

## Unit-II

### **FLOW CONTROL**

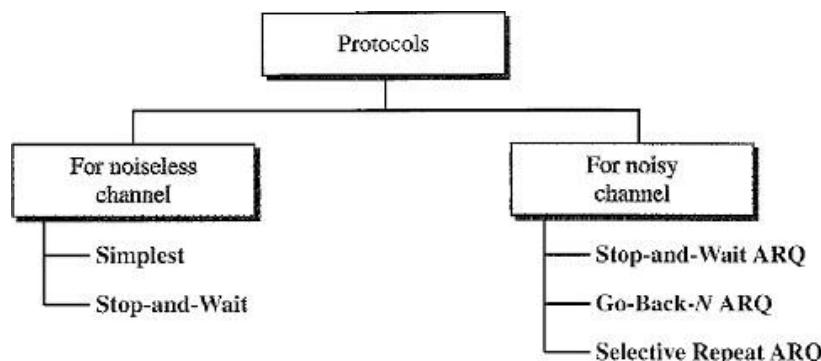
- Flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgment from the receiver.
- The flow of data must not be allowed to overwhelm the receiver.
- Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data.
- The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily.
- Incoming data must be checked and processed before they can be used.
- The rate of receiving data is slower than the rate of transmission of data.
- For this reason, each receiving device has a block of memory, called a *buffer*, reserved for storing incoming data until they are processed.
- If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

### **ERROR CONTROL**

- Error control is both error detection and error correction.
- It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the **Retransmission** of those frames by the sender. At any time an error is detected in an exchange, specified frames are retransmitted. This process is called **Automatic Repeat Request (ARQ)**.

### **PROTOCOLS**

The protocols are normally implemented in software by using the common programming languages.



### **Noiseless Channels**

Noiseless Channel is an ideal channel in which no frames are lost, duplicated, or corrupted. There are two protocols for this type of channel:

1. Simplest Protocol
2. Stop and Wait Protocol

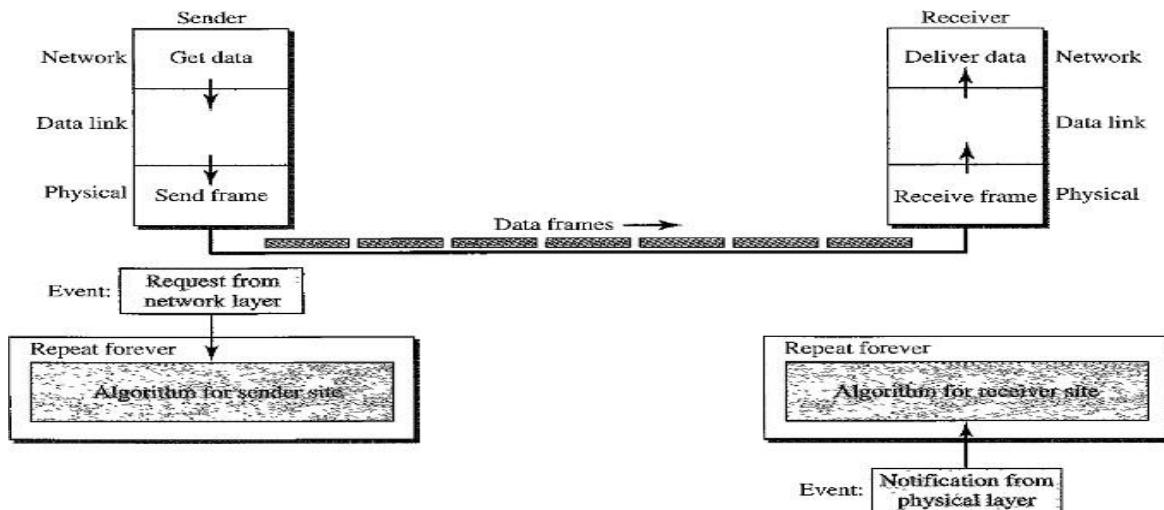
**Note:** In real world there are no Noiseless channels.

### **Simplest Protocol**

- Simplest Protocol has no flow control and no error control mechanism.
- It is a unidirectional protocol in which data frames are traveling in only one direction-

from the sender to receiver.

- It is applicable when the receiver can never be overwhelmed with incoming frames that are sent by the sender.
- The receiver can receive any frame with a very short span of processing time.
- The data link layer of the receiver immediately removes the header from the frame and delivers the data packet to its network layer.



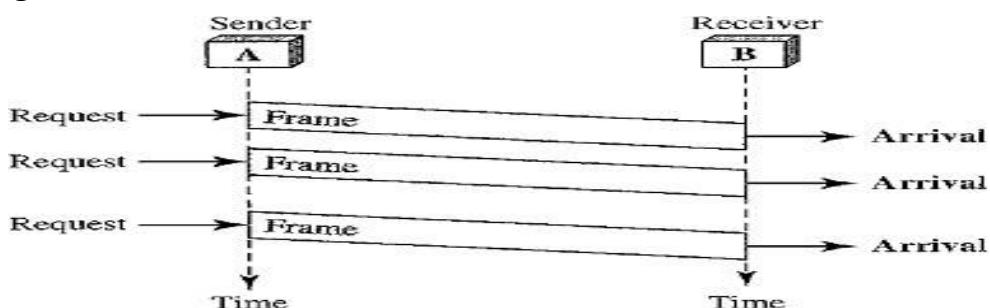
### Procedure:

- The data link layer at the **Sender site** gets data from its network layer, makes a frame out of the data, and sends it.
- The data link layer at the **Receiver site** receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer.
- The data link layers of the sender and receiver provide transmission services for their network layers.
- The data link layers use the services provided by their physical layers (such as signaling, multiplexing, and so on) for the physical transmission of bits.
- There is no need for flow control in Simplest Protocol scheme.

