Go-Back-*N* Protocol (GBN)

- To improve the efficiency of transmission (to fill the pipe), multiple packets must be in transition while the sender is waiting for acknowledgment.
- The key to Go-back-*N* is that we can send several packets before receiving acknowledgments, but the receiver can only buffer one packet. We keep a copy of the sent packets until the acknowledgments arrive.

Sequence Numbers:

- The sequence numbers are modulo 2*m*,

Acknowledgment Numbers:

- **In the Go-Back-*N* protocol, the acknowledgment number is cumulative and defines the sequence number of the next packet expected to arrive.**

Send Window:

- The send window is an imaginary box covering the sequence numbers of the data packets that can be in transit or can be sent.
- The send window at any time divides the possible sequence numbers into four regions. The first region, left of the window, defines the sequence numbers belonging to packets that are already acknowledged. The sender does not worry about these packets and keeps no copies of them.
- The second region, colored, defines the range of sequence numbers belonging to the packets that have been sent, but have an unknown status. The sender needs to wait to find out if these packets have been received or were lost. We call these *outstanding* packets.
- The third range, white in the figure, defines the range of sequence numbers for packets that can be sent; however, the corresponding data have not yet been received from the application layer.
- Finally, the fourth region, right of the window, defines sequence numbers that cannot be used until the window slides.

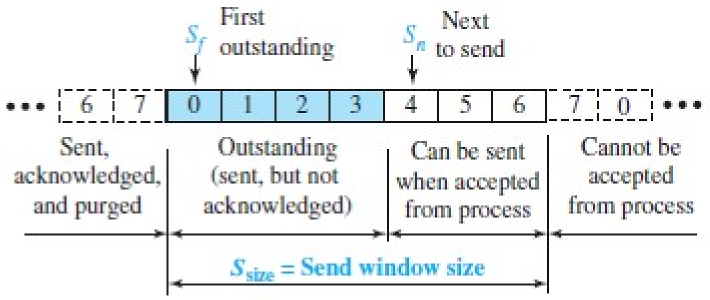

S_f First outstanding
S_n Next to send
6 7 0 1 2 3 4 5 6 7 0
Sent, acknowledged, and purged
Outstanding (sent, but not acknowledged)
Can be sent when accepted from process
Cannot be accepted from process
S_{size} = Send window size

Fig: Send window for Go – Back – N.

Receive Window:

- The receive window makes sure that the correct data packets are received and that the correct acknowledgments are sent. In Go-Back-*N*, the size of the receive window is always 1.
- The receiver is always looking for the arrival of a specific packet. Any packetarriving out of order is discarded and needs to be resent.

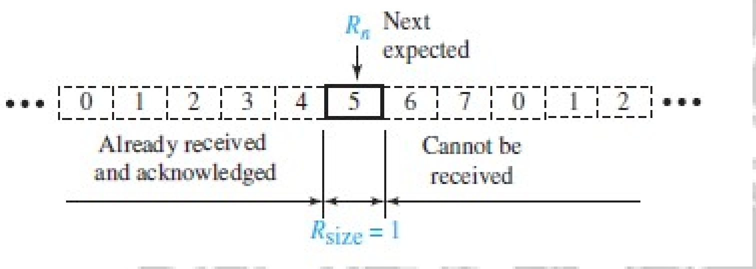


Fig: Receive window for Go- back –N.

- **The receive window is an abstract concept defining an imaginary box of size 1 with a single variable *Rn*. The window slides when a correct packet has arrived; sliding occurs one slot at a time.**

Timers:

- Although there can be a timer for each packet that is sent, in our protocol we use only one. The reason is that the timer for the first outstanding packet always expires first. Were send all outstanding packets when this timer expires.

Resending packets:

- When the timer expires, the sender resends all outstanding packets.

FSMs

Sender

- The sender starts in the ready state, but thereafter it can be in one of the two states :*ready* or *blocking*.

Ready state. Four events may occur when the sender is in ready state.

a. If a request comes from the application layer, the sender creates a packet with the sequence number set to *Sn*. A copy of the packet is stored, and the packet is sent. If the window is full, *Sn* = (*Sf* +*S*size) modulo 2*m*, the sender goes to the blocking state.
b. If an error-free ACK arrives with ackNo related to one of the outstanding packets, the sender slides the window (set *Sf* = ackNo), and if all outstanding packets are acknowledged (ackNo =

Sn), then the timer is stopped. If all outstanding packets are not acknowledged, the timer is restarted.

