EECS 489 - FA 24
Discussion 5

Assignment 2, Video Streaming, Reliable Transport

Logistics

Assignment 2

Due: Friday, Oct. 11 (right before Fall Break)

- Groups of 1-3 people
- 3 AG submits per day
- 3 late days across all 3 remaining projects
- **Order of magnitude longer and more difficult than Assignment 1!**

How to Develop:

- Develop on AWS like Project 1; you will have to set up a VNC Client on your computer, as we need a GUI for this project.

A2 Prep: select() Demo

https://github.com/mosharaf/eecs489/tree/w24/Discussion/discussion-3-code

Credit: Zack Goldston (WN24 EECS 489 GSI)

Video Streaming

Q1 HTTP Streaming

Consider a simple HTTP streaming model.

Q is the number of bits that must be buffered before the client application begins playout.

r is the video consumption rate.

x is the bits sending rate whenever the client buffer is not full.

Also assume that $x < r$.

What will the video output behavior look like?

Q1 HTTP Streaming

Consider a simple HTTP streaming model.

Q is the number of bits that must be buffered before the client application begins playout.

r is the video consumption rate.

x is the bits sending rate whenever the client buffer is not full.

Also assume that $x < r$.

What will the video output behavior look like?

Alternate between playout and freeze.

Q1 HTTP Streaming

Consider a simple HTTP streaming model.

Q is the number of bits that must be buffered before the client application begins playout.

r is the video consumption rate.

x is the bits sending rate whenever the client buffer is not full.

Also assume that $x < r$.

Suppose the buffer starts out empty. How long will it be before the video can begin playout?

Q1 HTTP Streaming

Consider a simple HTTP streaming model.

Q is the number of bits that must be buffered before the client application begins playout.

r is the video consumption rate.

x is the bits sending rate whenever the client buffer is not full.

Also assume that $x < r$.

Suppose the buffer starts out empty. How long will it be before the video can begin playout?

Q / x

Q1 HTTP Streaming

Consider a simple HTTP streaming model.

Q is the number of bits that must be buffered before the client application begins playout.

r is the video consumption rate.

x is the bits sending rate whenever the client buffer is not full.

Also assume that $x < r$.

Assume the current buffer size is z ($> Q$). How long will the playout last?

Q1 HTTP Streaming

Consider a simple HTTP streaming model.

Q is the number of bits that must be buffered before the client application begins playout.

r is the video consumption rate.

x is the bits sending rate whenever the client buffer is not full.

Also assume that $x < r$.

Assume the current buffer size is z ($> Q$). How long will the playout last?

z / (r - x)

Reliable Transport

Review: Reliable Transport

Motivation:

- IP provides a best-effort packet delivery service – this means that packets sometimes get corrupted or dropped!
- We need some way to build reliable transport on top of an unreliable network.

Key Question:

- How does sender figure out what to send / resend?

Review: Reliable Transport

Common Techniques:

- Checksums
- Sequence numbers
- Acknowledgements upon receiving packet
- Sender-side timers to trigger resending packets

Review: Sliding Window

A **window** is a set of adjacent sequence numbers. If the window size is **n** and the window starts at sequence number **s**, it will consist of packets

`s, s + 1, s + 2, …, s + n - 1.`

A larger window enables more throughput, as more packets can be in-flight at once.

A window that is too large leads to the sender/receiver having to buffer and keep track of too many packets, which can create slowness.

Review: Sliding Window

To communicate that a packet has been received, the receiver sends an ACK (acknowledgement) back to the sender for every packet it receives.

- **Cumulative ACKs:** The receiver sends an ACK containing the sequence number of the next packet it expects.
- **Selective ACKs:** The receiver just sends back the sequence number of the packet it just received.

Selective ACKs convey more information, but the receiver has to keep track of a lot more state. *Why?*

Go-Back-N vs. Selective Repeat

Go-Back-N	Selective Repeat
• Used with sliding window • Receiver only accepts packets in-order, and discards out-of-order packets (i.e. does not ACK nor buffer). • Cumulative ACKs. • Sender maintains one timer (for first outstanding ACK) and resends **n** packets if the timer expires.	• Used with sliding window • Receiver accepts any packet that is in its window, possibly buffering until it can recreate them in-order. • Selective ACKs. • Sender maintains one timer per packet, resending just that packet if ACK is not received in time.

