Announcements:

- Assignment 1 Solutions Posted
So far: rdt3.0

- Acknowledgment and retransmission
  - Reliable delivery
- Sequence numbers
  - Duplication detection
- Timeout and retransmit
  - Deal with packet losses
- Stop-n-wait
  - one packet at a time
- Problem: efficiency
Performance of rdt3.0: stop-n-wait

- rdt3.0 works, but performance is terribad
- example: $R=1$ Gbps, $RTT=30$ ms, $L=1000$Byte packet:

\begin{align*}
U_{sender} & : \text{utilization} - \text{fraction of time sender busy sending} \\
U_{sender} & = \frac{\frac{L}{R}}{RTT + \frac{L}{R}} = \frac{0.008}{30.008} = 0.00027
\end{align*}
Performance of rdt3.0: stop-n-wait

- rdt3.0 works, but performance stinks
- example: R=1 Gbps, RTT=30 ms, L=1000Byte packet:

- 1KB pkt every 30 msec -> 33kB/sec throughput over 1 Gbps link
- network protocol limits use of physical resources!
Pipelined protocols

Pipelining: sender allows multiple, “in-flight”, yet-to-be-ACK’ed pkts

- What about the range of sequence numbers then??
- What about buffering at receiver??

Two generic forms of pipelined protocols:

*go-Back-N* and *selective repeat*
Go-Back-N

Sender side
- 'N' packets can be sent w/o being ACK'ed,
- Fills “sliding window”
- Single timer: oldest non-ACK'ed packet
- At timeout(n), retransmit packet n and all successive packets in window

Receiver side
- Cumulative ACK
- Discard out-of-order packets, no buffering
Selective Repeat

Sender side
- 'N' packets can be sent w/o being ACK'ed,
- Fills “sliding window”
- 'N' timers: every packet in window
- At timeout(n), retransmit packet n
- ACK for bottom of window? Advance window to the next un-ACK'ed packet

Receiver side
- Selective ACK
- ACK and Buffer out-of-order segments (if in window)
- Discard out-of-order packets (if ahead of window)
- ACK and Discard out-of-order packets (if behind window)
- In-order packet? ACK and deliver, then deliver all sequential buffered packets and advance window!
Selective repeat in action

pkt0 sent
0 1 2 3 4 5 6 7 8 9
pkt1 sent
0 1 2 3 4 5 6 7 8 9
pkt2 sent
0 1 2 3 4 5 6 7 8 9
pkt3 sent, window full
0 1 2 3 4 5 6 7 8 9

pkt0 rcvd, delivered, ACK0 sent
0 1 2 3 4 5 6 7 8 9
pkt1 rcvd, delivered, ACK1 sent
0 1 2 3 4 5 6 7 8 9

pkt3 rcvd, buffered, ACK3 sent
0 1 2 3 4 5 6 7 8 9

ACK0 rcvd, pkt4 sent
0 1 2 3 4 5 6 7 8 9
ACK1 rcvd, pkt5 sent
0 1 2 3 4 5 6 7 8 9

pkt2 TIMEOUT, pkt2 resent
0 1 2 3 4 5 6 7 8 9

pkt4 rcvd, buffered, ACK4 sent
0 1 2 3 4 5 6 7 8 9
pkt5 rcvd, buffered, ACK5 sent
0 1 2 3 4 5 6 7 8 9
pkt2 rcvd, pkt2, pkt3, pkt4, pkt5 delivered, ACK2 sent
0 1 2 3 4 5 6 7 8 9

ACK3 rcvd, nothing sent
0 1 2 3 4 5 6 7 8 9
GBN/SR comparison

Go-Back-N
- Sliding window
  - N packets in the pipe
- Cumulative ACK
- Single timer (oldest un-ACK'ed packet)
- Discard out-of-order packets
- Retransmit all successive packets at timeout

