GLS University

Faculty of Computer Application & IT

SY BCA
Semester - IV
2024-2025

210301404
Data Communication & Networks (DCN)
(Core Subject)

Unit 2
Multiplexing and Demultiplexing

Supplementary Reading :

1. Forouzan, B. A. (2001). Data Communication and Networking. Tata McGraw Hill Education Private Limited.
2. Godbole, A. S. (2002). Data Communication and Computer Networks. Tata McGraw-Hill Companies.

Topics to be Covered :

Concept of Multiplexing and Demultiplexing

- Types of Multiplexing
  - o FDM
  - o TDM
  - o WDM
- FDM versus TDM

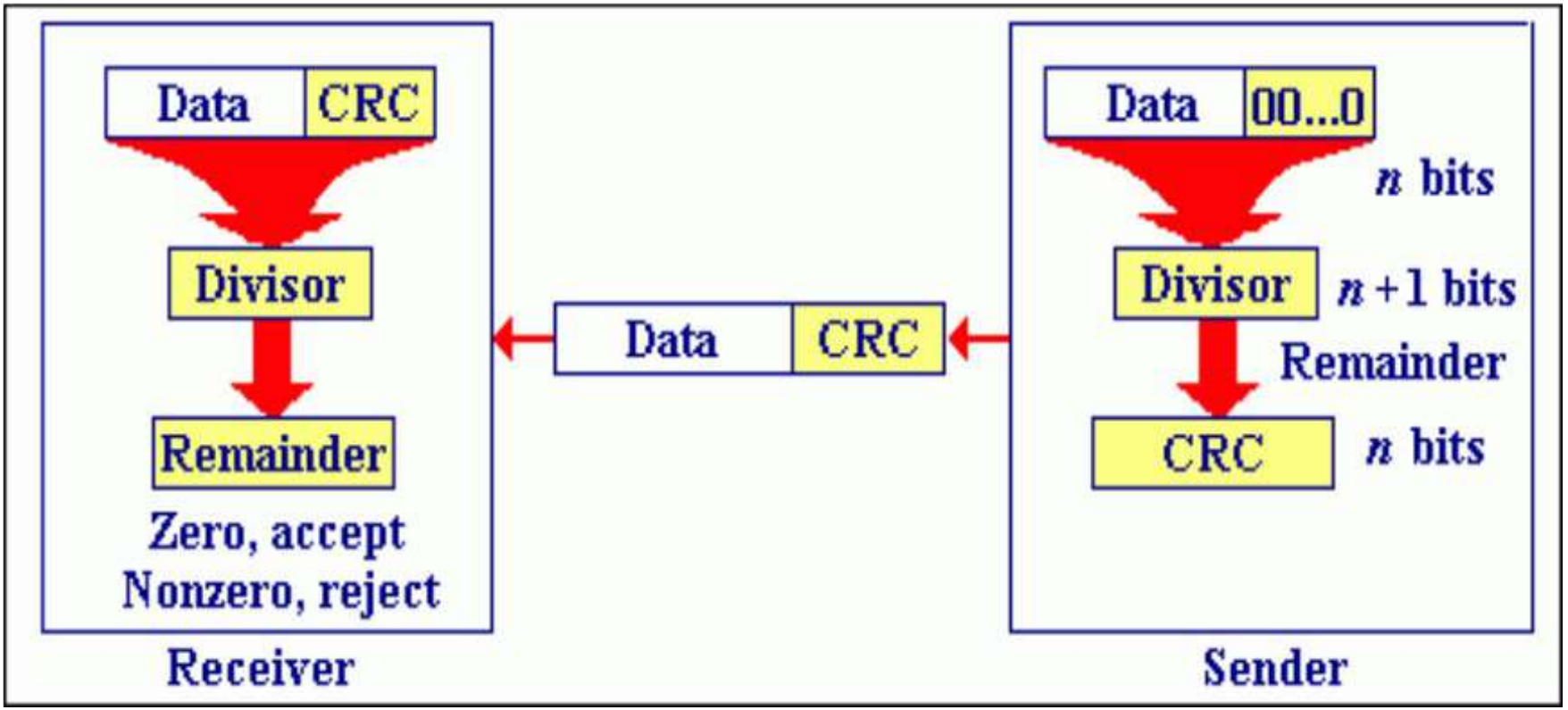
Transmission Errors: Detection and correction

- Introduction
- **Error classification**
  - o Delay Distortion
  - o Attenuation
  - o Noise
- **Types of Error**
- **Error Detection**
  - o Checksum
  - o VRC
  - o LRC
  - o CRC
- **Recovery from errors**
  - o Stop and Wait
  - o Go back n
  - o Sliding Window

Error Detection

- **Cyclic Redundancy Check(CRC):**
- In Cyclic Redundancy Check(CRC) a sequence of redundant overhead bits called CRC or CRC remainder is added to the end of the data to be transmitted.
- The CRC is so calculated that it can be perfectly divided by a second predecided number. At the receiver, the arriving data is divided by the same predecided number.
- If this division produces a zero remainder, the transmission is considered as error-free.
- In such a case, the incoming data is accepted by the receiver. If there is a remainder, it means that the transmission is in error and therefore the arriving data must be rejected.

Error Detection



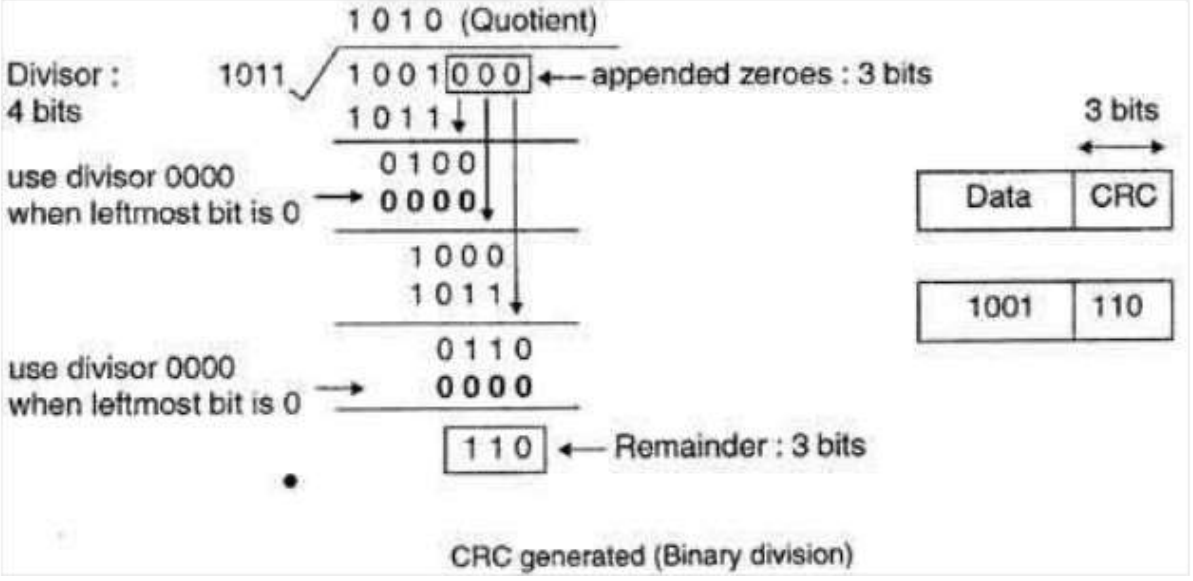
Error Detection

Cyclic Redundancy Check(CRC):

1)Data unit 1001000 is divided by 1011.

