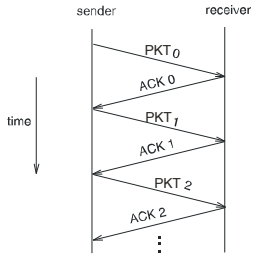
Stop-and-Wait (Idle RQ)

- Sender sends only one packet at a time



Stop-and-Wait (cont'd)

- Problem: Keeping the pipe full (*i.e.* maintain high link utilization)
- Example: Assuming packet size of 1KB, 1.5Mbps link, 40ms RTT
- BxD ~ 8 packets.

  Stop-and-wait uses about 1/8 of the link's capacity. Want the sender to be able to transmit up to 8 packets before having to wait for an ACK

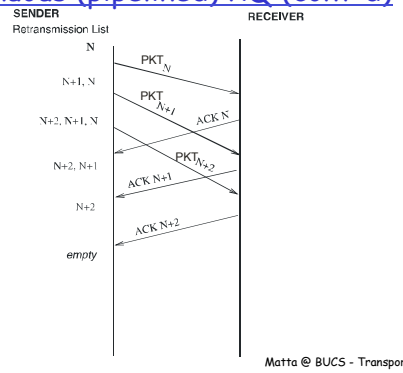
  What is the effective throughput?

  Answer: about 0.2 Mbps!!

Continuous RQ (pipelining)

- Achieves higher link utilization than stop-and-wait
- Sender sends multiple packets without waiting for an ACK
- In practice, there is a limit for flow control
- Sender needs more memory to buffer outstanding unacked packets

Continuous (pipelined) RQ (cont'd)



Pipelined RQ (cont'd)

Two retransmission strategies:

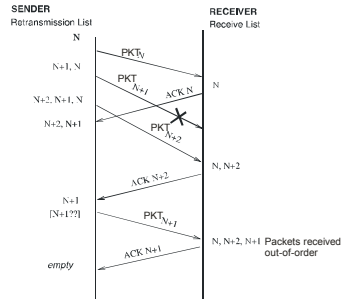
- Selective Repeat: Only corrupted/lost packets are retransmitted
- Go-Back-N: Packets received correctly may be retransmitted

Selective Repeat

- Packets transmitted continually (when available) without waiting for ACK, up to K outstanding unACKed packets
- A different sender timer associated with each unACKed packet
- Receiver:
  - ignores (implicit retransmission) or NAKs (explicit retransmission) missing/corrupted packets
  - ACKs correct (possibly out-of-order) packets
  - buffers out-of-order packets so as to deliver packets in-order to higher layer
- Sender:
  - on timeout or NAK for packet N, or ACK for packet > N, just retransmit N

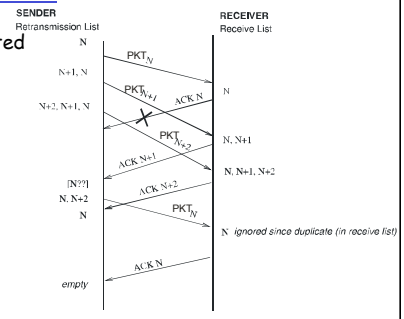
Selective Repeat, Implicit Retransmission

- Example: lost/ corrupted packet



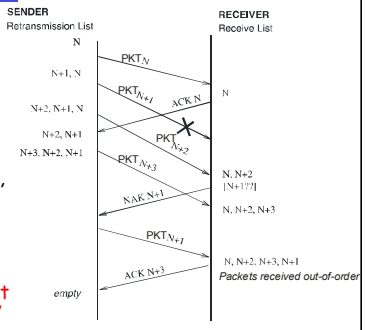
Selective Repeat, Implicit Retransmission

- Lost / corrupted ACK



Selective Repeat, Explicit Retransmission

- Example: lost / corrupted packet
- ACK for packet N implicitly acknowledges up through N (i.e. ***cumulative*** ACK)
- While in NAK state, receiver does not ACK - why?
- A timer associated with NAK?
- What happens if pkt N+3 also gets lost / corrupted?

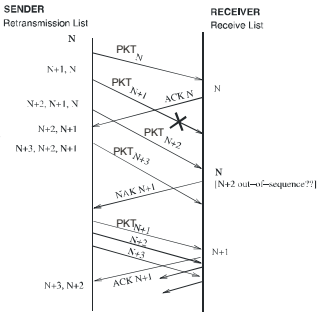


Go-Back-N

- Unlike Selective Repeat, Go-Back-N saves receiver buffering by requiring packets to arrive in-order
- As in Selective Repeat:
  - Packets transmitted continually (when available) without waiting for ACK, up to K outstanding unACKed packets
  - A different sender timer associated with each unACKed packet, although a single timer implementation for Go-Back-N is common
  - Receiver ignores or NAKs missing/corrupted packets
- Unlike Selective Repeat:
  - Receiver ACKs only correctly received and in-order packets, passes them to higher layer
  - On timeout or NAK for packet N, sender retransmits from N all over again (all outstanding packets)

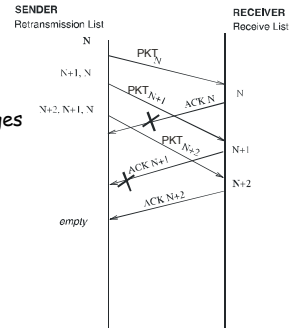
Go-Back-N (cont'd)

- Example: lost / corrupted packet
- A timer associated with NAK, or if not and the NAK is lost, what will happen?



Go-Back-N (cont'd)

- Example: lost / corrupted ACK
- ACK for packet N implicitly acknowledges up through N (i.e. ***cumulative*** ACK)



Pros and Cons of Go-Back-N

- No receiver buffering with Go-Back-N
- Saves resources at receiver
- Avoids large bursts of packet delivery to higher layers
- Simplicity in buffering and protocol processing at sender and receiver, e.g. can easily detect duplicates if an out-of-sequence packet is received
- Consumes more capacity by retransmitting correctly received packets
- Tradeoff between buffering/processing complexity and capacity