2.1 Framing:

Framing in the data link layer separates a message from one source to a destination by adding a sender address & a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt. Although the whole message could be packed in one frame, that is not normally done. One reason is that a frame can be very large, making flow & error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message. When a message is divided into smaller frames, a single-bit error affects only that small frame.

Frames can be of fixed or variable size.

i) Fixed-size framing:

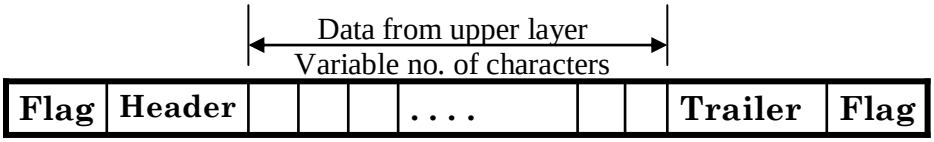
In this there is no need for defining the boundaries of the frames, the size itself can be used as a delimiter.
Ex: ATM wide area network which uses frames of fixed size called cells.

ii) Variable-size framing:

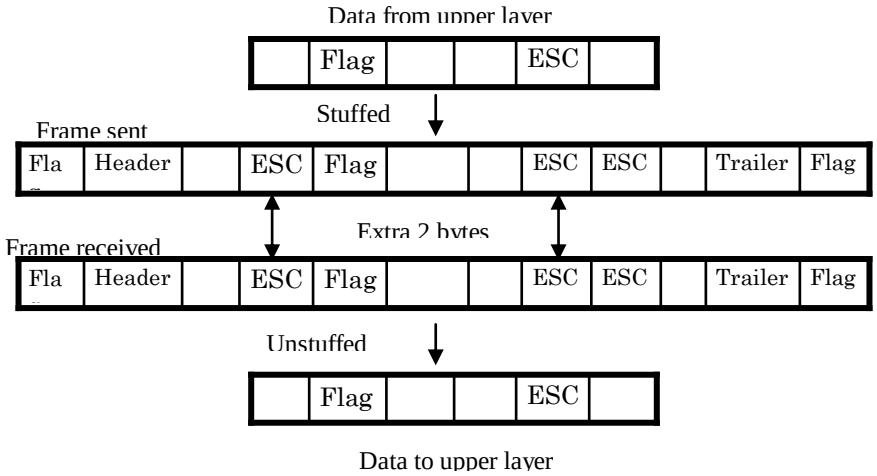
In this, we need a way to define the end of the frame and the beginning of the next.
Ex: LAN

Historically 2 approaches were used for variable size framing: Character-oriented & bit-oriented.

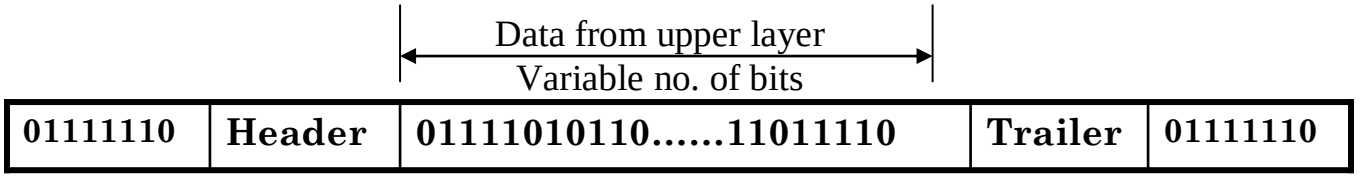
a) Character-oriented approach:



Character oriented protocol – Byte Stuffing & De-stuffing

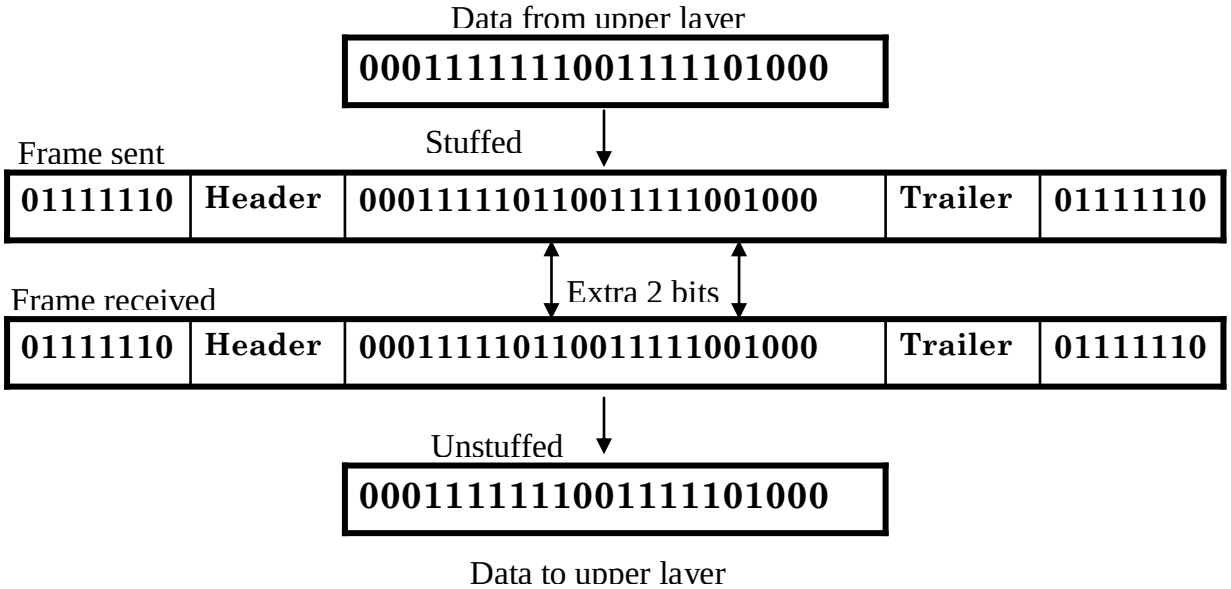


b) Bit-oriented approach:



Bit oriented protocol – Bit Stuffing & De-stuffing

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver doesn't mistake the pattern 01111110 for a flag



2.2 Flow & Error Control:

Flow Control:

- Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data.
- The flow of data must not be allowed to overwhelm the receiver.
- The receiver must be able to tell the sender to halt transmission until it is once again able to receive.
- Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgement.

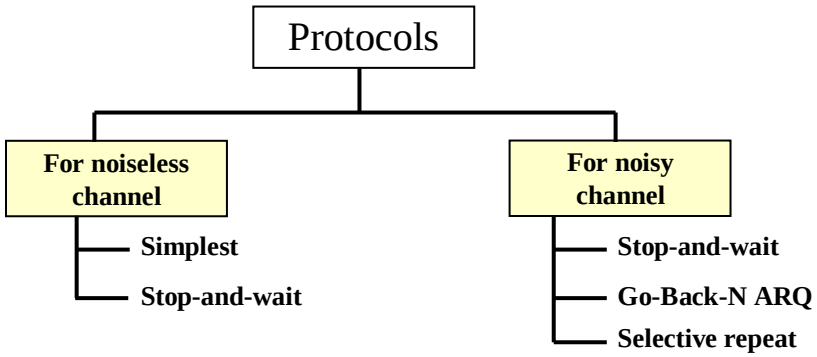
Error Control:

- Errors occur due to noises in the channel.
- Error control is both error detection & error correction.
- In the data link layer error control refers to methods of error detection and retransmission.
- To sender should add certain amount of redundant bits to the data, based on which the receiver will be able to detect errors.

2.3 PROTOCOLS:

- A protocol is a set of rules that govern data communication.
- A protocol defines what, how it is communicated, and when it is communicated.
- The key elements of a protocol are syntax, semantics & timing.
- Syntax refers to the structure or format of the data, i.e., the order in which they are presented.
- Semantics refers to the meaning of each section of bits.
- Timing refers to when data should be sent & how fast they can be sent.
- Protocols are implemented in software by using any of the common programming languages.

The protocols in the data link layer are classified as



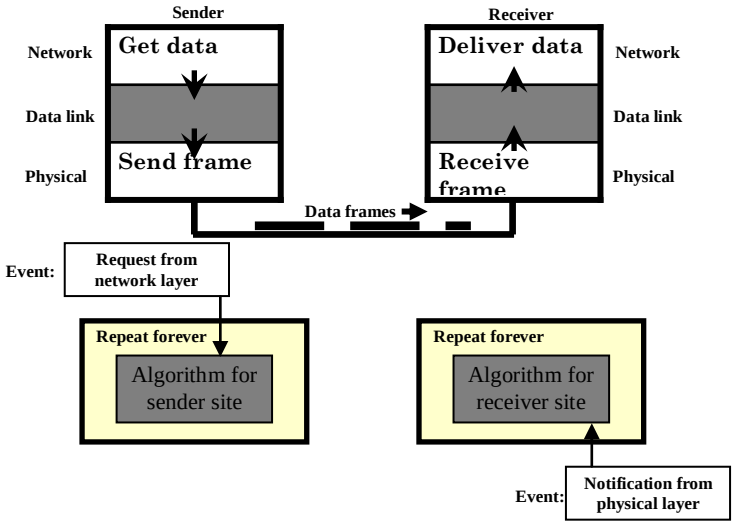
2.4 Noiseless channels:

i) Simplest protocol:

Assumptions in this protocol are:

- Data transfer is unidirectional.
- Both sender & receiver network layers are always ready.
- Processing time can be ignored.
- Infinite buffer space is available.
- Frames are never damaged or lost.

