
    MIN(delay x bandwidth, available buffer space at receiver)

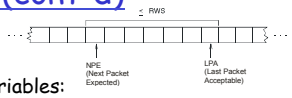
Sliding Window



Sender:

- Assign sequence number to each packet (**SeqNum**)
- Maintain three state variables:
  - send window size (**SWS**)
  - last acknowledgment received (**LAR**)
  - last packet sent (**LPS**)
- Maintain invariant: **LPS – LAR ≤ SWS**
- When ACK arrives, advance **LAR**, thereby opening window
- Buffer up to **SWS** packets

Sliding Window (cont'd)



Receiver:

- Maintain three state variables:
  - receive window size (**RWS**)
  - last packet acceptable (**LPA**)
  - next packet expected (**NPE**)
- Maintain invariant: **LPA - NPE + 1 ≤ RWS**
- Packet **SeqNum** arrives:
  - if NPE ≤ SeqNum ≤ LPA → accept
  - if SeqNum < NPE or SeqNum > LPA → discarded
- Send ACK/NAK

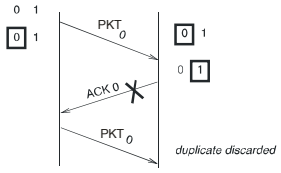
Sliding Window (cont'd)

- With Go-Back-N, **RWS** = 1
- With Selective Repeat, **RWS** = **SWS**.

  Receiver can then maintain sequence numbers of packets that the sender can send, and so can detect whether a received packet is new or duplicate

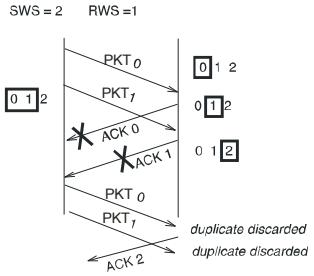
Sequence Numbers

- **SeqNum** field is finite; sequence numbers wrap around
- The size of the sequence number space must be larger than the number of outstanding packets
- Stop-and-Wait: sequence numbers {0, 1}



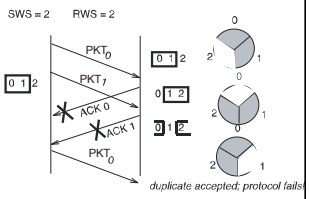
Sequence Numbers (cont'd)

- Go-Back-N: sequence numbers {0, 1, … , **SWS**}



Sequence Numbers (cont'd)

- Selective Repeat: sequence numbers {0, 1, … , **SWS**} is not sufficient
- Size of sequence number space must be at least **SWS+RWS = 2 SWS**
- Intuitively, **SeqNum** ``slides'' between two halves of sequence number space

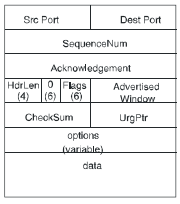


End-to-End Challenges

Based basically on go-back-n sliding window protocol, but it's challenging!

- Potentially different RTT
  - need adaptive timeout mechanism
- Potentially long delay in network
  - need to be prepared for arrival of very old packets
- Potentially different buffering at destination
  - need to accommodate different amounts of buffering
- Potentially different network capacity
  - need to be prepared for network congestion

TCP Segment Format



- Demultiplexing; each connection identified with 4-tuple:
  - **<SrcPort, SrcIPAddr, DstPort, DstIPAddr>**
- Basically, Go-back-n sliding window operating at byte (not segment) level
  - Acknowledgment, SequenceNum, AdvertisedWindow
  - Piggybacking ACK on data segments destined for sender improves network utilization
- Flags
  - SYN, FIN, RESET, PUSH, URG, ACK

TCP Reliability & Flow Control

- In practice:
  - storing ***out-of-order*** bytes
  - using ***one timer*** for all unacked bytes
  - using ***duplicate ACK*** to fast retransmit
  - On retransmission, ***only one segment retransmitted***
- A new version, SACK, is more like Selective-repeat
- At sender:
  - ***LastByteSent – LastByteAcked ≤ AdvertizedWindow***
  - If zero, sender keeps sending 1-byte data segments