Exercise Session 5: Transport Layer - Solutions

COM-208: Computer Networks

In most of the problems, you will be asked to complete a diagram, stating all exchanges, events and actions taken by the transport layers of a sender and a receiver. For each exchanged segment, indicate whether data or acknowledgment (ACK) segment, its sequence (SEQ) or ACK number, and whether the segment was lost. Also, indicate any timeout events and changes to the sender or receiver window.

We define the “duration of a file transfer” as the amount of time that elapses from the moment the sender transmits the first bit of the file until the moment the receiver receives the last bit of the file. Whenever computing the duration of a file transfer, assume that segment headers and ACK segments have insignificant size (i.e., their transmission delay is insignificant).

Go-Back-N in the presence of data-segment loss

End-system A is communicating with end-system B using a Go-Back-N (GBN) protocol with window size $N = 4$ and valid sequence numbers ranging from 0 to 10. The network between the two end-systems may drop segments, but it never reorders or corrupts segments. Suppose B has received and acknowledged data segments with SEQs 0 and 1, and it is expecting data segment with SEQ 2. Instead, it receives data segments with SEQs 4, 5, 2, and 3 (in this order). What has happened? Which (data or ACK) segments have been lost? How does B respond upon receiving each data segment?

Complete the diagram in Figure 1. We have already filled in some of the information: segment 2 was lost and A timed out waiting for an ACK to segment 2.

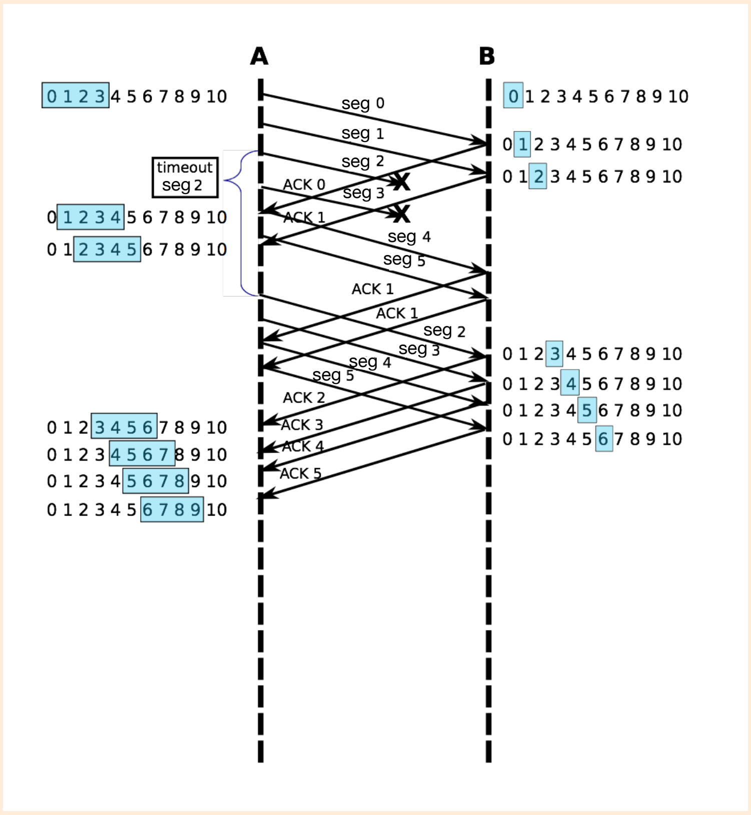

Figure 1

Now also with ACK-segment loss

End-system A communicates with end-system B using a GBN transport-layer protocol with sender window size $N = 4$ and valid sequence numbers ranging from 0 to 10. The network between them may drop segments, but it never reorders or corrupts segments. End-system B receives data segments with the following SEQ numbers (in this order): 0, 1, 3, 4, 5, 2, 3, 4, 5. End-system A receives ACK segments with the following ACK numbers (in this order): 1, 1, 1, 2, 3.

Complete the diagram in Figure 2. We have already filled in some of the information to help you get started: end-system A sent data segments with SEQs 0 and 1, which reached end-system B; ACK 0 was lost.

Figure 2 shows one solution to this question; there may exist other plausible solutions.