Design:



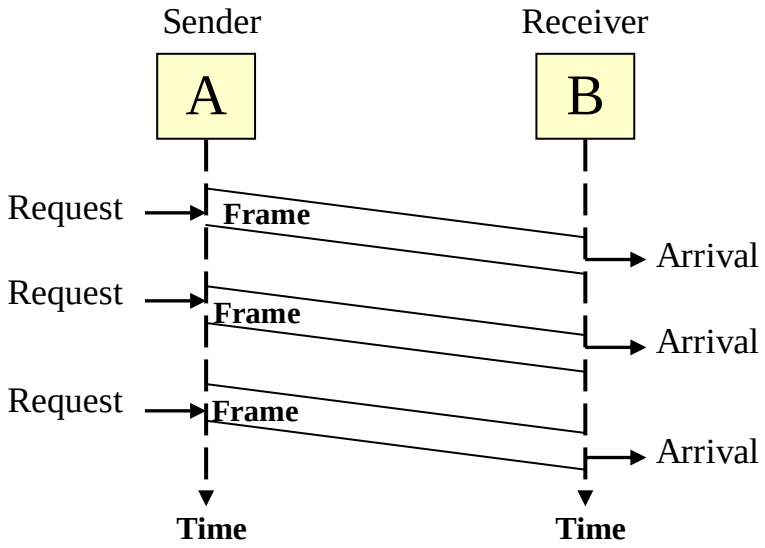
Sender-site algorithm for the Simplest Protocol:

```
While(true)                    //Repeat forever
{
  waitForEvent();                    //Sleep until an event occurs
  if(event(requestTosend))           //there is a packet to send
    {
      GetData();                     //get data from n/w layer
      MakeFrame();                   //make a frame
      SendFrame();                   //send the frame
    }
}
```

Receiver-site algorithm for the Simplest Protocol:

```
While(true)                    //Repeat forever
{
  waitForEvent();                    //Sleep until an event occurs
  if(event(ArrivalNotification)) //data frame arrived
    {
      ReceiveFrame();          //receive frame from the physical layer
      ExtractData();                 //extract data from a frame
      deliverData();                 //deliver the data to the n/w layer
    }
}
```

Flow diagram to illustrate simplest protocol:

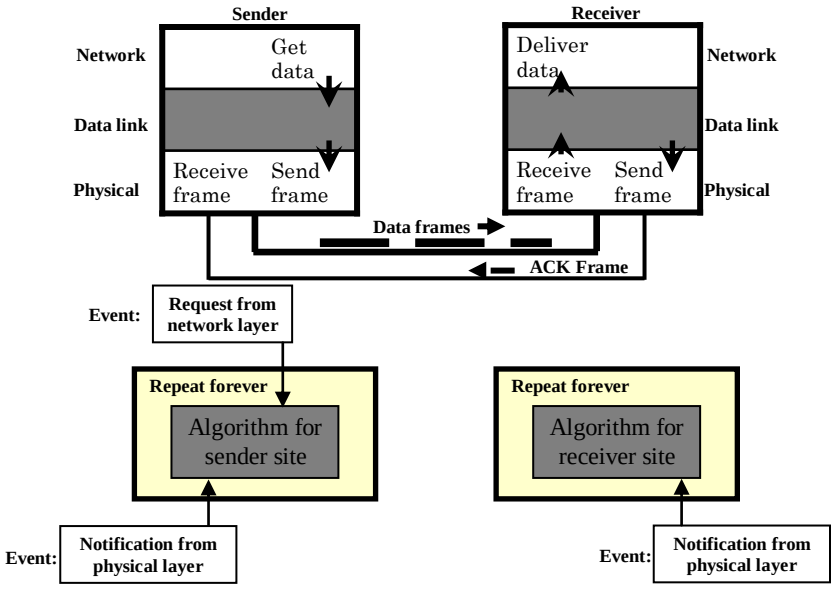


ii) Stop - & - Wait Protocol:

Assumptions in this protocol are:

- Data transfer is unidirectional.
- Both sender & receiver network layers are always ready.
- Receiver doesn't have enough storage space.
- Receiver is slower than sender in processing.
- Frames are never damaged or lost.

Design:



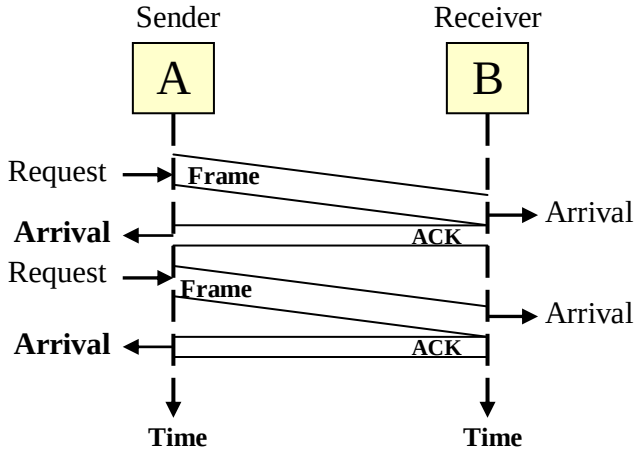
Sender-site algorithm for the Stop-&-wait Protocol:

```
While(true)                    //Repeat forever
Cansend = true                 //Allow the first frame to go
{
  waitForEvent();                      //Sleep until an event occurs
  if(event(requestTosend) AND cansend) //there is a packet to send
    {
      GetData();                       //get data from n/w layer
      MakeFrame();                     //make a frame
      SendFrame();                     //send the frame
      cansend = false;         //can't send until ACK arrives
    }
  waitForEvent();                      //Sleep until an event occurs
  if(Event(ArrivalNotification))//an ACK has arrived
   {
     ReceiveFrame();           //Receive the ACK frame
      cansend = true;
    }
}
```

Receiver-site algorithm for the Stop-&-Wait Protocol:

```
While(true)                    //Repeat forever
{
  waitForEvent();                      //Sleep until an event occurs
  if(event(ArrivalNotification)) //data frame arrived
    {
      ReceiveFrame();          //receive frame from the physical layer
      ExtractData();                   //extract data from a frame
      deliverData();                   //deliver the data to the n/w layer
      SendFrame();                     //Send an ACK frame
     }
}
```

Flow diagram to illustrate Stop-&-Wait protocol:



2.5 Noisy channels:

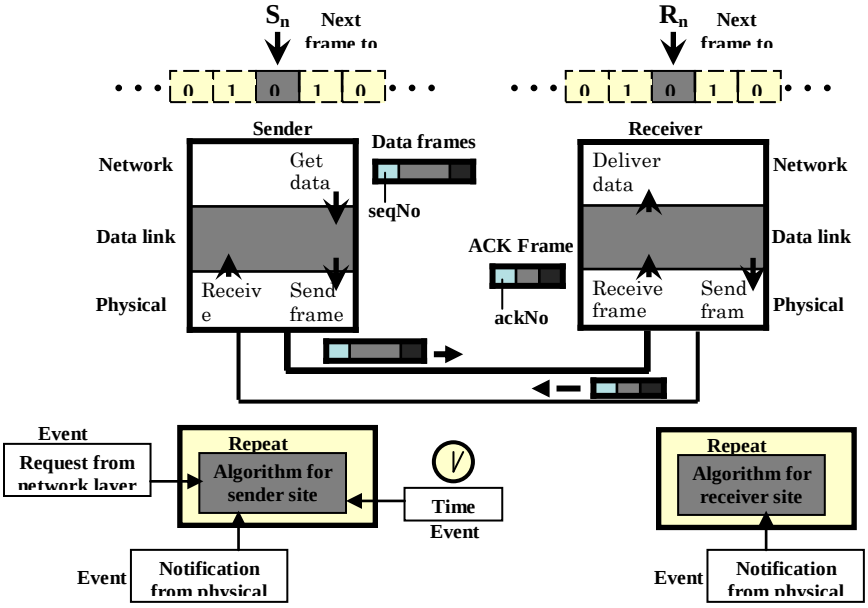
Assumptions are:

- Data transfer is unidirectional, but half-duplex link.
- Both sender & receiver network layers are always ready.
- Receiver doesn't have enough storage space.
- Receiver is slower than sender in processing.
- Frames are damaged or lost because of the noises in the channel.

i) Stop - & - Wait Automatic Repeat Request (ARQ):

- Adds a simple error control mechanism to the Stop-&-Wait protocol.
- Error detection & retransmission is used for error control.
- Sending device keeps a copy of the last frame transmitted until acknowledgement for that frame is received.
- Data frames uses Sequence numbers, to avoid duplication of frames.
- Range of sequence numbers = 2m – 1.
- ACK frame uses Acknowledgement number, & always announces the sequence number of the next frame expected.
- Timers are used in case of loss of ACK frames.

Design:



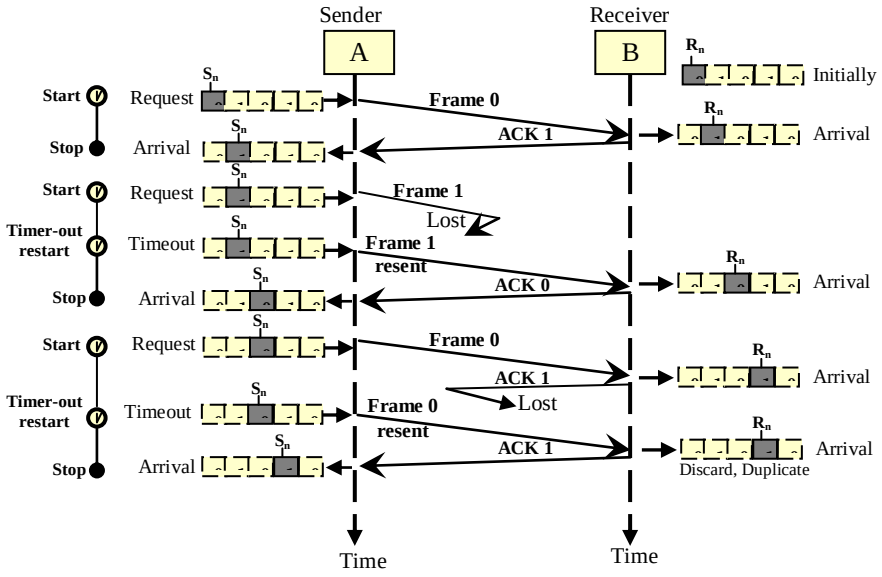
Sender-site algorithm:

```
Sn = 0;                          //Frame 0 should be sent first
Cansend = true                   //Allow the first request to go
While(true)                      //Repeat forever
{
  waitForEvent();                          //Sleep until an event occurs
  if(event(requestTosend) AND cansend) //there is a packet to send
    {
      GetData();                 //get data from n/w layer
      MakeFrame(Sn);             //make a frame
      StoreFrame(Sn);                  //copy of frame
      SendFrame(Sn);             //send the frame
      StartTimer();
      Sn = Sn + 1;                         //Mod-2 addition
      cansend = false;           //can't send until ACK arrives
    }
  waitForEvent();                          //Sleep until an event occurs
if(Event(ArrivalNotification))             //an ACK has arrived
   {
     ReceiveFrame(ackNo);                  //Receive the ACK frame
      if( not corrupted AND ackNo == Sn) // valid ACK
        {
          Stoptimer();
          purgeframe(Sn-1);
          cansend = true;
        }
    }
If(Event(Timeout))
  {
    StartTimer();
    ResendFrame(Sn-1);
  }
}
```

Receiver-site algorithm:

```
Rn = 0;                          //Frame 0 expected to arrive first
While(true)                      //Repeat forever
{
  waitForEvent();                          //Sleep until an event occurs
  if(event(ArrivalNotification)) //data frame arrived
    {
      ReceiveFrame();            //receive frame from the physical layer
      if(Corrupted(Frame))
         sleep();
      if(seqNo == Rn)
        {
          ExtractData();                   //extract data from a frame
          DeliverData();                   //deliver the data to the n/w layer
          Rn = Rn +1;
        }
       SendFrame();                        //Send an ACK frame
    }
}
```

Flow diagram:

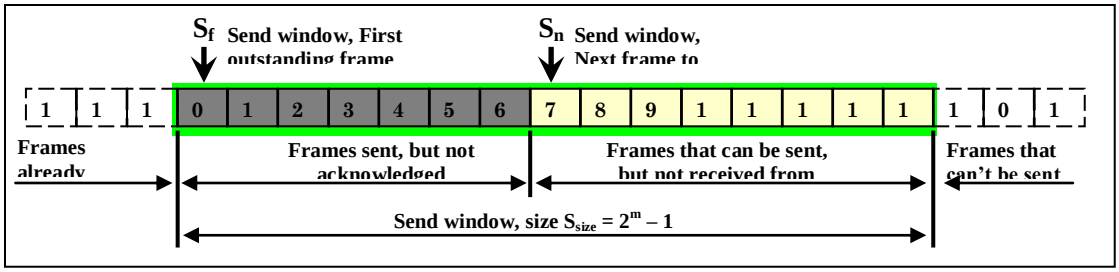


ii) Go-Back-N ARQ Protocol:

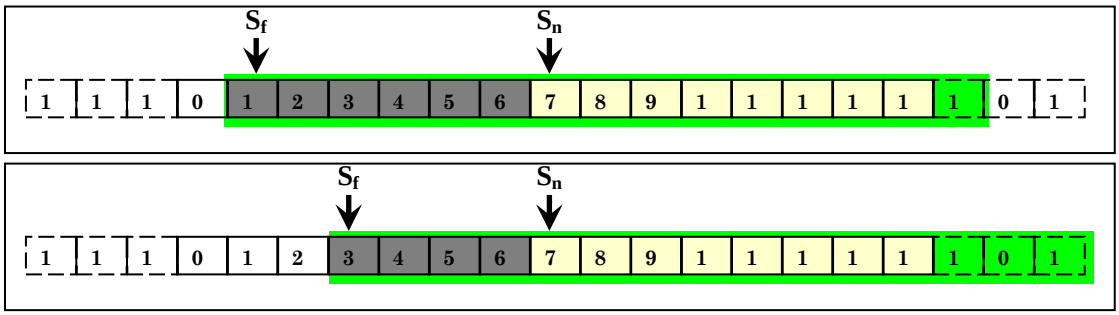
- Bandwidth-Delay product.
- Pipelining.
- Sequence numbers range from 0 to 2m – 1.
- Sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the Sender & Receiver.
- Timers – timer for the first outstanding frame always expires first, we resend all outstanding frames when this timer expires.
- Acknowledgement – sends ACK for safe and in order frames, receiver will be silent for corrupted & out of order frames.
- Resending a frame – when the timer expires, the sender resends all the outstanding frames.

Send window for Go-Back-N ARQ

a) Send window before sliding

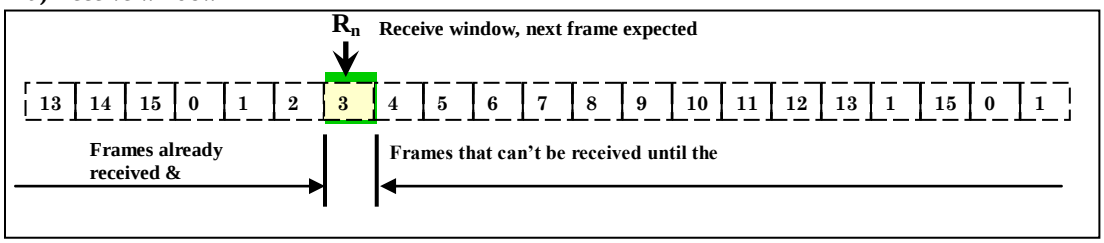


b) Send window after sliding

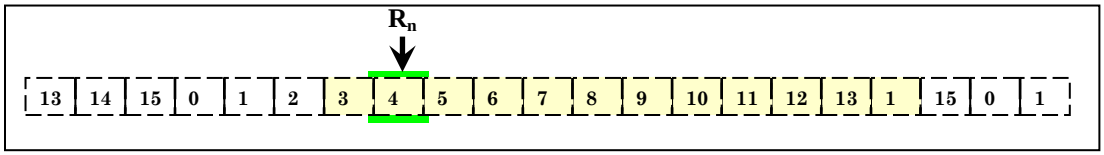


Receive window for Go-Back-N ARQ

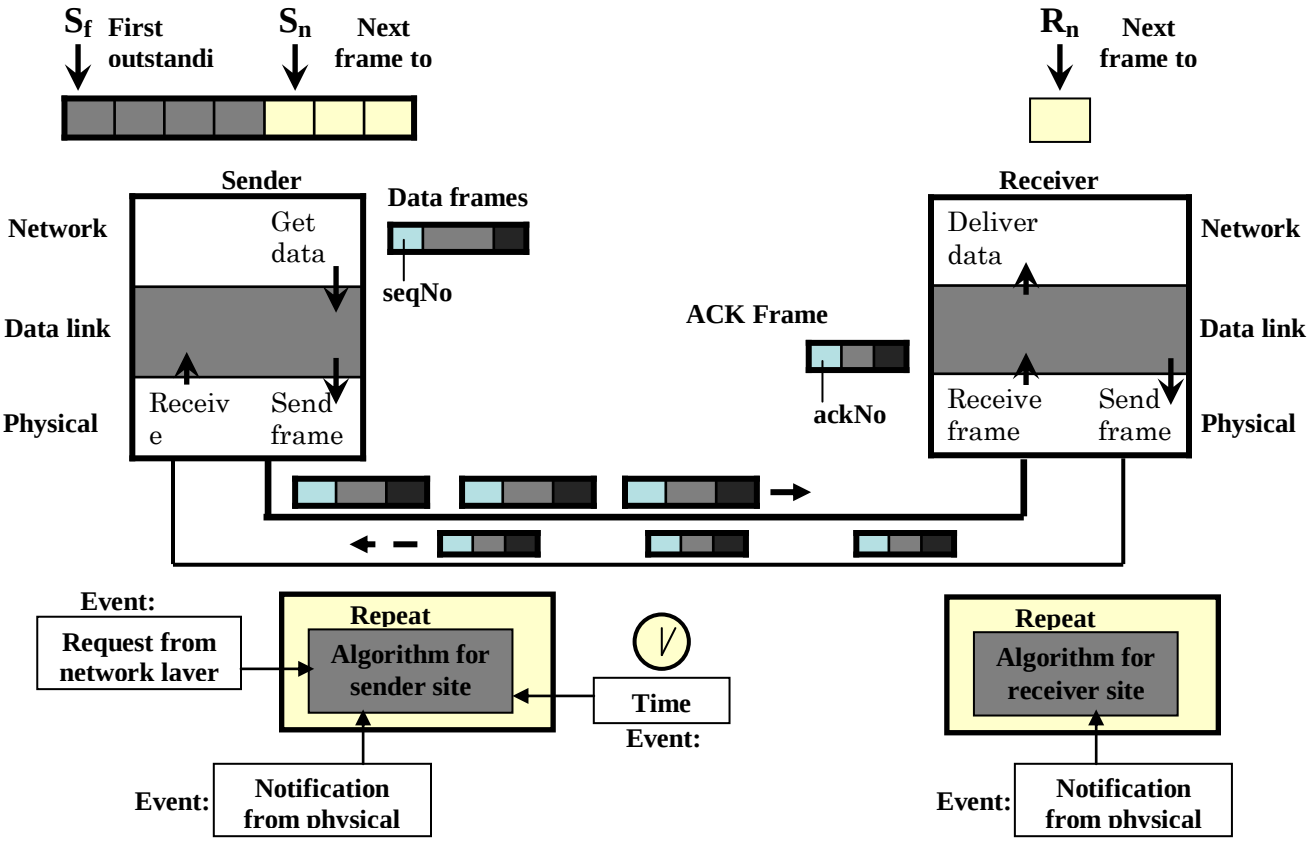
a) Receive window



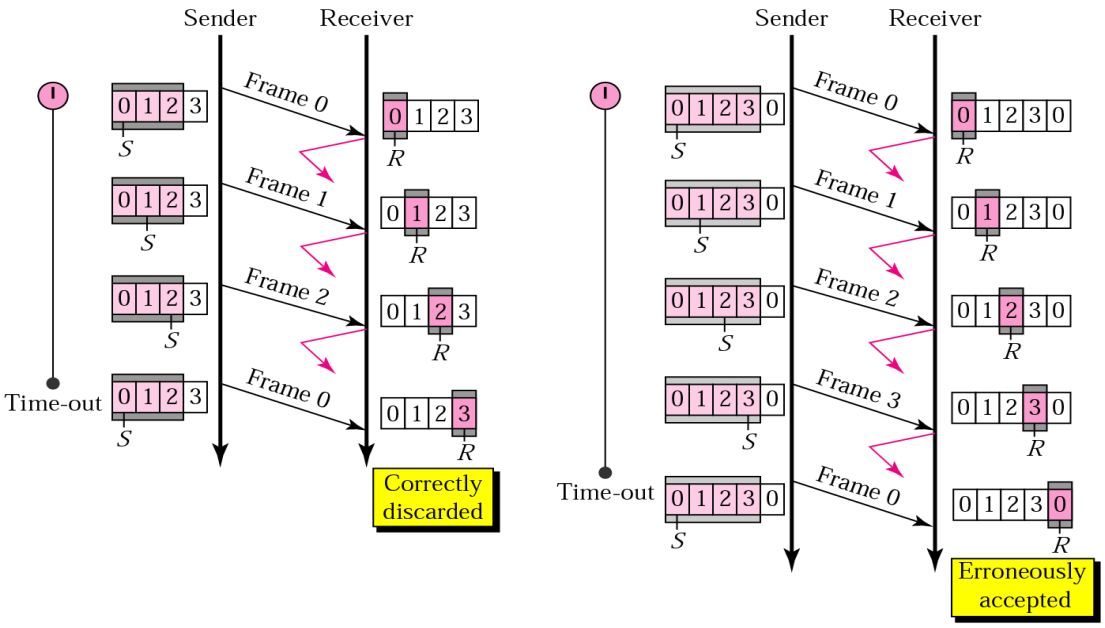
b) Window after sliding



Design:



Window size for Go-Back-N ARQ:



a. Window size < 2^m b. Window size = 2^m

Sender-site algorithm:

```
Sw = 2^m – 1;
Sf = 0;
Sn = 0;                              //Frame 0 should be sent first
While(true)                          //Repeat forever
{
  waitForEvent();                          //Sleep until an event occurs
  if(event(requestTosend)) //there is a packet to send
    {
      if(Sn – Sf >= Sw)              //If window is full
         Sleep();
      GetData();                           //get data from n/w layer
      MakeFrame(Sn);                 //make a frame
      StoreFrame(Sn);                       //copy of frame
      SendFrame(Sn);                 //send the frame
      StartTimer();
      Sn = Sn + 1;                         //Mod-2 addition
      if(timer not running);               //can't send until ACK arrives
         StartTimer();
    }
if(Event(ArrivalNotification))             //an ACK has arrived
   {
     Receive (ACK);                  //Receive the ACK frame
      if(corrupted (ACK))
        Sleep();
       if((ackNo > Sf && (ackNo <= Sn)))
       While(Sf  <= ackNo)
          {
           Purgeframe(Sn-1);
           Sf = Sf + 1;
          }
        StopTimer();
    }
If(Event(Timeout))
  {
    StartTimer();
    Temp = Sf;
    While(Temp < Sn)
       {
         SendFrame(Sf);              //sendframe(temp);
          Sf = Sf + 1;               //temp = temp+1;
        }
  }
}
```

Receiver-site algorithm :

```
Rn = 0;                                  //Frame 0 expected to arrive first
While(true)                        //Repeat forever
{
```