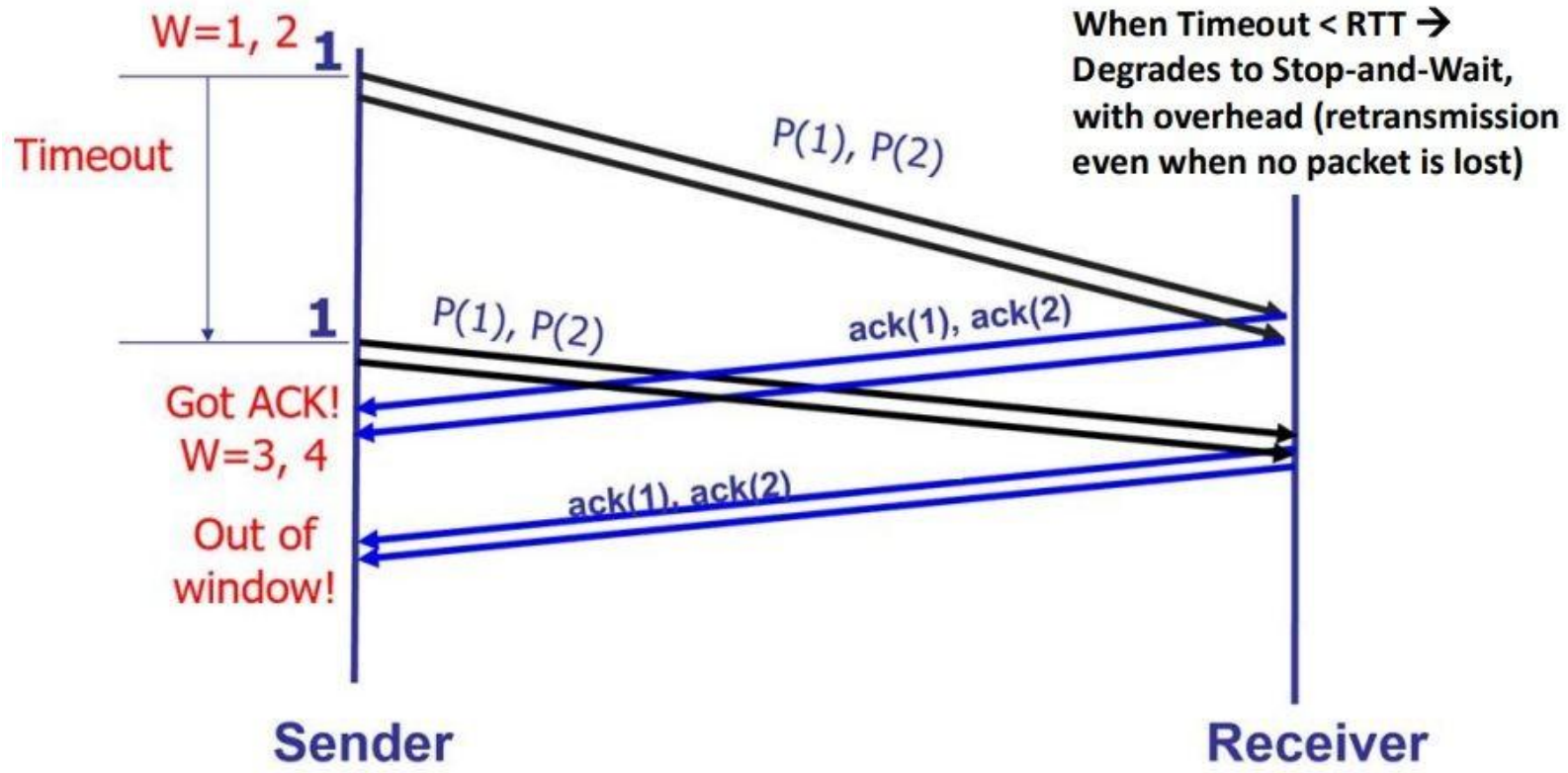
Q2: Selective Repeat

Suppose a sliding window with Selective Repeat (SR) reliable transport protocol is being used. Is it possible for a sender to receive an ACK for a packet that falls outside of its current window?

Q2: Selective Repeat

Suppose a sliding window with Selective Repeat (SR) reliable transport protocol is being used. Is it possible for a sender to receive an ACK for a packet that falls outside of its current window?

Yes.



Q3 Go-Back-N

Suppose a sliding window with Go-Back-N (GBN) reliable transport protocol is being used. Is it possible for a sender to receive an ACK requesting a sequence number that is less than the start of the sliding window?