Simplest protocols are implemented as an Event driven Procedure.

- The procedure at the sender site is constantly running; there is no action until there is a request from the network layer.
- The procedure at the receiver site is also constantly running, but there is no action until notification from the physical layer arrives.
- Both procedures are constantly running because they do not know when the corresponding events will occur.

### Flow diagram



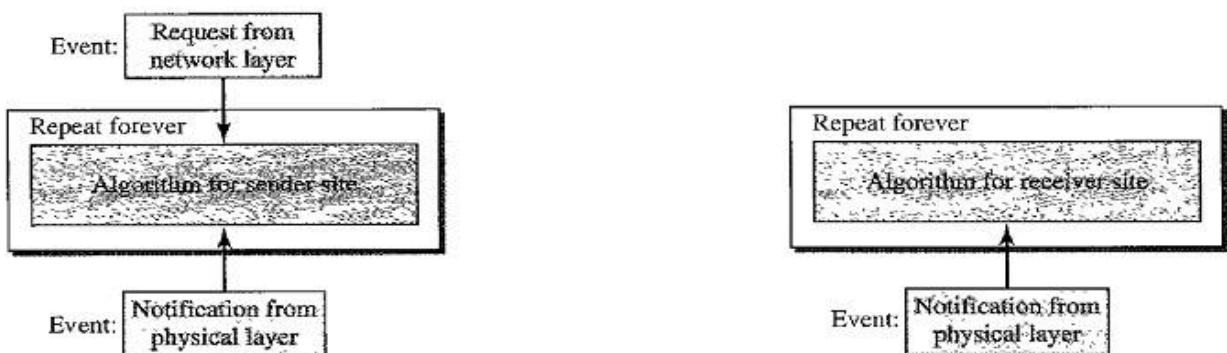
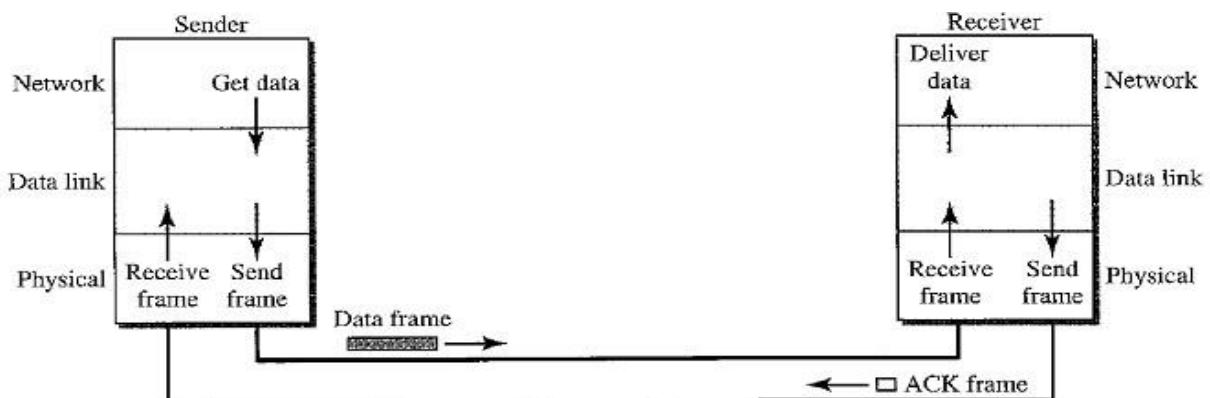
- The sender sends a sequence of frames without even thinking about the receiver.
- To send three frames, three events occur at the sender site and three events at the receiver site.