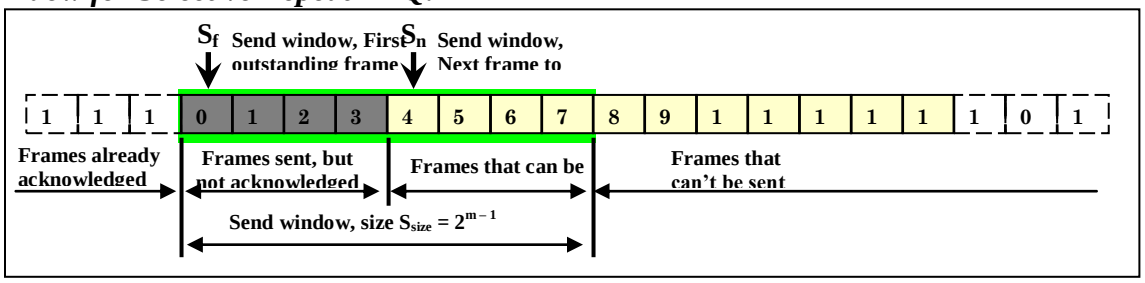
```
  waitForEvent();                    //Sleep until an event occurs
  if(event(ArrivalNotification))     //data frame arrived
    {
      ReceiveFrame();          //receive frame from the physical layer
      if(Corrupted(Frame))
         sleep();
      if(seqNo == Rn)
        {
          DeliverData();               //deliver the data to the n/w layer
          Rn = Rn +1;          //Slide window
          SendACK(Rn);
        }
    }
}
```

Stop-and-Wait ARQ is actually a Go-Back-N ARQ in which there are only two sequence numbers and the send window size is 1.
i.e., m = 1; $2^m - 1 = 1$

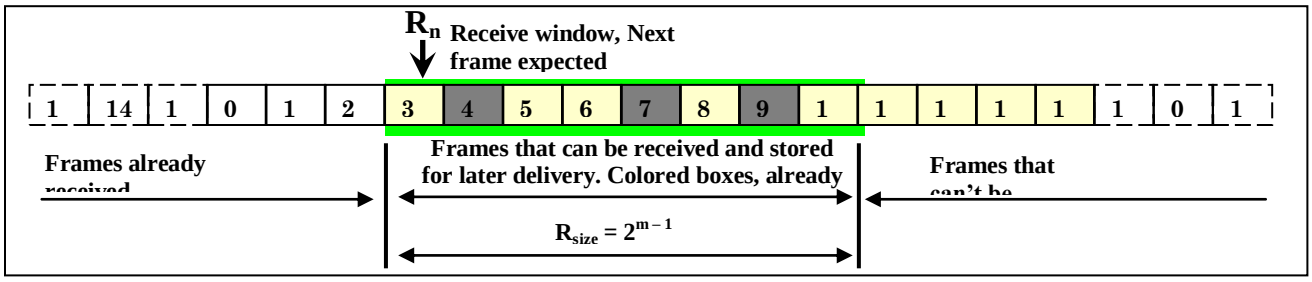
iii) Selective Repeat ARQ Protocol:

- In a noisy link frame has a higher probability of damage, means resending of multiple frames.
- This uses more bandwidth & slows down the transmission.
- Instead of sending N frames when just one frame is damaged, a mechanism is required to resend only the damaged frame.
- Sequence numbers range from 0 to 2m – 1.
- Both send & receive window size = 2(m – 1).

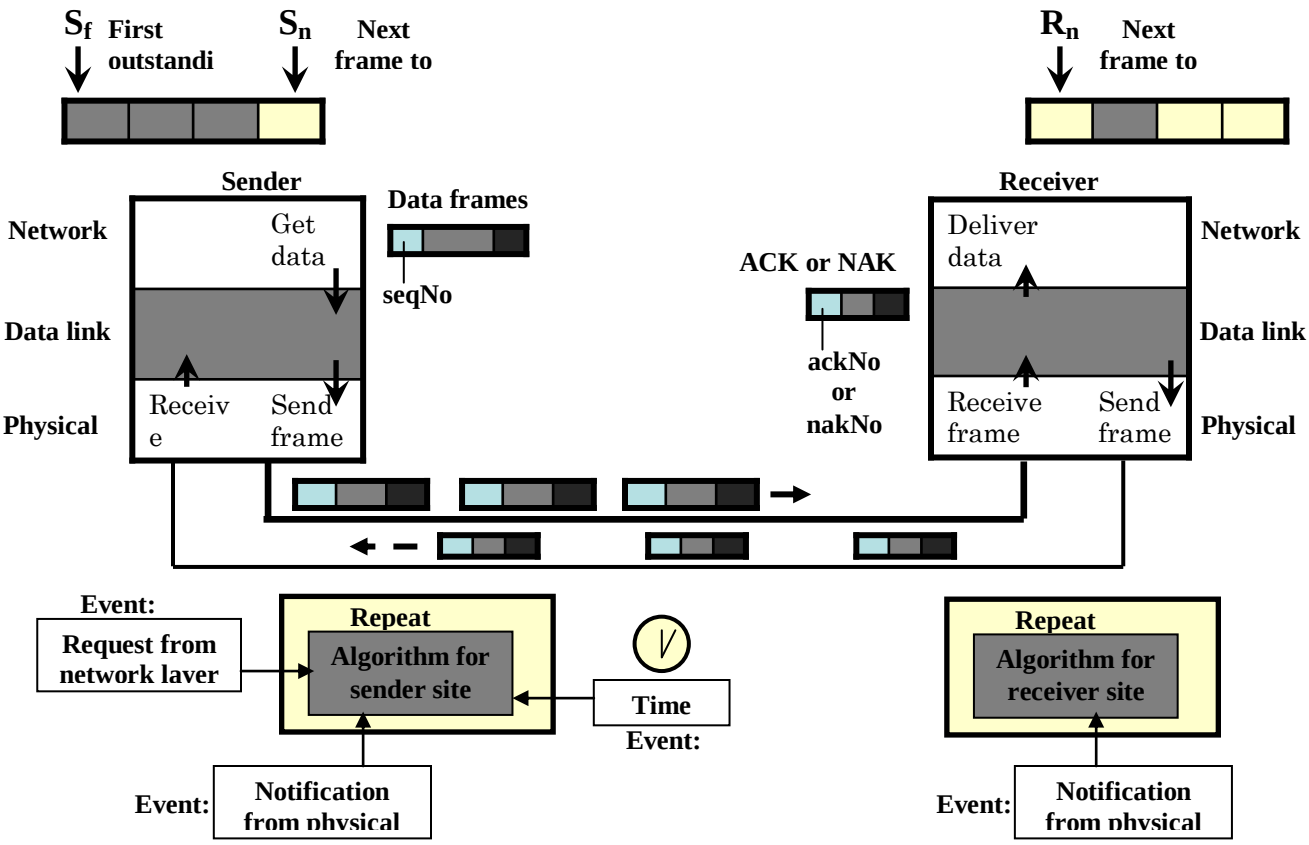
Send window for Selective Repeat ARQ:



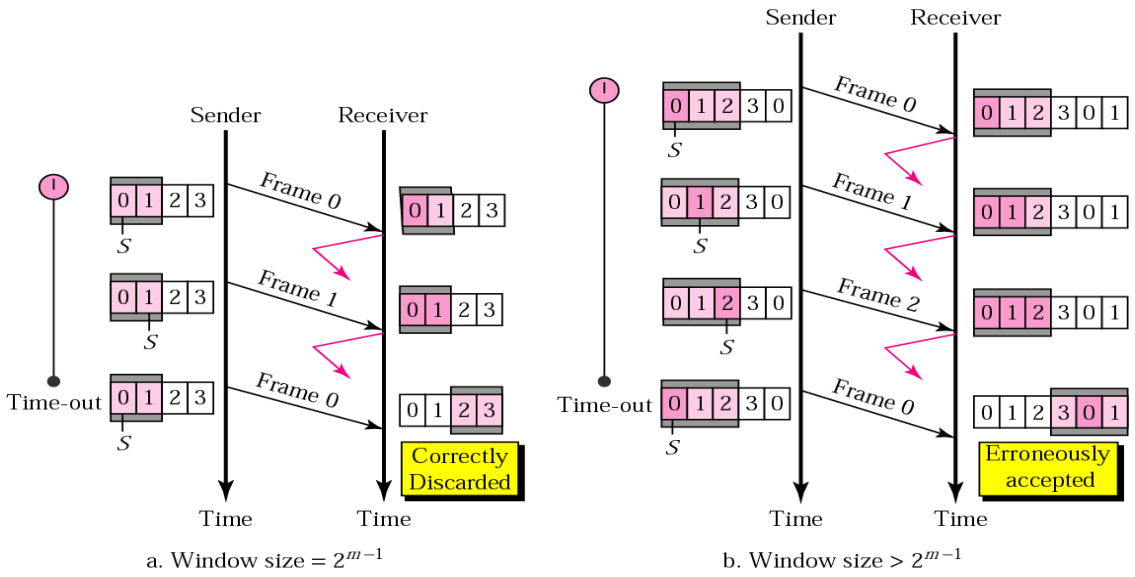
Receive window for Selective Repeat ARQ:



Design:



Window size for Selective Repeat ARQ:



Sender-site algorithm:

```
Sw = 2m-1 ;
Sf = 0;
Sn = 0;                          //Frame 0 should be sent first
While(true)                      //Repeat forever
{
  waitForEvent();                //Sleep until an event occurs
  if(event(requestTosend)) //there is a packet to send
    {
      if(Sn – Sf >= Sw)          //If window is full
         Sleep();
      GetData();                 //get data from n/w layer
      MakeFrame(Sn);             //make a frame
      StoreFrame(Sn);                 //copy of frame
      SendFrame(Sn);         //send the frame
      Sn = Sn + 1;
      StartTimer(Sn);
}
if(Event(ArrivalNotification))   //an ACK has arrived
   {  Receive (frame);           //Receive the ACK frame
      if (corrupted (frame))
        Sleep();
       if (FrameType == NAK)
          if (nakNo between Sf & Sn)
             {  resend(nakNo);
                StartTimer(nakNo);
             }
          if (FrameType == ACK)
             if (ackNo between Sf & Sn)
```

```
            {   While (Sf  <= ackNo)
                        {
                         Purge (Sf);
                         StopTimer (Sf);
                         Sf = Sf + 1;
                        }
             }
       }
If (Event (Timeout (t)))
  {
    StartTimer(t);
    SendFrame(t);
   }
}
```

Receiver-site algorithm :

