UNIT - II DATA LINK LAYER:

SYLLABUS:

1. **DESIGN ISSUES**:
   - **1.1.** Services provided to network layer
     - a. Framing
     - b. Character Count
     - c. Character/Byte Count
     - d. Bit Stuffing
   - **1.2.** Error Control
   - **1.3.** Flow Control

2. **ERROR DETECTION AND CORRECTION**
   - **2.1.** Error Detection Codes
     - a. Parity Check
     - b. Checksum.
     - c. Cyclic Redundancy Check.
     - d. Hamming Code
   - **2.2.** Error- Correction Codes
     - a. Hamming Codes.
     - b. Binary Convolutional Codes

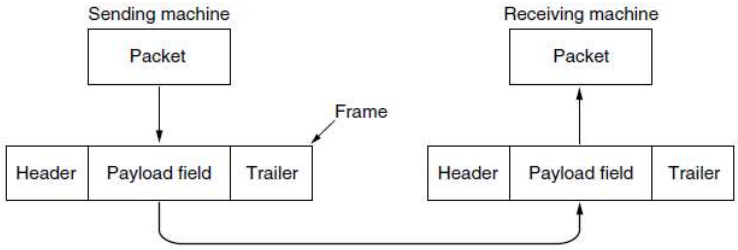
3. **ELEMENTARY DATA LINK PROTOCOLS**:
   - **3.1.** Simplex Protocol
   - **3.2.** A Simplex Stop and Wait Protocol for an Error-Free Channel
   - **3.3.** A Simplex Stop and Wait Protocol for Noisy Channel.

4. **SLIDING WINDOW PROTOCOLS**:
   - **4.1.** A One-Bit Sliding Window Protocol
   - **4.2.** A Protocol using Go-Back-N
   - **4.3.** A Protocol using Selective Repeat.

5. **MEDIUM ACCESS SUB LAYER:**
   - **5.1.** The Channel Allocation Problem
   - **5.2.** Multiple Access Protocols:
     - a. ALOHA
     - b. Carrier Sense Multiple Access Protocols
     - c. Collision Free Protocols.
   - **5.3.** Wireless LAN Protocols
   - **5.4.** Data Link Layer

INTRODUCTION:

- The data link layer uses the services of the physical layer to send and receive bits over communication channels. It has a number of functions, including:
  - Providing a well-defined service interface to the network layer.
  - Dealing with transmission errors.
  - Regulating the flow of data so that slow receivers are not swamped by fast senders.
- To accomplish these goals, the data link layer takes the **packets** it gets from the network layer and encapsulates them into **frames** for transmission.
- Each frame contains a Header, Packet (from Network layer), and Trailer



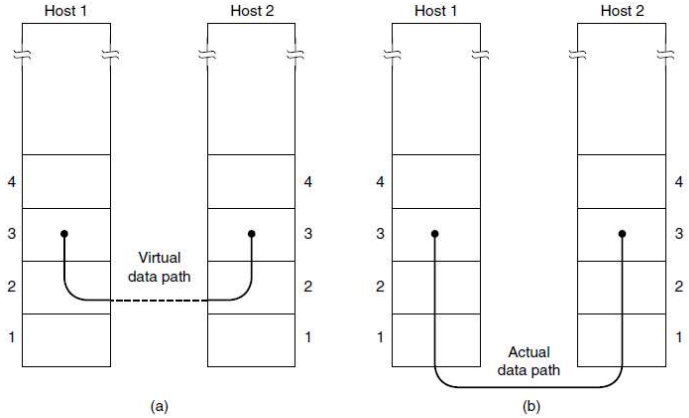
1. DESIGN ISSUES:

The following are the design issues of the Data-Link Layer:

1.1. Services provided to the Network Layer
1.2. Framing
1.3. Error Control
1.4. Flow Control

1.1. Services provided to the Network Layer

- The function of the data link layer is to provide services to the network layer.
- Fig(a) is the virtual data path between network layers of Host1 and Host2
- Fig(b) is the actual data path between them.



- The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine

- Three possibilities are:
  - *Unacknowledged connectionless service*: Unacknowledged connectionless service consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. If a frame is lost due to noise on the line, no attempt is made to detect the loss or recover from it in the data link layer

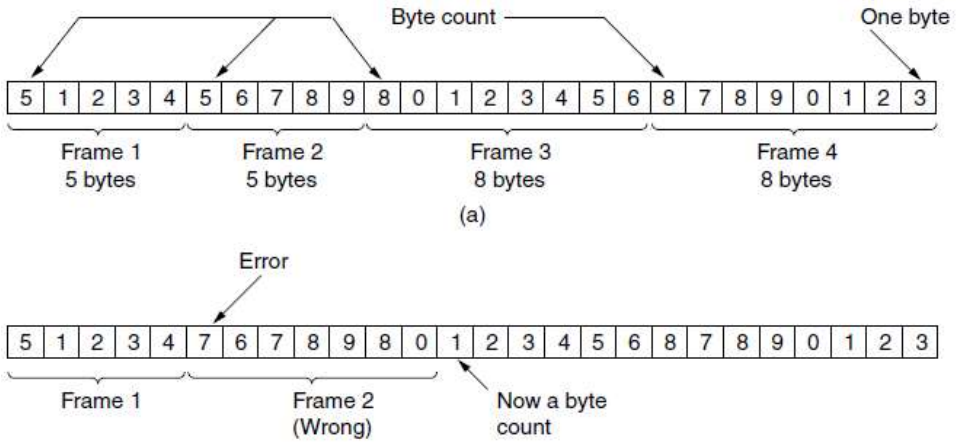
  - *Acknowledged connectionless service*: Each frame sent is individually acknowledged. the sender knows whether a frame has arrived correctly or been lost, if it has not arrived within a specified time interval, it can be sent again. This service is useful over unreliable channels, such as wireless systems.

