CHAPTER 23

PRACTICE SET

Questions

Q23-1. The transport-layer packets are encapsulated in the datagram at the network layer. The router through which the datagrams need to pass to reach their destination may be congested and drop the packets.

Q23-2. If a transport layer protocol such as TCP uses a timer and resends some packets that have not arrived at the destination on time, it may happen that both the original and the resent packet arrive at the destination.

Q23-3. We check each protocol one by one:

- **a.** The protocol can be Stop-and-Wait with the receive window size of 1 and the send window size of 1.
- **b.** The protocol can also be Go-Back-*N* with the receive window size of 1 and the send window size of *n* packets.
- **c.** The protocol cannot be Selective-Repeat because the size of both windows should be the same (1), which means the protocol is Stop-and-Wait, not Selective-Repeat.

Q23-4. The organization needs to select a port number from the registered range, 1024 to 49,151, and register that port number with ICANN. If the port number is already in use, ICANN informs the organization to choose another port number in this range.

Q23-5. The networks need to be carefully designed to make the time between the two wraparounds as long as possible. For example, in a protocol that uses the sequence number field of size 3 ($m = 3$), every $2^m = 8$ packets have the same sequence number. If the previous packet with sequence number x (or its accidentally created duplicate) is still wandering in the network arrives at the destination, the receiver may confuse this with the expected new packet, also with sequence number *x*.

Q23-6. The transport-layer packets are encapsulated in the datagram at the network layer. Each IP packet may travel a different route and arrives at the destination with a different delay. If a packet encounters more delay than the next packet, it will be received out of order.

Q23-7. The answer is no. Host-to-host and process-to-process communication are needed because each computer in the Internet is designed to do multiple tasks: to run multiple application-layer programs.

Q23-8. Although this can be done for client processes, it is very inefficient, if not impossible, for the server processes. When a server process starts running on the server, its number should be advertised by the server to all possible clients that need to contact that process.

Q23-9. Although any port number can be used for the client and server and they can be the same in this private communication, it is recommended to follow the division specified by ICANN:

a. The client port number should be chosen from the dynamic range, 49,152 to 65,535.

b. The server port number also should be chosen from the dynamic range, 49,152 to 65,535.

c. It is advisable to choose different port numbers for the server and the client to be able to better debug the programs.

Q23-10. There can be several packets still in transit. The size of the receive window is chosen to be the same as the size of the send window to accommodate out-of-order packets until a set of packets all in order arrives. The protocol does not want to deliver the out-of-order packets to the application-layer protocol.

Q23-11. The rest of the packets ($2^m - 2$) are supposed to be in transit, filling the pipe. The size of the receive window is chosen to be 1 to accept only one packet, the one expected, and not out-of order packets. The receiver cannot be overwhelmed because it holds only one packet in its window. When the only packet in the window is consumed by the upper-layer protocol, the receive window slides to make it possible to receive the next packet in transit. If any packet in transit arrives before the window slides, it is discarded.

Q23-12. We check each protocol one by one:

a. The protocol cannot be the Stop-and-Wait because the size of the send window should be 1.

b. The protocol can be the Go-Back-*N* with the send window size of 20 and the receive window size of 1 packet.

c. The protocol can be Selective-Repeat with the send window size of 20 and the receive window size of 20.

Q23-13. We describe the advantage and disadvantage of each first:

a. The advantage of using the Go-Back-*N* protocol is that we can have a larger send window size. We can send more packets before waiting for their acknowledgment. The disadvantage of using this protocol is that the receive window size is only 1. The receiver cannot accept and store the out-of-order received packets; they will be discarded. Discarding of the out-of-order packets means resending these packets by the sender, resulting in congestion of the network and reducing the capacity of the pipe. So the advantage seen by a larger send window may disappear by filling the network with resent packets.

