CS5222 Computer Networks and Internets

Tutorial 5 (Week 5)

Prof Weifa Liang

Weifa.liang@cityu.edu.hk

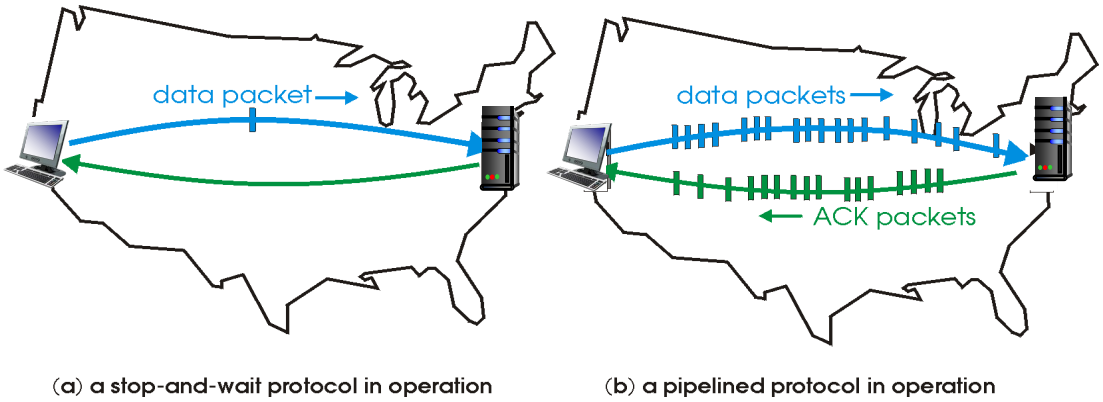
Slides based on book *Computer Networking: A Top-Down Approach.*

Pipelined protocols

pipelining: sender allows multiple, “in-flight”, yet-to-be-acknowledged packets

- range of sequence numbers must be increased
- buffering at sender and/or receiver
- two generic pipelined protocols:

  - *Go-Back-N*
  - *Selective Repeat*



(a) a stop-and-wait protocol in operation (b) a pipelined protocol in operation

Pipelined protocols: overview

Go-back-N:

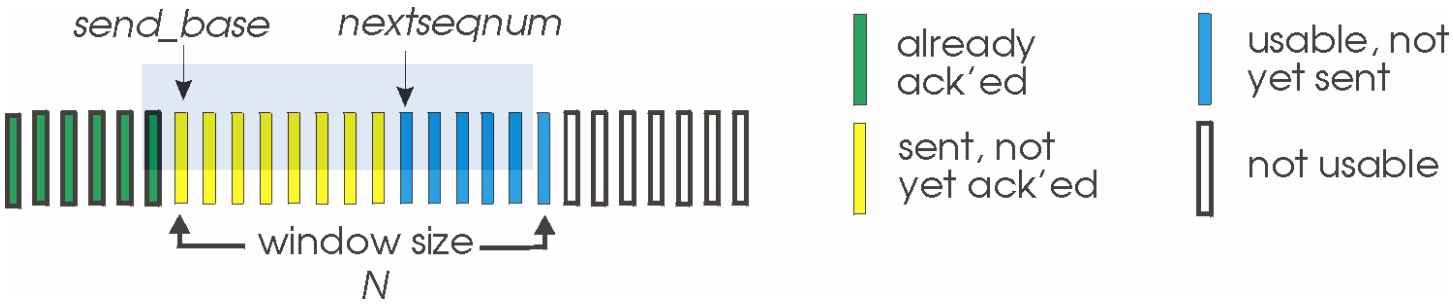
- sender can have up to N unacked packets in pipeline
- receiver only sends *cumulative ack*
  - Does not ack packet if there is a gap
- sender has a timer for the oldest unacked packet
  - when the timer expires, retransmit *all* unacked packets

Selective Repeat:

- Sender can have up to N unacked packets in pipeline
- receiver sends *individual ack* for each received packet
- sender maintains timer for each unacked packet
  - when timer expires, retransmit only that unacked packets

Go-Back-N: sender

- sender: "window" of up to *N*, consecutive transmitted but unACKed pkts
  - *k*-bit seq # in pkt header