  - *Acknowledged connection-oriented service*: the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order. Connection-oriented service thus provides the network layer processes with the equivalent of a reliable bit stream.

- When connection-oriented service is used, transfers go through three distinct phases. In the first phase, the connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not. In the second phase, one or more frames are actually transmitted. In the third and final phase, the connection is released, freeing up the variables, buffers, and other resources used to maintain the connection.

a. **Framing:**
- The usual approach is for the data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted
- When a frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with.
- But, breaking up the bit stream into frames is more difficult. 3 Possible methods to form good frames are:
  - Byte count.
  - Flag bytes with byte stuffing.
  - Flag bits with bit stuffing.

b. **Byte count:**
- ✓ Uses a field in the header to specify the number of bytes in the frame. When the data link layer at the destination sees the byte count, it knows how many bytes follow and hence where the end of the frame is. Figure below shows the faming method.



- ✓ In above figure, message with frame size of 5,5,8,8 are transmitted.
  Frame 1, There are 5 bytes of data including the frame size
  Frame 2, There are 5 bytes of data including the frame size
  Frame 3, There are 8 bytes of data including the frame size
  Frame 4, There are 8 bytes of data including the frame size

- ✓ The problem with this algorithm is shown at receiver end in the above figure. After transmission If the byte-count 5 of frame2 changes to 7, the frame2 is treated as 7 bytes whereas it was 5 bytes during transmission. Because of which the start of the Frame3 is not identifiable.

c. **Flag bytes with byte stuffing:**

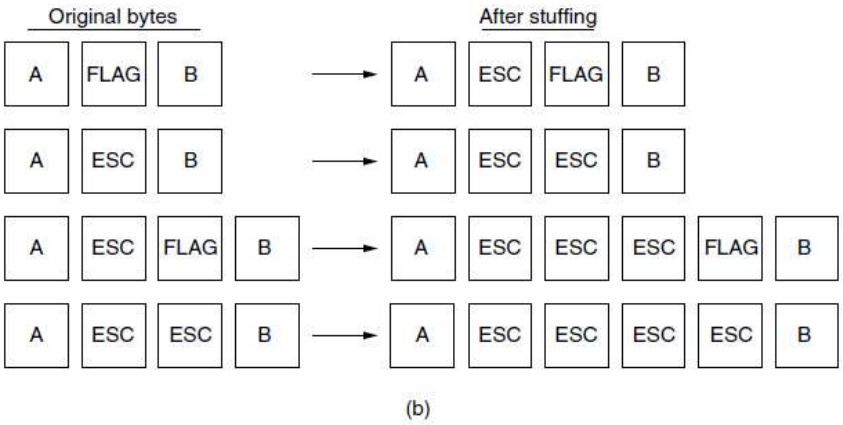
The same byte, called a **flag byte**, is used as both the starting and ending delimiter, it is as shown in fig(a).

(a)

Two consecutive flag bytes indicate the end of one frame and the start of the next. Thus, if the receiver ever loses synchronization it can just search for two flag bytes to find the end of the current frame and the start of the next frame

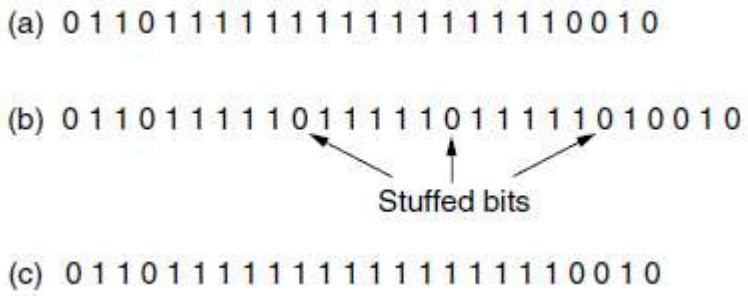
What if the FLAG bytes occur in the data?
One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each ''accidental'' flag byte in the data. Thus, a framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it. The data link layer on the receiving end removes the escape bytes before giving the data to the network layer. This technique is called **byte stuffing**.

what happens if an escape byte occurs in the middle of the data?
At the receiver, the first escape byte is removed, leaving the data byte that follows it. Example is as shown in fig(b)



d. **Bit Stuffing:**

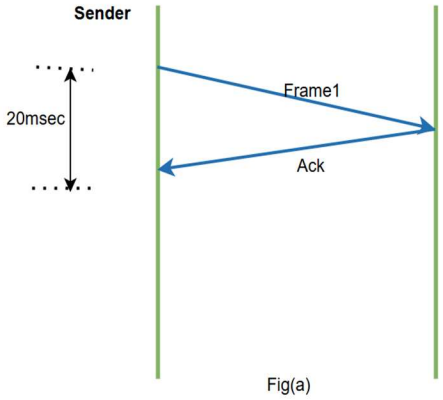
- Framing can also be done at the bit level, so frames can contain an arbitrary number of bits made up of units of any size
- Each frame begins and ends with a special bit pattern. Here we use the bit pattern as 01111110 i.e., 7E Hexa-decimal.
- 01111110 is now the flag byte.
- **What if the Flag byte (01111110) is in the message signal.**
  - Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This **bit stuffing** is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data.
  - When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs (i.e., deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110
  - Fig(a) is original data, Fig(b) is Bit-stuffed data at transmitter end, fig(c) is unstuffed data at receiver end