### **Stop-and-Wait Protocol**

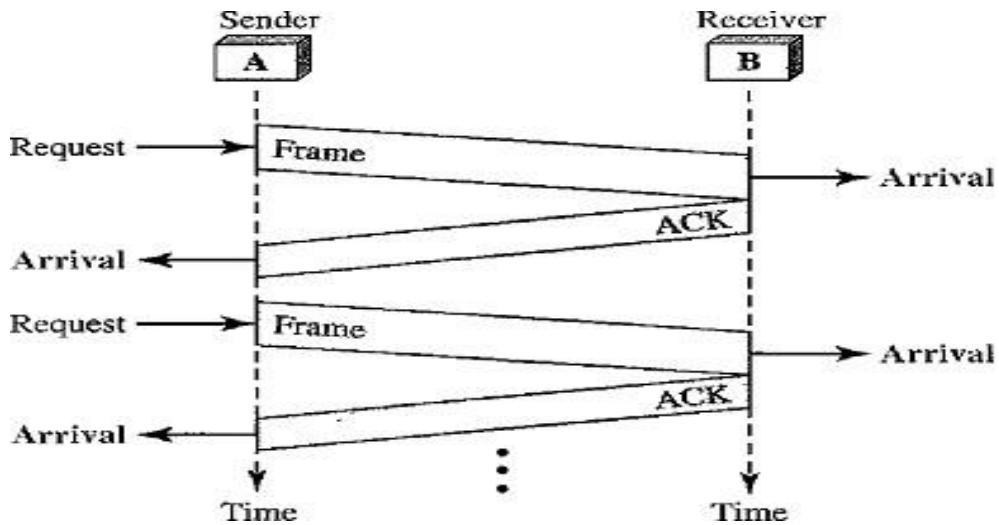
- Stop-and-Wait Protocol implements **Flow Control Mechanism**.
- If there are multiple senders sending data to one receiver, the receiver receiving data rate is slower than the sender's data rate. Then receiver is overwhelmed by the sender's data.
- To prevent the receiver from becoming overwhelmed with frames, the receiver needs to tell the sender to slow down. (i.e.) There must be feedback from the receiver to the sender.
- In Stop-and-Wait Protocol the sender sends one frame, stops until it receives confirmation (ACK's) from the receiver and then sends the next frame.

### **Design Procedure:**

At any time, there is either one data frame on the forward channel or one ACK frame on the reverse channel. Therefore need a half-duplex link.



### **Flow Diagram**



The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame.

#### Sender site:

Here two events can occur:

1. **Request Event:** At the sender site the request event comes from the Network layer to Data link layer, whether Sender wants to receive the packet from the network layer.
2. **Arrival Notification:** At the sender site the arrival notification event comes from the physical layer to Data link layer.

After a frame is sent, the data link layer must ignore another network layer request until that frame is acknowledged.

- If we are designing error free frames then two arrival events cannot happen one after another.
- The requests from the network layer may happen one after another without an arrival event in between. We need somehow to prevent the immediate sending of the data frame.
- canSend( )** method is used for this purpose, it may be either true or false.
- When a frame is sent, the variable is set to false to indicate that a new network request cannot be sent until **canSend** is true.
- When an ACK is received, **canSend** is set to true to allow the sending of the next frame.

#### Receiver-site

There is only one event can happen at the receiver site is the **Arrival Notification** from the Physical layer to indicate that a frame is arrived from the sender site.

After the data frame arrives, the receiver sends an ACK frame to acknowledge the receipt and allow the sender to send the next frame.

#### Disadvantages of Stop and Wait Protocol

1. If the receiver does not respond when there is an error then sender doesn't know which frame has to be resending.
2. In Stop and Wait protocol the sender does not send next frame until it receives ACK from receiver. If the ACK has lost then sender does not know when to send the frame.

#### **NOISY CHANNELS**

In Noisy Channels we need to implement both Flow control and Error Control Mechanisms

in the Protocols.

### **Stop-and-Wait Automatic Repeat Request (ARQ)**

This Protocol uses Acknowledgements and Sequence Numbers to implement Flow and Error Control mechanisms. The corrupted and lost frames need to be resent in this protocol.

There are 2 Questions arises:

#### **1. If the receiver does not respond when there is an error, how the sender knows which frame to resend?**

**Solution:** The sender keeps a copy of the sent frame. At the same time, it starts a timer.

If the timer expires and there is no ACK for the sent frame then “**3 actions**” to be performed

- i. The frame is resent
- ii. The copy is held
- iii. The timer is restarted.

Note: Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.

#### **2. What if an ACK frame is also lost?**

**Solution:**

Since an ACK frame can also be corrupted and lost, it needs a **Sequence Number** and Redundancy Bits.

The ACK frame for this protocol has a sequence number field.

In this protocol, the sender simply discards a corrupted ACK frame or ignores an out-of-order one.

### **Sequence Numbers**

- Sequence Number is the number that is given to the data frame.
- A field is added to the data frame to hold the sequence number of that frame.
- The sequence numbers are based on modulo-2 arithmetic.

**Range:** Consider the field is m bits long, the range of sequence numbers are from 0 to  $2^m - 1$ , and then the same number are repeated.