Selective Repeat
- Sliding window
  - N packets in the pipe
- Selective ACK
- One timer for each packet
- Buffer out-of-order packets (within window)
- Retransmit only lost or delayed packet at timeout
Selective repeat: dilemma

Example:
- Seq #’s: 0, 1, 2, 3
- Window Size = 3

- Receiver sees no difference in two scenarios! Even though,
  - (a) is a retransmitted packet
  - (b) is a new packet
- In (a), receiver incorrectly passes old data as new

Q: What relationship between sequence # size and window size to avoid duplication problem??
Duplication problem: illustration

- Let’s assume window size $W = 2$ for illustration.
- Consider a scenario where:
  Only ACK 1 is lost
- Next, assume the Sequence Number (SN) space $n = W = 2$
  That is, SN pattern is $0, 1, 0, 1, 0, 1, ...$
**Duplication problem: illustration**

Scenario: $n=W=2$; only ACK 1 is lost

Receiver thinks of retransmission of 2nd Packet (Pkt1, SN=1) as a 4th/new Pkt (Pkt3,SN=1) ➔ Duplication detection problem!
Duplication problem: illustration

Recap:

- When window size ($W$) = Sequence Number space ($n$),
  - there is a duplication problem
  - receiver can’t tell a new transmission from a retransmission of a lost packet

- Let’s now see what happens if we increase $n$ by 1:
  - Now $n=3$ and SN pattern is: 0,1,2,0,1,2,0...

- Let’s revisit the scenario again
**Duplication problem: illustration**

**Scenario:** \( n=3; W=2; \) ACK 1 is lost

---

Receiver still thinks of retransmission of 2nd Packet (Pkt1, SN=1) as a new Pkt (Pkt3, SN=1)

\( \Rightarrow \) Increasing \( n \) by 1 didn’t solve the duplication detection problem this time!!

Chapter 3, slide: 47
Recap:

- Increasing SN space $n$ from 2 to 3 didn’t solve the duplication problem yet!

- Let’s now see what happens if we again increase $n$ by 1: Now $n=4$ and SN pattern is: $0,1,2,3,0,1,2,3,0…$

- Let’s revisit the scenario with $n=4$ and see
Scenario: \( n=4; W=2; \) ACK 1 is lost

Receiver now drops this retransmission of 2\(^{nd}\) Packet (Pkt1, SN=1) since its SN falls outside the window

\( \Rightarrow \) Duplication detection problem is now solved when \( n=2W=4 \)

Chapter 3, slide: 49
Recap:

- Increasing SN space $n$ from 2 to 4 did solve the duplication problem.

- That is, when SN space = twice window size ($n = 2W$) the duplication problem was solved.

- Hence, in this case:
  
  when $W \leq n/2$, the problem is solved.

- Can we generalize?
  
  Yes, $W \leq n/2$ must hold in order to avoid duplication problem.
Suppose \( \frac{1}{2} n < W < n \)

- \( W \) segments with SN in \([0, W-1]\) are sent, but their ACKs are not received yet

\( \text{pktts already sent, but ACKs not received yet} = \text{sender's sliding window} \)
Receiver: Now assume:

- receiver received all $W$ segments ($SN$ in $[0,W-1]$)
- receiver sent all $W$ ACKs, one for each received segment
- receiver slides its window: Sliding window ($SN$ of expected segments) from: $W$ to $x = 2W-n-1$

Hint: $(n-1-W+1) + (x-0+1) = W \Rightarrow x = 2W-n-1$
Consider worst case scenario: all $W$ ACKs are lost.

Sender will re-transmit all the first $W$ segments; i.e., with SN in $[0,W-1]$.

But receiver is expecting segments with SN in $[W,n-1] \cup [0,2W-n-1]$.

Retransmissions with SN falling within $[0,2W-n-1]$ will then be interpreted by receiver as new transmissions (⇒ dup problem).

To avoid dup problem, we must then have:

$$2W-n-1 \leq -1 \Rightarrow W \leq \frac{1}{2} n$$