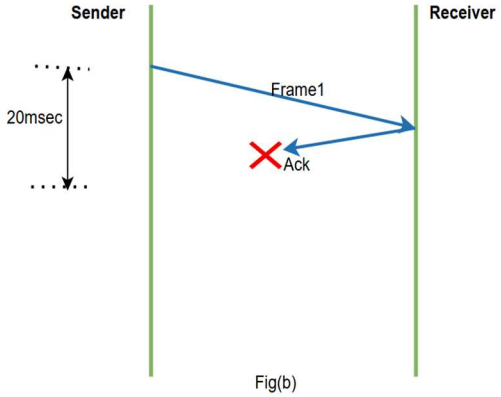
1.2. Error Control

Error control can be easily understood only when we know "*how to make sure all frames are eventually delivered to the network layer at the destination and in the proper order*".

- The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line. Typically, the protocol requires the receiver to send back special control frames that bear positive or negative acknowledgements about the incoming frames.
  - If the sender receives a positive acknowledgement for a frame, it knows the frame has arrived safely. Fig(a)
  - conversely, a negative acknowledgement indicates that something has gone wrong and the frame must be transmitted again. Fig(b)

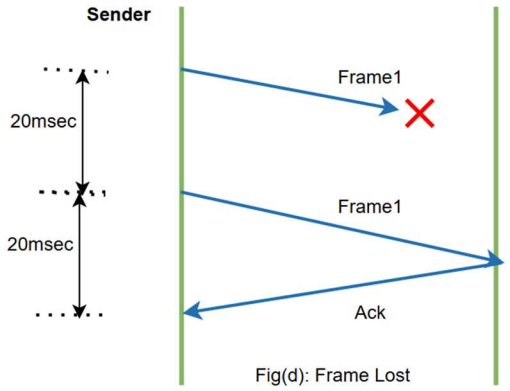


Fig(a)

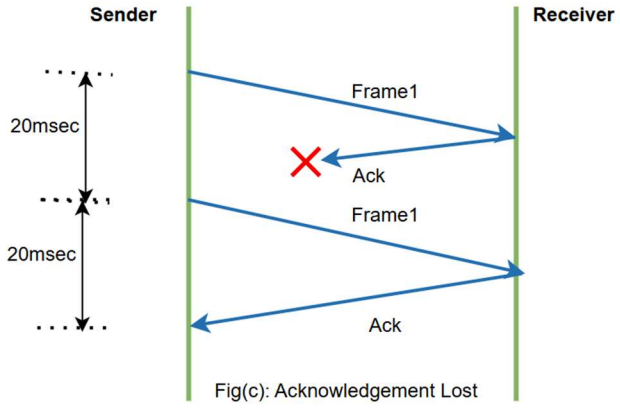


Fig(b)

- **What if the frame is lost? What if the acknowledgement is lost?**
  - This possibility is dealt with by introducing timers into the data link layer.
  - When the sender transmits a frame, it generally also starts a timer.
  - The timer is set to expire after an interval long enough for the frame to reach the destination, be processed there, and for the acknowledgement to propagate back to the sender.
  - Normally, the frame will be correctly received and the acknowledgement will be back before the timer runs out, in which case the timer will be cancelled.



Fig(d): Frame Lost



Fig(c): Acknowledgement Lost

1.3. Flow Control:

Another key design issue with the data link layer is, **what if the transmitter transmits the frames faster than the receiver can accept them?**

This can be prevented in 2 approaches.

i. ***Feedback-based flow control***, the receiver sends back information to the sender, permitting it to send more data, or at least informs the sender how the receiver is doing.

ii. ***Rate-based flow control***, the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver

2. ERROR DETECTION AND CORRECTION

Communication channels like optical fibre in telecommunications networks, have tiny error rates so that transmission errors are a rare occurrence. But other channels, especially wireless links and aging local loops, have error rates that are orders of magnitude larger.

One strategy is to include enough redundant information to enable the receiver to deduce what the transmitter actually transmitted; this strategy is **Error- Correction Codes** also referred as **Forward-Error Correction**.