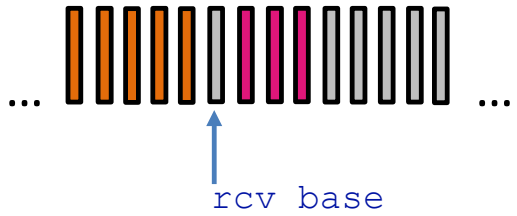


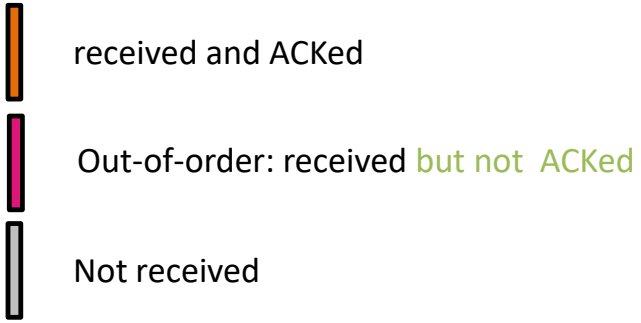
- *cumulative ACK:* ACK(*n*): ACKs all packets up to, including seq # *n*
  - on receiving ACK(*n*): move window forward to begin at seq *n+1*
- timer for the oldest in-flight/unacked packet
- *timeout(n):* retransmit packet *n* and all higher seq # packets in the window

Go-Back-N: receiver side

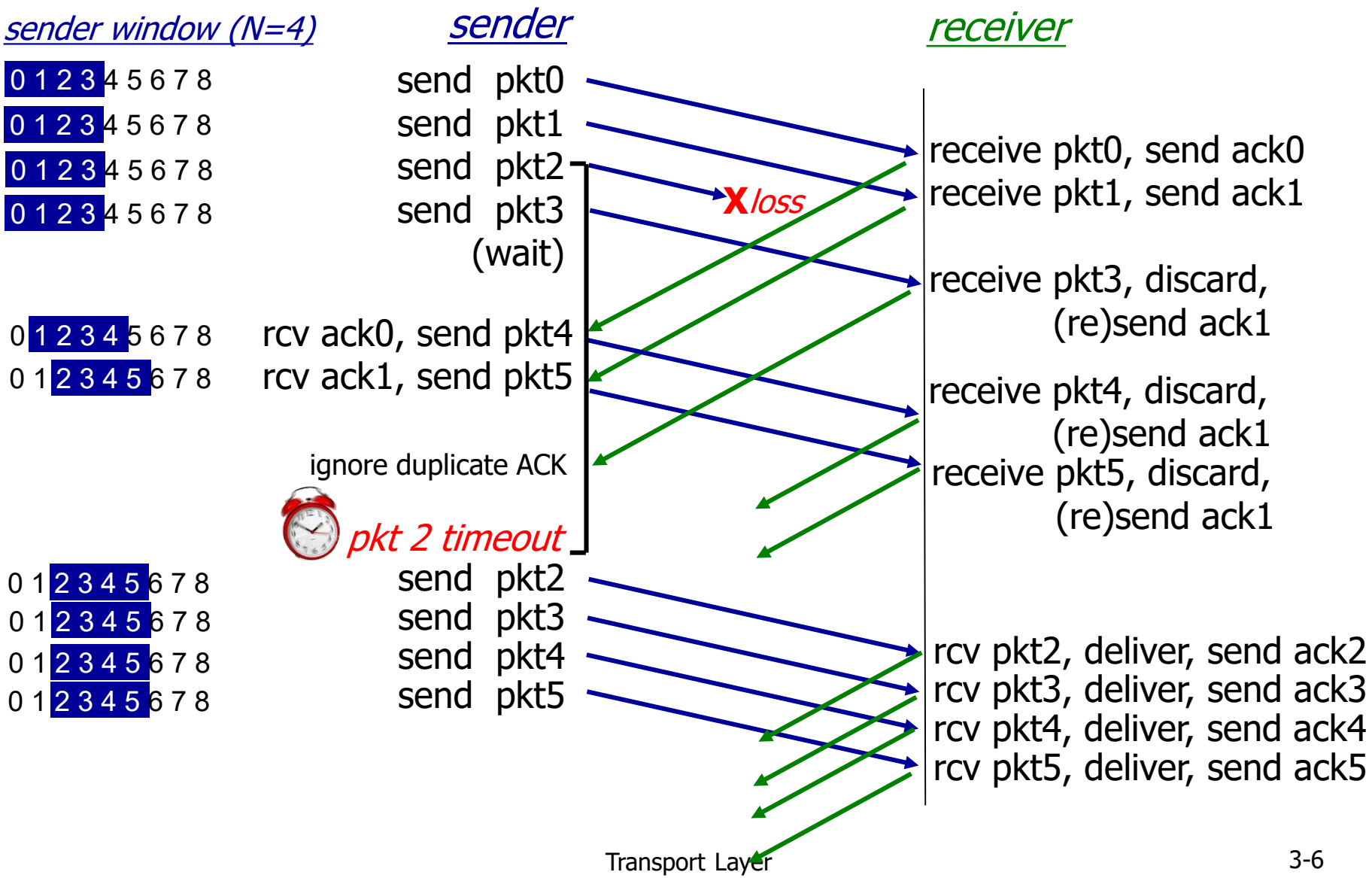
- ACK-only: always send ACK for correctly-received packet so far, with highest *in-order* seq #
  - may generate duplicate ACKs
  - need only remember `rcv_base`
- on receipt of out-of-order packet:
  - either discard (i.e. don't buffer) or buffer: depends on implementation!
  - or re-ACK pkt with the highest in-order seq #