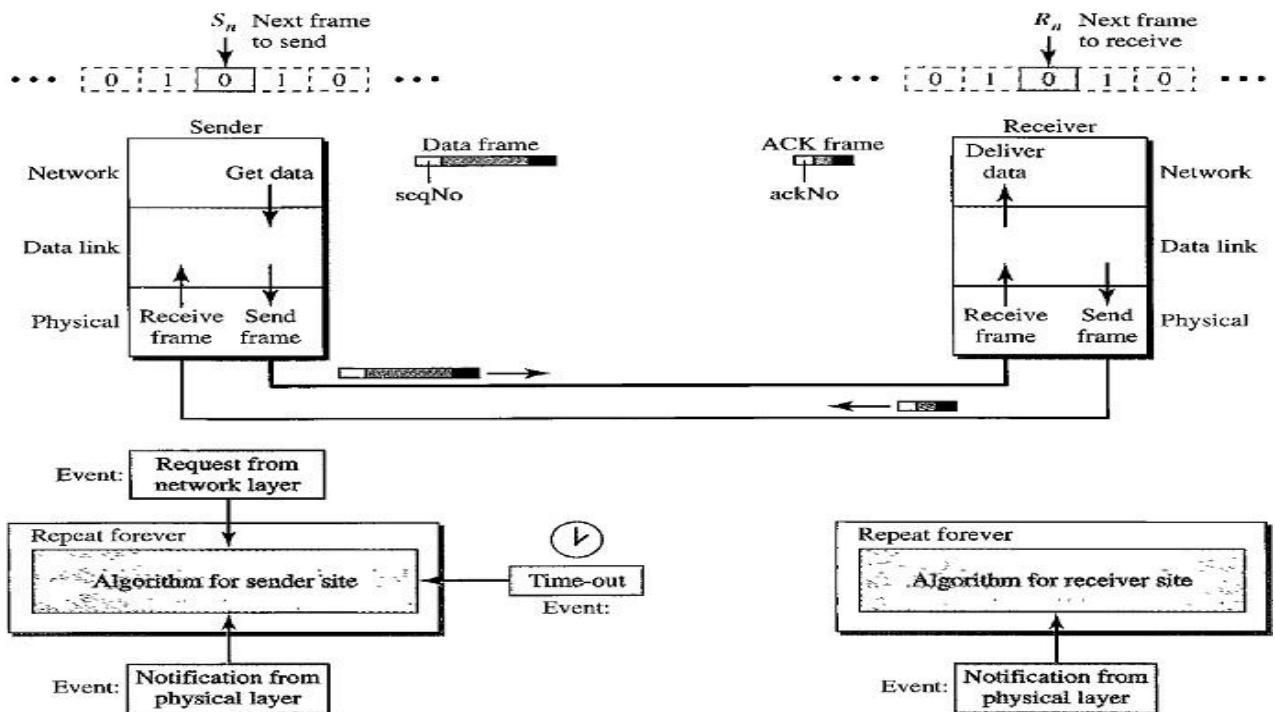
### **Acknowledgment Numbers**

The acknowledgment numbers always announce the sequence number of the next frame expected by the receiver.

Example:

1. If frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1 (meaning frame 1 is expected next).
2. If frame 1 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 0 (meaning frame 0 is expected).

## Design



## Procedure:

- The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame.
- The design contains:
  - **seqNo:** Sequence Number
  - **ackNo:** Acknowledgement Number
  - **$S_n$ :** Sender Control variable- that holds the sequence number for the next frame to be sent (0 or 1).
  - **$R_n$ :** Receiver Control Variable- that holds the number of the next frame expected.
- When a frame is sent, the value of  $S_n$  is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa.
- When a frame is received, the value of  $R_n$  is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa.
- $S_n$  variable points to the slot that matches the sequence number of the frame that has been sent, but not acknowledged.
- Sequence numbers are modulo-2, the possible numbers are 0,1,0,1,0,1,.....and so on.
- $R_n$  points to the slot that matches the sequence number of the expected frame.
- Three events can happen at the sender site; one event can happen at the receiver site.

**There are 3 Events can happen at Sender site.**

1. RequestToSend (which comes from network layer to data link layer)
2. ArrivalNotification (which comes from physical layer to data link layer)
3. TimeOut

### **Event 1: Request to send**

- Event RequestToSend is used to send the frame to receiver.
- Before the frame is sent, it is stored in the buffer. We need at least one buffer to hold this

frame until sender make sure that it is received safe and sound.

- The copy is used for resending a corrupt or lost frame.
- Sender has to prevent the network layer from making a request before the previous frame is received safe and sound.

### **Event 2: Arrival Notification**

- If the frame is not corrupted then three actions will be done:
  1. The ackNo of the ACK frame matches the sequence number of the next frame to send.
  2. Timer is stopped
  3. Purge the copy of the data frame that was saved.
- If the frame is corrupted then it ignore arrival notification event and wait for the next event to happen.

### **Event 3: Time Out**

- After each frame is sent, a timer is started.
- When the timer expires the frame is resent and the timer is restarted.

