Sliding Window 2
int sockfd1, sockfd2;
struct sockaddr_in server;
struct sockaddr_in client;
unsigned int clen;
int nums[3];
int i;
int max;

sockfd1 = socket(AF_INET, SOCK_STREAM, 0);

memset(&server, 0, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons(25000);

bind(sockfd1, (struct sockaddr *)&server, sizeof(struct sockaddr_in));

listen(sockfd1, 5);
while (1) {
    sockfd2 = accept(sockfd1, (struct sockaddr *) &client, &clen);
    recv(sockfd2, nums, 3*sizeof(int), 0);
    for (i = 0; i < 3; i++) nums[i] = htonl(nums[i]);
    int max = nums[0];
    for (i = 1; i < 3; i++) if (nums[i] > max) max = nums[i];
    max = htonl(max);
    send(sockfd2, &max, sizeof(int), 0);
    close(sockfd2);
}

close(sockfd1);
Go-Back-N

- **Sender Window Size**
  - Up to SWS unacknowledged frames
  - Keep track of last frame sent (LFS) and last ACK received (LAR)
  - SWS $\geq$ LFS - LAR

- **Receiver Window Size**
  - RWS = 1

- **Retransmission**
  - Suppose a time out occurs on frame n the frames n through LFS are retransmitted
GBN: sender extended FSM

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

if (base == nextseqnum)
    stop_timer
else
    start_timer

rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
base = getacknum(rcvpkt)+1
if (base == nextseqnum)
    stop_timer
else
    start_timer
```
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!
  - re-ACK pkt with highest in-order seq #
Go-Back-N

- Finite Sequence Numbers
- If there is an $n$ bit field for sequence number then there are $2^n$ sequence numbers
- Maximum SWS must less than $2^n$
- Maximum SWS is one less than the number of sequence numbers
Selective Repeat

• Accept frames out of order but pass frames to higher layers in order
• Must buffer out of order frames
• Receiver Window Size (RWS)
  – Buffer space for received frames
  – Can hold frames received out of order before passing in order frames to higher layer
  – Largest Acceptable Frame (LAF)
  – Last frame received (LFR) in order
  – RWS >= LAF-LFR
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

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 #

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

• Finite Sequence Numbers
• If n bit field for sequence number then there are $2^n$ sequence numbers
• Maximum SWS must less than or equal to $2^{n-1}$
• Maximum SWS is half of the number of sequence numbers