Stop-and-Wait Protocol

- Connection-oriented protocol called the **Stop-and-Wait protocol,** which uses both flow and error control. Both the sender and the receiver use a sliding window of size 1. The sender sends one packet at a time and waits for an acknowledgment before sending the next one.
- To detect corrupted packets, we need to add a checksum to each data packet. When a packet arrives at the receiver site, it is checked. If its checksum is incorrect, the packet is corrupted and silently discarded.
- The silence of the receiver is a signal for the sender that a packet was either corrupted or lost. Every time the sender sends a packet, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next packet (if it has one to send).
- If the timer expires, the sender resends the previous packet, assuming that the packet was either lost or corrupted. This means that the sender needs to keep a copy of the packet until its acknowledgment arrives.

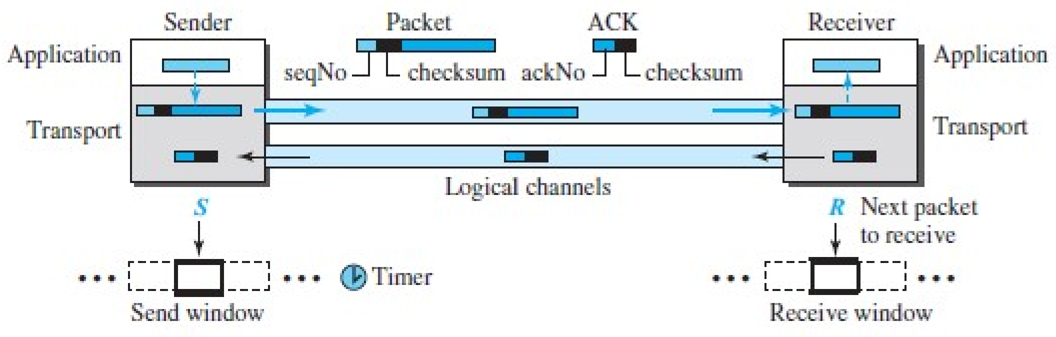

Fig: Stop – and – wait protocol.

- The Stop-and-Wait protocol is a connection-oriented protocol that provides flow and error control.

Sequence Numbers:

- To prevent duplicate packets, the protocol uses sequence numbers and acknowledgment numbers. The packet arrives safe and sound at the receiver site; the receiver sends an acknowledgment.
- The acknowledgment arrives at the sender site, causing the sender to send the next packet numbered $x + 1$.

- The packet is corrupted or never arrives at the receiver site; the sender resends the packet (numbered *x*) after the time-out.
- The packet arrives safe and sound at the receiver site; the receiver sends an acknowledgment, but the acknowledgment is corrupted or lost.
- The sender resends the packet (numbered *x*) after the time-out. Note that the packet here is a duplicate. The acknowledgment numbers always announce the sequence number of the *next packet expected* by the receiver.
- **In the Stop-and-Wait protocol, the acknowledgment number always announces, inmodulo-2 arithmetic, the sequence number of the next packet expected.**

FSMs:

- FSMs for the Stop-and-Wait protocol. Since the protocol is aconnection-oriented protocol, both ends should be in the *established* state beforeexchanging data packets. The states are actually nested in the *established* state.

Sender

- The sender is initially in the ready state, but it can move between the ready and blockingstate. The variable *S* is initialized to 0.

➢ ***Ready state.***

- When the sender is in this state, it is only waiting for one event to occur. If a request comes from the application layer, the sender creates a packet with the sequence number set to *S*. A copy of the packet is stored, and the packet is sent. The sender then starts the only timer. The sender then moves to the blocking state.

➢ ***Blocking state.***

- When the sender is in this state, three events can occur:
    a. If an error-free ACK arrives with the ackNo related to the next packet to be sent, then the timer is stopped. Finally, the sender moves to the ready state.
    b. If a corrupted ACK or an error-free ACK with the ackNo ≠ (S + 1) modulo 2arrives, the ACK is discarded.
    c. If a time-out occurs, the sender resends the only outstanding packet and restarts the timer.