Q3 Go-Back-N

Suppose a sliding window with Go-Back-N (GBN) reliable transport protocol is being used. Is it possible for a sender to receive an ACK requesting a sequence number that is less than the start of the sliding window?

Yes, for the same reason.

Q4 (N)ACK

Consider a reliable data transfer protocol that uses only negative acknowledgments (NACK). Suppose the sender sends data only **infrequently**.Would a NACK-only protocol be preferable to a protocol that uses ACKs? Why?

- ACK: send ACK upon packet arrival
- NACK: send NACK upon packet loss

Hint: How is packet loss detected?

Q4 (N)ACK

Consider a reliable data transfer protocol that uses only negative acknowledgments (NACK). Suppose the sender sends data only **infrequently**.Would a NACK-only protocol be preferable to a protocol that uses ACKs? Why?

- ACK: send ACK upon packet arrival
- NACK: send NACK upon packet loss

Hint: How is packet loss detected?

No. In a NACK only protocol, the loss of packet x is only detected by the receiver when packet x+1 is received. If there is a long delay between the transmission of x and the transmission of x+1, then it will be a long time until x can be recovered, under a NACK only protocol

Q5 (N)ACK

Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NACK-only protocol be preferable to a protocol that uses ACKs? Why? (Assuming ACK/NACK is never lost)

- ACK: send ACK upon packet arrival
- NACK: send NACK upon packet loss

Q5 (N)ACK

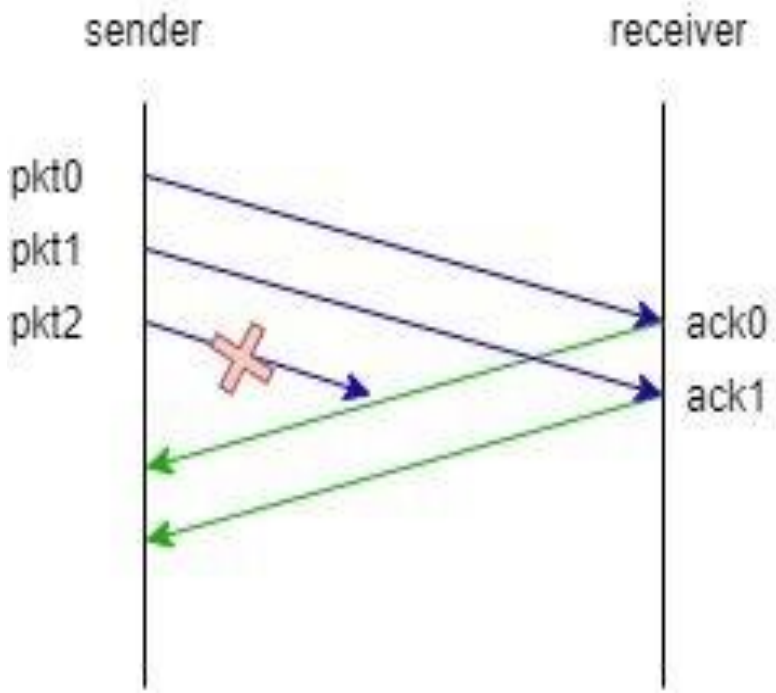
Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NACK-only protocol be preferable to a protocol that uses ACKs? Why? (Assuming ACK/NACK is never lost)

- ACK: send ACK upon packet arrival
- NACK: send NACK upon packet loss

Yes. If data is being sent often, then recovery under a NACK only scheme could happen quickly. Moreover, if errors are infrequent, then NACKs are only occasionally sent (when needed), while ACKs would be sent for every packet received.