2) During this process of division, whenever the leftmost bit of dividend or remainder is 0, we use a string of Os of same length as divisor. Thus in this case divisor 1011 is replaced by 0000.

3)At the receiver side, data received is 1001110.

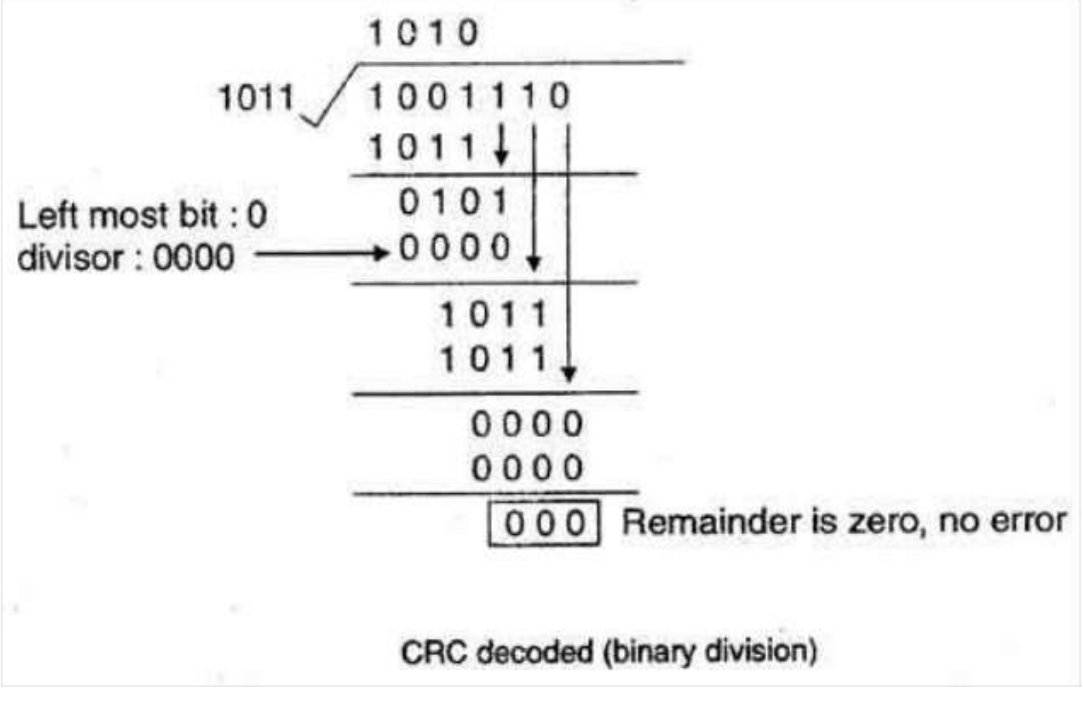


Error Detection

- **Cyclic Redundancy Check(CRC):**

4) This data is again divided by a divisor 1011.

5)The remainder obtained is 000; it means there is no error. Next example: message: 1100101, polynomial: 11011



CRC decoded (binary division)

Steps:

- **Cyclic Redundancy Check(CRC):**

n : Number of bits in data to be sent from sender side.
k : Number of bits in the key obtained from generator polynomial.

Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

The binary data is first augmented by adding k-1 zeros in the end of the data
Use modulo-2 binary division to divide binary data by the key and store remainder of division.
Append the remainder at the end of the data to form the encoded data and send the same

Receiver Side (Check if there are errors introduced in transmission)

Perform modulo-2 division again and if the remainder is 0, then there are no errors.

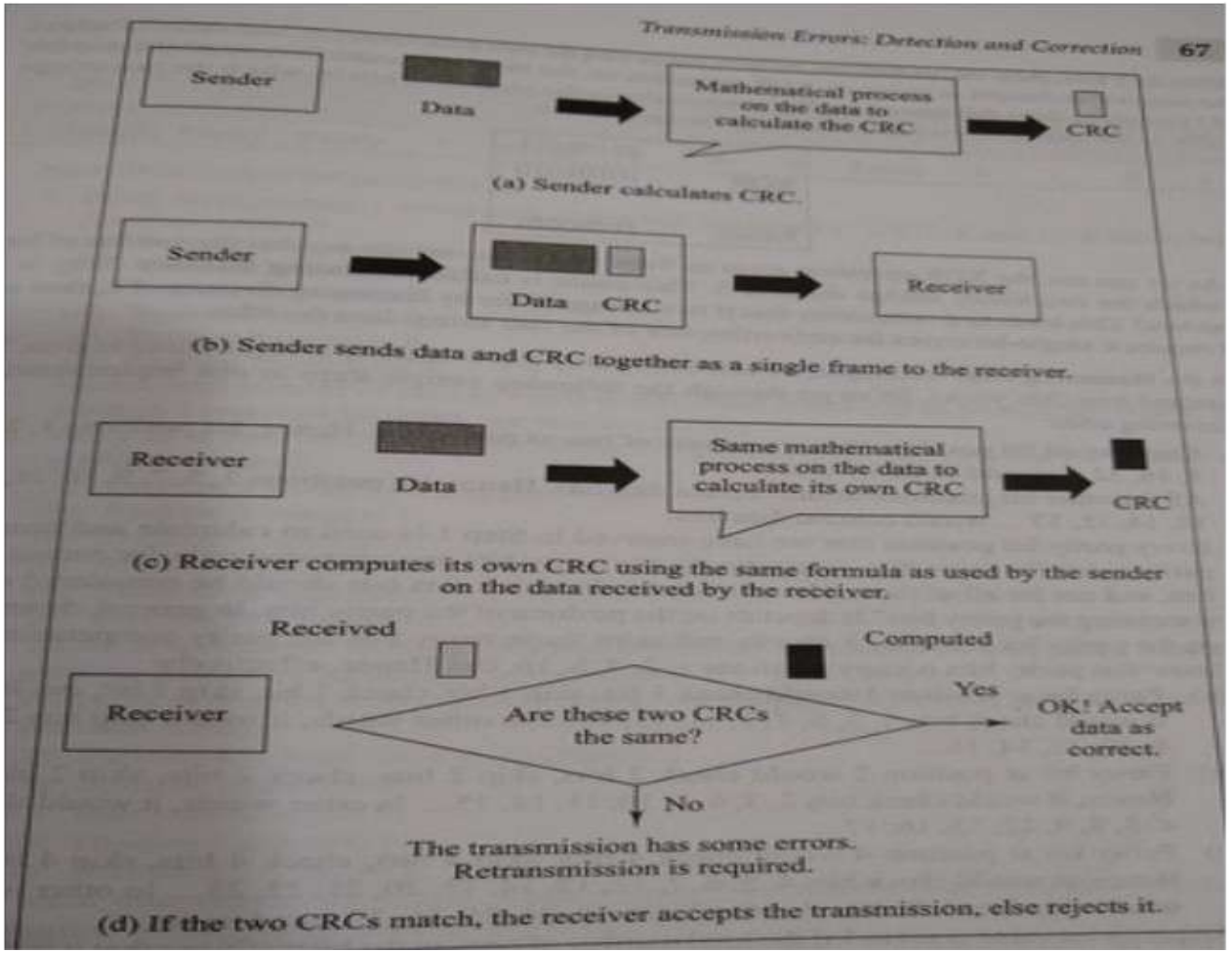
Steps:

Modulo 2 Division:

- The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. Just that instead of subtraction, we use XOR here.
- In each step, a copy of the divisor (or data) is XORed with the k bits of the dividend (or key).
- The result of the XOR operation (remainder) is (n-1) bits, which is used for the next step after 1 extra bit is pulled down to make it n bits long.
- When there are no bits left to pull down, we have a result. The (n-1)-bit

remainder which is appended at the sender side.

Error Detection



Recovery from Errors

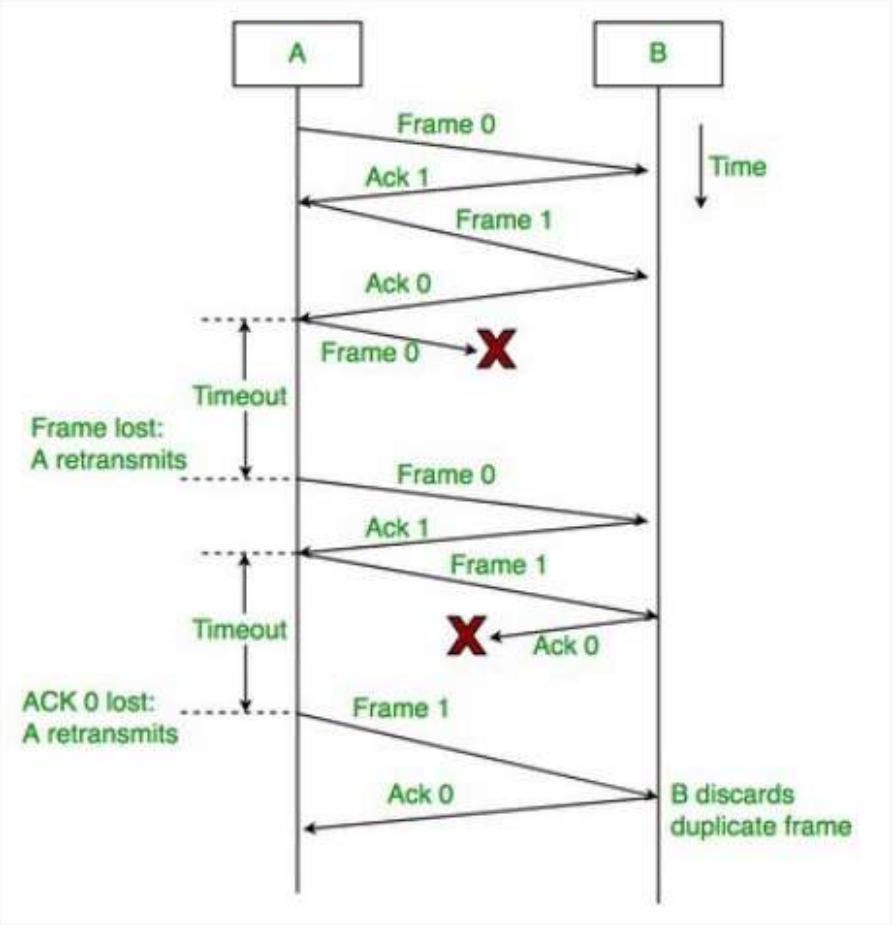
- Once error detected there are multiple ways of recovery from errors.
- If we follow a scheme of necessarily sending some form of acknowledgement, the receiver will send a positive Acknowledgement (ACK) back to the sender if every thing was ok.
- If any error were found, it will send a negative acknowledgement (NAK) to the sender.
- The sender would wait until it receives an ACK or NAK and would then decide whether to retransmit the same chunk or block or frame of data or not, and only
  then send the next block