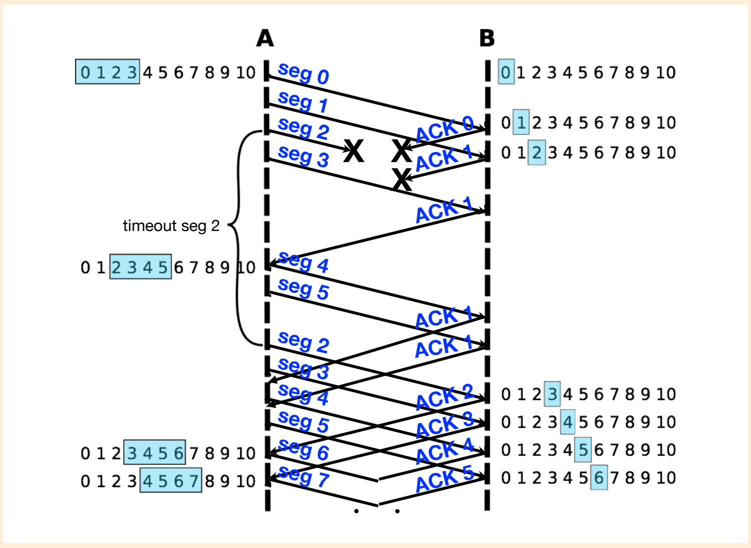

Figure 2

A more challenging scenario

End-system A communicates with end-system B using a GBN transport-layer protocol with sender window size $N = 3$. The network between them may drop segments, but it never reorders or corrupts segments. A wants to send a file to B. It splits the file into 8 data segments with SEQs from 0 to 7. In the entire duration of the file transfer, A receives ACK segments with the following ACK numbers (in this order): 1, 1, 4, 6, 7.

Complete the diagram in Figure 3. We have already completed some of the information to help you get started: A sends data segments with SEQs 0 and 1, which reach B. B sends ACKs 0 and 1, but only ACK 1 reaches A.

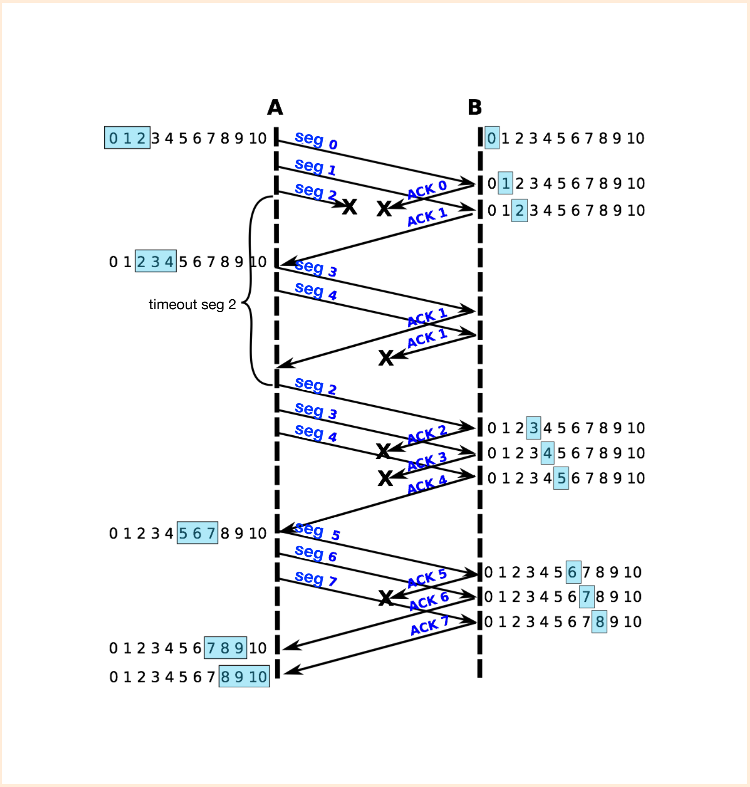

Figure 3

Thinking creatively about Go-Back-N

An end-system wants to send 4 data segments over a network that may drop segments, but it never reorders or corrupts segments.

- In the GBN protocol, what mechanism is used by the sender to detect the loss of data segments?

Retransmission timer.

- Are there situations where the sender has sufficient information that could be used to detect a segment loss sooner and, thus, recover faster? How would you improve the sender algorithm? What is the minimum amount of time needed to detect the loss of a data segment? Think about the pattern of acknowledgments received by the sender when a data segment is lost.

Yes, duplicate ACKs indicate that the data segment after the ACKed segment was lost. These are generated when one or more segments with sequence numbers higher than the lost segment are received.