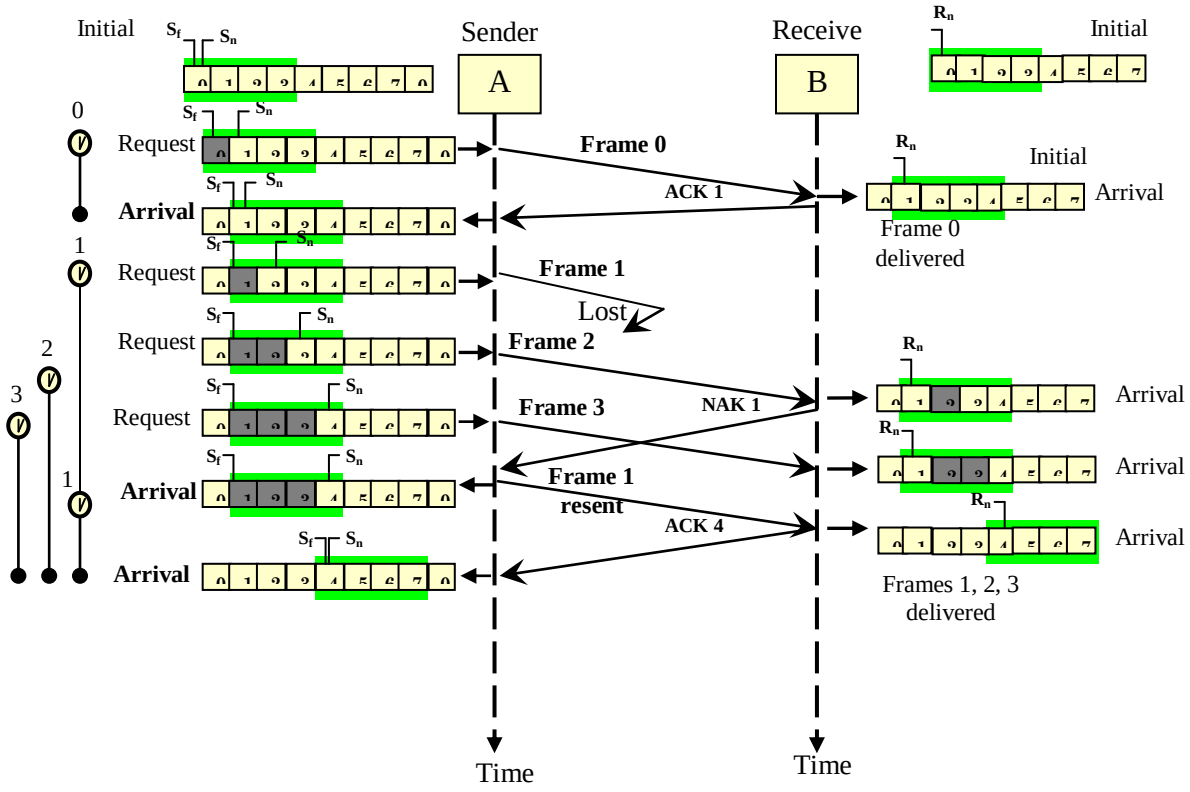
```
Rn = 0;                    //Frame 0 expected to arrive first
Naksent = false;
AckNeeded = false;
Repeat( for all slots)
     Marked (slot) = false;

While(true)                        //Repeat forever
{
  waitForEvent();                          //Sleep until an event occurs
  if(event(ArrivalNotification))           //data frame arrived
    {
      ReceiveFrame();
      if(Corrupted(Frame) && NOT NakSent)
        {
         SendNAK(Rn);
          Naksent = true;
         sleep();
         }
if(seqNo < > Rn && NOT Naksent)
   {   SendNAK(Rn);
       Naksent = true;
       if((seqno in window) && (!Marked(seqno)))
         {  StoreFrame(seqNo);
            Marked(seqNo) = true;
            while(Marked(Rn))
              {  DeliverData(Rn);              //deliver the data to the n/w layer
                 purge(Rn);
                 Rn = Rn +1;           //Slide window
                 AckNeeded = true;  }
            if(AckNeeded)
```

```
                { SendACK(Rn);
                  AckNeeded = false;
                  NakSent = false;      }
            }
    }
    }
}
```

Piggybacking

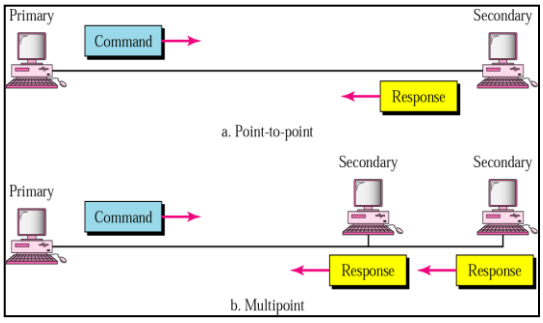
- When the data transfer is bidirectional, i.e., from node A to node B and from node B to node A, the control information also needs to flow in both the directions.
- Efficiency can be improved if the control information can be passed to the other end along with the data itself which is flowing to that end. This concept is called as piggybacking.

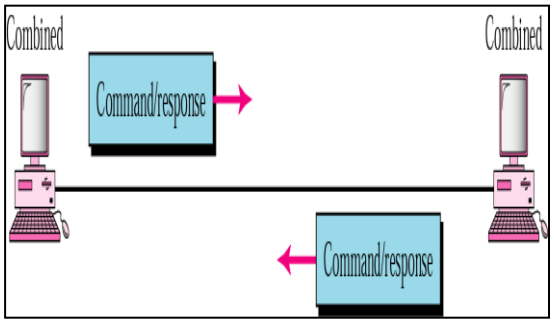
Flow diagram:



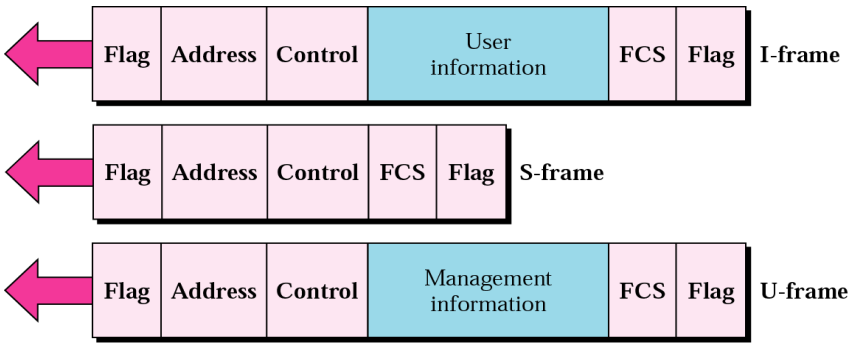
2.6 High-level Data Link Control (HDLC):

- HDLC is a bit-oriented protocol for communication over point-to-point and multipoint links.
- It implements the ARQ mechanisms.
- HDLC provides two common transfer modes.

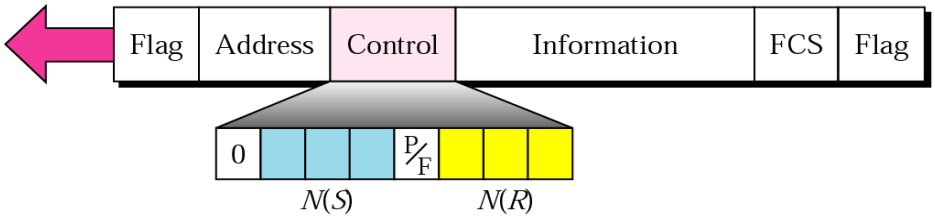




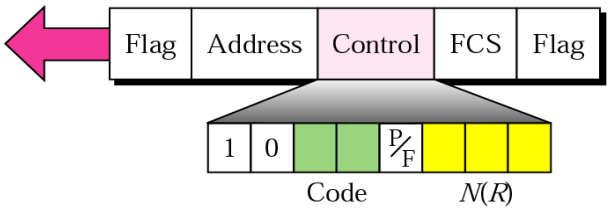
Frame format :



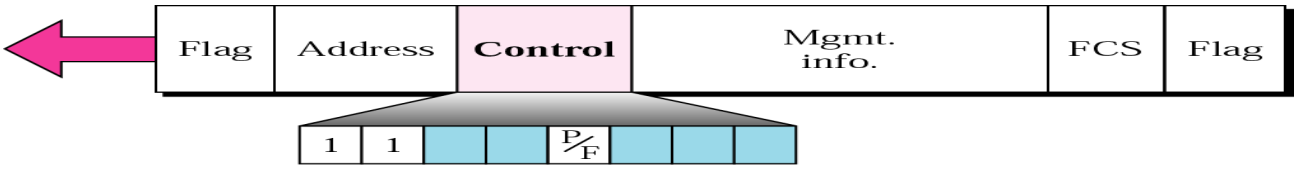
Information frames (I-Frames) :



Supervisory frames (I-Frames) :



Unnumbered frames (I-Frames) :



Code	*Command*	*Response*	*Meaning*
00 001			**Set normal response mode**
11 011	**SNRME**		**Set normal response mode (extended)**
11 100	**SABM**	**DM**	**Set asynchronous balanced mode or disconnect mode**
11 110	**SABME**		**Set asynchronous balanced mode (extended)**
00 000	**UI**	**UI**	**Unnumbered information**
00 110		**UA**	**Unnumbered acknowledgment**
00 010	**DISC**	**RD**	**Disconnect or request disconnect**
10 000	**SIM**	**RIM**	**Set initialization mode or request information mode**
00 100	**UP**		**Unnumbered poll**
11 001	**RSET**		**Reset**
11 101	**XID**	**XID**	**Exchange ID**
10 001	**FRMR**	**FRMR**	**Frame reject**

Recommended questions:

1. What are the three protocols considered for noisy channels?
2. Define framing and the reason for its need.
3. Compare and contrast the G0-Back-N ARQ protocol with Selective repeat ARQ.
4. Briefly describe the services provided by the data link layer.
5. Define piggybacking and its uses.
6. Compare and contrast byte-stuffing and bit-stuffing. Which technique is used in byte-oriented protocols? Which technique is used in bit-oriented protocols?
7. Which all the protocols of data link layer uses pipelining concept?
8. Compare and contrast flow control and error control.
9. Compare and contrast HDLC with PPP. Which one is byte-oriented; which one is bit-oriented?
10. Explain the reason for moving from stop-and-wait ARQ protocol to the Go-Back-N ARQ protocol.