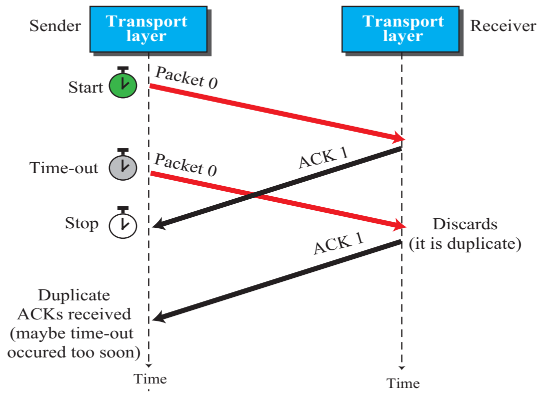
b. The advantage of using the Selective-Repeat protocol is that the receive window can be much larger than 1. This allows the receive window to store the out-of-order packets and avoids resending them to congest the network. The disadvantage of this protocol is that the send window size is half of the Go-Back-*N*, which means that we can send fewer packets before waiting for the acknowledgment.

We can conclude that if the bandwidth-delay product of the network is large, the reliability is good, and the delay is low, we should choose the Go-Back-*N* protocol to use more of the network capacity. On the other hand, if the bandwidth-delay product is small, or the network is not very reliable, or the network creates long delays, we need to use Selective-Repeat.

Q23-14. Sequence numbers use modulo 2^m arithmetic. This means that if a packet has the sequence number x, 2^m packets need to be passed to see the same sequence number if each packet uses only one sequence number (not in TCP, where the sequence number defines the number of the first byte in the packet).

Problems

P23-1. The following figure shows the case. It happens when an ACK is delayed and the time-out occurs. The sender resends the packet that is already acknowledged by the receiver. The receiver discards the duplicate packet, but resends the previous ACK to inform the sender that there is a delay in the network.



P23-2. Since port numbers have local jurisdiction and can be repeated, the division distinguishes between client processes and server processes to avoid confusion when a host runs client and server processes at the same time. For example, assume host A is running a client process with the port number *x*, which has sent out a request and is waiting for a response from the corresponding server. Host B sends a request to a server process with the port number *x*, which is received by host A. Host A erroneously passes the request to the client process with port number *x*, assuming that this is the response that the client process is waiting for. If client and server were using different port numbers, *x* and *y* for example, the request received by host A would be dropped because no server with port number *y* was running. ICANN has also divided the server port numbers into two groups. The well-known port numbers are recognized through the whole Internet society; the registered port numbers are those that do not have a universal jurisdiction yet.

P23-3. We first calculate the average round-trip time (RTT) and the number of packets in the pipe before finding the sizes of the windows, the value of *m*, and the time-out value.

a. Average RTT $= 2 \times (5{,}000\text{ Km}) / (2 \times 10^8) = 50$ ms.

b. The bandwidth-delay product = 1 Gbps × 50 ms = 50,000,000 bits.

c. The bandwidth-delay product = 50,000,000 bits / 50,000 bits = 1000 packets.

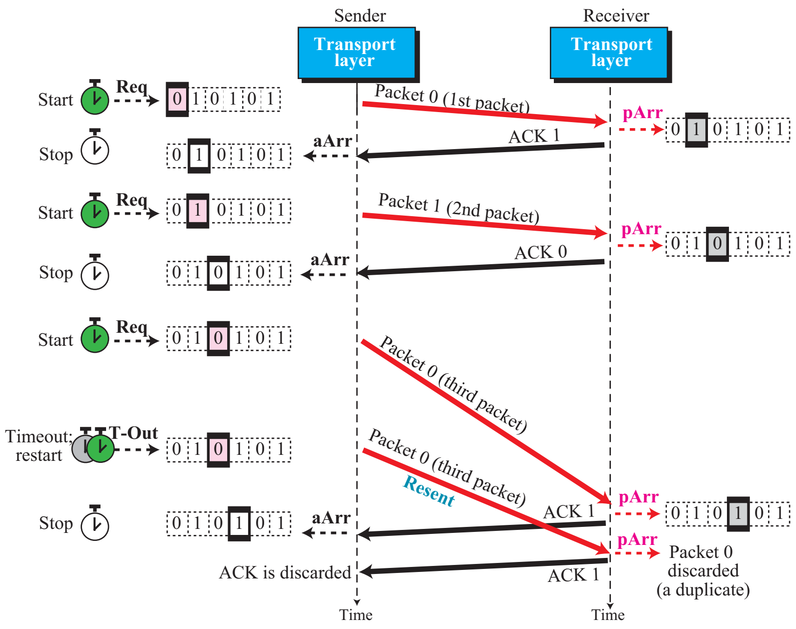
d. The maximum send window size should be 1000 to allow not more than 1000 packets in the pipe.

e. The maximum receive window size should also be 1000 packets.