### **Receiver-site:**

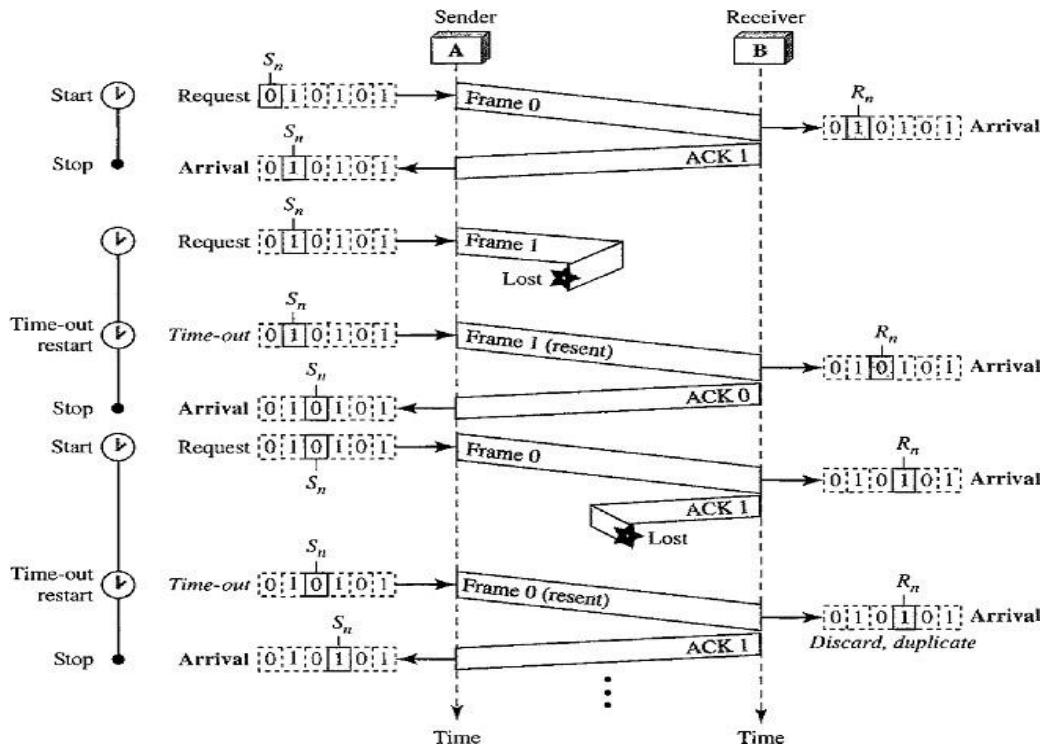
Only **One Event** happens at the receiver site.

- First, all arrived data frames that are corrupted are ignored.
- If the seqNo of the frame is the one that is expected ( $R_n$ ), the frame is accepted, the data are delivered to the network layer, and the value of  $R_n$  is incremented.

### **Note:**

- Even if the sequence number of the data frame does not match the next frame expected, an ACK is sent to the sender.
- This ACK reconfirms the previous ACK instead of confirming the frame received.
- This is done because the receiver assumes that the previous ACK might have been lost; the receiver is sending a duplicate frame.
- The resent ACK may solve the problem before the time-out does it.

## Flow diagram



- Frame 0 is sent and acknowledged.
- Frame 1 is lost and resent after the time-out.
- The resent frame 1 is acknowledged and the timer stops.
- Frame 0 is sent and acknowledged, but the acknowledgment is lost.
- The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.

## Pipelining

- In networking a task is often begun before the previous task has ended. This is known as pipelining.
- There is no pipelining in Stop-and-Wait ARQ because sender need to wait for a frame to reach the destination and be acknowledged before the next frame can be sent.
- Go-Back-N ARQ and Selective Repeat ARQ use Pipelining, because several frames can be sent before sender receives the ACK about previous frame.
- Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay product.

## Go-Back-N Automatic Repeat Request (ARQ)

In this protocol

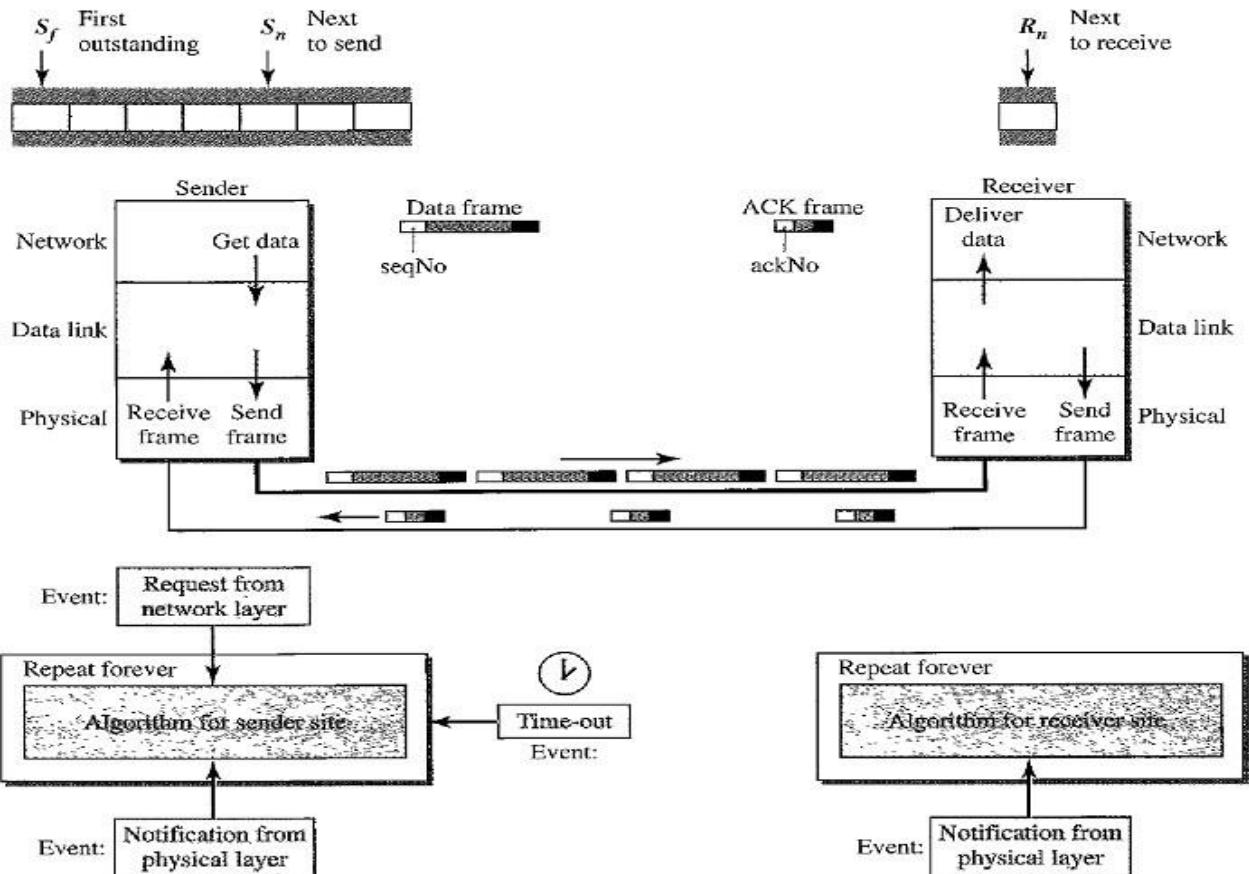
- Sender can send several frames before receiving acknowledgments.
- Sender keep a copy of these frames until the acknowledgments arrive.