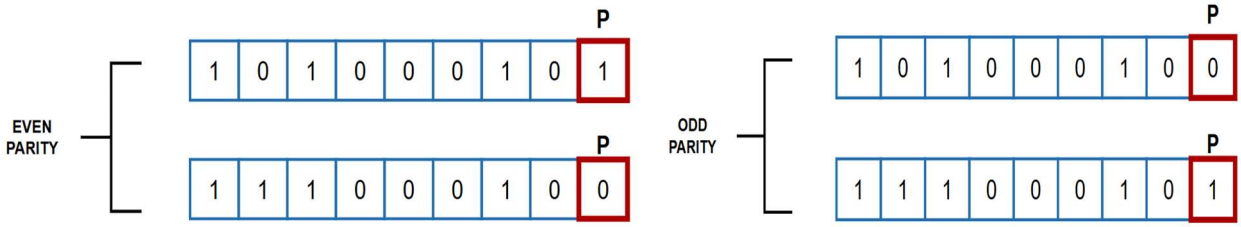
Another strategy is to include only enough redundancy to allow the receiver to deduce that an error has occurred and request for a re-transmission, this strategy is **Error-Detection Codes**

2.1. Error-Detection Code:

We will discuss about three error-detection codes:

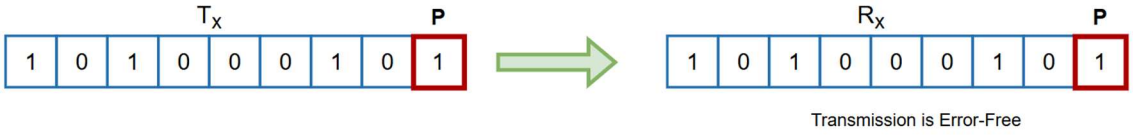
1. *Parity*
2. *Checksum*
3. *Cyclic Redundancy Code (CRC)*

Parity: In this error-detection scheme, a parity bit is added to the frame. The parity bit is chosen such that the number of 1s in the frame is even (or odd). Consider a frame with 8 bits of information and we add an extra bit to represent the parity.

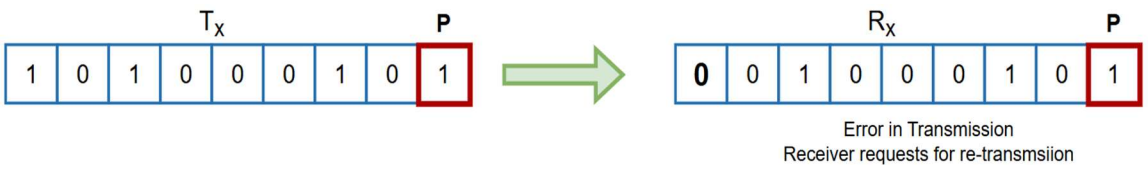


Let Transmitter and receiver decide on even-parity data transmission

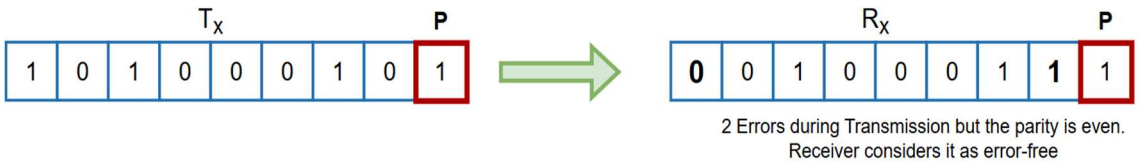
i. *No bits modified*: None of the bits are modified during transmission.



ii. *One of the bits is modified:* During transmission, one of the bits is modified. Let the MSB is modified i.e., 1 becomes 0, then the parity is odd. In this situation the receiver requests for a re-transmission. (In some situations, the parity bit itself may get modified)



iii. *Two- bits modified:* Let any of the two (MSB and LSB) transmitted bits are modified. In this situation, the receiver checks for parity and finds the total no. of 1s are even. The receiver will consider it as the error-free message though the bits are modified. We can conclude that, **this error-detection code is valid only for one bit error**.



Checksum: The algorithm for checksum is as follows

- Split the m-bit message into n frames of same length (bit capacity)
- Add the first frame with the second frame. If a carry is generated, add the carry to the LSB of sum.
- Perform the above step until the last frame and note the final sum.
- To get the checksum at T_x, Invert the sum.
- This checksum is transmitted along with all the n frames
- At R_x, all the frames are added including the checksum (T_x)
- The resulting sum is now inverted to get the checksum at R_x
- If Checksum at $R_x = 0$, its error-free transmission. Else, it is error transmission

Example is as shown below: Consider a 48-bit message divided into 8-bit per frame.

42-Bit Message 100010111100111100011110100110110101010000111110

Frames to be transmitted

F1	1	0	0	0	1	0	1	1
F2	1	1	0	0	1	1	1	1
F3	0	0	0	1	1	1	1	0
F4	1	0	0	1	1	0	1	1
F5	0	1	0	1	0	1	0	0
F6	0	0	1	1	1	1	1	0

Frames actually transmitted

F1	1	0	0	0	1	0	1	1
F2	1	1	0	0	1	1	1	1
F3	0	0	0	1	1	1	1	0
F4	1	0	0	1	1	0	1	1
F5	0	1	0	1	0	1	0	0
F6	0	0	1	1	1	1	1	0
CHECKSUM	0	1	0	1	1	0	0	0

Checksum at Transmitter

	1	0	0	0	1	0	1	1
	1	1	0	0	1	1	1	1
1	0	1	0	1	1	0	1	0
								1
	0	1	0	1	1	0	1	1
	0	0	0	1	1	1	1	0
	0	1	1	1	1	0	0	1
	1	0	0	1	1	0	1	1
1	0	0	0	1	0	1	0	0
								1
	0	0	0	1	0	1	0	1
	0	1	0	1	0	1	0	0
	0	1	1	0	1	0	0	1
	0	0	1	1	1	1	1	0
sum	1	0	1	0	0	1	1	1
checksum	0	1	0	1	1	0	0	0

Checksum at Receiver

	1	0	0	0	1	0	1	1
	1	1	0	0	1	1	1	1
1	0	1	0	1	1	0	1	0
								1
	0	1	0	1	1	0	1	1
	0	0	0	1	1	1	1	0
	0	1	1	1	1	0	0	1
	1	0	0	1	1	0	1	1
1	0	0	0	1	0	1	0	0
								1
	0	0	0	1	0	1	0	1
	0	1	0	1	0	1	0	0
	0	1	1	0	1	0	0	1
	0	0	1	1	1	1	1	0
	1	0	1	0	0	1	1	1
	0	1	0	1	1	0	0	0
sum	1	1	1	1	1	1	1	1
Checksum	0	0	0	0	0	0	0	0

Since the checksum at receiver is all 0s, we can say that the data is transmitted without errors.