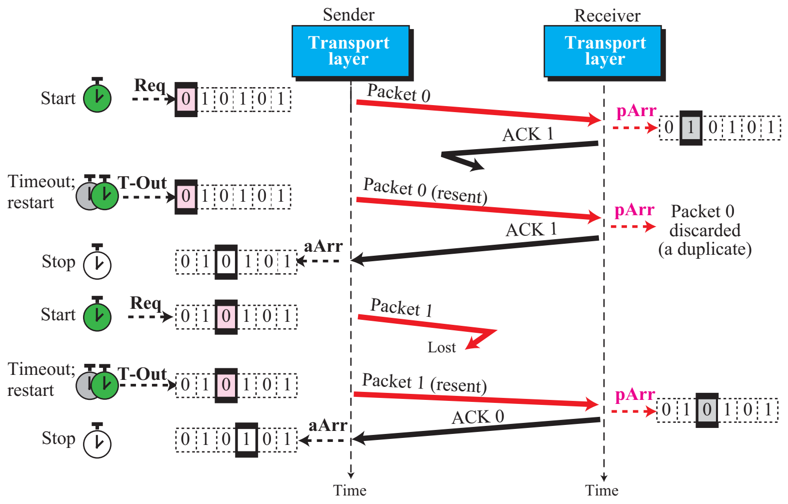
f. We know that the (window size) $\leq (2^{m-1})$ or 1000 $\leq (2^{m-1})$. This means that we need to choose $(m - 1)$ to be at least 10 or $m = 11$. The sequence numbers are then 0 to 2047.

g. The timeout value should be at least the average RTT = 50 ms to avoid early retransmission of the packets and to prevent congestion.

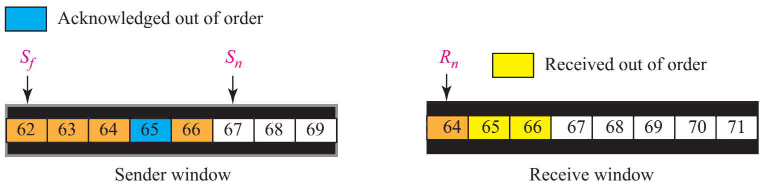
P23-4. The following figure shows the flow diagram:



P23-5. See the following figure:



P23-6. The following figure shows the situation.



a. The sender has sent packets 62 to 66; the receiver is expecting only packet 64 in this set (packets 62 and 63 have already been received and acknowledged). This means there is only one pending packet, packet 64.

b. The receiver has sent ACKs 62, 63, 65 and 66, but the sender has received only ACK 65. This means ACKs 62, 63, and 66 are pending.

P23-7. In each case we first define the bandwidth-delay product (BDP) in bits and then find it in the number of packets:

a. BDP = 1 Mbps × 20 ms = 20,000 bits = 20 packets

b. BDP = 10 Mbps × 20 ms = 200,000 bits = 100 packets

c. BDP = 1 Gbps × 4 ms = 4,000,000 bits = 400 packets

P23-8. We assume each event is independent.

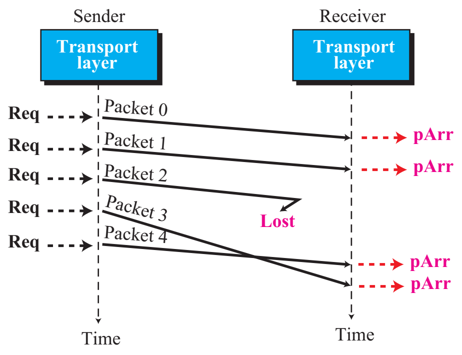
a. seqNo = 0.

b. seqNo = 1.

c. Deliver the packet, slide the window forward (R = 0), and send an ACK with ackNo = 1.

d. Discard the packet, and send an ACK with ackNo = 1.