Receiver view of sequence number space:





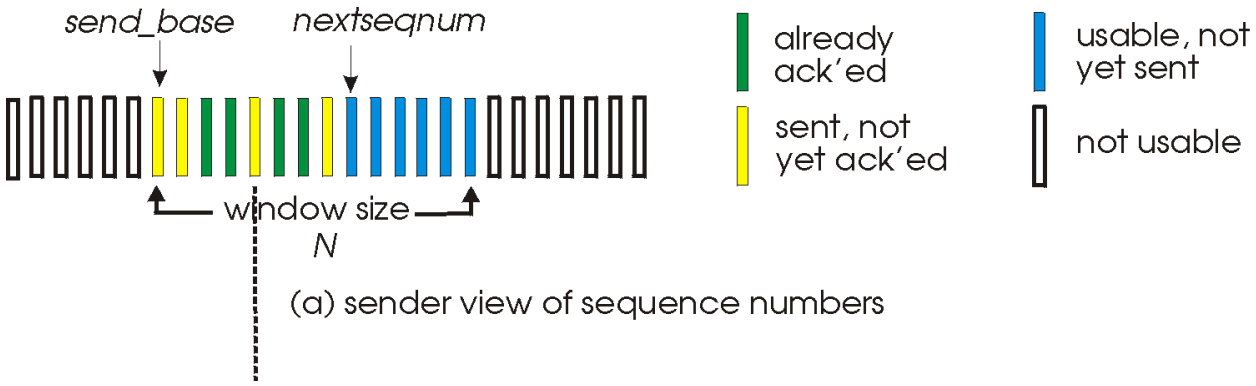
Go-back-N in action



Selective repeat

- receiver *individually* acknowledges all correctly received packets
  - buffers packets, as needed, for eventual in-order delivery to the upper layer
- sender times-out/retransmits individually unACKed packets
  - sender maintains a timer for *each* unACKed pkt
- sender window
  - *N* consecutive seq #s
  - limits seq #s of sent, unACKed packets

Selective repeat: sender, receiver windows



Selective repeat: sender and receiver

sender

packet from above:

- if next available seq # in window, send packet

timeout(*n*):

- resend packet *n*, restart timer

ACK(*n*) in [sendbase,sendbase+N]:

- mark packet *n* as received
- if packet with seq # *n* is the smallest unACKed packet, advance the sender window base to next unACKed seq #

receiver

packet *n* in [rcvbase, rcvbase+N]

- send ACK(*n*)
- out-of-order: buffer the packet
- in-order: deliver (also deliver buffered, in-order packets), advance the receiver window to next not-yet-received packet

