DATA-LINK LAYER PROTOCOLS

Four protocols have been defined for the data-link layer to deal with flow and error control: Simple, Stop-and-Wait, Go-Back-N, and Selective- Repeat.

Simple Protocol:

- Our first protocol is a simple protocol with neither flow nor error control. We assume that the receiver can immediately handle any frame it receives.
- In other words, the receiver can never be overwhelmed with incoming frames.

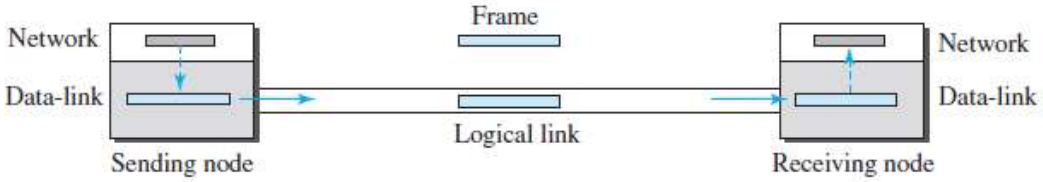


Fig 6: Simple Protocol.

- The data- link layer at the sender gets a packet from its network layer, makes a frame out of it, and sends the frame.
- The data-link layer at the receiver receives a frame from the link, extracts the packet from the frame, and delivers the packet to its network layer.
- The data- link layers of the sender and receiver provide transmission services for their network layers.

FSMs

- The sender site should not send a frame until its network layer has a message to send.
- The receiver site cannot deliver a message to its network layer until a frame arrives.
- Each FSM has only one state, the ready state. The sending machine remains in the ready state until a request comes from the process in the network layer.
- When this event occurs, the sending machine encapsulates the MESSAZGE in a frame and sends it to the receiving machine.
- The receiving machine remains in the ready state until a frame arrives from the sending machine
- When this event occurs, the receiving machine decapsulates the message out of the frame and delivers it to the process at the network layer.

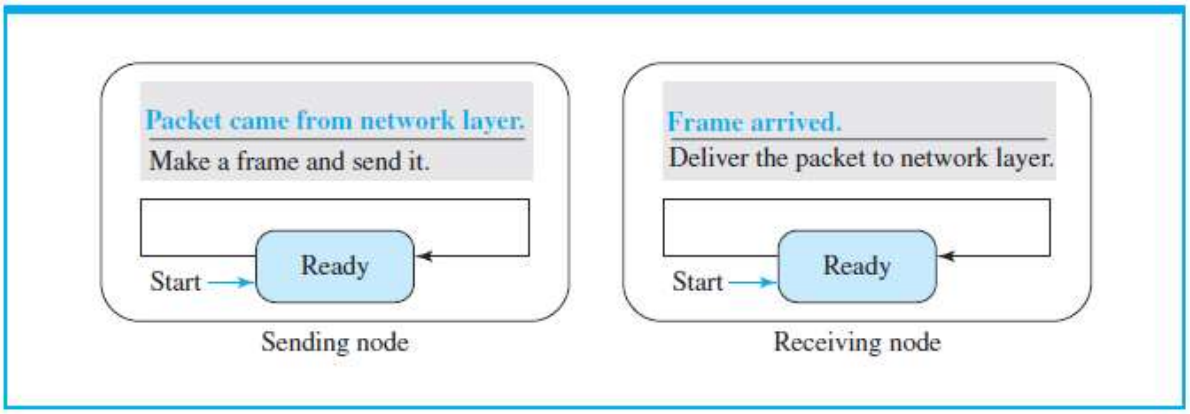


Fig: FSMs for the simple protocol

Stop and Wait Protocol

- Our Second protocol is called the Stop-and-Wait protocol, which uses both flow and error control
- In this protocol, the sender sends one frame at a time and waits for an acknowledgement before sending the next one.
- To detect corrupted frames, we need to add a CRC to each data frame.
- When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded.
- The silence of the receiver is a signal for the sender that a frame was either corrupted or lost.
- Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives

before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send).

- If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted.
- This means that the sender needs to keep a copy of the frame until its acknowledgment arrives.
- When the corresponding acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready.

