Reliable Transmission: Pipelined Transfer

Srinidhi Varadarajan
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
Go-Back-N

Sender:

- k-bit seq # in pkt header
- “window” of up to N, consecutive unack’ed pkts allowed

- ACK(n): ACKs all pkts up to, including seq # n - “cumulative ACK”
  - may receive duplicate ACKs (see receiver)
- timer for each in-flight pkt
- timeout(n): retransmit pkt n and all higher seq # pkts in window
GBN: sender extended FSM

```
rdt_send(data)
    if (nextseqnum < base+N) {
        compute chksum
        make_pkt(sndpkt(nextseqnum)), nextseqnum, data, chksum
        udt_send(sndpkt(nextseqnum))
        if (base == nextseqnum)
            start_timer
            nextseqnum = nextseqnum + 1
    }
    else
        refuse_data(data)

rdt_rcv(rcv_pkt) && notcorrupt(rcvpkt)
    base = getacknum(rcvpkt) + 1
    if (base == nextseqnum)
        stop_timer
    else
        start_timer
    timeout
        start_timer
        udt_send(sndpkt(base))
        udt_send(sndpkt(base+1))
        udt_send(sndpkt(nextseqnum-1))
```

```
GBN: receiver extended FSM

receiver simple:

- **ACK-only**: always send ACK for correctly-received pkt with highest *in-order* seq #
  - may generate duplicate ACKs
  - need only remember `expectedseqnum`

- **out-of-order pkt**:
  - discard (don’t buffer) -> no receiver buffering!
  - ACK pkt with highest in-order seq #
GBN in action

sender

send pkt0
send pkt1
send pkt2
send pkt3 (wait)
rcv ACK0
send pkt4
rcv ACK1
send pkt5
pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5

receiver

rcv pkt0
send ACK0
rcv pkt1
send ACK1
rcv pkt3, discard
send ACK1
rcv pkt4, discard
send ACK1
rcv pkt5, discard
send ACK1
rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3
GBN (corrected)

**Original Timeout**
- send pkt0
- send pkt1
- send pkt2
- send pkt3 (wait)
- rcv pkt1

**Timer reset 1**
- pkt2 timeout
- send pkt2
- send pkt3
- send pkt4
- send pkt5

**Timer reset 2**
- send pkt5

**Sender**
- rcv pkt0
- send ACK0
- rcv pkt1
- send ACK1
- rcv pkt3, discard
- send ACK1

**Receiver**
- rcv pkt4, discard
- send ACK1
- rcv pkt5, discard
- send ACK1
- rcv pkt2, deliver
- send ACK2
- rcv pkt3, deliver
- send ACK3
Problems with GBN

- Retransmits entire sender window on timeout
  - Can cause excessive retransmissions
  - Problem is exacerbated for networks with large “memory”, i.e. large delay bandwidth product

- Receiver throws away any out of order packets, even if they are received correctly.
  - Forces retransmission
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 not received
  - sender timer for each unACKed pkt
- sender window
  - N consecutive seq #’s
  - again limits seq #’s of sent, unACKed pkts
Selective repeat: sender, receiver windows

(a) sender view of sequence numbers

(b) receiver view of sequence numbers
Selective repeat

**sender**

- **data from above:**
  - if next available seq # in window, send pkt
  - else hold packet

**timeout(n):**
- resend pkt n, restart timer

**ACK(n) in [sendbase,sendbase+N]:**
- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #
- Transmit any pending packets

**receiver**

- **pkt n in [rcvbase,rcvbase+N-1]:**
  - send ACK(n)
  - out-of-order: buffer
  - in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

- **pkt n in [rcvbase-N,rcvbase-1]:**
  - ACK(n)

- otherwise:
  - ignore
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
ACK0 rcvd, pkt4 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
ACK1 rcvd, pkt5 sent
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

pkt4 rcvd, buffered, ACK4 sent
0 1 2 3 4 5 6 7 8 9

pkt2 rcvd, deliver pkts 2, 3, 4
ACK2 sent
0 1 2 3 4 5 6 7 8 9

pkt5 rcvd, delivered, ACK5 sent
0 1 2 3 4 5 6 7 8 9
Selective repeat: dilemma

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)

Q: what relationship between seq # size and window size?
Out of Order Delivery

- What happens if the network delivers packets out of order
  - Send order != receive order

- Need a much larger sequence space
  - Why?

- TCP faces the same problem. IP does not guarantee in-order packet delivery
  - TCP assumes max. packet lifetime of 3 minutes. Seq. numbers should be reused within 3 minutes. RFC 1323
Reading

Required