P23-9. The figure on the next page shows the outline. Note that since the simple protocol provides no error control, if a packet is lost, the receiving process is responsible for finding a solution. The transport layer is not even aware that this has happened. The packets may also be delivered out of order to the receiving process. The responsibility again is on the receiving process to reorder the packets.



P23-10. We assume each event is independent.

a. seqNo = 15.

b. Five packets with seqNos 10, 11, 12, 13, and 14 are resent.

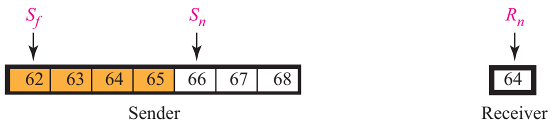
c. $S_f = 10$; $S_n = 15$. Packet with seqNo = 13 is marked as acknowledged.

d. Size of the window depends on what is set at the beginning.

e. $S_f = 17$ and $S_n = 21$. The machine moves to the ready state.

f. An ACK with ackNo = 16 is sent. Packet with seqNo = 16 is delivered to the application layer. $R_n = 17$.

P23-11. The following figure shows the situation.



a. If the receiver expects a packet with sequence number 64 and packets with sequence numbers 62 to 65 are already sent but not acknowledged, it means that two packets with sequence numbers 64 and 65 are in transit from the sender to the receiver.

b. If the sender expects the acknowledgment for packet 62, but the value of R_n = 64, it means that the ACK packets with acknowledgment numbers 62 and 63 are in transit from the receiver to the sender.

P23-12. We first calculate the average round-trip time (RTT) and the number of packets in the pipe before finding the sizes of the windows, the value of *m*, and the time-out value.

a. Average RTT $= 2 \times (10{,}000 \text{ Km}) / (2 \times 10^8) = 100$ ms.

b. The bandwidth-delay product = 100 Mbps × 100 ms = 10,000,000 bits.

c. The bandwidth-delay product = 10,000,000 bits /100,000 bits = 100 packets.

d. The maximum send window size should be 100 to allow not more than 100 packets in the pipe.

e. The maximum receive window size is 1 in Go-Back-*N*.

f. We know that the (send window size) $\leq (2^m - 1)$ or $100 \leq (2^m - 1)$. This means that we need to choose *m* to be at least 7. The sequence numbers are then 0 to 127.

g. The time-out value should be at least the average RTT = 100 ms to avoid early retransmission of the packets and to prevent congestion.

P23-13. We assume each event is independent.

a. seqNo = 15.

b. Five packets with seqNos set to 10, 11, 12, 13, and 14 are to be resent.

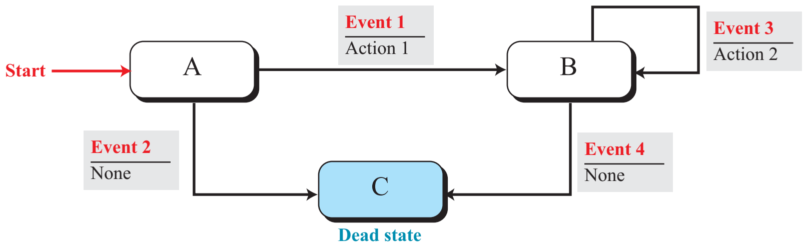
c. $S_f = 13$ and $S_n = 15$.

d. The size of the window remains the same. Max $W_{size} = 64 - 1 = 63$.

e. $S_f = 18$ and $S_n = 21$. Next state = ready.

f. $R_n = 17$. Action: message is delivered and an ACK with ackNo = 17 is sent.