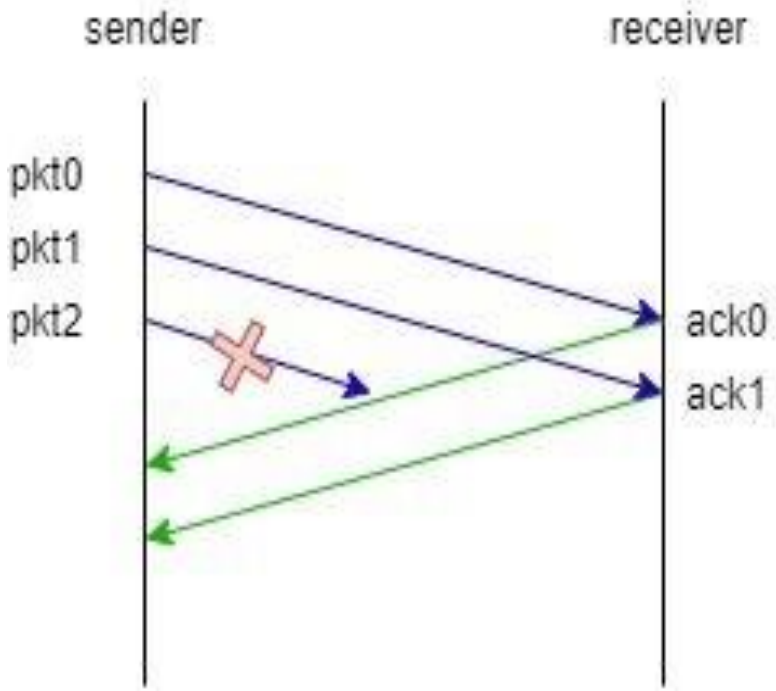
Q6 Sliding Window Protocols

Consider the sliding window protocol in the following figure. Does this figure indicate that GBN is being used, SR is being used, or there is not enough information to tell?



Q6 Sliding Window Protocols

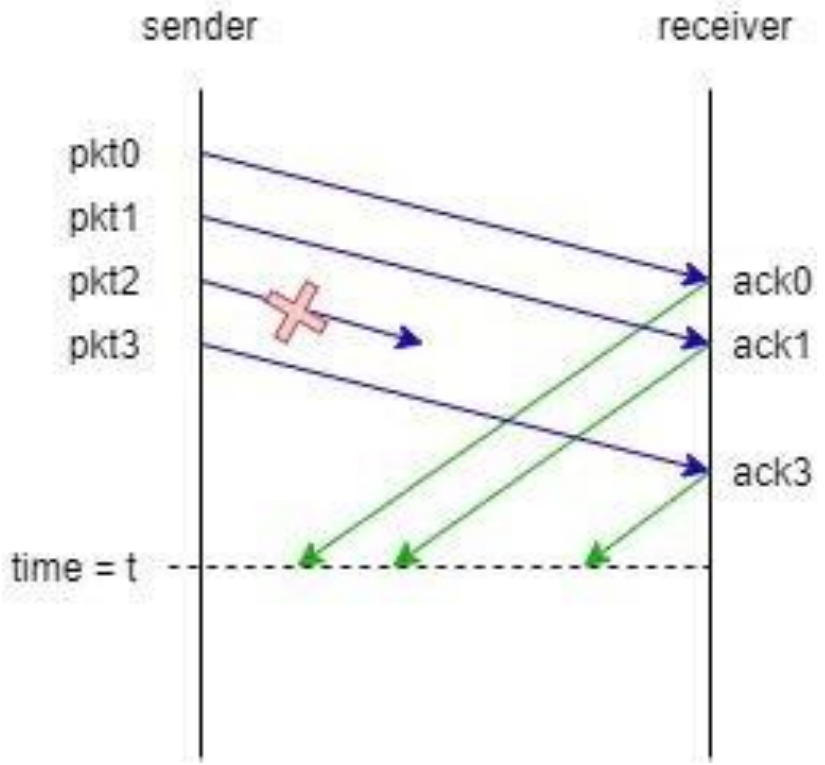
Consider the sliding window protocol in the following figure. Does this figure indicate that GBN is being used, SR is being used, or there is not enough information to tell?



There is not enough information to tell, since both GBN and SR will individually ACK each of the first two messages as they are received correctly.