The sender could be modified to retransmit immediately after a duplicate acknowledgment is received, without waiting for the timer to expire. The detection time of a loss would be reduced, in the best case, to one RTT.

- Can you identify any problem with the modified sender algorithm if the network starts reordering segments?

Yes. If there is segment reordering in the network on the forward path, the aforementioned approach can cause problems because the receiver sends duplicate ACKs for every out-of-order segment received. Even if there is no loss, the sender would incorrectly conclude that some segments were lost and retransmit, wasting bandwidth.

- Propose a solution that makes the modified algorithm work better in the case where the network can reorder segments.

One plausible solution is to modify the algorithm to react to triple or quadruple duplicate ACKs, as opposed to duplicate ACKs. The network may reorder consecutively transmitted segments, so the algorithm must account for it by allowing some "slack time". In other words, the algorithm must conclude segment loss only when it has a strong indication. Such an indication is receiving three or more duplicate ACKs, as it is less likely that further apart transmitted segments got reordered. Note that we cannot indefinitely increase the slack time, as there is a trade-off between the accuracy and the latency of detecting segment loss.

Thinking about different transport-layer protocols

End-system A is connected to end-system B over a link that has, in each direction, propagation delay d_{prop} and transmission rate R. The link is reliable and does not drop, reorder, or corrupt segments. A wants to send a file of size F to B.

- What is the minimum duration of the file transfer (regardless of the transport-layer protocol used)? Assume that the entire file fits in a single segment.

It consists of:

- The transmission delay of the file, which is the time push all ths bits of the file into the link: $\frac{F}{R}$.
- The propagation delay of the link, which is the time for the last bit of the file to reach B: d_{prop}.

In total: $\frac{F}{R} + d_{prop}$.

- Suppose that the file-transfer application uses a Stop-and-Wait transport-layer protocol What is the duration of the file transfer? Assume data segments of size P and that P perfectly divides F (i.e., $\frac{F}{P}$ is an integer).

The answer is the same as the one for the next question, assuming sender window size $N = 1$. This is because Stop-and-Wait protocol is identical to Go-Back-N (GBN) with $N = 1$.

- Suppose that the file-transfer application uses a Go-Back-N (GBN) protocol with sender window size N. What is the duration of the file transfer? How does your answer compare to the result you found for the Stop-and-Wait protocol? Assume data segments of size P and that P perfectly divides F. Assume that N perfectly divides the total number of segments $\frac{F}{P}$ that A transmits during the file transfer.

A quick recap: In GBN with sender window size N, the sender can transmit up to N segments without waiting for an ACK.

We identify two cases:

1. When B's first ACK arrives at A, A is still busy transmitting the first N segments.

   In this case, A never has to wait for B's ACKs to arrive, it keeps transmitting segments one after the other until it has transmitted the entire file. Hence, the duration of the file transfer is $\frac{F}{R} + d_{prop}$, the same as in the first question.

   In more detail: A can transmit up to N unacknowleged segments. By the time it finishes transmitting the Nth segment, it has already received the ACK for the 1st segment, which means that A can immediately transmit the $N + 1$st segment. By the time it finishes transmitting the $N + 1$st segment, it has already received the ACK for the 2nd segment, which means that A can immediately transmit the $N + 2$nd segment. And so on. So, A is never "blocked" waiting for B's ACKs, it can keep transmitting.

   This case occurs when $\frac{N \cdot P}{R} \geq \frac{P}{R} + 2 \cdot d_{prop}$: The term on the left is the amount of time it takes for A to transmit the first N segments. The term on the right is the amount of time it takes for A to transmit one segment, wait for it to reach B, and receive B's ACK.

2. A finishes transmitting the first N segments before B's first ACK arrives at A.

   In this case, the file transfer happens in "rounds"; a round begins when A starts transmitting the first segment of a window. There are $M = \frac{F}{P \cdot N}$ such rounds, because A needs to send $\frac{F}{P}$ segments, and it sends N segments per round.

   Figure 4 illustrates: Each round expecpt for the last one lasts $\frac{P}{R} + 2 \cdot d_{prop}$: the time for A to transmit the first segment in the current round, wait for it to reach B, and receive B's ACK for that segment. The last round lasts only $\frac{N \cdot P}{R} + d_{prop}$: the time for A to transmit the first segment in the current round and wait for it to reach B. In total: $(M - 1) \cdot \left(\frac{P}{R} + 2 \cdot d_{prop}\right) + \frac{N \cdot P}{R} + d_{prop}$.