## Design of Go-Back-N ARQ

The idea is similar to Stop-and-Wait ARQ, multiple frames can be in transit in the forward direction, and multiple acknowledgments in the reverse direction.

The difference is that the send window allows us to have as many frames in transition as

there are slots in the send window.



This protocol uses the following concepts:

1. Sequence Numbers
2. Sliding Window
3. Timers
4. Acknowledgement
5. Retransmission (Resending a Frame)

### Sequence Numbers

- It is a number given to each frame included in the header.
- The header of the frame allows  $m$  bits for the sequence number, the sequence numbers range from 0 to  $2^m-1$ .
- The sequence numbers are repeated after the number  $2^m-1$  (i.e) the sequence numbers are modulo- $2^m$

Example:

If  $m=4$ , the sequence numbers are 0,1,2,3,4,5.....14,15,0,1,2,3,4,5,6,.....

### Sliding Window

Sliding Window is an abstract concept that defines the sender and receiver needs to deal with only part of the possible sequence numbers.

There are 2 types of windows are used:

- i. Send Window
- ii. Receive Window

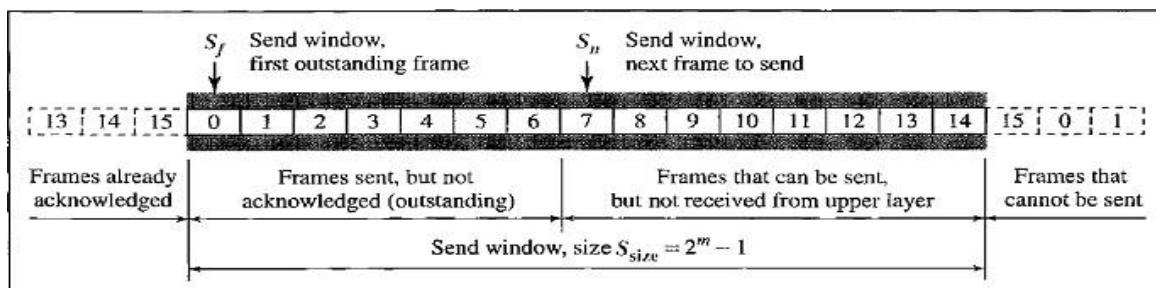
The range that is the concern of sender is called the send sliding window or Send Window  
The range that is the concern of receiver is called receive sliding window or receive window.

### Send Window

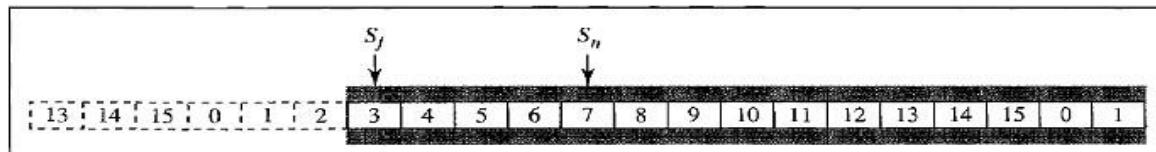
- The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit.
- In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent.
- The maximum size of the send window is  $2^m - 1$ .

Example: Let us take  $m=4$ . Size of window=15.

Consider the below figure:



a. Send window before sliding



b. Send window after sliding

The window at any time divides the possible sequence numbers into four regions.