Unit 3: Multiple Accesses Hrs: 06

Syllabus:

Random access, Controlled access, channelization

Recommended readings:

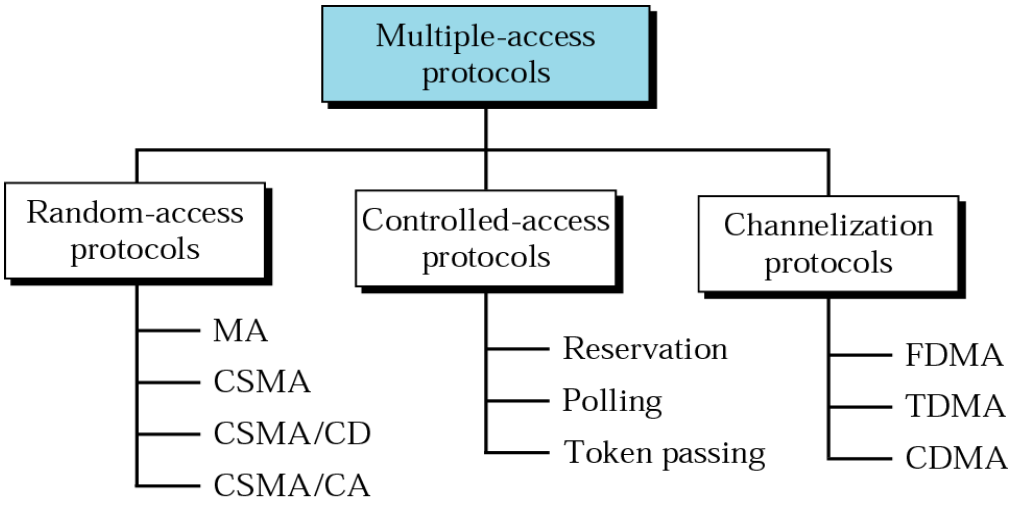
Text Book: Data Communication & Networking, B Forouzan, 4e, TMH

Chapter 12 – page 363 to 396

3.1 Introduction:

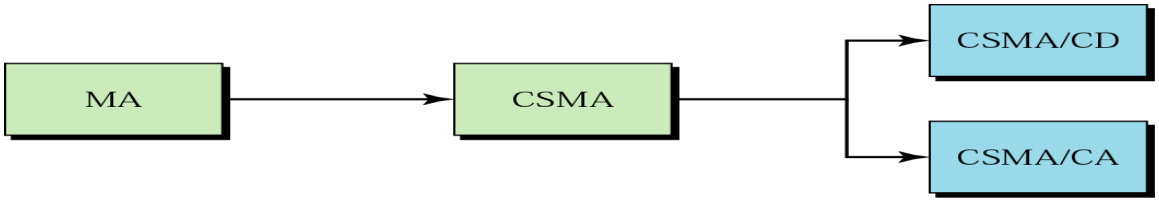
- Data link layer has two sub layers: Upper sub layer – Data link Control & Lower sub layer – Multiple access Control.
- The upper sub layer is responsible for flow and error control.
- The lower sub layer is responsible for multiple access resolution.
- When the nodes are connected using a dedicated link, lower sub layer is not required, but when the nodes are connected using a multipoint link (broadcast), multiple access resolution is required.

Taxonomy of Multiple access protocols

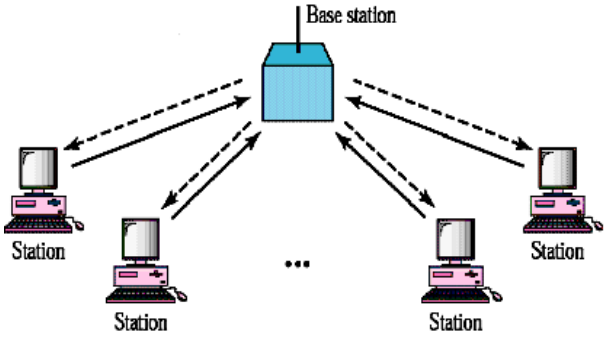


3.2 RANDOM ACCESS

- No station is superior to another station and none is assigned control over another.
- No station permits, or does not permit, another station to send.
- At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send.
- Transmission is random among the stations.
- Each station has the right to the medium without being controlled by any other station. All stations compete with one another to access the medium. Random access methods are also called as *contention methods*.
- If more than one station tries to send, there is an access conflict – collision, frames will be either destroyed or modified.



i) ALOHA

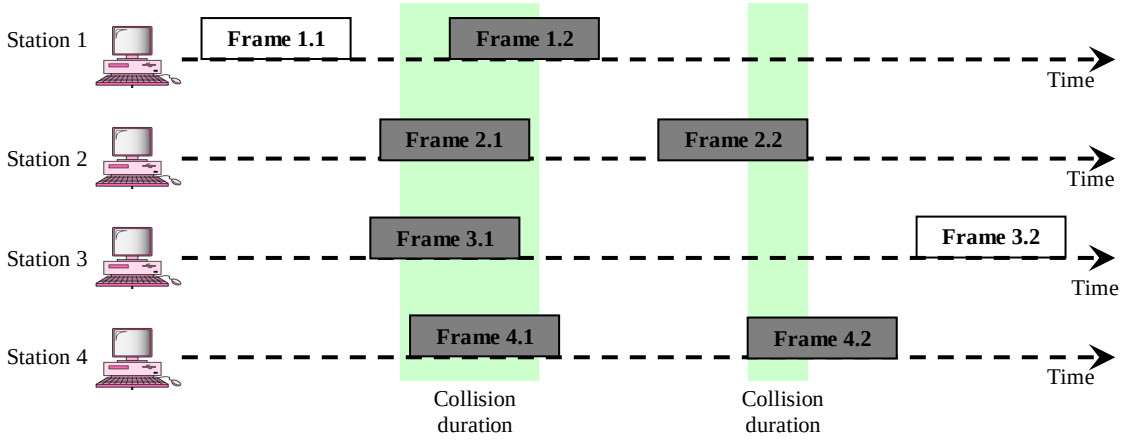


- The earliest random access method, was developed at the university of Hawaii in early 1970.
- It was designed for a radio (wireless) LAN, but it can be used on any shared medium.
- The medium is shared between the stations. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled.

a) PURE ALOHA

- The original ALOHA protocol is called pure ALOHA.
- Each station sends a frame whenever it has a frame to send.
- Since there is only one channel to share, there is possibility of collision between frames from different stations.

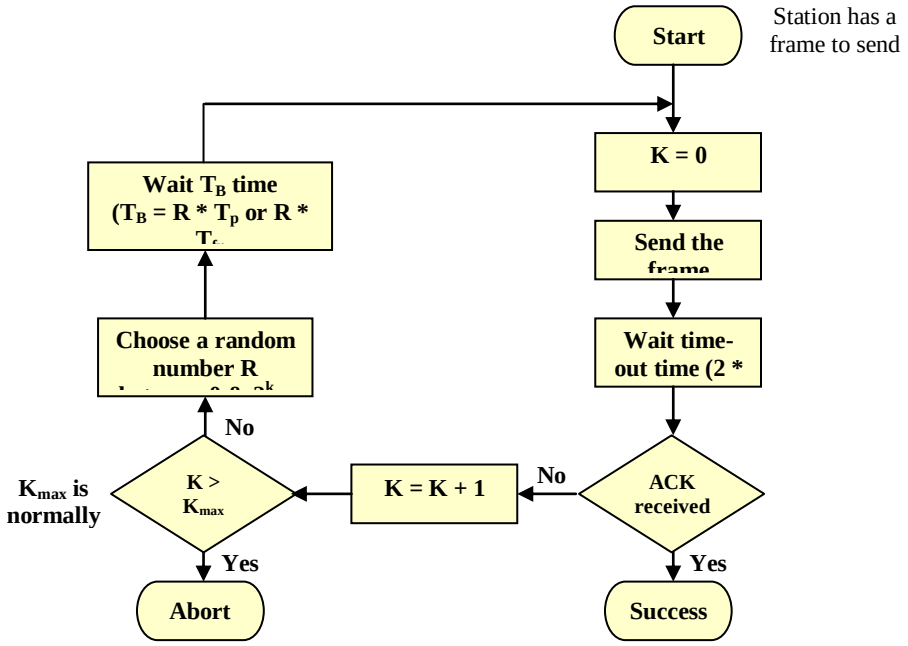
Frames in a pure ALOHA network



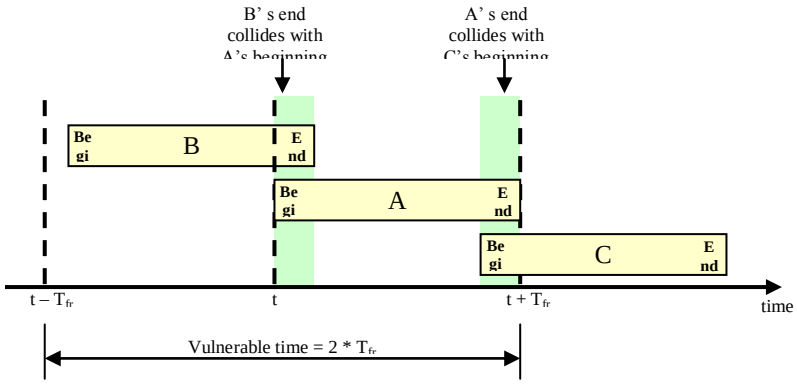
- Even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed.

- Retransmission of frames are required for the destroyed frames.
- The *pure ALOHA* protocol relies on acknowledgements from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgement.
- If the acknowledgement does not arrive after a time-out period, the station assumes that the ACK has been destroyed and resends a frame.
- A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again.
- *Pure ALOHA* dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. This time is called back-off time TB. This randomness will help avoid more collisions.
- To prevent congesting the channel with retransmitted frames, *pure ALOHA* dictates that after a maximum number of retransmission attempts Kmax, a station must give up and try later.

Procedure for pure ALOHA protocol



Vulnerable time for pure ALOHA



Throughput

G = average number of frames generated by the system during one frame transmission time.

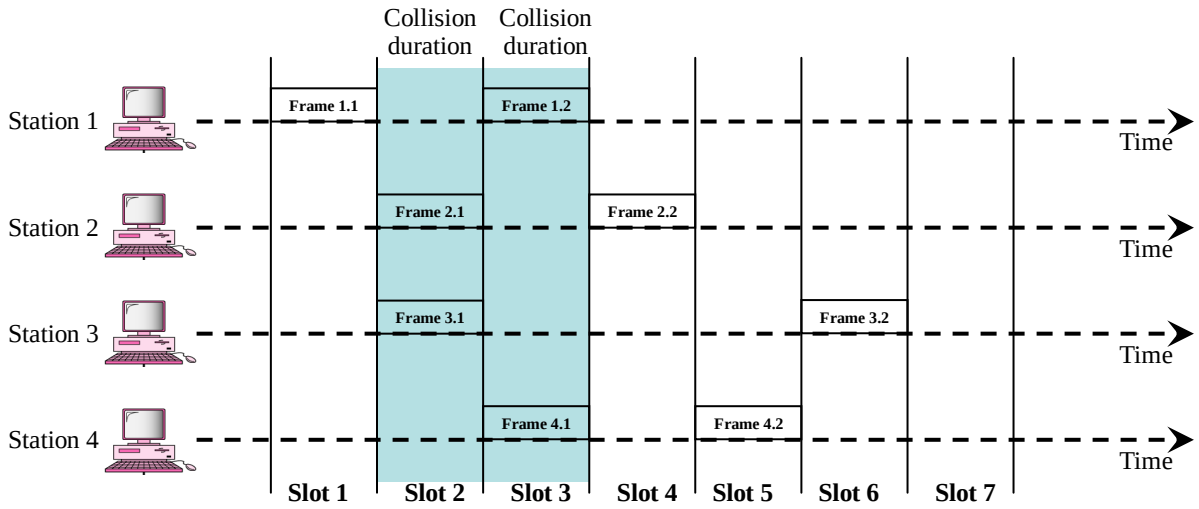
Successful transmissions for pure ALOHA is

$$S = G * e^{-2G}$$

The maximum throughput, Smax is for G = ½. i.e., Smax = 0.184 = 18.4%.