P23-14. The following figure shows the states and events:



P23-15. The domain of IP addresses is universal. A device directly connected to the Internet needs a unique IP address. The domain of port numbers is local; they can be repeated. Two computers running the HTTP server process use the same well-known port number (80); two computers running the HTTP client process can use the same ephemeral port number.

P23-16. The wraparound depends on the value of m.

a. In the Stop-and-Wait protocol, $m = 1$, every $2^m = 2$ packets have the same sequence number.

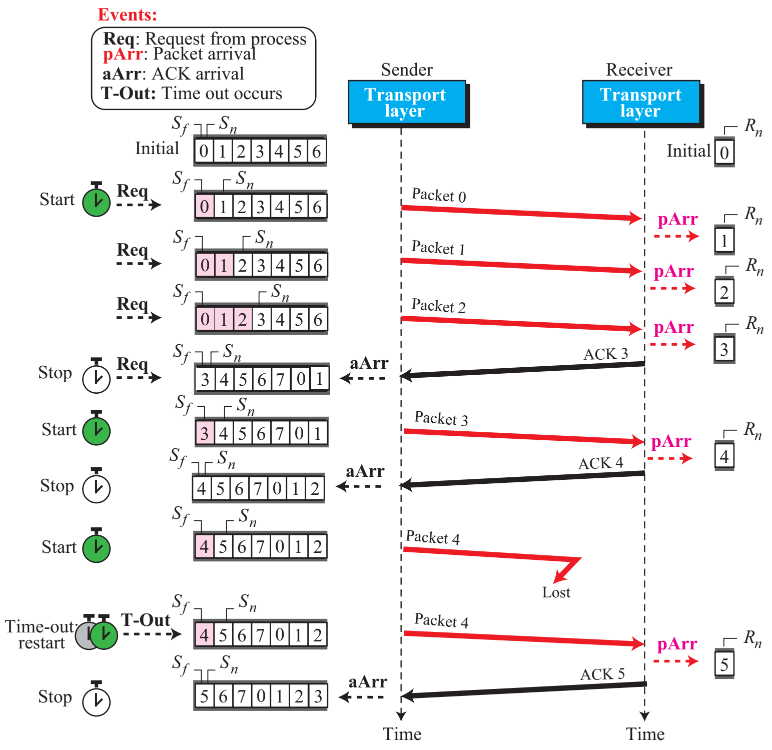
b. In the Go-Back-N protocol with $m = 8$, every $2^m = 256$ packets have the same sequence number.

c. In the Selective-Repeat protocol with $m = 8$, every $2^m = 256$ packets have the same sequence number.

P23-17.

Protocol	Max Send W_{size}	Max Receive W_{size}
Stop-and-Wait:	1	1
Go-Back-N:	$2^5 - 1 = 31$	1
Selective-Repeat:	$2^5 / 2 = 16$	$2^5 / 2 = 16$

P23-18. See the following figure:



P23-19. The sequence number of any packet can be found using the following relation:

$$\textbf{seqNo = (starting segNo + packet number} - \textbf{1) mod } 2^m$$

in which m is the number of bits used to define the sequence number. The sequence number in this case is