Stop and wait

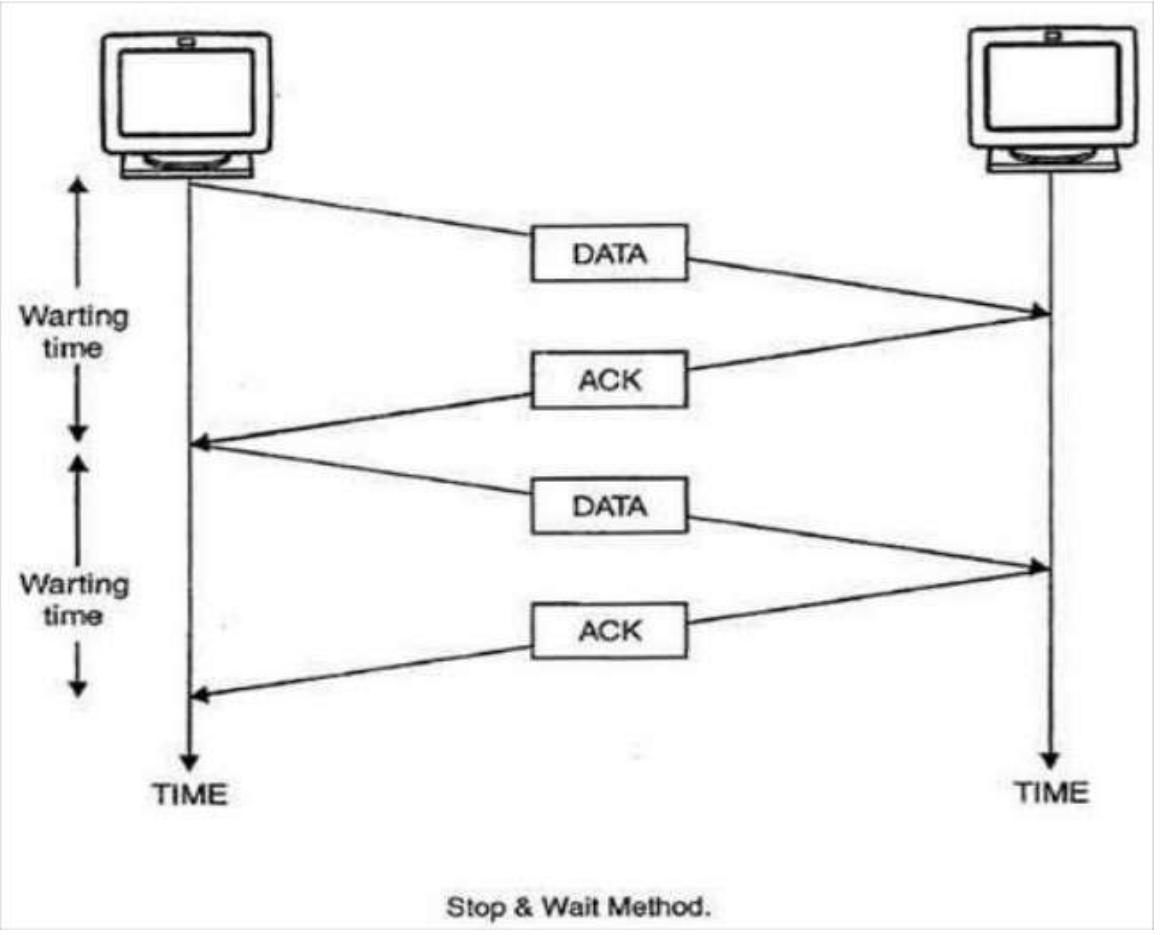
- This is very simple method where receiver sends one frame of data and necessarily waits for an acknowledgement (ACK) from the receiver before sending the next frame.
- Only after the sender receives an acknowledgement for a frame then it sends the next frame.
- Thus the transmission always takes the form: **data- ACK-data ACK…**etc. where the data frames are sent by the sender and the ACK frames are sent by the receiver, back to the sender.

Stop and wait

- The stop and wait approach is pretty simple to implement.
- Every frame must be individually acknowledges before the next frame can be transmitted.
- There is also lies its drawback , since the sender must receive each acknowledgement before it can transmit next frame, it makes the transmission very slow.



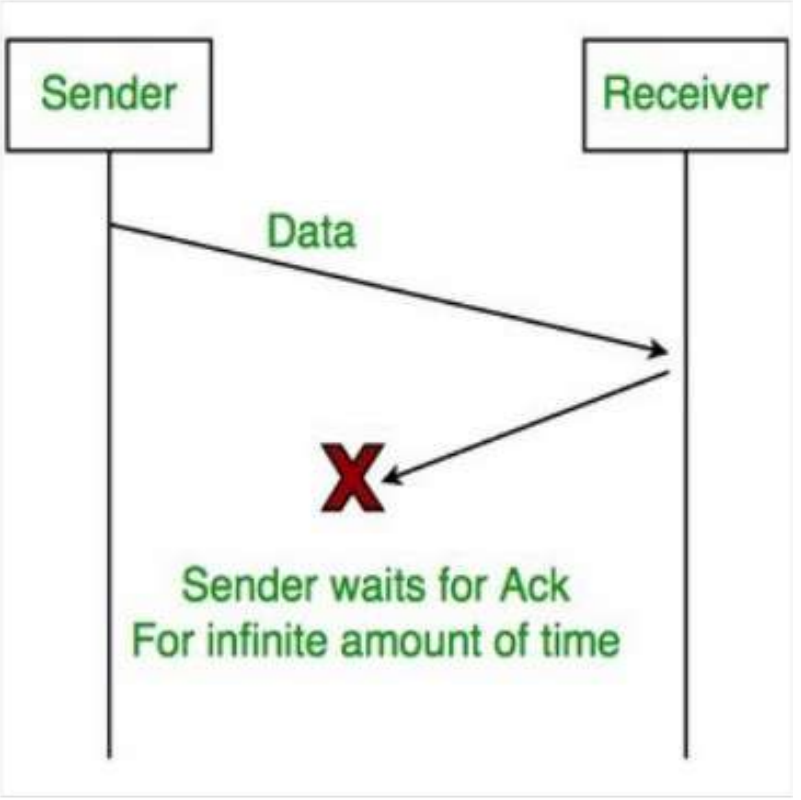
Stop and wait



Stop & Wait Method.