1. The first region (the left side of the window) defines the sequence numbers belonging to frames that are already acknowledged. The sender don't need to keep copies of the frames.
2. The second region defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. These frames are called outstanding frames.
3. The third range defines the range of sequence numbers for frames that can be sent. The corresponding data packets have not yet been received from the network layer.
4. The fourth region defines sequence numbers that cannot be used until the window slides.

The window uses three variables define its size and location at any time.

- i.  $S_f$  defines the sequence number of the first (oldest) outstanding frame.
- ii.  $S_n$  holds the sequence number that will be assigned to the next frame to be sent.
- iii.  $S_{size}$  defines the size of the window, which is fixed in our protocol.

The acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame.

In the above figure frames 0, 1, and 2 are acknowledged, so the window has slid to the right three slots.

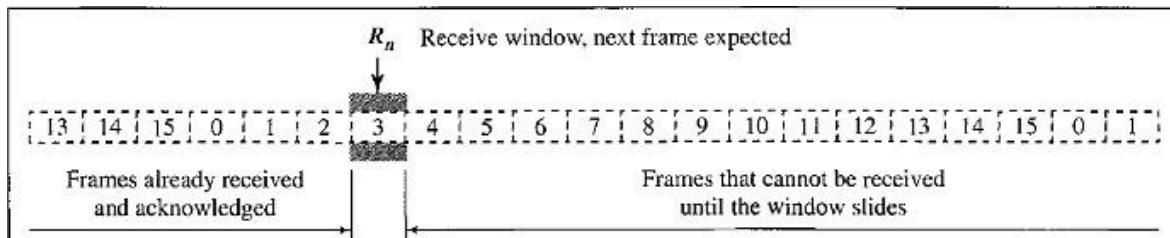
### Receive Window

- The receive window makes sure that the correct data frames are received and that the correct acknowledgments are sent.
- The size of the receive window is always 1.
- The receiver is always looking for the arrival of a specific frame.
- Any frame arriving out of order is discarded and needs to be resent.

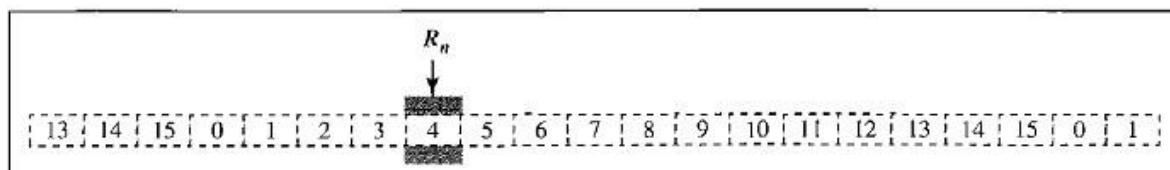
There is only one variable required, that is  $R_n$

$R_n$ : defines next frame expected, belongs to Receive Window.

- The sequence numbers to the left of the window belong to the frames already received and acknowledged.
  - The sequence numbers to the right of this window define the frames that cannot be received.
  - Any received frame with a sequence number in these two regions is discarded.
  - Only a frame with a sequence number matching the value of  $R_n$  is accepted and acknowledged. (i.e.  $S_n = R_n$ )
- The receive window slides only one slot at a time, when the



a. Receive window



b. Window after sliding

correct frame is received the window slides.

## Timers

The timer for the first outstanding frame always expires first. We send all outstanding frames when this timer expires.

## Acknowledgment

- The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order.
- If a frame is damaged or frame is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting.
- The silence of the receiver causes the timer of the unacknowledged frame at the sender site to expire. This causes the sender to go back and resend all frames, beginning with the one with the expired timer.
- The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

## Retransmission (Resending a Frame)

When the timer expires, the sender resends all outstanding frames.

Example:

- Suppose the sender has already sent frame 6, but the timer for frame 3 expires.

