Reliable Data Transfer

Kurose & Ross, Chapter 3.4 (5th ed.)

Many slides adapted from:
J. Kurose & K. Ross
Computer Networking: A Top Down Approach (5th ed.)
Addison-Wesley, April 2009.
Copyright 1996-2010, J.F Kurose and K.W. Ross, All Rights Reserved.

Exercise (Kurose & Ross, 5th ed.)

- rdt 3.0 is correct only under a FIFO channel assumption.
  - Correct = guarantees reliable transmission. Data sent by sender is exactly the data reconstructed in the receiver side.
  - Show a case where a non-FIFO channel (i.e., one that can cause packet reordering) causes rdt 3.0 to deliver incorrect data.
Exercise (Kurose & Ross, 5th ed.)
- The sender of rdt 3.0 simply ignores all received packets that are either in error or have the wrong value in the acknum field of an ack packet.
- Suppose that in such circumstances, rdt 3.0 were simply to transmit the current data packet.
- Would the protocol still work?
- Would it be more or less efficient than before?

Exercise (Kurose & Ross, 5th ed.)
- Would it be more or less efficient than before?
- Depends on the length of the sender timeout, compared to the expected channel delay.
- If the timeout is very long, then the immediate retransmit can save us the long wait until the timeout expires.
- However, premature timeouts can cause a pathologies.

Exercise (Kurose & Ross, 5th ed.)
- Would the protocol still work?
- Yes. A retransmission is exactly what would happen if the ack was completely lost instead of garbled.
- The receiver can't even distinguish between the two events.

Exercise (Kurose & Ross, 5th ed.)
- Would it be more or less efficient than before?
- It depends on the length of the sender timeout, compared to the expected channel delay.
- If the timeout is very long, then the immediate retransmit can save us the long wait until the timeout expires.
- However, premature timeouts can cause a pathologies.

Performance of rdt3.0
- rdt3.0 works, but performance stinks
- ex: 1 Gbps link, 15 ms prop. delay, 8000 bit packet:
  \[ d_{max} = \frac{L}{R} = \frac{8000\text{bits}}{10^6\text{bps}} = 8\text{microseconds} \]
- \[ U_{sender} = \frac{L}{RTT + L/R} = \frac{0.008}{30.008} = 0.00027 \]
- \[ U_{sender} = \frac{L}{RTT + L/R} = \frac{0.008}{30.008} = 0.00027 \]
- if RTT=30 msec, 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-acknowledged pkts
- range of sequence numbers must be increased
- buffering at sender and/or receiver

+ two generic forms of pipelined protocols: go-Back-N, selective repeat

Pipelining: increased utilization

Go-Back-N

Sender:
- k-bit seq # in pkt header
- "window" of up to N, consecutive unack'ed pkts allowed
- sender can have up to N unack'ed packets in pipeline
- rcvr only sends cumulative acks
  - doesn't ack packet if there's a gap
- sender has timer for oldest unack'd packet
  - if timer expires, retransmit all unack'ed packets

Selective Repeat: big pic
- sender can have up to N unack'ed packets in pipeline
- rcvr sends individual ack for each packet
- sender maintains timer for each unack'ed packet
  - when timer expires, retransmit only unack'ed packet

GBN: sender extended FSM

GBN: receiver extended FSM

ACK-only: always send ACK for correctly-received pkt with highest in-order seq #
- may generate duplicate ACKs
- need only remember expected seq num
- out-of-order pkt:
  - discard (don't buffer) → no receiver buffering!
  - Re-ACK pt with highest in-order seq #
Selective Repeat

- Receiver individually acknowledges all correctly received packets.
- Buffers packets, as needed, for eventual in-order delivery to the upper layer.
- Sender only resends packets for which ACK not received.
- Sender timer for each unACKed packet.
- Sender window:
  - $N$ consecutive sequence numbers.
  - Again limits sequence numbers of sent, unACKed packets.

Selective Repeat: Sender, Receiver Windows

Sender view of sequence numbers:
- Window size $N$.
- Window base $send_base$.
- New window size $new_size$.
- Already acknowledged and sent, not yet acknowledged.
- Usable, not yet sent.
- Not usable.

Receiver view of sequence numbers:
- Window size $N$.
- Window base $recv_base$.
- Out of order (buffered) but already acknowledged.
- Expected, not yet received.
- Acceptable (within window).
- Not usable.

Selective Repeat in Action

Example:
- Sequence numbers: 0, 1, 2, 3.
- Window size: 3.
- Receiver sees no difference in two scenarios.
- Incorrectly passes duplicate data as new.
- What relationship between sequence number size and window size?
Minimal sequence range

- Assume we want to use a sender window of size \( N \).
- What is the minimal number of unique sequence numbers we should allow to prevent such errors?
- The cyclic sequence number should never cause the sender and receiver’s window to ambiguously overlap.
- In FIFO channels:
  - GBN: \( N + 1 \)
  - SR: \( 2N \)
  - Proof: on-board

Exercise (Kurose & Ross, 5th ed.)

- Are the following statements true or false?
  - With SR, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.
    - True. Suppose sender has a window size of 3.
      - Time \( t_1 \): it sends packets 1, 2, 3.
      - Time \( t_2 > t_1 \): sender times out and retransmits 1, 2, 3.
      - Time \( t_3 > t_2 \): receiver gets the duplicates and retransmits 1, 2, 3.
      - Time \( t_4 > t_3 \): sender gets the ack sent at \( t_2 \), advances its window to 4, 5, 6.
      - Time \( t_5 > t_4 \): sender receives the acks sent at \( t_2 \), that fall outside of its current window.
    - With GBN, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.
      - True, with the same scenario as described above. Only need to replace the selective acks with cumulative acks.

Exercise (Kurose & Ross, 5th ed.)

- Are the following statements true or false?
  - rtd 3.0 is the same as SR with a sender and receiver window size of 1.
    - Both are true. With a window size of 1, SR, GBN, and the rtd 3.0 are functionally equivalent.
      - The window size of 1 precludes the possibility of out-of-order packets (within the window).
      - A cumulative ACK is just an ordinary ACK in this situation, since it can only refer to the single packet within the window.
  - rtd 3.0 is the same as GBN with a sender and receiver window size of 1.
    - Incorrect. Let the sender send packets 1, \( \ldots \), \( m - 1 \). All received correctly, but all acks are lost.

Exercise

- Recall the GBN receiver: assume it is waiting for packet \( m \) (i.e., it received all the packets up to \( m - 1 \) inclusive).
  - If \( n < m \) the receiver discards the packet and does not send an ack. Otherwise, operate as before.
    - Correct. If \( n > m \) was received, but the receiver is waiting for \( m \), it means we have a gap. The sender will eventually timeout for \( m \), and resend packet \( n \) then.
  - Deadlock.
  - Incorrect. Let the sender send packets 1, \( \ldots \), \( m - 1 \). All received correctly, but all acks are lost.
    - The receiver waits for packet \( m \).
    - But whenever the sender times-out expires, it resends packets 1, \( \ldots \), \( m - 1 \).
    - Receiver discards them and does not ack.