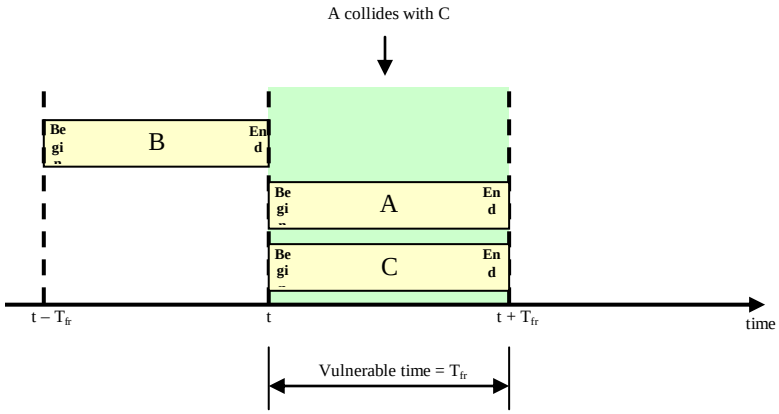
b) SLOTTED ALOHA

Frames in a slotted ALOHA network



Slotted ALOHA Vulnerable time = T_{fr}

Vulnerable time for slotted ALOHA



Throughput

G = average number of frames generated by the system during one frame transmission time.

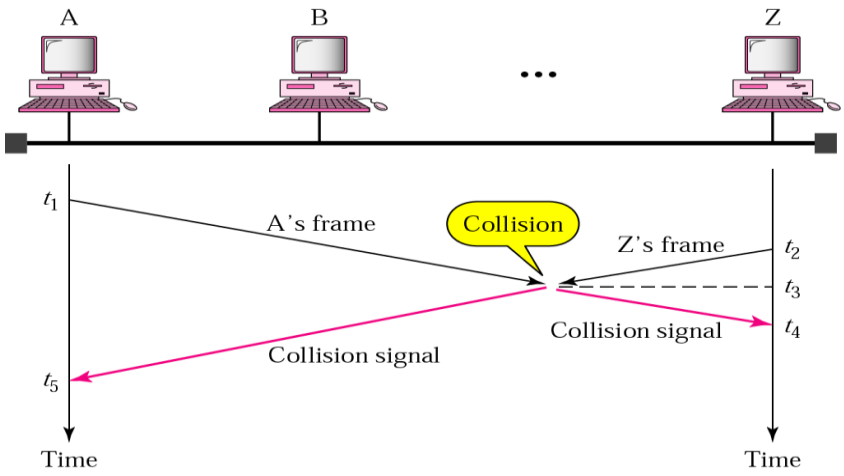
Successful transmissions for pure ALOHA is

$$S = G * e^{-G}$$

The maximum throughput, Smax is for G =1 i.e., Smax = 0.368 = 36.8%.

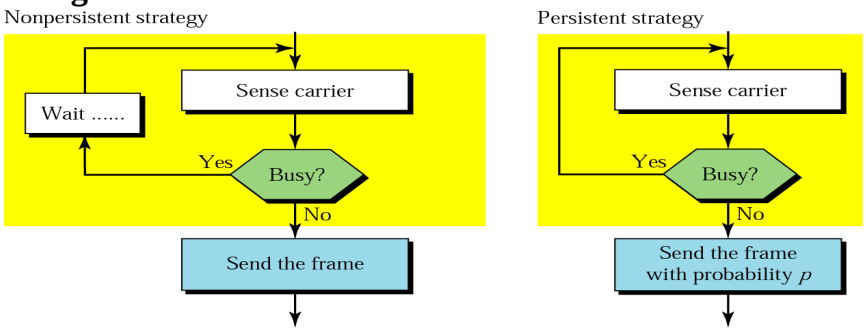
C) Carrier Sense Multiple Access (CSMA)

The chance of collision can be reduced if a station senses the medium before trying to use it. CSMA requires that each station first listen to the medium before sending. CSMA is based on the principle " sense before transmit" or "listen before talk".



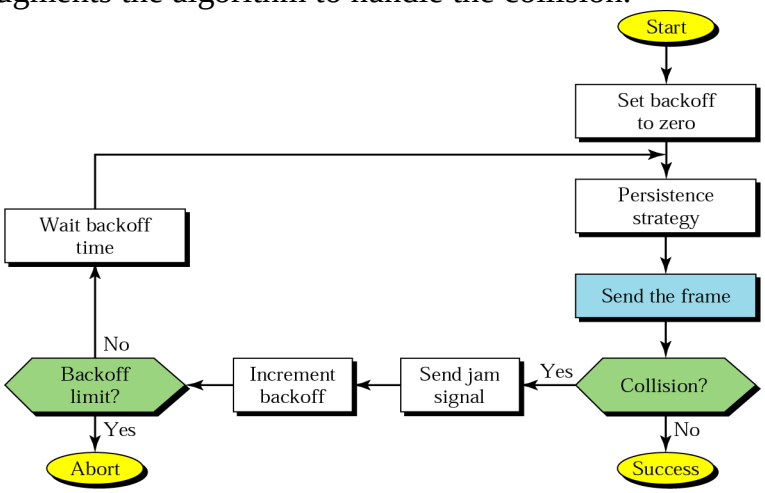
CSMA can reduce the possibility of collision, but it can't eliminate it. The reason for this is shown in the above figure, a space and time model of a CSMA network. Stations are connected to a shared channel. The possibility of collision still exists because of the propagation delay.

Persistence strategies



C) Carrier Sense Multiple Access with collision detection (CSMA/CD)

The CSMA method does not specify the procedure following the collision. CSMA with collision detection augments the algorithm to handle the collision.

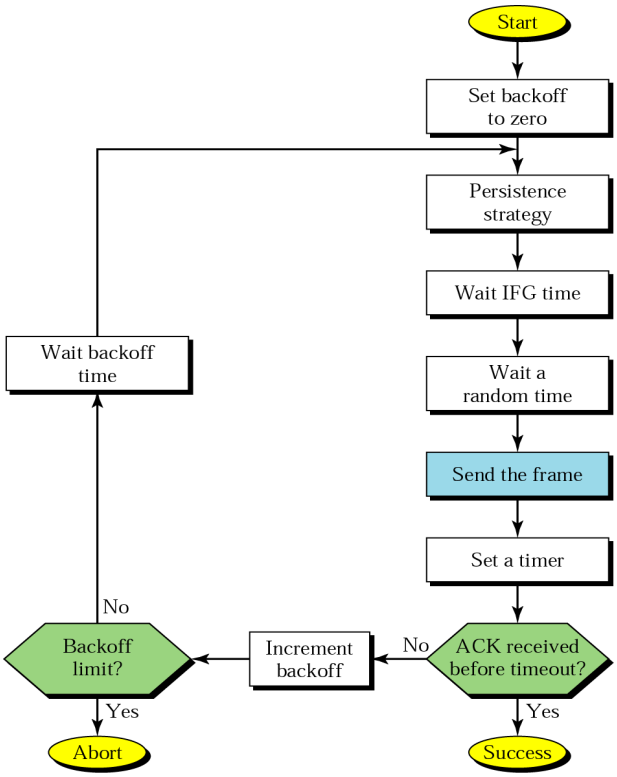


In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

D) Carrier Sense Multiple Access with collision avoidance (CSMA/CA)

The basic idea behind CSMA/CD is that a station needs to be able to receive while transmitting to detect a collision. When there is no collision, the station receives one signal; its own signal. When there is a collision, the station receives two signals; its own signal and the signal transmitted by the second station.

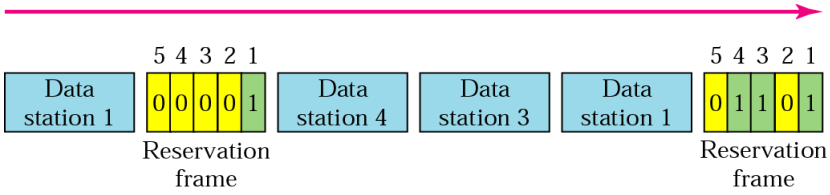
Collisions are avoided by deferring transmissions even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the interframe space or IFS. Even though the channel may appear idle when it is sensed, a distant station's signal has not yet reached this station. The IFS time allows the front of the transmitted signal by the distant station to reach this station.



3.3 CONTROLLED ACCESS

In controlled access, the stations consult one another to find which station has the right to send. A station can't send unless it has been authorized by other stations.

a) Reservation:

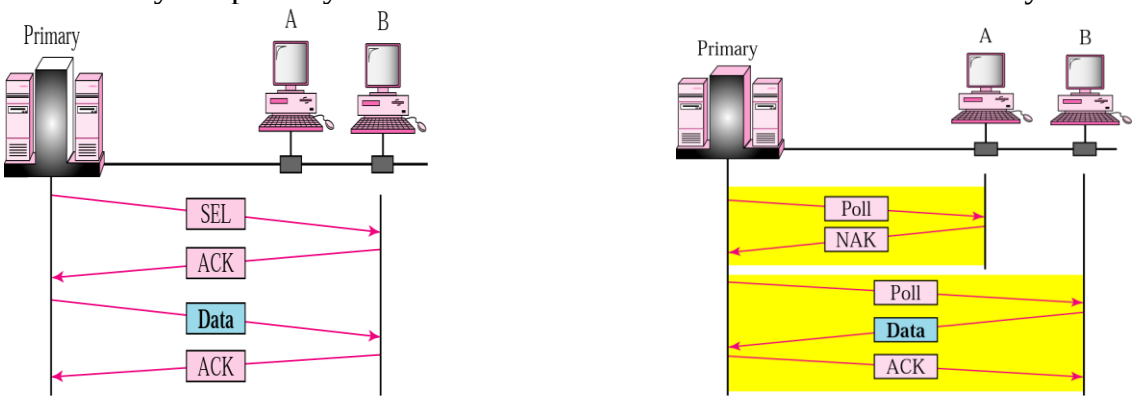


In the reservation method, a station needs to make a reservation before sending the data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in the interval.

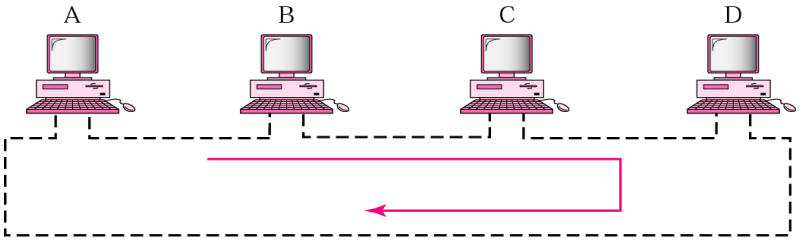
b) Polling:

Polling works with topologies in which one device is designated as a primary station and the other devices are secondary stations. All data exchanges must be made through the primary device even when the ultimate destination is a secondary device.