Cyclic Redundancy Check (CRC):

- CRC is the most powerful error-detection code, also called a polynomial code
- Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only. A *k*-bit frame is regarded as the coefficient list for a polynomial with *k* terms, ranging from x^{k-1} to x^0. Such a polynomial is said to be of degree $k - 1$.
- For example, 110010 has 6 bits and represents a six-term polynomial

$$1x^5 + 1x^4 + 0x^3 + 0x^2 + 1x^1 + 0x^0$$

- **Algorithm**:
  - Message = **m-bits**
  - Generate Polynomial = **k-bits**
  - Append **(k-1) 0s** to the m-bit message
  - Evaluate the (k-1) CRC bits by **modulo-2 division**. Here, the divisor is a k-bit polynomial
  - Append the **evaluated CRC bits to the m-bit message** and transmit.
  - At the receiver, the received message is once again modulo-2 divided by a k-bit polynomial
  - If the **reminder is all 0s**, then the message is error-free. Else, it received an error message

To better understand CRC, let us transmit a 6-bit frame 101101, with polynomial coefficients of 1101. Here, k=4, so k-1, i.e., three 0s, are appended to the frame.

The frame now is 101101000. The following table shows the process of the Cyclic Redundancy Check mechanism

TRANSMITTER

```
1101) 1 0 1 1 0 1 0 0 0 (1100
      1 1 0 1
      0 1 1 0 0
        1 1 0 1
        0 0 0 1 1 0 0
              1 1 0 1
              0 0 0 1 0
```

The redundant bits are evaluated as 010

RECEIVER

```
1101) 1 0 1 1 0 1 0 1 0 (1100
      1 1 0 1
      0 1 1 0 0
        1 1 0 1
        0 0 0 1 1 0 1
              1 1 0 1
              0 0 0 0 0
```

The reminder at the receiver is 0.

error-free transmission

Consider another example with an error message transmitted.

TRANSMITTER

```
1101) 1 0 1 1 0 1 0 0 0 (1100
      1 1 0 1
      0 1 1 0 0
        1 1 0 1
        0 0 0 1 1 0 0
              1 1 0 1
              0 0 0 1 0
```

The redundant bits are evaluated as 010

RECEIVER

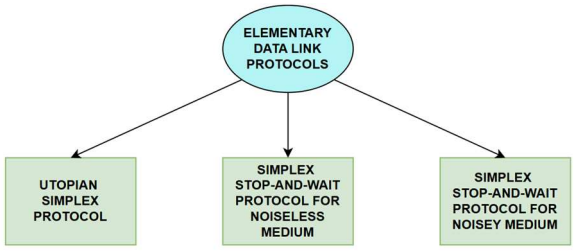
```
1101) 1 0 1 0 0 1 0 1 0 (11011
      1 1 0 1
      0 1 1 1 0
        1 1 0 1
        0 0 1 1 1 0
              1 1 0 1
              0 0 1 1 1 0
                    1 1 0 1
                    0 0 1 1
```

The reminder at the receiver is 0011

error transmission

3. ELEMENTARY DATA LINK PROTOCOLS:

- When we study the various Elementary data link protocols, we will assume that there is an unreliable channel over which frame loss occurs during transmission.
- To recover frames lost during transmission, the sender starts the internal clock each time a frame is sent.
- The sender will wait for some time, and if no reply is received within the predetermined time, the clock times out, and the data link layer will receive an interrupt signal.
- There are some methods or procedures executed by the protocol that turn the timer on and off, respectively.
- The clock is reset only after an interval is reached.
- The data link layer decides when to enable the network layer to send packets to prevent it from swamping packets with them.



3.1. Utopian Simplex Protocol:

- A utopian simplex protocol is a simple protocol because it does not worry about whether something is going right or wrong on the channel.
- In this protocol, data is transmitted in only one direction. Therefore, it is unidirectional.
- No matter what is happening in the network, the sender and receiver are always ready to communicate. So they also ignore the delay in processing.
- This protocol is just a consideration so that there is infinite buffer space available on the sender and receiver.

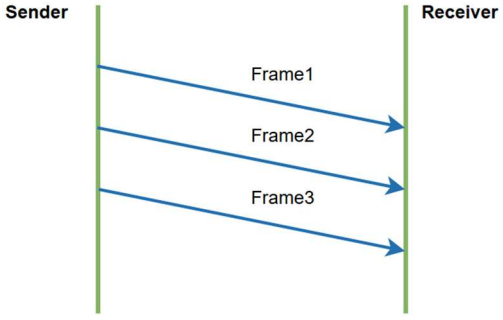


Fig: Utopian Simplex Protocol

- It is an unrealistic protocol, or you can say it is an unrestricted protocol.
- In this protocol, the channel used between layer-2 of the sender and receiver never discards or damages the frame during communication.
- Steps:
  - In protocol, two entities are sender and receiver, who communicate with each other over a channel.
  - The sender process and receiver process are running at the data link layer of the sender's machine and the receiver's machine, respectively.