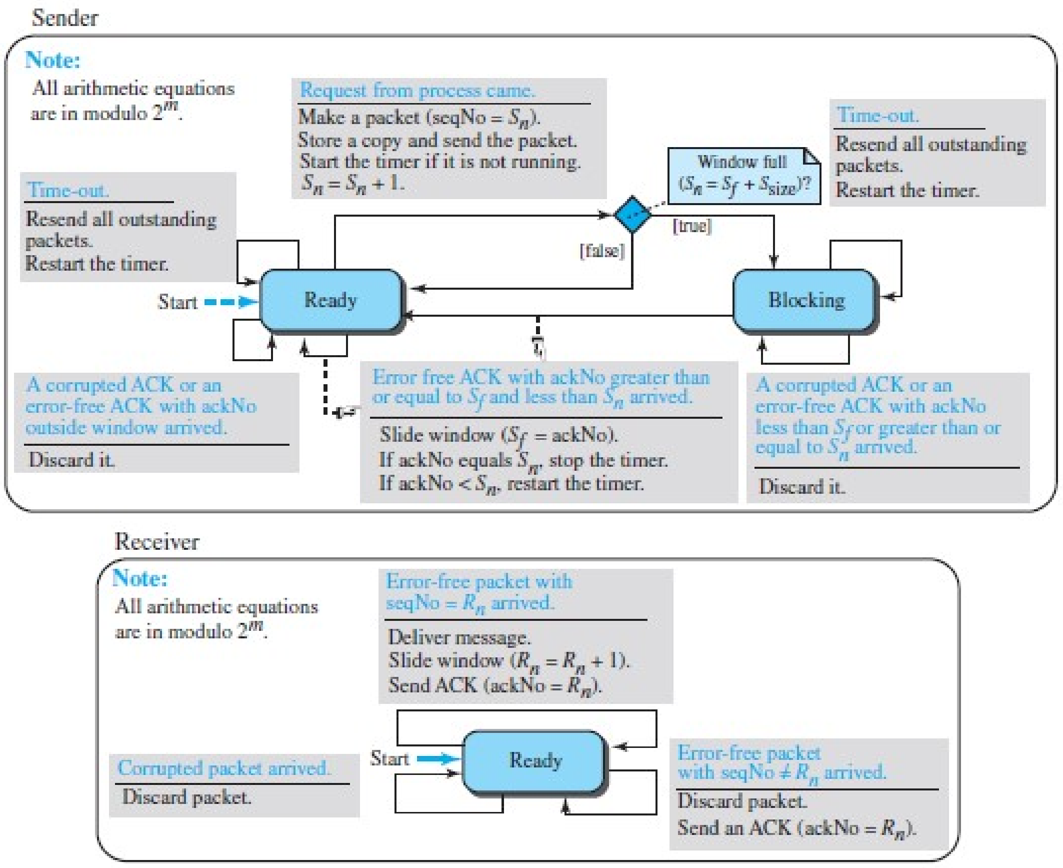

Sender
Note: All arithmetic equations are in modulo 2^m.
Request from process came.
Make a packet (seqNo = S_n). Store a copy and send the packet. Start the timer if it is not running. $S_n = S_n + 1$.
Window full ($S_n = S_f + S_{size}$)?
[true]
[false]
Time-out. Resend all outstanding packets. Restart the timer.
Start
Ready
Blocking
A corrupted ACK or an error-free ACK with ackNo outside window arrived. Discard it.
Error free ACK with ackNo greater than or equal to S_f and less than S_n arrived.
Slide window (S_f = ackNo). If ackNo equals S_n, stop the timer. If ackNo < S_n, restart the timer.
A corrupted ACK or an error-free ACK with ackNo less than S_f or greater than or equal to S_n arrived. Discard it.
Receiver
Note: All arithmetic equations are in modulo 2^m.
Error-free packet with seqNo = R_n arrived.
Deliver message. Slide window ($R_n = R_n + 1$). Send ACK (ackNo = R_n).
Corrupted packet arrived. Discard packet.
Start
Ready
Error-free packet with seqNo ≠ R_n arrived. Discard packet. Send an ACK (ackNo = R_n).

Fig: FSMs for the Go-Back-N protocol

c. If a corrupted ACK or an error-free ACK with ackNo not related to the outstanding packet arrives, it is discarded.

d. If a time-out occurs, the sender resends all outstanding packets and restarts the timer.

Blocking state. Three events may occur in this case:

a. If an error-free ACK arrives with ackNo related to one of the outstanding packets, the sender slides the window (set *Sf* = ackNo) and if all outstanding packets are acknowledged (ackNo = *Sn*), then the timer is stopped.

b. If all outstanding packets are not acknowledged, the timer is restarted. The sender then moves to the ready state.

c. If a corrupted ACK or an error-free ACK with the ackNo not related to the outstanding packets arrives, the ACK is discarded.

d. If a time-out occurs, the sender sends all outstanding packets and restarts the timer.