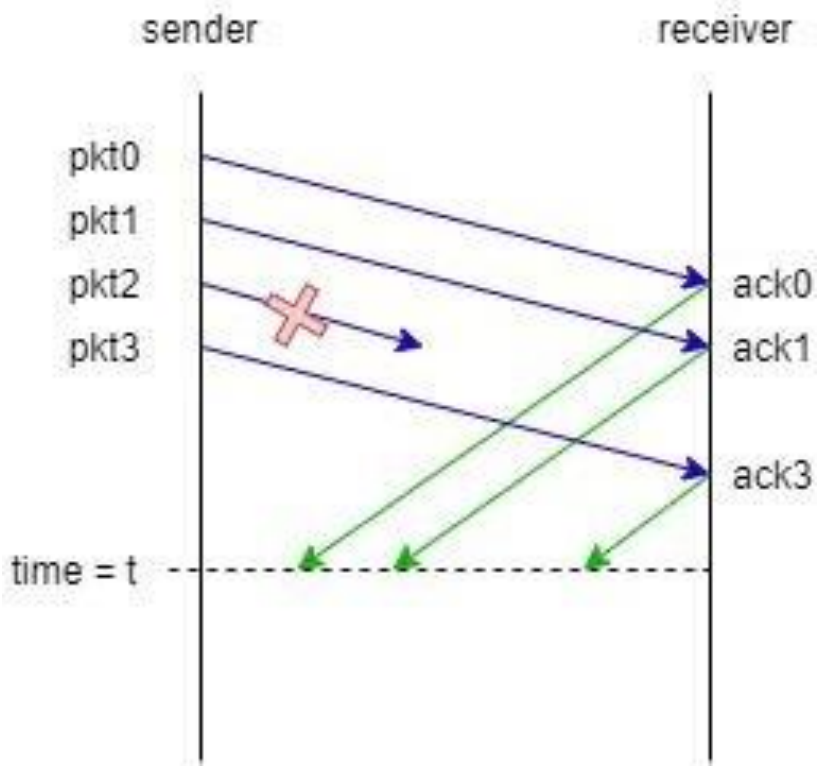
Q7 Sliding Window Protocols

Consider the sliding window protocol in the following figure. Does this figure indicate that GBN is being used, SR is being used, or there is not enough information to tell?



Q7 Sliding Window Protocols

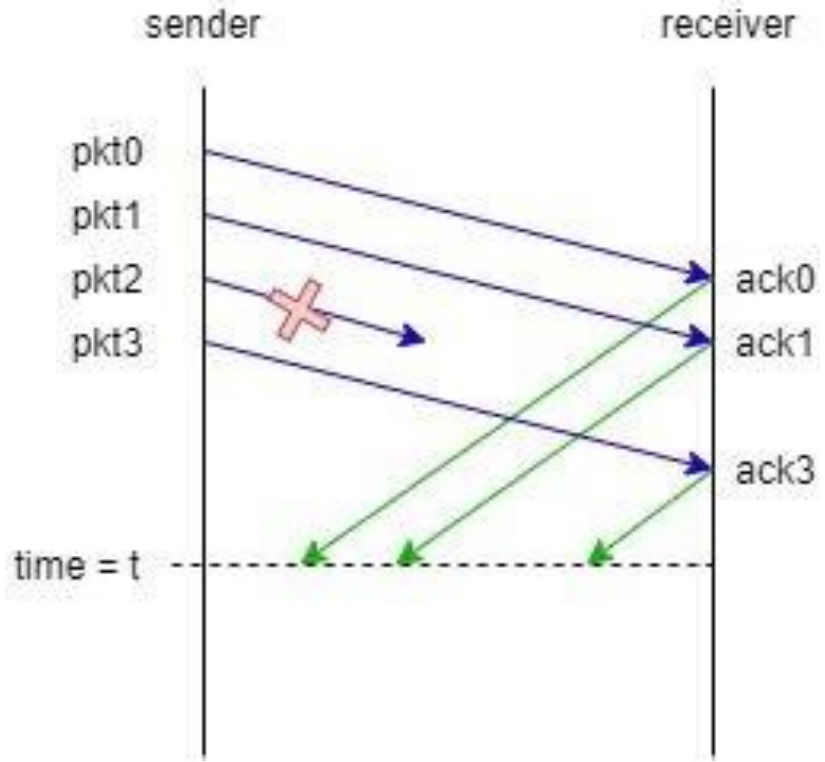
Consider the sliding window protocol in the following figure. Does this figure indicate that GBN is being used, SR is being used, or there is not enough information to tell?



This must be the SR protocol since pkt3 is acked even though pkt2 was lost. GBN will discard pkt3 if pkt2 was dropped.