- Sequence number and acknowledgment number are not used. Only the undamaged frame arrival process is going on.

3.2. Simplex Stop-and-Wait Protocol for Noiseless Channel

- In a stop-and-wait protocol, the sender stops after sending a frame to the receiver and waits for an acknowledgment before sending another frame.
- We here assume a noiseless channel that is error-free on which the frame is never damaged or corrupted.
- Here the channel is error-free but does not control the flow of data.
- Using the simplex stop-and-wait protocol, we can prevent the sender from flooding the receiver with frames faster than the receiver can process them.
- To prevent flooding on the receiver side, one solution is to enable the receiver to process frames back-to back by adding a buffer of sufficient size.
- We can enhance the processing capabilities of the receiver so that it can quickly pass the received frame to the network layer. But it's still not a general solution.
- Common solutions for addressing flooding issues on the receiver side, providing feedback to the sender to reduce the flow rate at the receiver.
- So that, in the simplex stop-and-wait protocol, the receiver sends a dummy frame back to the sender after the packet is sent over the network layer, asking the sender to send the next frame.
- Frames can be transmitted to or received from the sender or receiver, so the simplex stop-and-wait protocol is bidirectional.

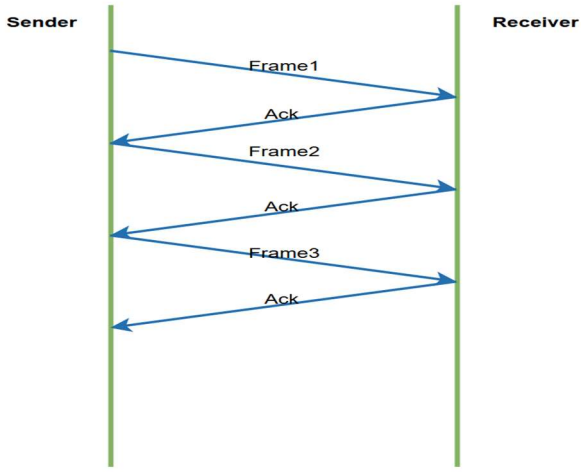


Fig: Stop-and-Wait Protocol Noiseless Channel

3.3. Simplex Stop-and-Wait Protocol for a Noisy Channel

- On a noisy channel, the receiver has only a limited buffer capacity and a limited processing speed, so the protocol prevents the sender from flooding the receiver with data too fast to handle it.
- In rare cases, the frame sent by the sender may be damaged in such a way that the checksum is correct, causing this and all other protocols to fail. To avoid this situation, a timer is added.

- Suppose, receiver's acknowledgment is lost during transmission, the sender will wait for acknowledgment for some time, and after the timeout, the sender will send the frame again. This process is repeated until the frame arrives and the acknowledgment is received from the receiver.
- The data link layer is responsible for flow and error control. Therefore, when the sender's network layer transmits a series of packets to the data link layer, the data link layer transmits the packets through the receiver's data link layer to the network layer

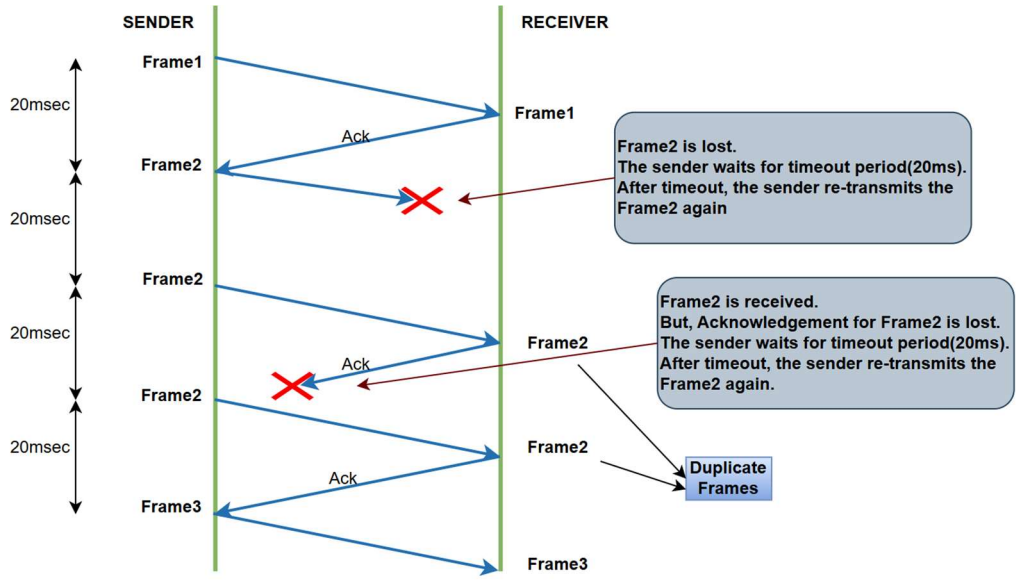


Fig: Stop-and-Wait Protocol Noisey Channel

4. SLIDING WINDOW PROTOCOL

- The sliding window protocol is a data link layer protocol that is useful in the sequential and reliable delivery of the data frames.
- Using the sliding window protocol, the sender can send multiple frames at a time.
- The sliding window protocol uses a mechanism of sequence numbers.
- The sender associates a sequence number to the data frames so that the receiver can use this sequence number to arrange the frames in order if any frame was re-transmitted.
- The sequence number also helps the receiver identify the loss of damaged packets.
- When the receiver receives the frame, it sends back an ACK (acknowledgment) to the sender.
- The ACK lets the sender know that a particular frame is received by the receiver correctly.
- There are two types of sliding window protocols namely - Go-Back-N ARQ, and Selective Repeat ARQ.