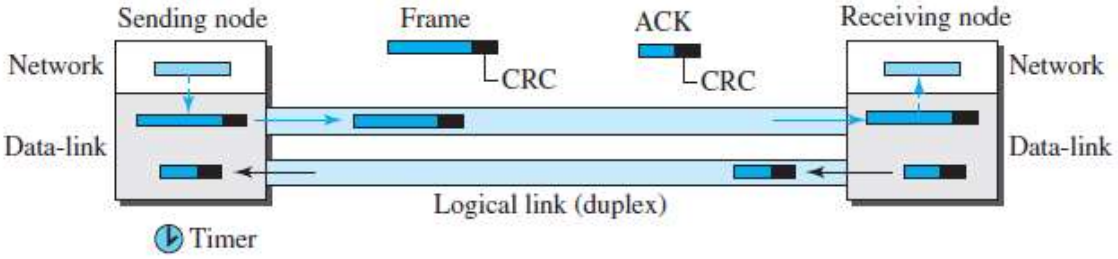


Fig : Stop-and-Wait protocol.

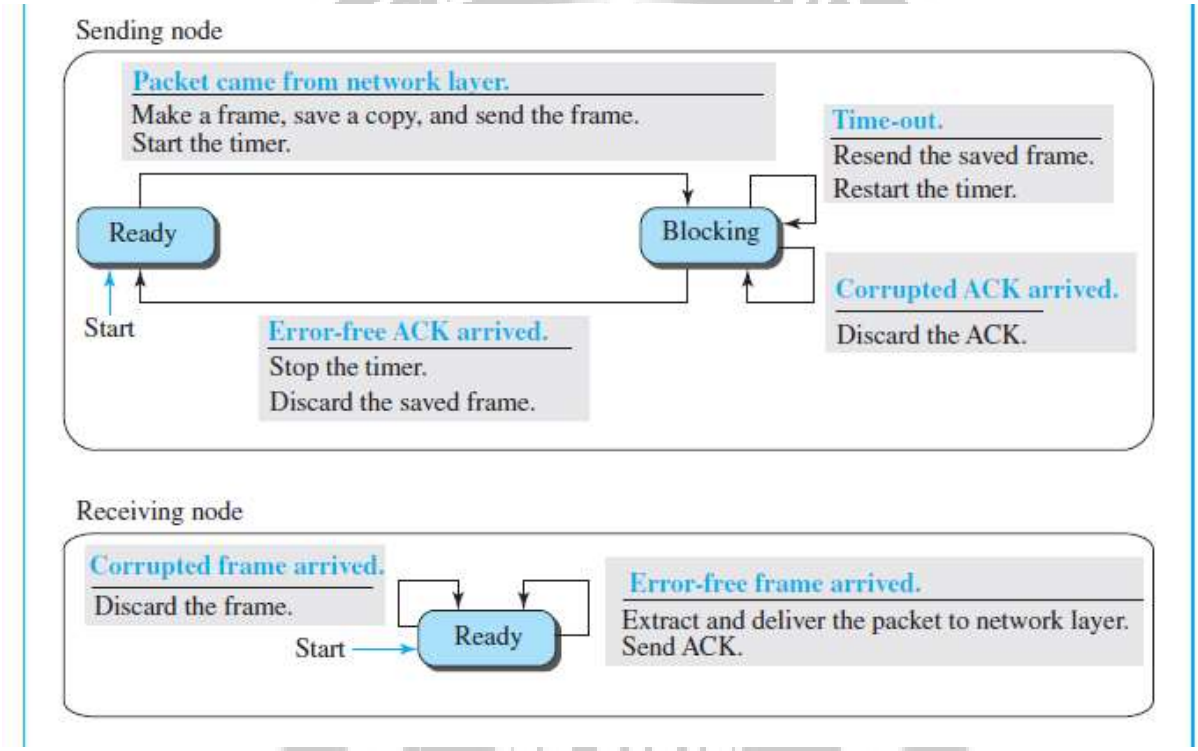


Fig: FSM for the Stop- and- wait protocol.

We describe the sender and receiver states below.

Sender States

- The sender is initially in the ready state, but it can move between the ready and blocking state.

Ready State.

- When the sender is in this state, it is only waiting for a packet from the network layer.
- If a packet comes from the network layer, the sender creates a frame, saves a copy of the frame, starts the only timer and sends the frame.
- Thesender then moves to the blocking state.

