Transport Layer III

Dmitri Loguinov
Texas A&M University

October 10, 2023
Chapter 3: Roadmap

3.1 Transport-layer services
3.2 Multiplexing and demultiplexing
3.3 Connectionless transport: UDP
3.4 Principles of reliable data transfer (cont)
3.5 Connection-oriented transport: TCP
   - Segment structure
   - Reliable data transfer
   - Flow control
   - Connection management
3.6 Principles of congestion control
3.7 TCP congestion control
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 and Selective Repeat
Pipelining: Increased Utilization

\[ U_{\text{sender}} = \frac{3 \cdot L / R}{\text{RTT} + L / R} = \frac{0.024}{30.008} = 0.0008 \]

Increases utilization by a factor of 3!
Go-Back-N (GBN)

Sender:
- **Window** of up to $N$ consecutive unack’ed pkts allowed
- A field in header that holds $k$ unique seq numbers

- ACK(n): ACKs all consecutive pkts up to & including seq # n (cumulative ACK)
  - Means packets 1…n have been delivered to application
- Timer for the oldest unacknowledged pkt (send_base):
  - Upon timeout: retransmit all pending pkts in current window (yellow in the figure); reset the timer
GBN: Sender Extended FSM

```c
rdt_send(data)

if (nextseqnum < base+N) {
    sndpkt[nextseqnum] = make_pkt(nextseqnum, data, chksum)
    udt_send(sndpkt[nextseqnum])
    if (base == nextseqnum) start_timer
    nextseqnum++
}
else refuse_data(data)

```
GBN: Receiver Extended FSM

- **ACK-only**: always send ACK for correctly-received pkt with highest *in-order* seq #
  - Duplicate ACKs during loss
  - Need only remember `expectedseqnum`

- **Out-of-order pkt**:
  - Discard → **no receiver buffering**!
  - Re-ACK pkt with highest in-order seq #
GBN in Action

Sender (N=4)

1
2
3
4

Receiver

ACK1, deliver
ACK2, deliver
ACK2, discard

5
6

ignore
ignore
ignore

ACK2, discard
ACK2, discard

3
4
5
6

ACK3, deliver
ACK4, deliver
ACK5, deliver
ACK6, deliver

timeout
Selective Repeat

- Receiver *individually* acknowledges all correctly received pkts
  - Buffers pkts, as needed, for eventual in-order delivery to upper layer
- Sender only resends pkts for which ACK was not received
  - Separate timer for each unACKed pkt
- Sender window
  - $N$ consecutive packets in $[\text{snd}_\text{base}, \text{snd}_\text{base}+N-1]$
Selective Repeat: Sender, Receiver Windows

**Sender (N=7)**
- snd_base
- nextseqnum
- sender window

**Receiver (N=7)**
- rcv_base
- receiver window

**Legend:**
- Teal: sent & acked
- Yellow: sent & not acked
- Blue: not sent & available
- White: not available
- Purple: received and delivered
- Red: received and buffered
- Gray: expected but not received
- Blue: available slot
**Selective Repeat**

**sender**

Data from above:
• If next available seq # in window, send pkt

Timeout(n):
• Resend pkt n, restart timer n

ACK(n) in [snd_base, snd_base+N-1]:
• Mark pkt n as received
• If n == snd_base, advance snd_base to the next unACKed seq #

**receiver**

Receive pkt n in [rcv_base, rcv_base+N-1]
• Send ACK(n)
• Out-of-order (n>rcv_base): buffer
• In-order (n == rcv_base): deliver, advance rcv_base to next not-yet-received pkt, deliver all buffered, in-order pkts

Pkt n in [rcv_base-N, rcv_base-1]
• ACK(n)
Otherwise:
• Ignore
Selective Repeat in Action (N=4)

timeout on pkt 2

ACK0

ACK1

ACK3

ACK4

ACK5

ACK2
**Selective Repeat: Dilemma**

Q: How many distinct seq #s are needed for window size N in selective repeat?

Example:
- Seq #’s: 0, 1, 2, 3
- Window size = 3
- Receiver sees no difference in two scenarios!
- Incorrectly passes duplicate data as new in (a)