   This case occurs when $\frac{N \cdot P}{R} < \frac{P}{R} + 2 \cdot d_{prop}$: The opposite of the above case.

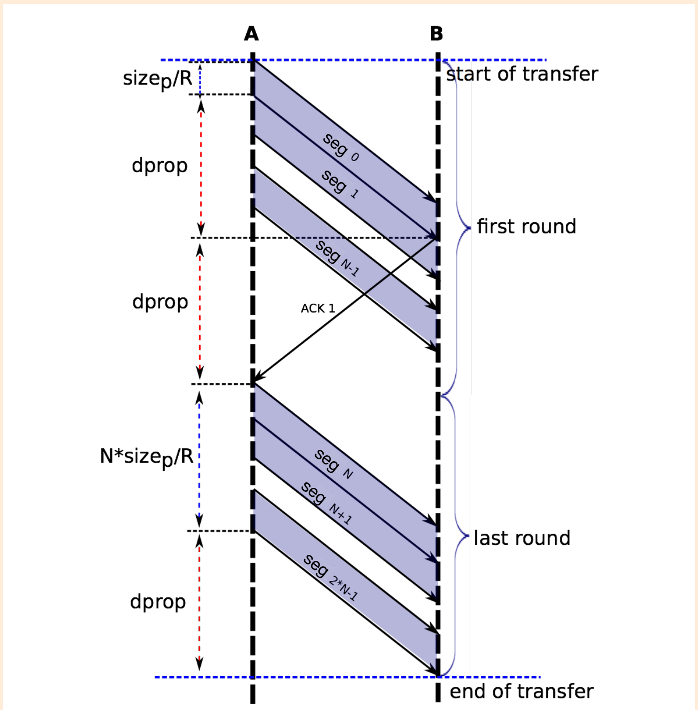

Figure 4: A sends $K = 2 \cdot N$ segments to B using GBN with window size N.

- Would you prefer a GBN or a Selective Repeat (SR) transport-layer protocol to perform this file transfer over this specific link? Justify your answer.

Both protocols perform equally well in this case, where there is no segment loss. The only difference is that GBN is a simpler protocol, and does not require the sender to maintain multiple timers, one for each unACKed segment.

Now with segment loss

An end-system wants to send 4 data segments over a network that may drop segments, but it never reorders or corrupts segments. Compute the time it takes for the sender to send all data segments and receive all acknowledgments in each of the following scenarios. If there are many possible answers, identify all of them. Assume that the round trip time between the sender and the receiver is $RTT = 100$ ms, the transmission delay of any single segment is negligible, and the retransmission timer duration is $TO = 400$ ms.

- The sender uses a GBN transport-layer protocol with window size $N = 4$. One data segment (from the sender to the receiver) is lost the first time it gets transmitted; no other segment is lost.

The retransmission timer will timeout at time TO for the lost data segment, and a retransmission will occur, which lasts another RTT. So, $\Delta T = TO + RTT = 500$ ms.

- The sender uses a GBN transport-layer protocol with window size $N = 4$. An ACK segment is lost the first time it gets transmitted; no other segment is lost.

GBN uses cumulative acknowledgments, so if the first, second or third ACK is lost, the fourth ACK "covers" for it and no retransmission is necessary; in each of these cases, $\Delta T = RTT = 100$ ms.

However if the fourth ACK is lost, there will be a retransmission; so, $T = TO + RTT = 500$ ms.

- The sender uses an SR transport-layer protocol with window size $N = 4$. A data segment is lost the first time it gets transmitted; no other segment is lost.

The retransmission timer will timeout at time TO for the lost data segment, and a retransmission will occur, which lasts another RTT. So, $\Delta T = TO + RTT = 500$ ms.

- The sender uses an SR transport-layer protocol with window size $N = 4$. An ACK segment is lost the first time it gets transmitted; no other segment is lost.

SR uses individual acknowledgments. The loss of the ACK will cause a timeout and a retransmission. So, $\Delta T = TO + RTT = 500$ ms.

A more challenging scenario

End-system A wants to send a file to end-system B using a transport-layer protocol that provides reliable data delivery with pipelining and sender window $N = 5$. A's transport layer splits the file in 10 data segments with SEQs from 0 to 9.

- The first data segment from A to B is lost the first time it is transmitted, while all other segments arrive at their destination uncorrupted and in order. Complete the two diagrams in Figure 5 in each of the following two cases: (i) A and B use GBN, and (ii) A and B use SR.