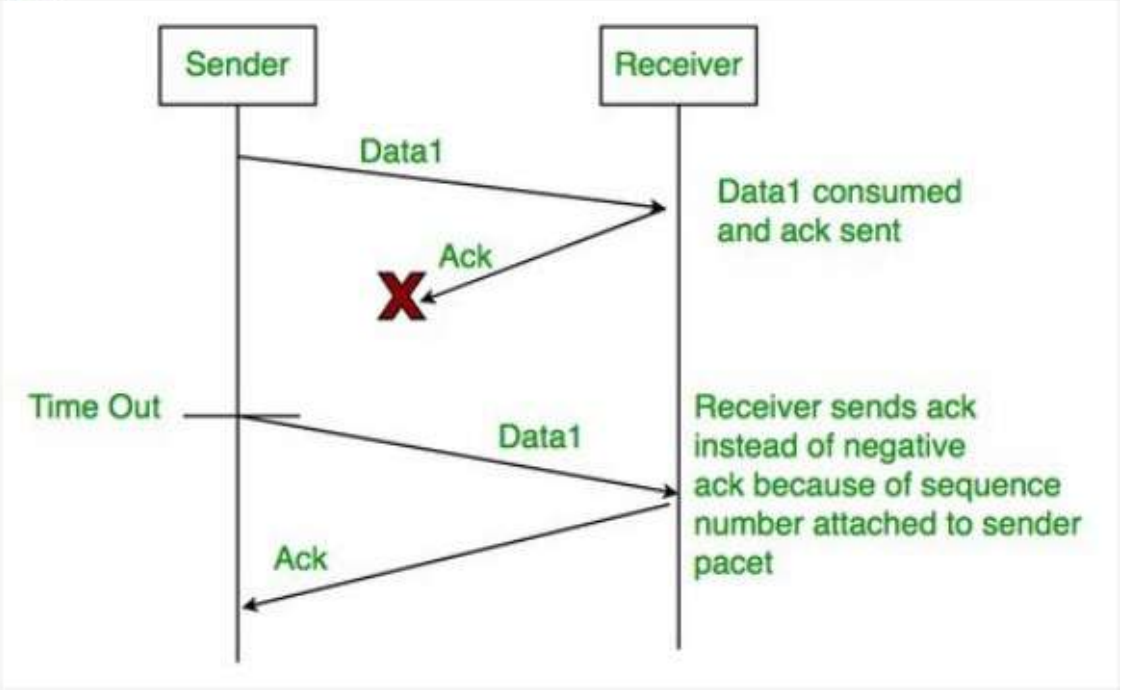
Stop and wait

- The sender sets a timer for every frame that it sends.
- If it does not receive any acknowledgement from the receiver before the timer expires, it sends the same frame again, with the new timer.
- However it could happen that the sender already half way through to the sender from the receiver.



Stop and wait

- In this case, the receiver receives two or more instances of the same frame. The receiver is equipped with the understanding that the duplication has to be rejected, in such a case.



Go-back-n

- InGo-back-n , the sender starts retransmission with the last acknowledged frame, even if the subsequent frames have arrived correctly at the receiver.
- For example, suppose the sender sends 5 frames to the receiver (i.e. the window size is 5).
- The frames 0,1,2 arrive correctly at the receiver, but the frame 3 is in error. The receiver send NAK for frame 3
- By the time the NAK reaches the sender, the sender may have already sent frames 4 and 5. however after receiving the NAK, the sender now retransmits frame 3 and all frame transmitted after frame 3.
- Assuming that this retransmission is successful the receiver would now acknowledge this correct transmission of frames 3, 4, 5 after discarding the duplicates for frames 4 and 5.

Go-back-n

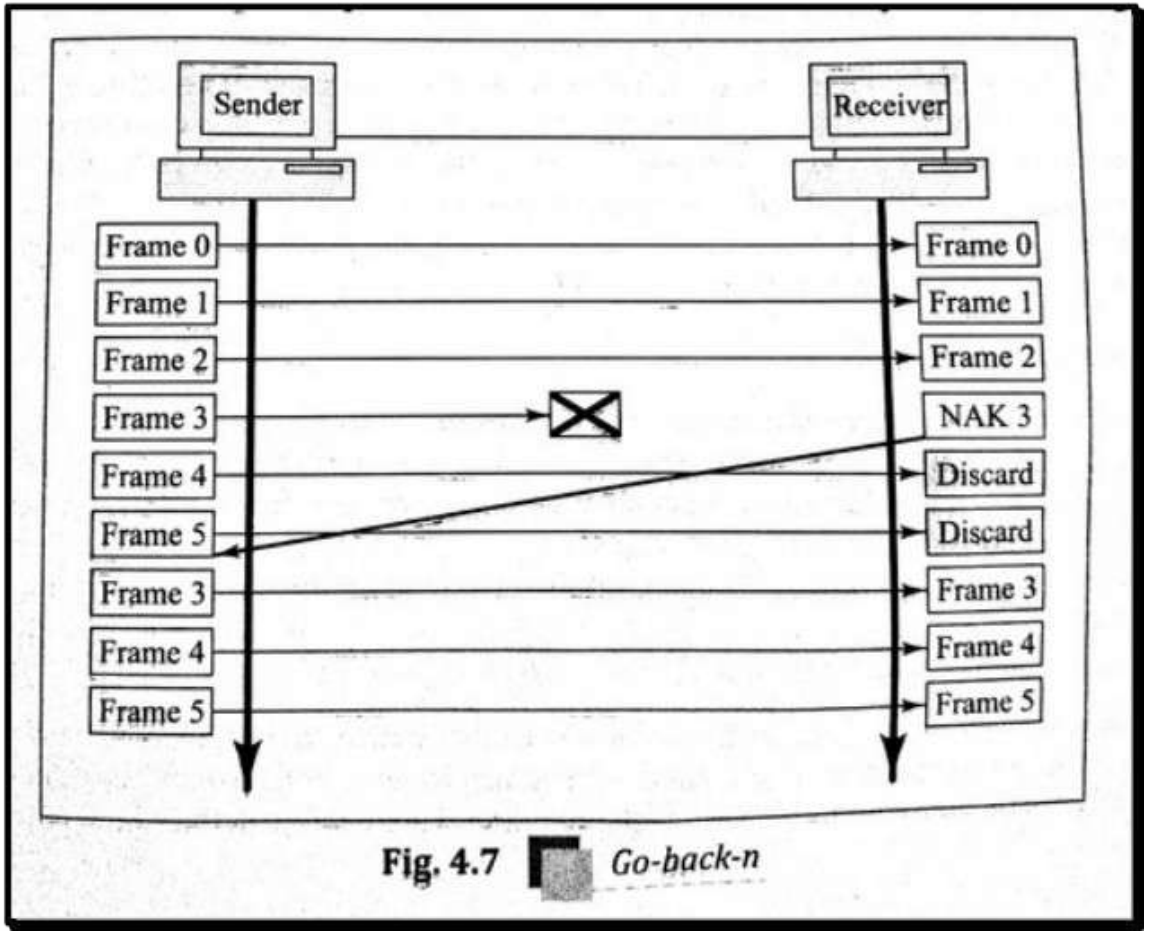


Fig. 4.7 *Go-back-n*

Go-back-n

- Let us consider another situation where the sender has transmitted all the frames and is actually waiting for an acknowledgement that has been lost on the way.
- The sender waits for some time and then retransmits the unacknowledged frames. The receiver detects this duplication, sends another acknowledgement and discards the redundant data.