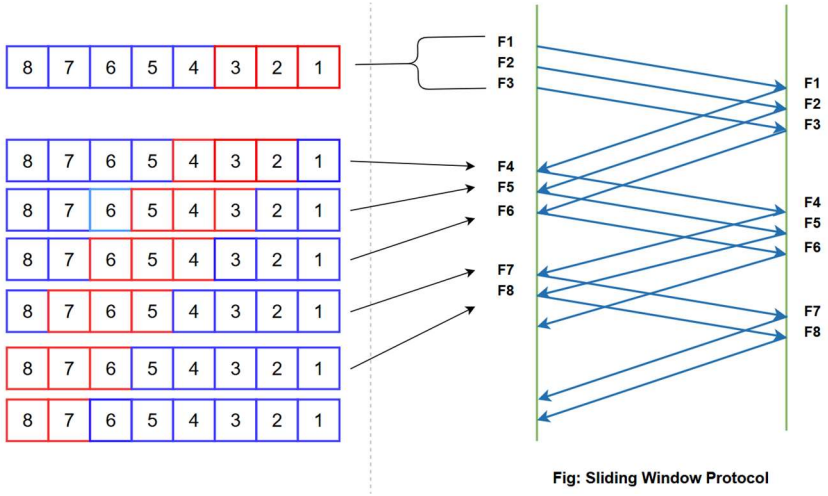


Fig: Sliding Window Protocol

4.1. A One-bit Sliding window protocol: is same as the Stop- and _Wait Protocol

4.2. A Protocol using Go-Back-N ARQ

- The Go-Back-N ARQ is one of the Sliding Window Protocol strategies that is used where reliable in-order delivery of the data packets is required.
- In the Go-Back-N ARQ, we use the concept of a timer. When the receiver receives the correct frame, it sends back an acknowledgment or ACK.
- Once the sender gets an ACK for a data frame, it shifts its window forward and sends the next frame.
- If the ACK is not received within the specified period, then all the data frames starting from the lost frame are retransmitted.
- In the Go-Back-N ARQ, the sender's window size is taken as N but the receiver's window size is always 1.
- Hence, the sender can send N data frames at a time but the receiver can receive only 1 frame at a time.
- The Go-Back-N ARQ is used for noisy channels or links and it manages the flow and error control between the sender and the receiver.
- Important Points regarding the Go-Back-N ARQ:

- In the Go-Back-N ARQ, we retransmit all the data frames starting from the lost or damaged frames.
- The ACK has the sequence number of the frame that helps the sender to identify the lost frame.
- The sender sets a timer for each frame so whenever the timer is over and the sender has not received any acknowledgment, then the sender knows that the particular frame is either lost or damaged.
- The ACK has the sequence number of the frame that helps the sender to identify the lost frame.
- The sender's window size is N.
- The receiver's window size is 1.
- The sender waits for the period specified by the timer before sending the frame again. Hence, the Go-Back-N ARQ is a slower protocol than the Selective Repeat ARQ.
- In Go-Back-N ARQ protocol, the efficiency is N/(1+2xa) where a is ratio of propagation delay vs the transmission delay and N is number of packets sent.
- Go-Back-N is quite easy to implement but here lot of bandwidth is used in case of high error rate for the retransmission of the entire window every time.