Receiver

- The receiver is always in the *ready* state. The only variable, *Rn*, is initialized to 0. Three events may occur:

a. If an error-free packet with seqNo = *Rn* arrives, the message in the packet is delivered to the application layer. The window then slides, *Rn* = (*Rn* + 1) modulo 2*m*.Finally an ACK is sent with ackNo = *Rn*.

b. If an error-free packet with seqNo outside the window arrives, the packet is discarded, but an ACK with ackNo = *Rn* is sent.

c. If a corrupted packet arrives, it is discarded.

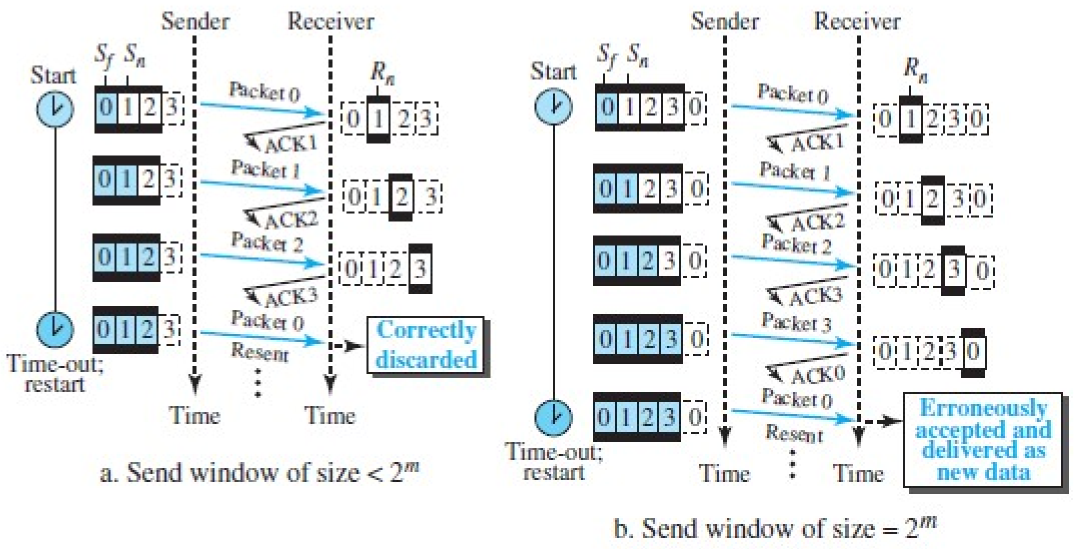
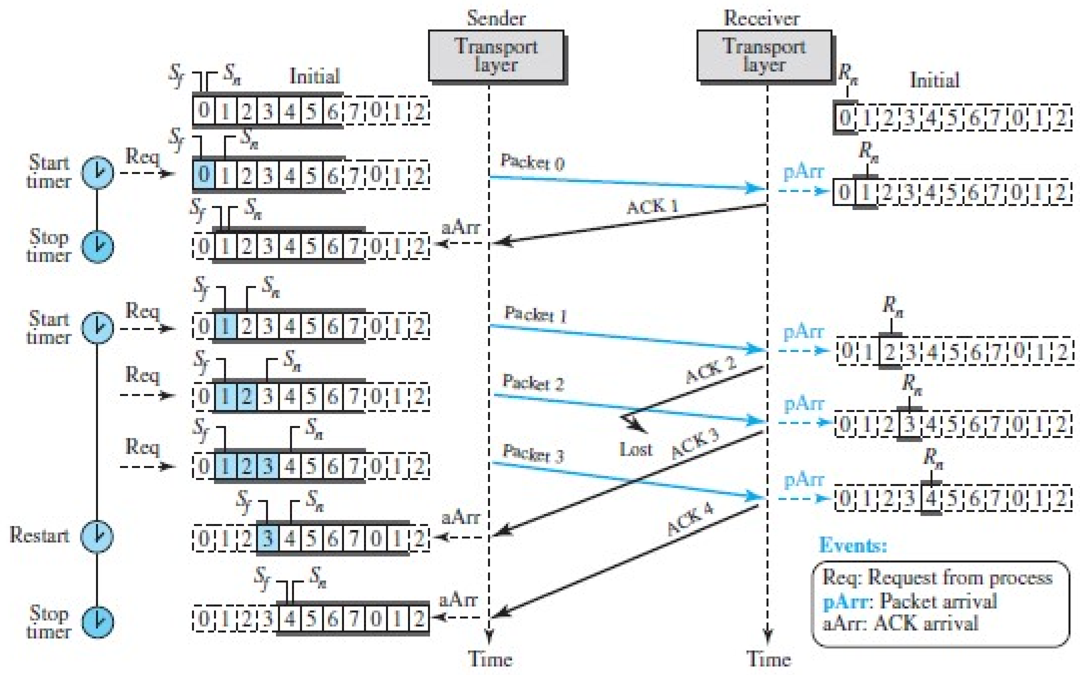


Fig: Send window size for Go – Back –N.



