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 (per-packet) RTT

$$K_{ideal} = RTT * C = \left\lceil \overset{5}{\cancel{40}}\,\cancel{m}s * \frac{1.5\cancel{M}bps}{1\cancel{K}\cancel{B} * \cancel{8}} \right\rceil = \lceil 7.5 \rceil \text{ packets} = 8$$

$$U_{S\&W} = \frac{1}{8}$$

$$Throughput_{S\&W} = \frac{1}{8} \cdot 1.5\,Mbps \simeq 0.2\,Mbps$$

20

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 (per-packet) 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: 1.5M/8 ~ 0.2 Mbps!!

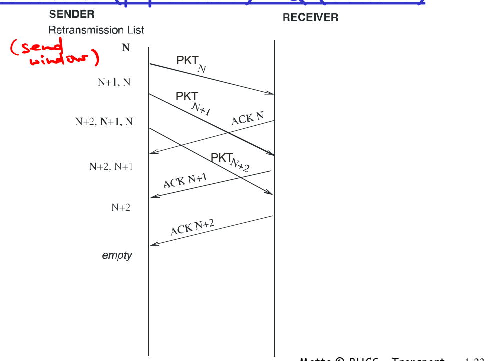
21

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

22

Continuous (pipelined) RQ (cont'd)



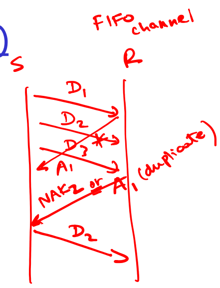
Matta @ BUCS - Transport 1-23

23

Pipelined RQ (cont'd)

Two retransmission strategies:

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



- NAK or duplicate ACK to improve utilization

Matta @ BUCS - Transport 1-24

24

Selective Repeat

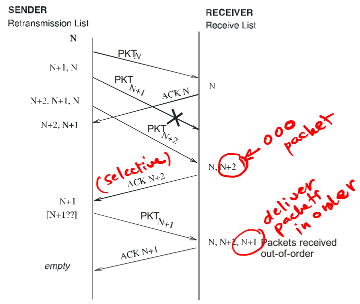
- Packets transmitted continually (when available) without waiting for ACK, up to K outstanding unACKed packets
- A different sender timer associated with each unACKed packet
- Receiver:
  - ignores (implicit retransmission) or NAKs (explicit retransmission) missing packets
  - ACKs correct (possibly 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

Matta @ BUCS - Transport 1-25

25

Selective Repeat, Implicit Retransmission

- Example: lost/corrupted packet

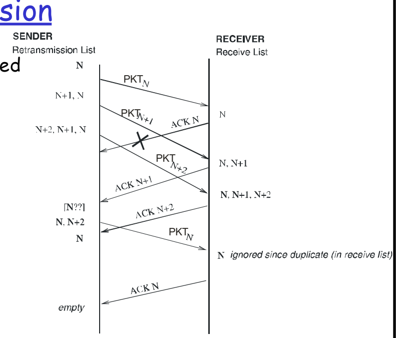


Matta @ BUCS - Transport 1-26

26

Selective Repeat, Implicit Retransmission

- Lost / corrupted ACK

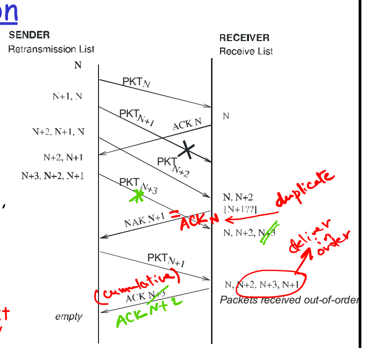


Matta @ BUCS - Transport 1-27

27

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?