Go-back-n

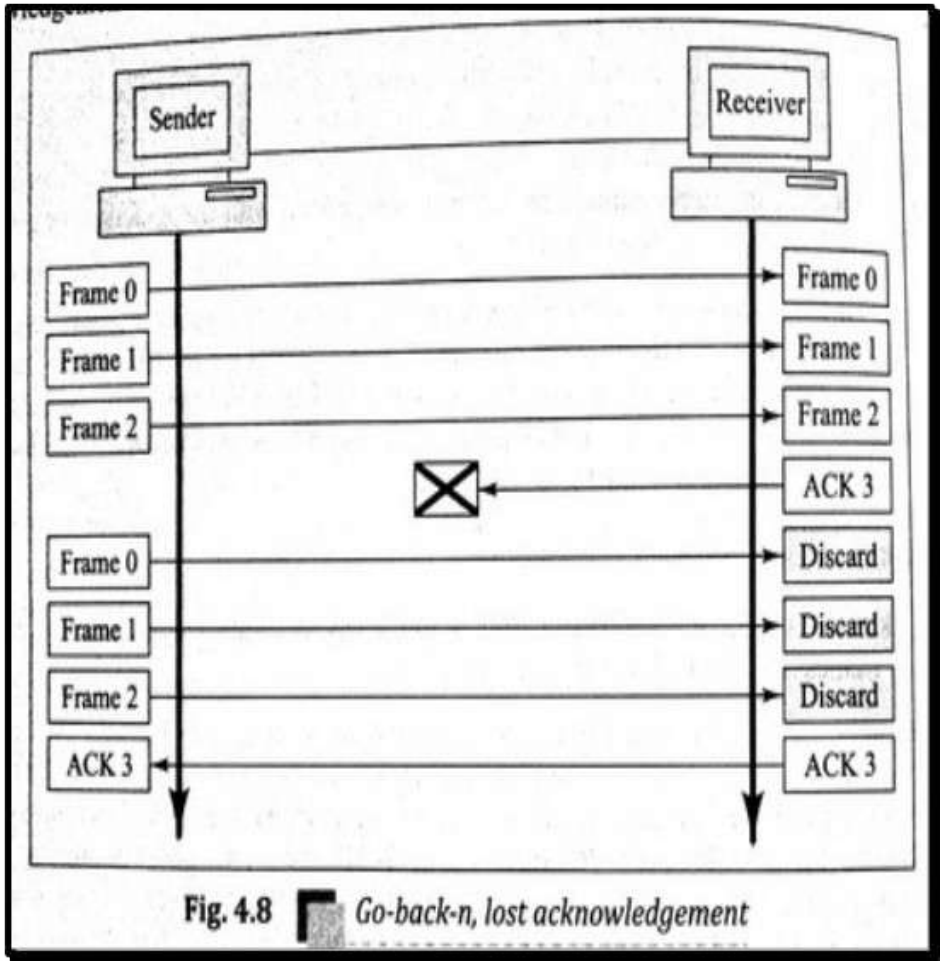


Fig. 4.8 *Go-back-n, lost acknowledgement*

Go-back-n

- Another possibility is when a damaged frame is received by the receiver. Here the receiver receives frames 0 and 1 correctly, but does not immediately acknowledge them.
- It receives frame 2 is in error, so it returns NAK 2. this informs the sender that frame 0 and 1 have been received correctly, but that frame 2 must be resent.
- However in this case, the receiver keeps accepting new frames and doesn't reject them.

After it receives the resent frame 2, the receiver will send ACK 5 implying that it is now ready to accept frame 5.

The processor needs lots of memory and processing logic to keep a track of all these things.

Go-back-n

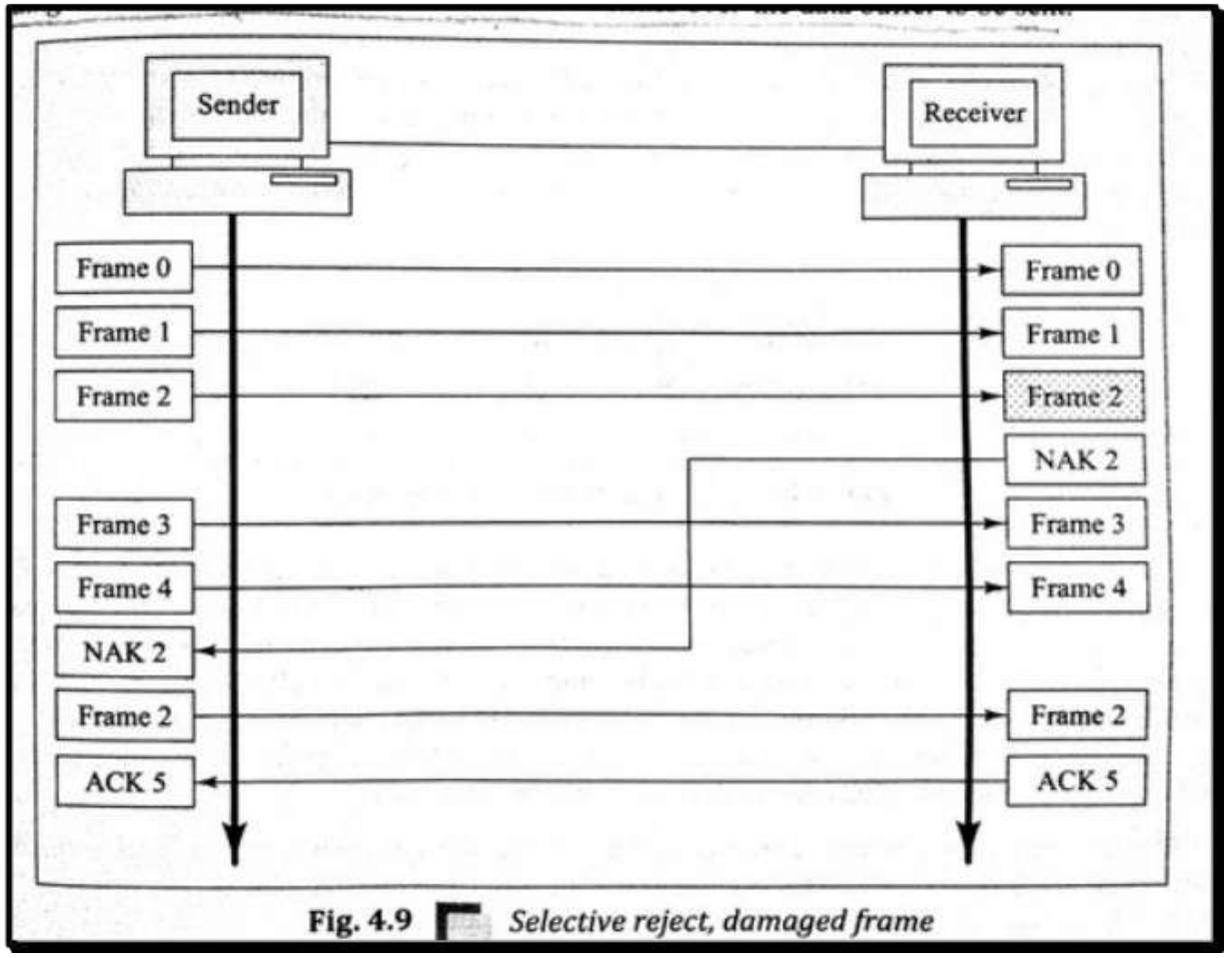


Fig. 4.9 *Selective reject, damaged frame*

Sliding Window

- The sliding window technique is a variation of the Go-Back-n technique.
- The data to be send is divided into frames.
- The sender must send a frame, wait for its acknowledgement and only after it receives that acknowledgement, send the next frame.
- A trick to improve efficiency would be to send multiple frames at a time, check the CRC of all the frames one by one, send the acknowledgement for all and request for the next set of frames. The sliding window is based on this philosophy.