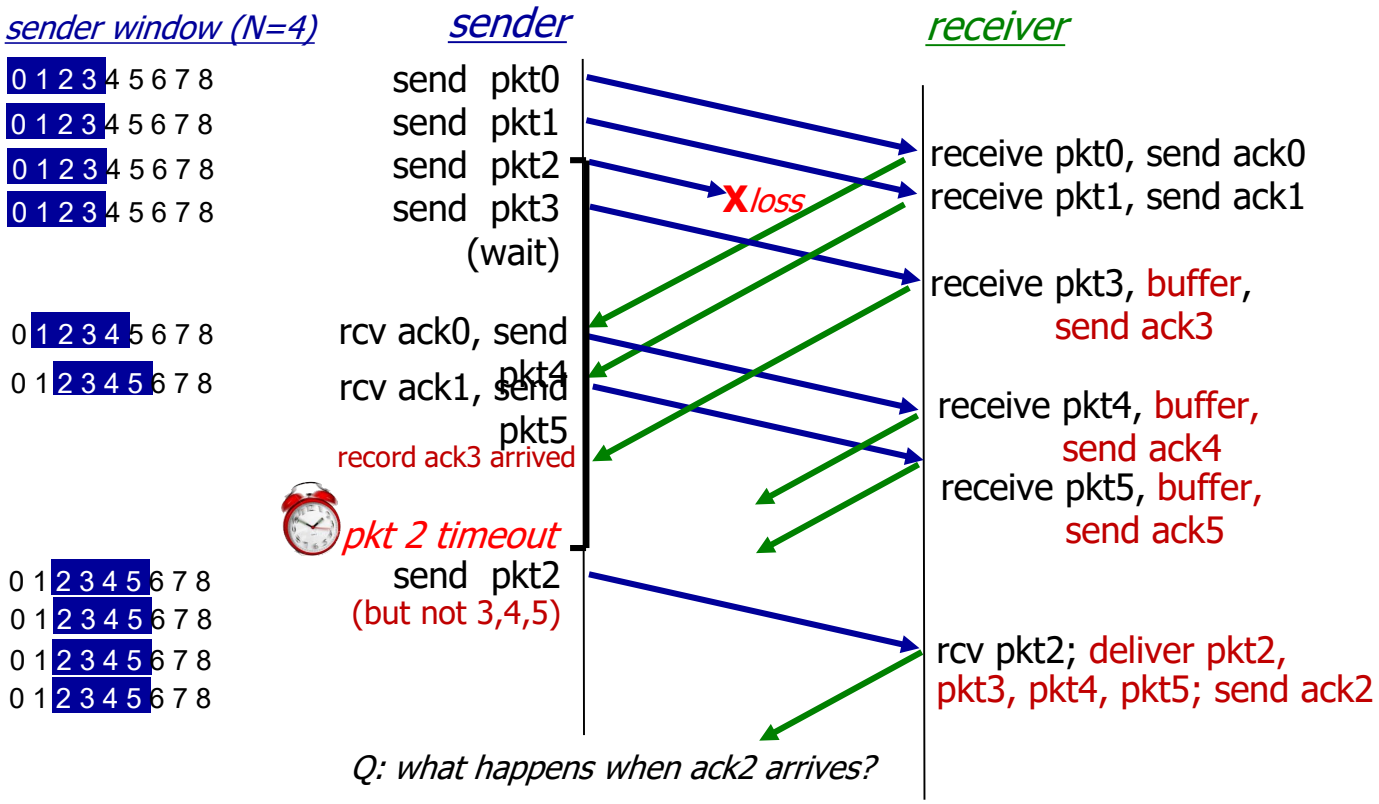
packet *n* in [rcvbase-N,rcvbase-1]

- ACK(*n*)

otherwise:

- ignore

Selective Repeat in action



Work on your questions

1. Suppose that the **Go-back-N** protocol is used. The sequence number space is {0, 1, 2, 3}. Suppose the **sender window size is 4** (where this is actually not allowed. For the illustration purpose, let us assume that the sender window size is 4). The sender sends packets 0, 1, 2, and 3. The receiver receives all the 4 packets in order and generates acks. Suppose that all acks are lost.

a) After the receiver sends out all acks, what will be the expected sequence number at the receiver side?

Answer: the expected sequence number at the receiver side is 0.

b) After the timeout interval for packet 0, the sender retransmits packet 0. Suppose that the receiver receives the packet. What will the receiver do?

Answer: The receiver will think it is an in-order-arrival and deliver it to the application. It will send ack 0.

2. Suppose that the **Go-back-N** protocol is used. The sequence number space is {0, 1, 2, 3}. **Suppose the sender window size is 3**. The sender sends packets 0, 1, and 2. The receiver receives all 3 packets in order and generates acks. Suppose that all acks are lost.