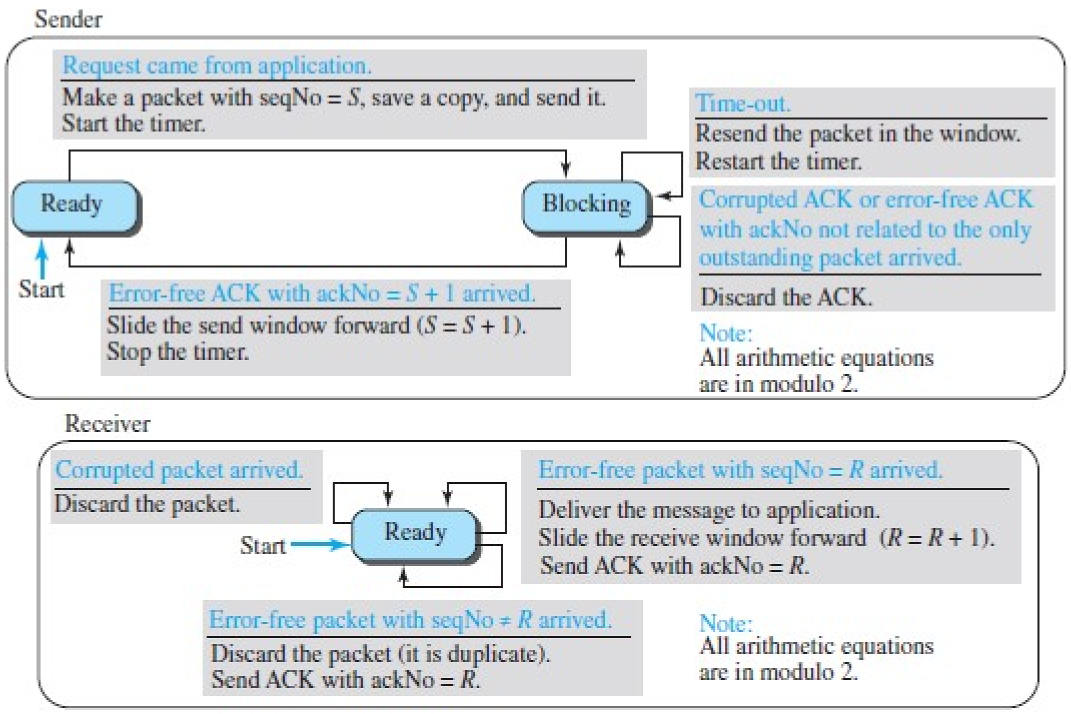
Sender
Request came from application.
Make a packet with seqNo = S, save a copy, and send it. Start the timer.
Time-out.
Resend the packet in the window. Restart the timer.
Ready
Blocking
Start
Corrupted ACK or error-free ACK with ackNo not related to the only outstanding packet arrived.
Discard the ACK.
Error-free ACK with ackNo = S + 1 arrived.
Slide the send window forward (S = S + 1). Stop the timer.
Note: All arithmetic equations are in modulo 2.
Receiver
Corrupted packet arrived.
Discard the packet.
Start
Ready
Error-free packet with seqNo = R arrived.
Deliver the message to application. Slide the receive window forward (R = R + 1). Send ACK with ackNo = R.
Error-free packet with seqNo ≠ R arrived.
Discard the packet (it is duplicate). Send ACK with ackNo = R.
Note: All arithmetic equations are in modulo 2.

Fig: FSMs for the Stop – and – wait protocol.

Receiver

- The receiver is always in the *ready* state. Three events may occur:

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

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

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

Efficiency:

- The Stop-and-Wait protocol is very inefficient if our channel is *thick* and *long*. By *thick,* we mean that our channel has a large bandwidth (high data rate); by *long,* we mean the round-trip delay is long.