Sliding Window

- In this technique, the underlying transmission mechanism defines an imaginary window consisting of maximum n frames to be sent at a time. The transmission mechanism allows the data to be transmitted at a time only up to the size of the window.
- The window defines how much data sender can send before it must wait to receive an acknowledgement from the receiver. The term sliding window is used because the data window slides over the data buffer to be sent.
- Fig shows a sliding window of size 8 frames. That is the sender can send eight frames before it must wait to receive an acknowledgement from the receiver.

Sliding Window

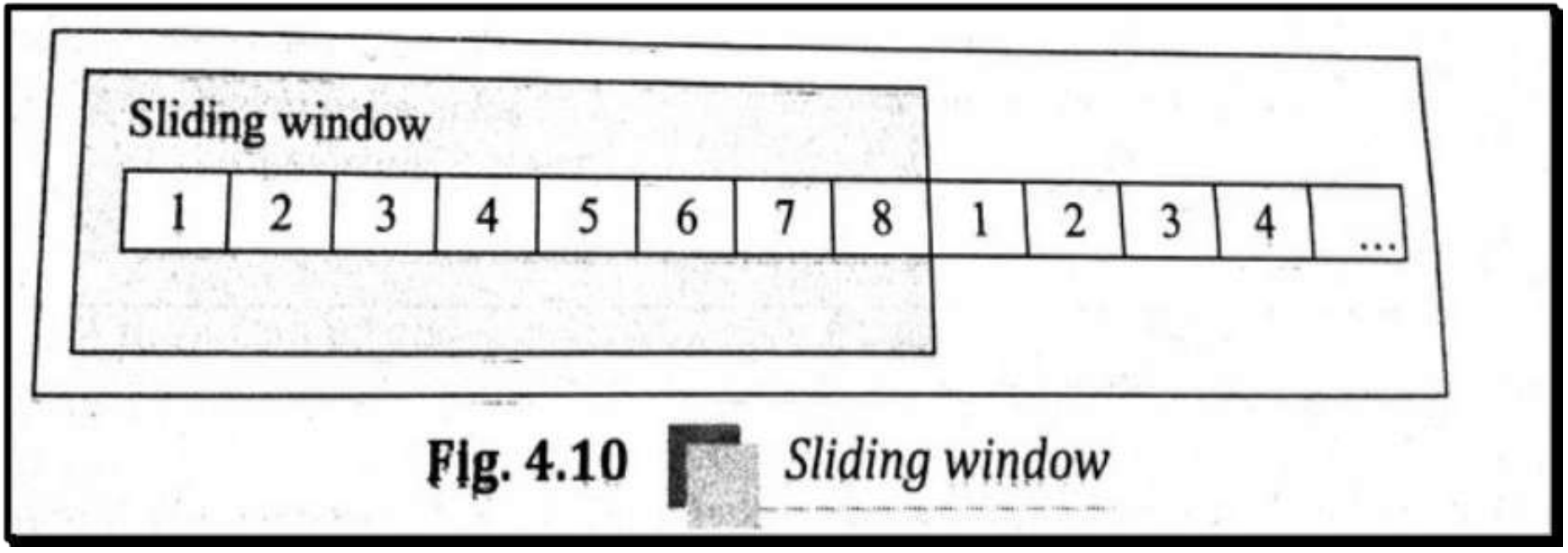


Fig. 4.10 *Sliding window*

Sliding Window

- The sender and receiver both maintain their own sliding windows.
- The sender sends the number of frames that it is allowed to send by its sliding window and then waits for the acknowledgement from the receiver.
- When receiver sends an acknowledgement back to the sender, it includes the number of next frame that is expect to receive.
- Thus if sender has sent frames numbered 1 to 3 to the receiver, assuming that the receiver has received them correctly, the receiver sends back an acknowledgement that includes number 4.
- Thus sender knows that frames 1 to 3 received correctly and proceeds to send frame 4.

Sliding Window

- The sender sends these eight frames and receives an acknowledgement for the first two frames.
- In such case, the sender slides the window two frame to the right and sends the 9^{th} and 10^{th} frames.
- Thus the receiver again has 8 frames. It can now check CRC for all, one by one, and if it founds an error in 7 and finds frame 3 to 6 ok, it will send the acknowledgement that includes the number 7.
- The sender now sends 8 frames from frame 7 onwards again.

Sliding Window

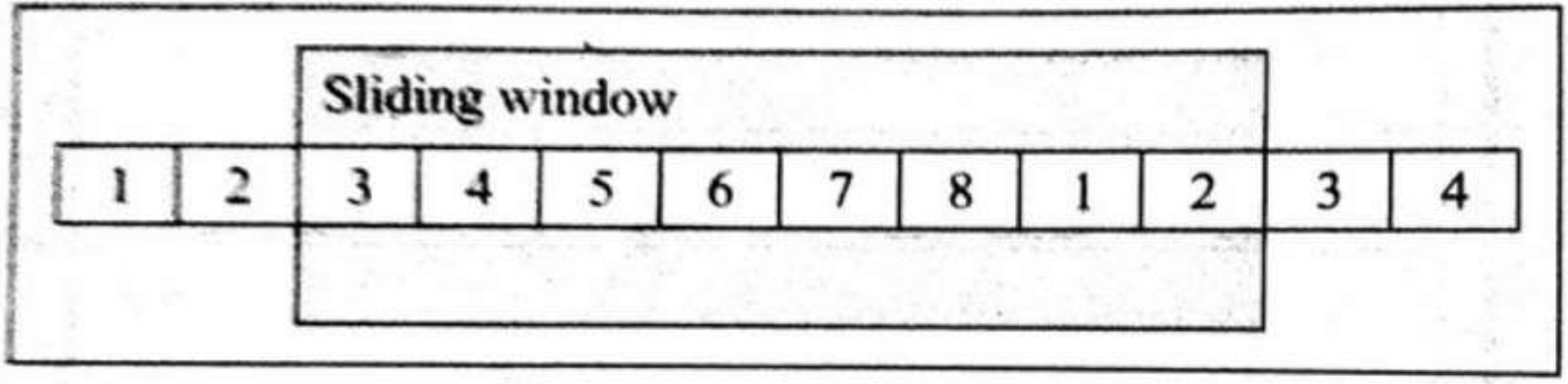


Fig. 4.11 *Sliding window mechanism*

Sliding Window

- It uses two buffers and one window to control the flow of data.
- The sender has buffer for storing data coming from the sending application program.
- The application program creates the data to be sent and writes it to this buffer. The sender imposes a window on the buffer and sends frames till all the frames have been sent.
- The receiver also has a buffer. The receiver receives the data, checks if any error and stores the correct ones in this buffer.
- The application program at the receiver‘s end then picks up the data from this receiver buffer