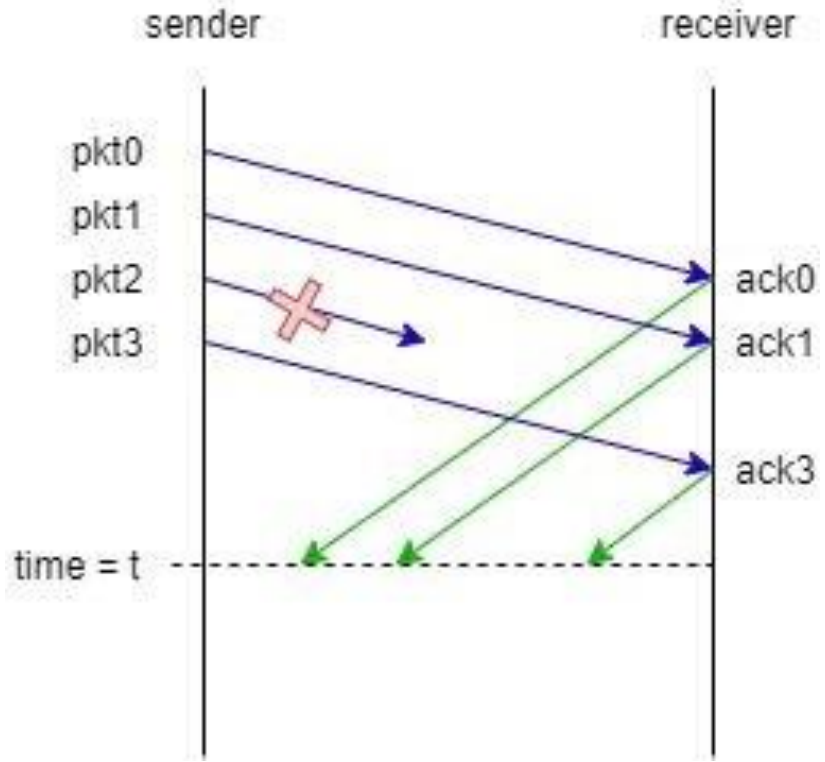
Q8 Sliding Window Protocols

Consider the sliding window protocol in the following figure. Suppose the window size = 5 for both sides. Show the positions of windows for the sender and receiver at time = t (no ack has been received).



Q8 Sliding Window Protocols

Consider the sliding window protocol in the following figure. Suppose the window size = 5 for both sides. Show the positions of windows for the sender and receiver at time = t (no ack has been received).



Sender: 0 1 2 3 4 5 6 7 8 …

Receiver: 0 1 2 3 4 5 6 7 8 …

TCP

Review: TCP

- Uses **cumulative ACKs** (like GBN)
- Receiver **buffers** out-of-order packets (like SR)

Q9 TCP

Suppose you are sending 100-byte TCP packets with a window size of 10 and you receive the following ACKs:

100, 200, 300, 300, 300, 300, 700, 800, 800, 1000

What is a possible order that the packets were received in? Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on.

Q9 TCP

Suppose you are sending 100-byte TCP packets with a window size of 10 and you receive the following ACKs:

100, 200, 300, 300, 300, 300, 700, 800, 800, 1000

What is a possible order that the packets were received in? Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on.

0 , 1 , 2 , 4 , 5 , 6 , 3 , 7 , 9 , 8

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

`200, 200, 100, 400`

Is it possible that all 5 packets were received?

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

`200, 200, 100, 400`

Is it possible that all 5 packets were received?

Yes; the ACK for one of the packets may have been lost.

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

`200, 200, 100, 400`

Is it possible that Packet 3 was not received?

No; as there is an ACK requesting byte 400, all bytes 0-399 must have been received.

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

`200, 200, 100, 400`

Is it possible that all packets were received in-order?

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

```
200, 200, 100, 400
```

Is it possible that all packets were received in-order?

No; having two ACKs requesting 200 indicates that there was some higher-sequence packet received when the receiver was waiting for packet 200.

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

`200, 200, 100, 400`

Can you tell which packet was received first?

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

```
200, 200, 100, 400
```

Can you tell which packet was received first?

Yes (Packet 0).

Suppose Packet 0 was not received first:

- We know that at most one ACK can be lost, as we sent 5 packets and received 4 ACKs.
- If Packet 0 was not received first, there must have been an ACK requesting Byte 0 that was lost (as we know Packet 0 was definitely received eventually).
- Then, that means that all 5 packets were received, as 5 total ACKs were sent out. So, Packet 4 must have been received.
- Thus, there must have been an ACK requesting Byte 500 when the fifth packet was received; this ACK cannot have been lost, as at most one ACK can be lost.

This creates a contradiction, as no ACK requesting Byte 500 was received by the sender.

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

```
200, 200, 100
```

Suppose all 5 packets were received.

Can you tell which packet was received first?

Q10 TCP

Suppose you send 5 100-byte packets with a window size of 5 using a reliability scheme similar to TCP. Assume Packet 0 has bytes 0-99, Packet 1 has bytes 100-199, and so on. Suppose you receive the following ACKs in this order:

```
200, 200, 100
```

Suppose all 5 packets were received.

Can you tell which packet was received first?

No; both of these are valid orders:

0, 1, 3, 2, 4 [ACK for Packet 2, Packet 4 lost]

3, 0, 1, 4, 2 [ACK for Packet 3, Packet 2 lost]

Thanks

Have a good one!