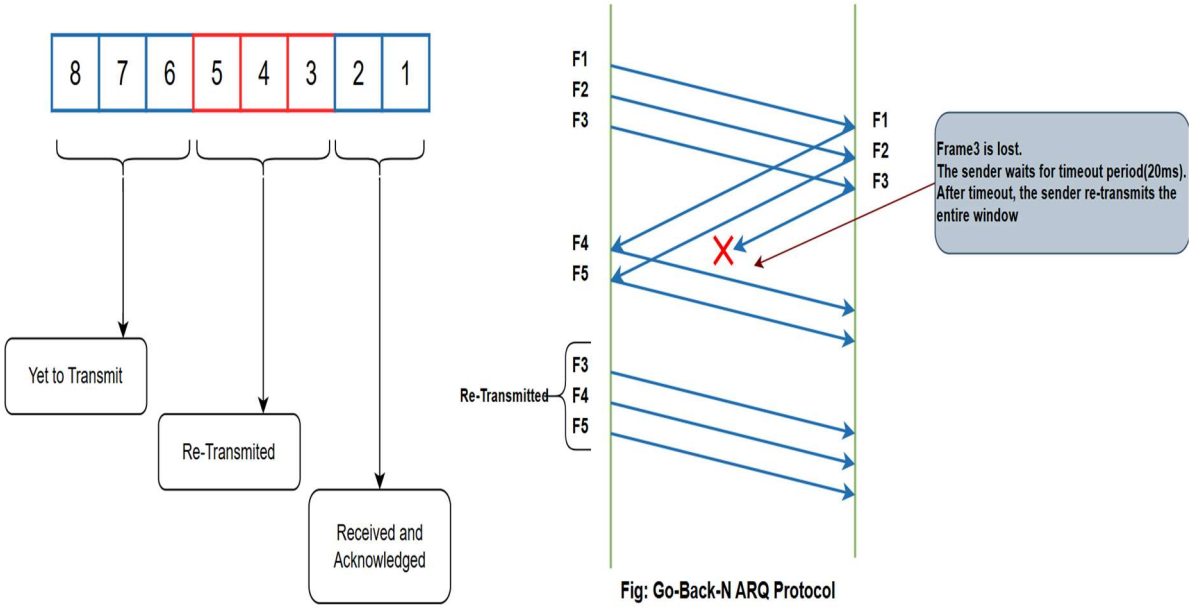
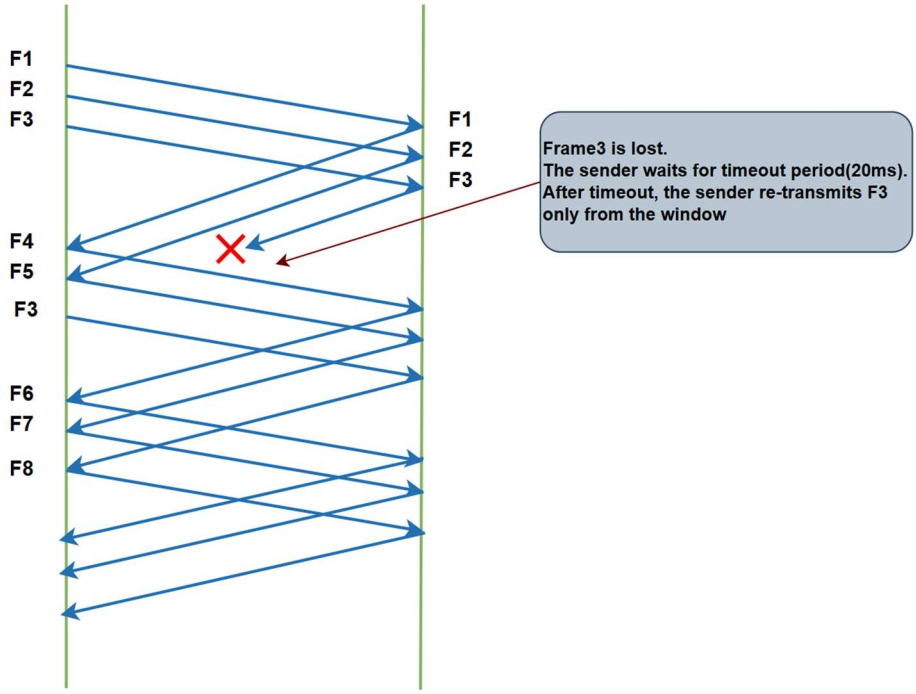


Fig: Go-Back-N ARQ Protocol

4.3. A protocol using Selective Repeat ARQ

- The selective repeat ARQ is one of the Sliding Window Protocol strategies that is used where reliable in order delivery of the data packets is required.
- The selective repeat ARQ is used for noisy channels or links and it manages the flow and error control between the sender and the receiver.
- In the selective repeat ARQ, we only retransmit the data frames that are damaged or lost.
- If any frame is lost or damaged then the receiver sends a negative acknowledgment (NACK) to the sender and if the frame is correctly received, it sends back an acknowledgment (ACK).
- As we only resend the selected damaged frames so we name this technique the Selective Repeat ARQ technique.
- The ACK and the NACK have the sequence number of the frame that helps the sender to identify the lost frame.
- ***Important Points regarding the Selective Repeat ARQ:***
  - In the selective repeat ARQ, we only resend the data frames that are damaged or lost.
  - If any frame is lost or damaged then the receiver sends a negative acknowledgment (NACK) to the sender and if the frame is correctly received, it sends back an acknowledgment (ACK).
  - The sender sets a timer for each frame so whenever the timer is over and the sender has not received any acknowledgment, then the sender knows that the particular frame is either lost or damaged.
  - As the sender needs to wait for the timer to expire before retransmission. So, we use negative acknowledgment or NACK.
  - The ACK and the NACK have the sequence number of the frame that helps the sender to identify the lost frame.
  - The receiver has the capability of sorting the frames present in the memory buffer using the sequence numbers.
  - The sender must be capable enough to search for the lost frame for which the NACK has been received.
  - The size of the sender's window is 2(m−1)2(m−1), where m is the number of bits used in the header of the packet to express the packet's sequence number.
  - The window size of the receiver is the same as that of the sender i.e. 2(m−1)2(m−1).
  - In Selective Repeat protocol, the efficiency is N/(1+2xa) where a is ratio of propagation delay vs the transmission delay and N is number of packets sent.
  - It is efficient as only the lost or broken packets need retransmission.
  - Let us briefly discuss ARQ. ARQ stands for Automatic Repeat Request. ARQ is an error-control strategy that ensures that a sequence of information is delivered in order and without any errors or duplications despite transmission errors and losses.



- **Fig: Selective Repeat ARQ Protocol**