Sliding Window

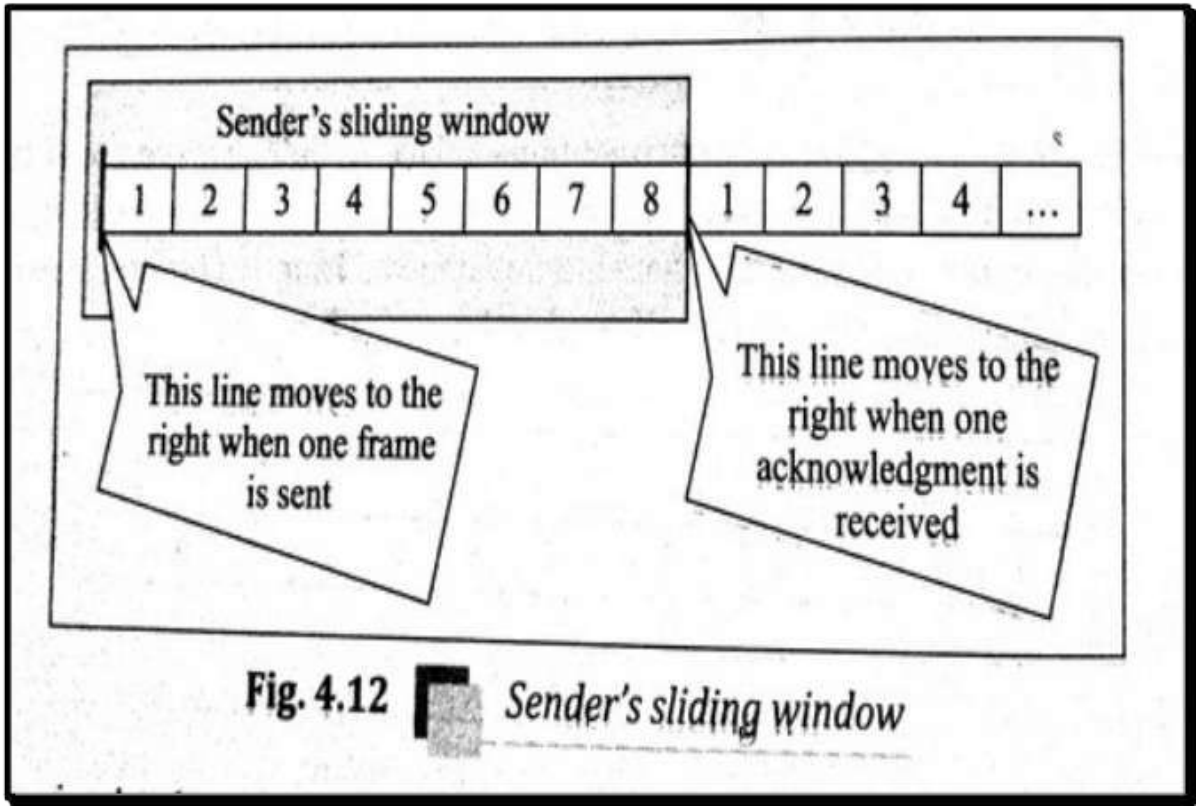


Fig. 4.12 *Sender's sliding window*

Sliding Window

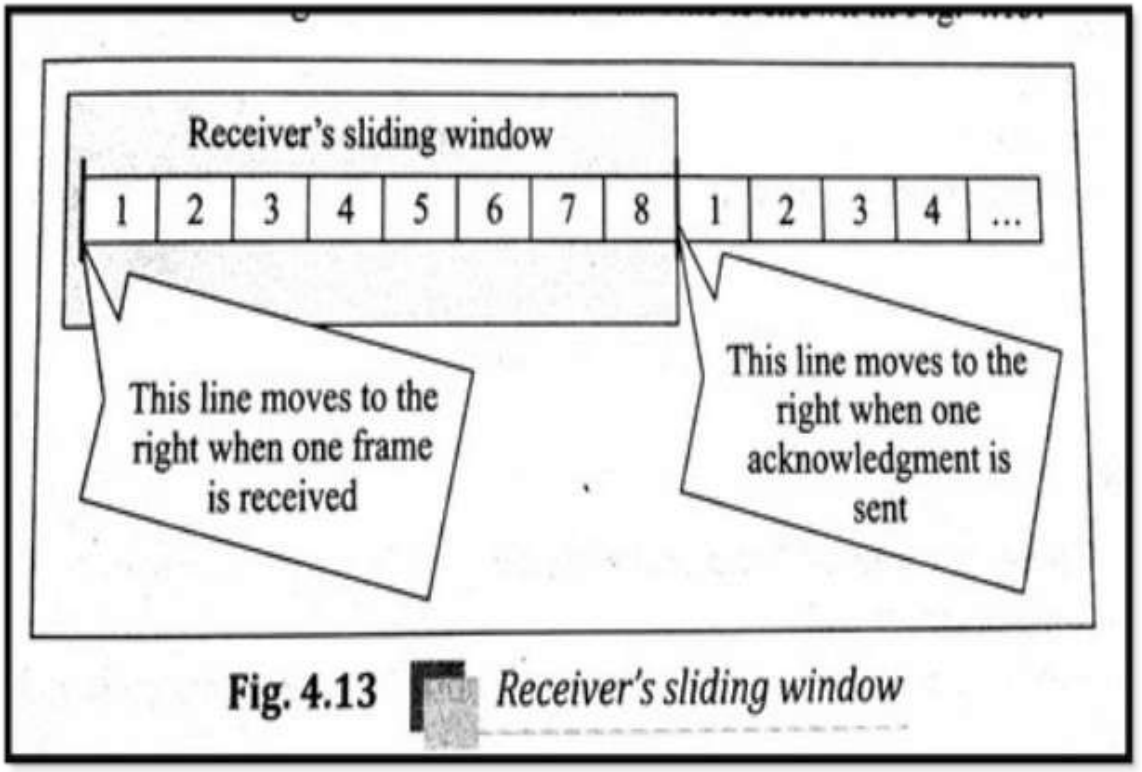


Fig. 4.13 *Receiver's sliding window*