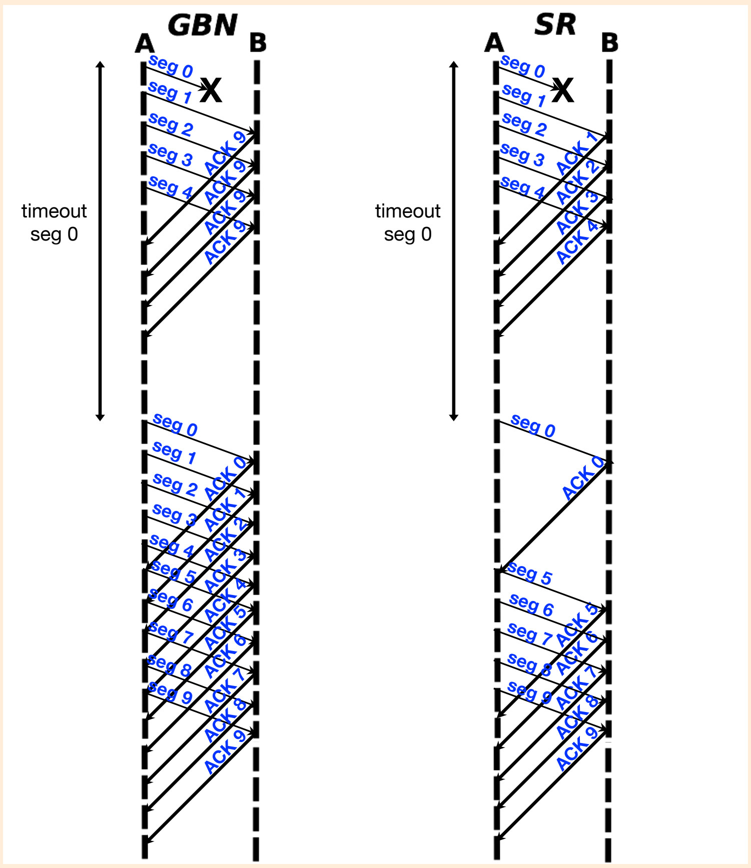

Figure 5

- The first 4 ACKs from B to A are lost the first time they are transmitted, while all other segments arrive at their destination uncorrupted and in order. Complete the two diagrams in Figure 6 in each of the following two cases: (i) A and B use GBN, and (ii) A and B use SR.

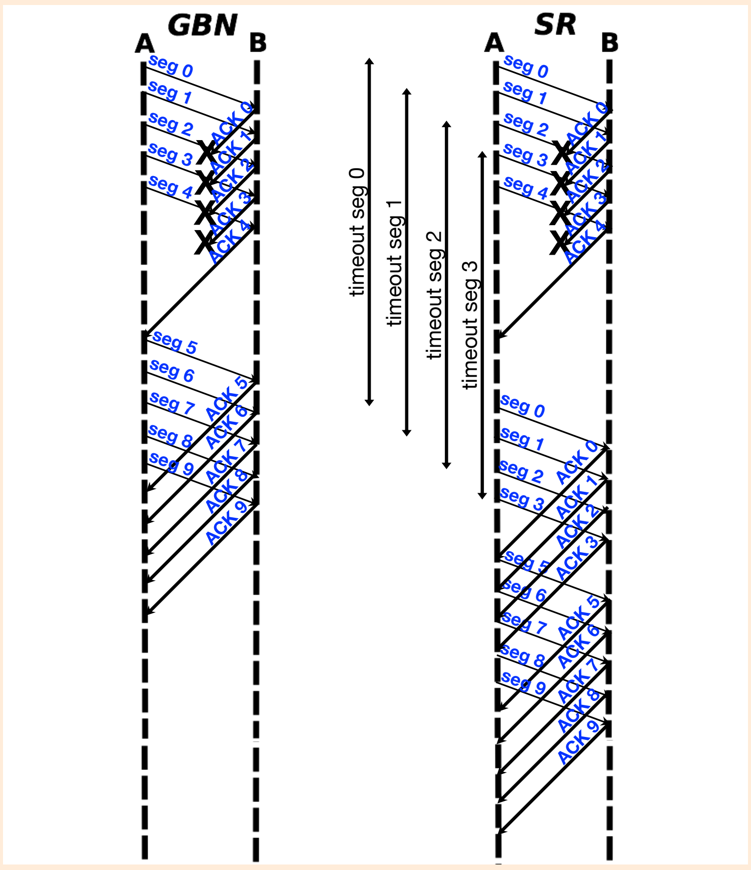

Figure 6

- Based on your answers to the previous two questions, can you draw a general conclusion about when one should prefer GBN over SR and vice versa? Justify your answer, e.g., if you state that one should prefer GBN in situation X, please say what metric GBN improves relative to SR in situation X.