Fig; shows an example of Go- back – N.

Example: 1

Figure 2.3 shows what happens when a packet is lost. Packets 0, 1, 2, and 3 are sent. However, packet 1 is lost. The receiver receives packets 2 and 3, but they are discarded because they are received out of order (packet 1 is expected). When the receiver receives packets 2 and 3, it sends ACK1 to show that it expects to receive packet 1. However, these ACKs are not useful for the sender because the ackNo is equal to *Sf* , not greater than *Sf*. So the sender discards them. When the time-out occurs, the sender resends packets 1, 2, and 3, which are acknowledged.

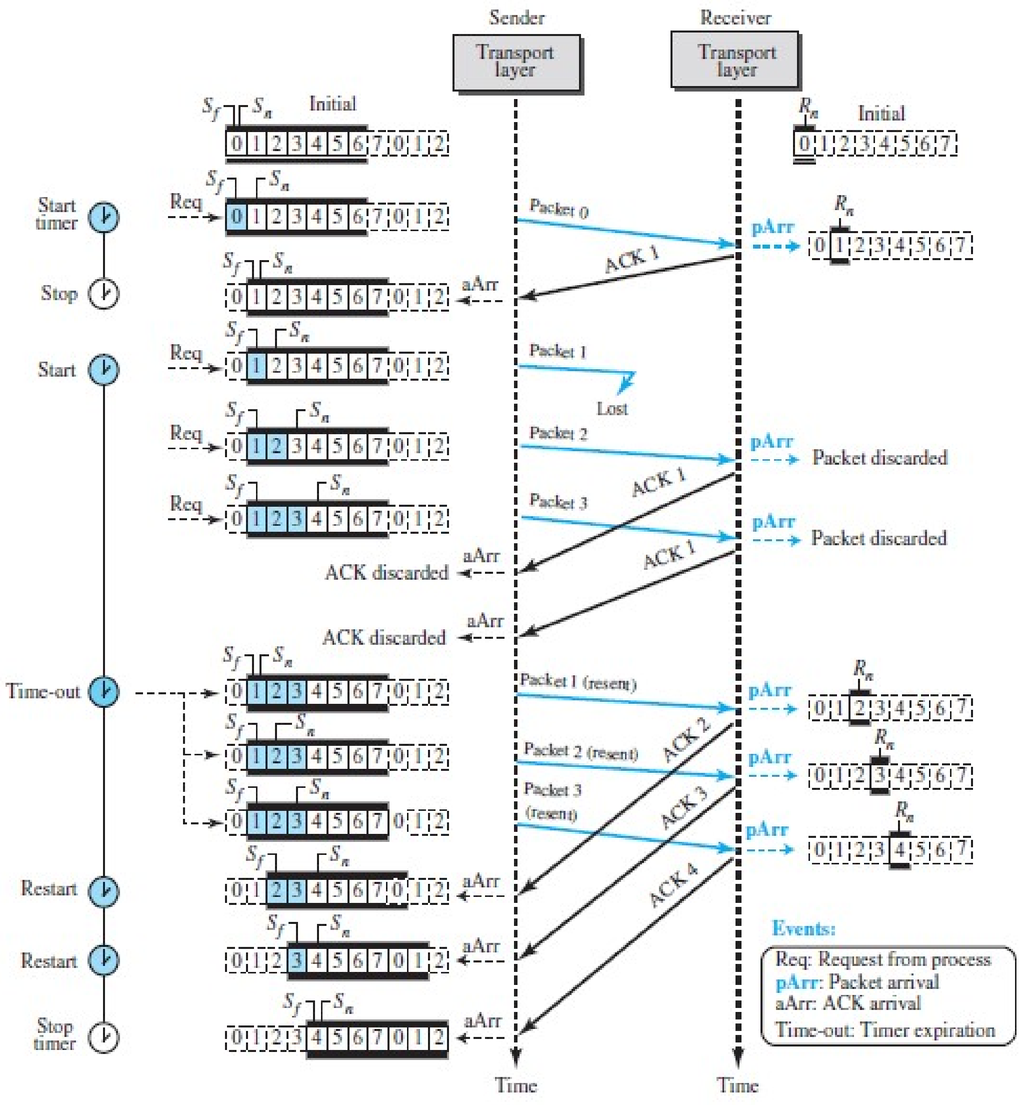


Fig: Flow diagram for Example 1