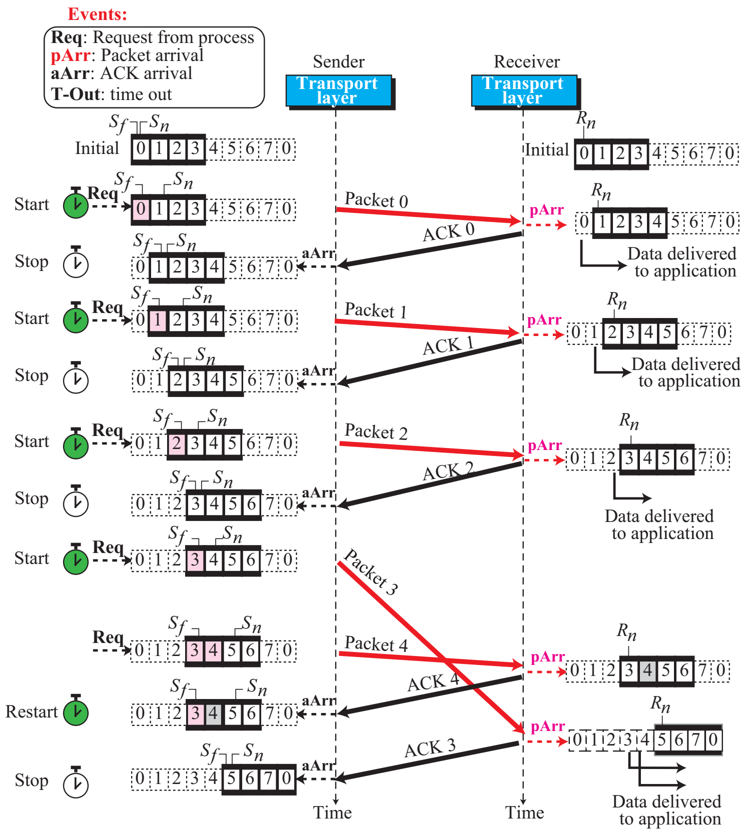
$$\text{seqNo} = (0 + 100 - 1) \bmod 2^5 = 99 \bmod 32 = 3$$

P23-20.

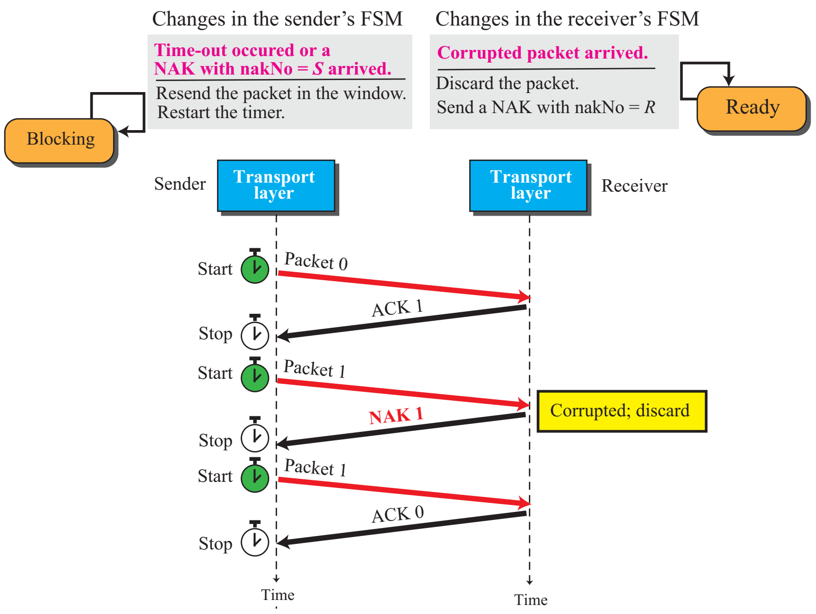
a. In the Go-Back-*N* protocol, since the receiver does not accept any out- of-order data packets, acknowledgments are cumulative, which means that the receiver does not have to acknowledge any single data packet received. Several packets can be acknowledged at a time, saving in the number of ACK packets. In this system, a single acknowledgment number defines the next packet that the receiver expects to receive to allow the sender to purge the whole range of packets whose sequence number is less than the acknowledgment number. For example, if the acknowledgment number is ackNo = 27, the sender can purge all of the outstanding packets whose sequence number is less than 27. The acknowledgment number clearly defines the first slot in the sender window after sliding.

b. In the Selective-Repeat protocol, the receiver can accept out-of-order packets and store them until the missing packets arrive. In this protocol, the acknowledgment number cannot be cumulative, because several packets before the one received may not have been received yet. Nor can't the acknowledgment define the next packet expected, because the next packet may be a packet with sequence number before or after the one received. The acknowledgment number exactly defines the sequence number of the packet received.