SR is expected to achieve higher sender utilization and throughput than GBN when there are few, independent segment losses.

GBN is expected to achieve higher sender utilization and throughput than SR when there is bursty loss on the reverse channel (many ACKs get lost back to back).

Reliable data delivery over unreliable networks

Table 1 describes different network channels between a sender A and a receiver B. E.g., in channel (3), there may be segment corruption only on the path from A to B, and there needs to be support for pipelining. There is no segment reordering in any of the channels.

channel ID	errors in $A \to B$	errors in $B \to A$	support for pipelining?
(1)	none	none	yes
(2)	corruption	none	no
(3)	corruption	none	yes
(4)	corruption &loss	none	yes
(5)	corruption & loss	corruption & loss	yes

Table 1: Description of network channel types.

For each channel, specify which of the mechanisms in Table 2 are necessary for providing reliable data delivery (no data loss and no data duplication).

You have the following constraints:

- Specify a mechanism only if it is necessary. E.g., if one can provide reliable data delivery over a given channel without checksums, you should not specify checksums for that channel.
- Do not specify timeout-based retransmissions if NACK-based retransmissions are sufficient.

Fill in Table 2 by writing "X" in a corresponding cell to select each necessary mechanism. Then justify your choice for each channel by explaining why the mechanisms you picked are necessary and sufficient.

channel ID	sequence numbers	checksums	NACK-based retransmissions	timeout-based retransmissions
(1)				
(2)		X	X	
(3)	X	X	X	
(4)	X	X		X
(5)	X	X		X

Table 2: Reliability mechanisms.

As a general principle:

- **Receivers need sequence numbers to detect duplicate segments**. A protocol's retransmission mechanism may create duplicate segments. For example, think of a protocol that uses timeouts to counter segment loss. If the timeout is too short or the ACK gets lost, the sender will timeout for a segment that correctly reaches the destination, and retransmit it. However, duplicate segments have the same sequence numbers as the original ones, and therefore, the receiver will be able to detect and drop them.

  Also, end-systems may need sequence numbers to identify exactly which segments the receiver is still missing. This is relevant when there can be multiple segments on the fly (e.g., due to pipelining), and any of them could be lost. Depending on the mechanism used, it is either the receiver that explicitly specifies the missing segments (NACKs), or it is the sender that infers the missing segments (the segments it has not received ACKs for, when a timeout occurs).

- **Receivers need checksums to detect whether a segment is corrupted or not**, so as to discard or accept it.

See Table 2 for the solution. The justification follows.

(1) No errors occur in this channel; there is never any need to retransmit segments.

(2) Sequence numbers are not needed, as at any point in time there can be only one segment on the fly (no pipelined segments).

    Checksums are needed for the receiver to detect corrupted segments.

    NACK-based retransmissions are sufficient: The receiver can explicitly notify the sender about a corrupted segment, since (i) all NACKs reach the sender (no errors on the $B \rightarrow A$ direction), and (ii) whenever the sender receives a NACK, it knows that the NACK corresponds the last segment that was transmitted.

(3) Same as in the previous channel, with the difference that sequence numbers are now required to distinguish among pipelined segments: From the point of view of the sender, any of the pipelined segments

could have been corrupted. So the receiver must specify in the NACK the sequence number of the segment that got corrupted.

(4) Like the previous channel, sequence numbers are needed to distinguish among pipelined segments; also, checksums are needed for the receiver to detect corrupted segments.

Unlike the previous channel, a NACK-based retransmission mechanism does not suffice: For example, when the last segment of a transfer is lost, there is no way for the receiver to infer it and notify the sender.

Note that unlike the previous channel where no unnecessary retransmissions occur, sequence numbers are also needed here to detect duplicate segments. These segments can be generated by our timeout-based retransmission mechanism if the timeout is too short.

(5) Same as in the previous channel. The reasoning is similar to the one we used for the symmetrical $A \rightarrow B$ channel.