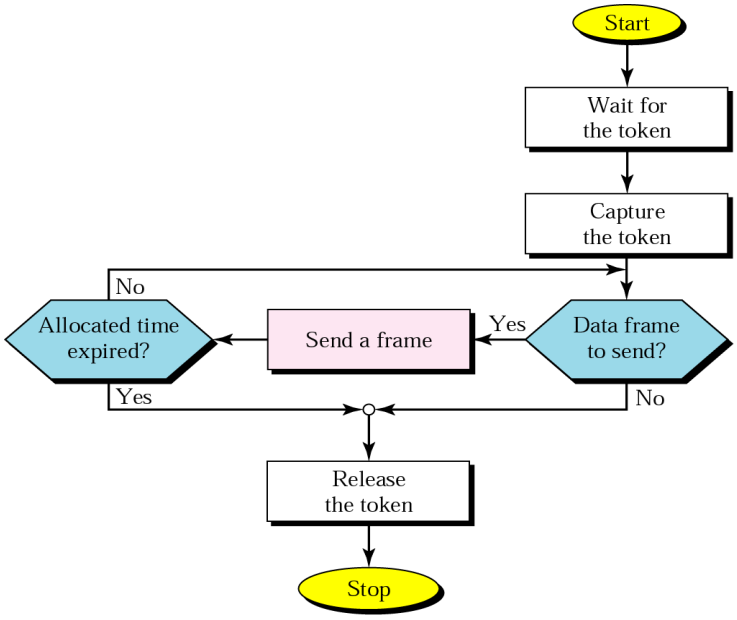
The select function is used whenever the primary device has something to send. The poll function is used by the primary device to solicit transmissions from the secondary device.



c) Token Passing:



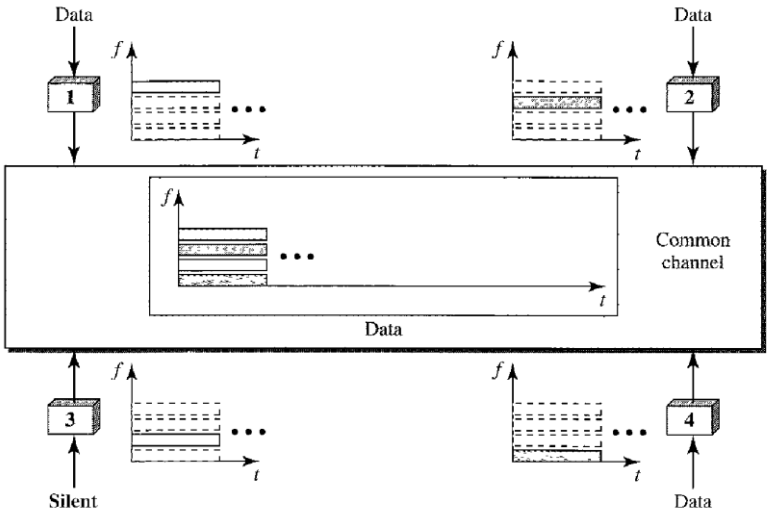
In the token-passing method, the stations in a network are organized in a logical ring. Token management is needed for this access method. Stations must be limited in the time they can have possession of the token. The token must be monitored to ensure it has not been lost or destroyed.



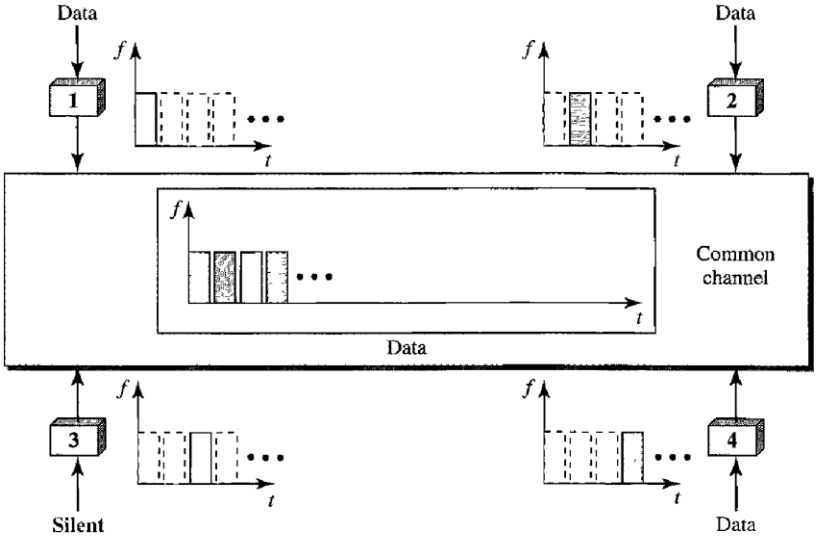
3.4 CHANNELIZATION

a) Frequency Division Multiple Access (FDMA)

In FDMA, the bandwidth is divided into channels. Each station is allocated a band to send its data. Each station also uses a band pass filter to confine the transmitter frequencies. To prevent station interferences, the allocated bands are separated from one another by small guard bands.



b) Time Division Multiple Access (TDMA)



In TDMA, the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot.

c) Code Division Multiple Access (CDMA)

CDMA differs from FDMA because only one channel occupies the entire bandwidth of the link and it differs from TDMA because all stations can send data simultaneously.

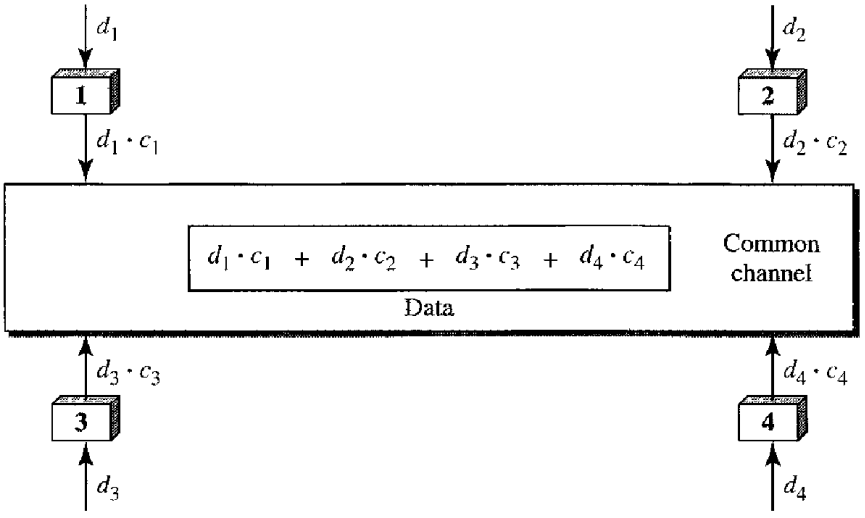
Analogy

Let us first give an analogy. CDMA simply means communication with different codes. For example, in a large room with many people, two people can talk in English if nobody else understands English. Another two people can talk in Chinese if they are the only ones who understand Chinese, and so on. In other words, the common channel, the space of the room in this case, can easily allow communication between several couples, but in different languages (codes).

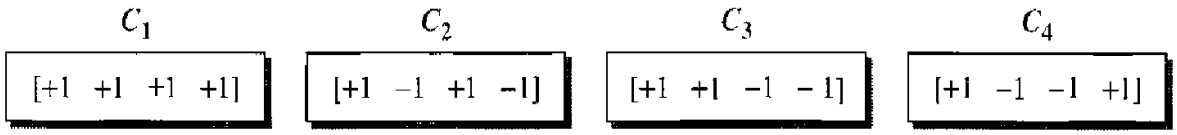
Idea

Let us assume we have four stations 1, 2, 3, and 4 connected to the same channel. The data from station 1 are d_1, from station 2 are d_2, and so on. The code assigned to the first station is c_1, to the second is c_2, and so on. We assume that the assigned codes have two properties.

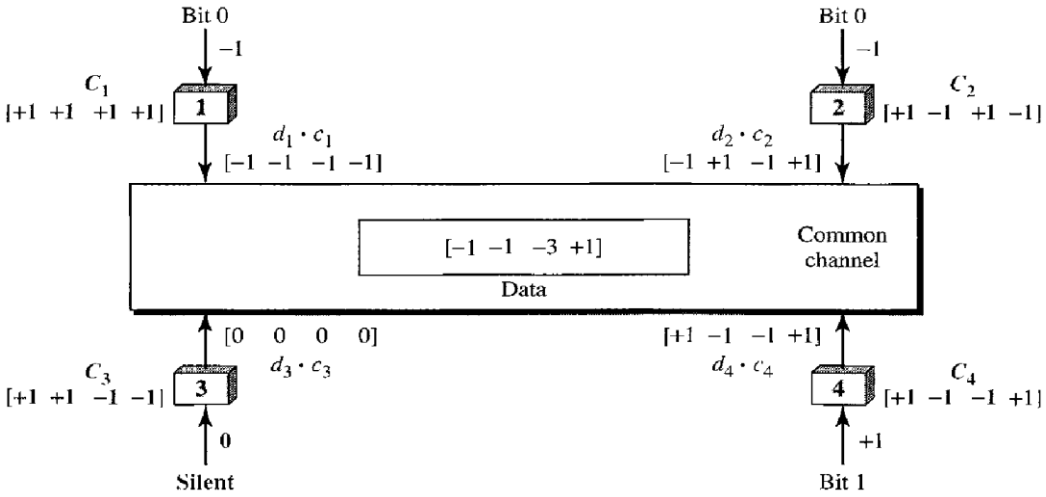
1. If we multiply each code by another, we get 0.
2. If we multiply each code by itself, we get 4 (the number of stations).



Chip sequences



Sharing channel in CDMA



1)
Find the chips for a network with

a. Two stations
b. Four stations

Solution

We can use the rows of W_2 and W_4 in Figure 12.29:

a. For a two-station network, we have [+1 +1] and [+1 −1].
b. For a four-station network we have [+1 +1 +1 +1], [+1 −1 +1 −1], [+1 +1 −1 −1], and [+1 −1 −1 +1].

2)
What is the number of sequences if we have 90 stations in our network?

Solution

The number of sequences needs to be 2^m. We need to choose $m = 7$ and $N = 2^7$ or 128. We can then use 90 of the sequences as the chips.

3)

Prove that a receiving station can get the data sent by a specific sender if it multiplies the entire data on the channel by the sender's chip code and then divides it by the number of stations.

Solution

Let us prove this for the first station, using our previous four-station example. We can say that the data on the channel $D = (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4)$. The receiver which wants to get the data sent by station 1 multiplies these data by c_1.

$$\begin{aligned} D \cdot c_1 &= (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1 \\ &= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 \\ &= d_1 \times N + d_2 \times 0 + d_3 \times 0 + d_4 \times 0 \\ &= d_1 \times N \end{aligned}$$

When we divide the result by N, we get d_1.

Recommended questions:

1. Define controlled access and list three protocols in this category.
2. Compare and contrast a random access protocol with a channelization protocol.
3. Define channelization and explain the three protocols.
4. List and explain the three categories of multiple access protocols.
5. Explain why collision is an issue in a random access protocol but not in controlled access or channelizing protocols.
6. Compare and contrast a random access protocol with a controlled access protocol.

Unit 4: **Hrs: 07**

Syllabus:

IEEE standards, standard Ethernet, changes in the standards, Fast Ethernet, Gigabit Ethernet, Wireless LAN IEEE 802.11

Recommended readings:

Text Book: Data Communication & Networking, B Forouzan, 4e, TMH

Chapter 13 – page 395 to 417
Chapter 14 – page 421 to 434