P23-21. The following figure shows the solutions:

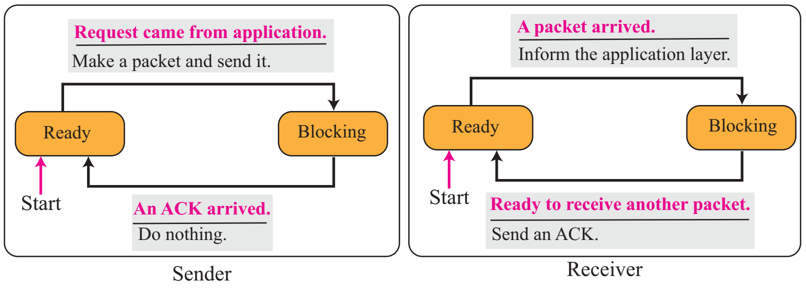


P23-22. In the receiver's FSM, we change the event when a corrupted packet arrives. Both changes are shown in the following figure. The figure shows only changes, the rest of the Figure 23.21 remain the same. An example follows the changes.



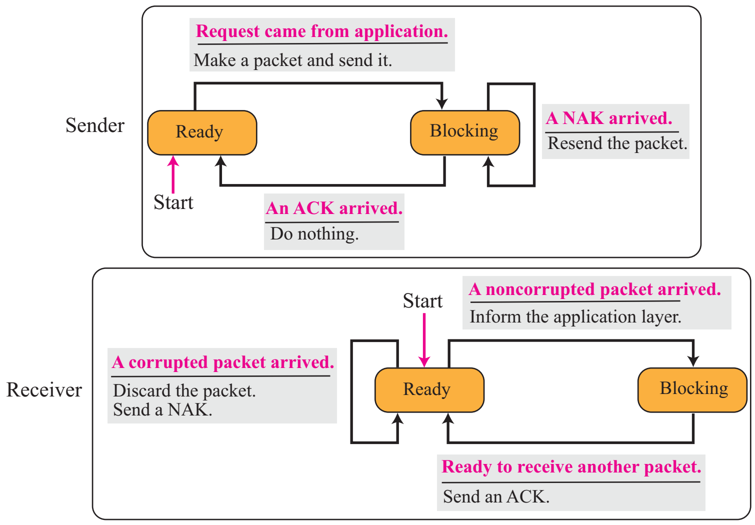
Changes in the sender's FSM
Time-out occured or a NAK with nakNo = S arrived.
Resend the packet in the window. Restart the timer.
Blocking
Changes in the receiver's FSM
Corrupted packet arrived.
Discard the packet. Send a NAK with nakNo = R
Ready
Sender
Transport layer
Transport layer
Receiver
Start
Packet 0
ACK 1
Stop
Start
Packet 1
Corrupted; discard
NAK 1
Stop
Start
Packet 1
ACK 0
Stop
Time
Time

P23-23. The following figure shows the states and events. The sender needs two states: ready and blocking. The receiver also needs these two states. To send a packet, the sender should be in the ready state and receives data from the application layer. The receiver accepts a packet from the sender when it is in the ready state. When the receiver becomes ready, it sends an ACK packet and moves to the ready state.



Request came from application.
Make a packet and send it.
Ready
Blocking
Start
An ACK arrived.
Do nothing.
Sender
A packet arrived.
Inform the application layer.
Ready
Blocking
Start
Ready to receive another packet.
Send an ACK.
Receiver

P23-24. The following figure shows the states and events. The sender needs two states: ready and blocking. The receiver also needs two states. To send a packet, the sender should be in the ready state and receives data from the application layer. When in the blocking state, if the sender gets a NAK, it resends the packet and remains in the blocking state. When the sender receives an ACK, it moves to the ready state and waits until data comes from the application layer. When the receiver is in the ready state, it can accept a packet. If the packet is corrupted, it sends a NAK and remains in that state.


Sender
Request came from application.
Make a packet and send it.
Ready
Blocking
A NAK arrived.
Resend the packet.
Start
An ACK arrived.
Do nothing.
Receiver
Start
A noncorrupted packet arrived.
Inform the application layer.
A corrupted packet arrived.
Discard the packet.
Send a NAK.
Ready
Blocking
Ready to receive another packet.
Send an ACK.