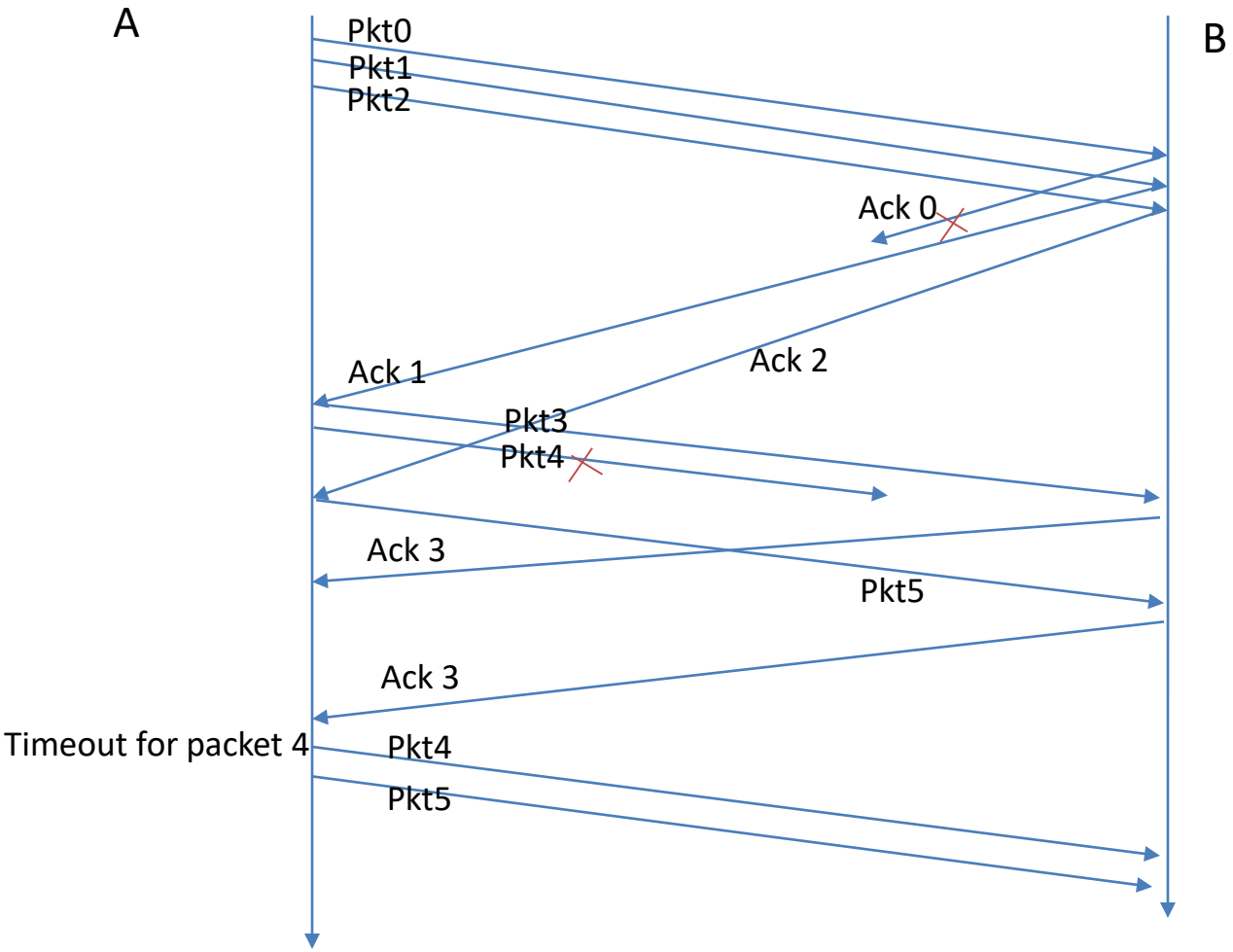
a) After the receiver sends out all acks, what will be the expected sequence number at the receiver side?

Answer: The expected sequence number at the receiver side is 3.