Matta @ BUCS - Transport 1-28

28

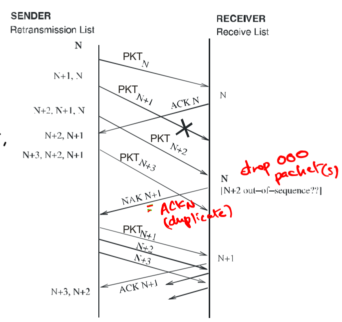
Go-Back-N

- 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 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)

29

Go-Back-N (cont'd)

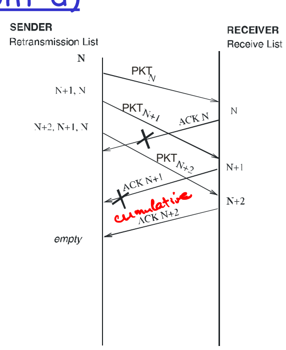
- Example: lost / corrupted packet
- A timer associated with NAK, or if not and the NAK is lost, what will happen?



30

Go-Back-N (cont'd)

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



31

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 link capacity by retransmitting correctly received packets
- Tradeoff between host buffering/processing complexity and link capacity

Matta @ BUCS - Transport 1-32

32

Flow Control

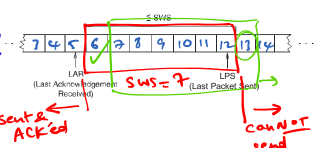
- Goal: control the flow of packets on the link so that receiver always has sufficient buffers to accept them until they can be processed
- Sliding Window:
  - Imposes a limit on the number of outstanding unACKed packets, i.e. length of retransmission list, called ***send window***
  - For stop-and-wait,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

    B×D efficiency   flow control

Matta @ BUCS - Transport 1-33

33

Sliding Window



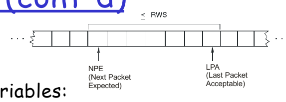
Sender:

- 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

Matta @ BUCS - Transport 1-34

34

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

35

Sliding Window (cont'd)

SWS = k

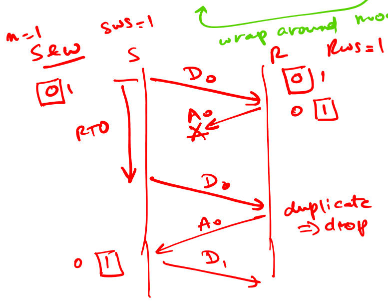
- With Go-Back-N, **RWS** = 1
- With Selective Repeat, **RWS** = **SWS**. Receiver can then maintain sequence numbers of packets that the sender can send, and so can detect whether a received packet is new or duplicate

36

Sequence Numbers

m-bit seq no. $\{0, 1, \ldots, 2^n - 1\}$

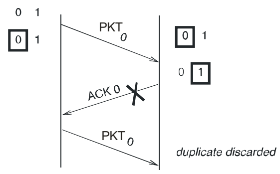
wrap around modulo 2^n



37

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}



Matta @ BUCS - Transport 1-38

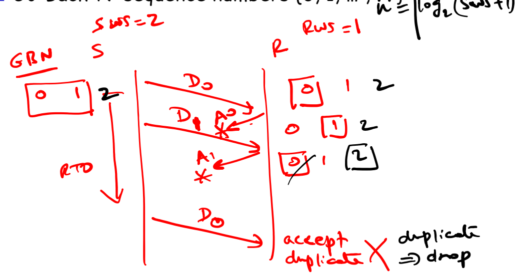
38

Sequence Numbers (cont'd)

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

min # seq nos = SWS + 1

$n \geq \lceil \log_2(SWS+1) \rceil$

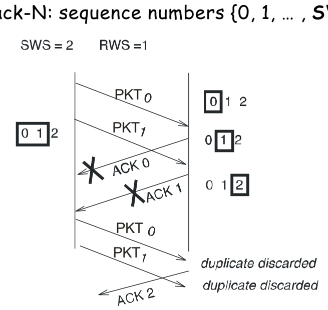


Matta @ BUCS - Transport 1-39

39

Sequence Numbers (cont'd)

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



Matta @ BUCS - Transport 1-40

40

Sequence Numbers (cont'd)

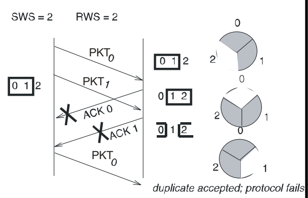
- Selective Repeat: sequence numbers {0, 1, … , **SWS**} is not sufficient

Matta @ BUCS - Transport 1-41

41

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



Matta @ BUCS - Transport 1-42

42