Blocking State.

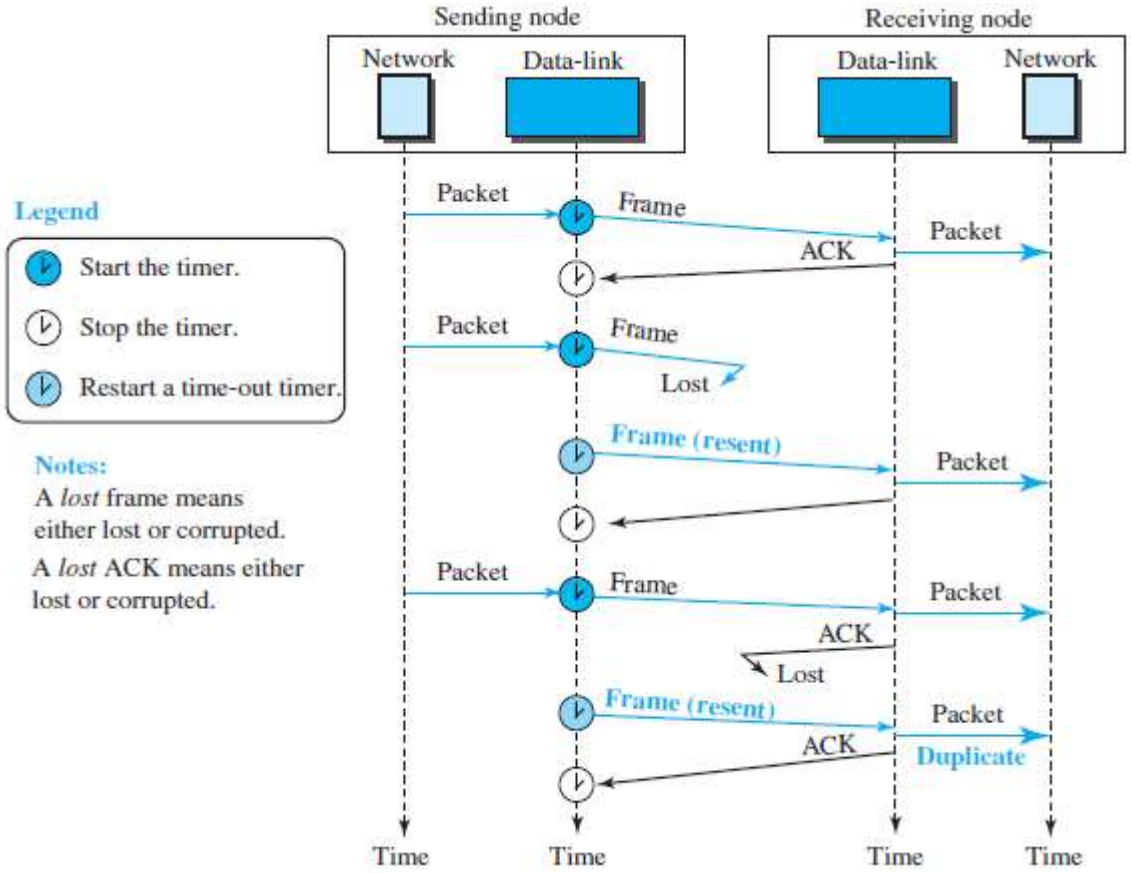
- When the sender is in this state, three events can occur:

a. If a time-out occurs, the sender resends the saved copy of the frame and restartsthe timer.
b. If a corrupted ACK arrives, it is discarded.
c. If an error-free ACK arrives, the sender stops the timer and discards the savedcopy of the frame. It then moves to the ready state.

Receiver

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

a. If an error-free frame arrives, the message in the frame is delivered to the network layer and an ACK is sent.
b. If a corrupted frame arrives, the frame is discarded.



Piggybacking

- The two protocols we discussed in this section are designed for unidirectional communication, in which data is flowing only in one direction although the acknowledgment may travel in the other direction.
- Protocols have been designed in the past to allow data to flow in both directions.

- However, to make the communication more efficient, the data in one direction is piggybacked with the acknowledgment in the other direction.