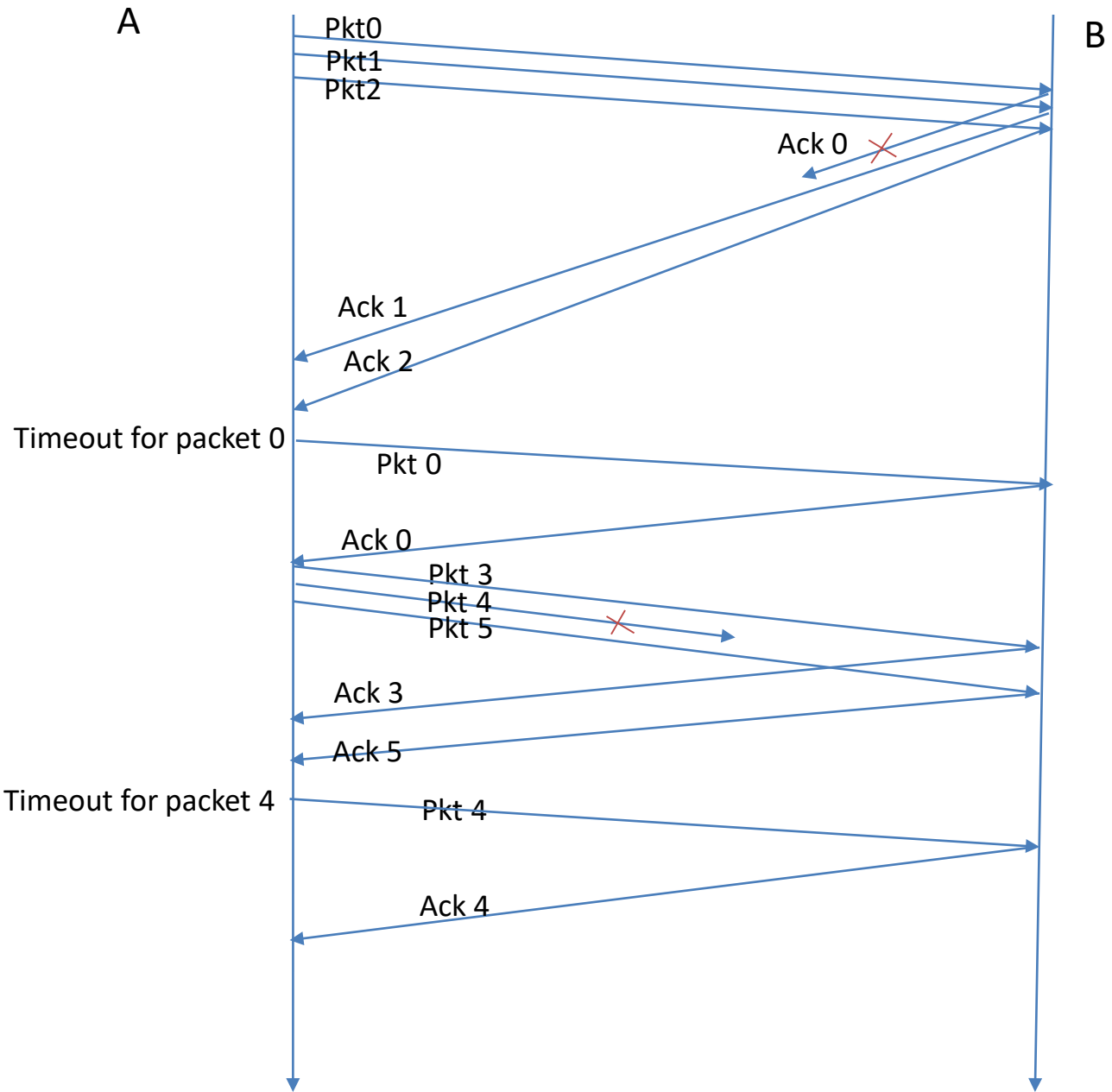
b) After the timeout interval for packet 0, the sender retransmits packet 0. Suppose that the receiver receives the packet. What will the receiver do?

Answer: the receiver will discard the packet and re-send ack 2.

3. Suppose that Host A and Host B use the **Go-Back-N protocol** with **window size N=3** and a long-enough range of sequence numbers. Assume Host A sends six application messages to Host B and that all messages are correctly received except for the first acknowledgement and the fifth data segment. Draw a timing diagram, showing data segments and the acknowledgements sent along with the corresponding sequence and acknowledge numbers, respectively. Assume that the first packet's sequence number starts with 0.



b) Use the **selective repeat** protocol for the problem:



4. Suppose that the **selective repeat** protocol is used. The sequence number space is {0, 1, 2, 3}. Suppose that the **sender window size is 2**. The sender sends packets with sequence numbers 0 and 1 in the beginning. The receiver receives packets with sequence numbers 0 and 1 and sends ack 0 and ack 1 respectively. However, both acks are lost.

a) After the receiver sends ack 0 and ack 1, what will be the receiver-base? What packets are expected/acceptable at the receiver side?

Answer: the receiver-base is 2.

The expected and acceptable packets are the packets with sequence numbers 2 and 3.

b) After the timeout interval for packet 0, the sender retransmits packet 0. Suppose that the receiver receives the packet. What does the receiver do when receiving that packet?

Answer: Since packet number 0 is not within the receiver window, but it is in the previous receiver window, the receiver knows that it is a previously acked packet.

The receiver will generate “ack 0” and ignore this packet.