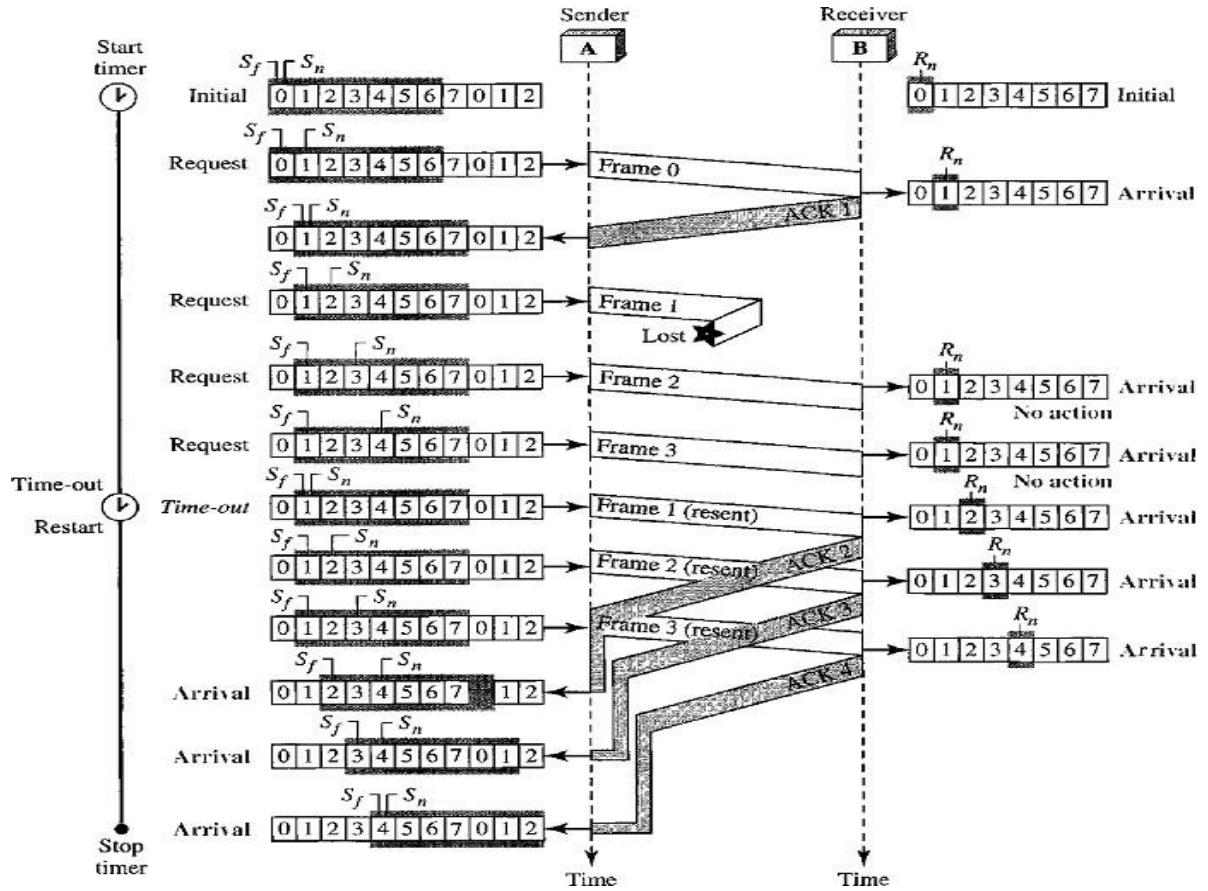
- This means that frame 3 has not been acknowledged.
- The sender goes back and sends frames 3, 4, 5, and 6 again.
- That is why the protocol is called **Go-Back-N ARQ**.

### **Flow Diagram**

The below shows what happens when a frame is lost.

- Frames 0, 1, 2, and 3 are sent.
- Frame 0 is acknowledged but **Frame 1** is lost.
- The receiver receives frames 2 and 3, but they are discarded because they are received out of order (frame 1 is expected).
- The sender receives no acknowledgment about frames 1, 2, or 3.
- Its timer finally expires. The sender sends all outstanding frames (1, 2, and 3) because it does not know whether the frame is lost or corrupted.
- Note that the resending of frames 1, 2, and 3 is the response to one single event.
- When the sender is responding to this event, it cannot accept the triggering of other events.
- This means that when ACK 2 arrives, the sender is still busy with sending frame 3.
- The physical layer must wait until this event is completed and the data link layer goes back to its sleeping state.
- Vertical line in the figure is to indicate the delay.

Note that before the second timer expires, all outstanding frames have been sent and the timer is stopped.



### Disadvantage with Go-Back-N ARQ

Go-Back-N ARQ protocol is very inefficient for a noisy link.

- Go-Back-N ARQ simplifies the process at the receiver site.
- The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames and they are simply discarded.
- In a noisy link a frame has a higher probability of damage; so it leads to the resending of multiple frames.
- This resending uses more bandwidth and slows down the transmission. This is a major disadvantage.

**Solution:** When there is just one frame is damaged, then only the damaged frame is resent, instead of resending from  $N^{\text{th}}$  frame. This will be achieved by using **Selective Repeat ARQ Protocol**.

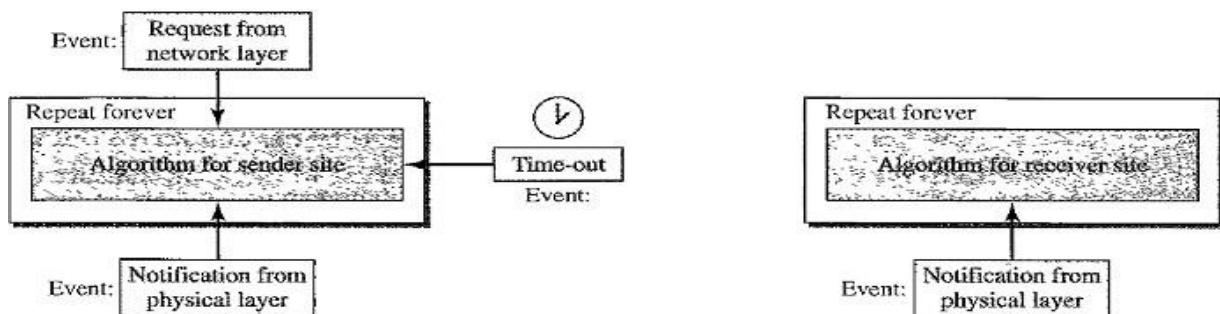
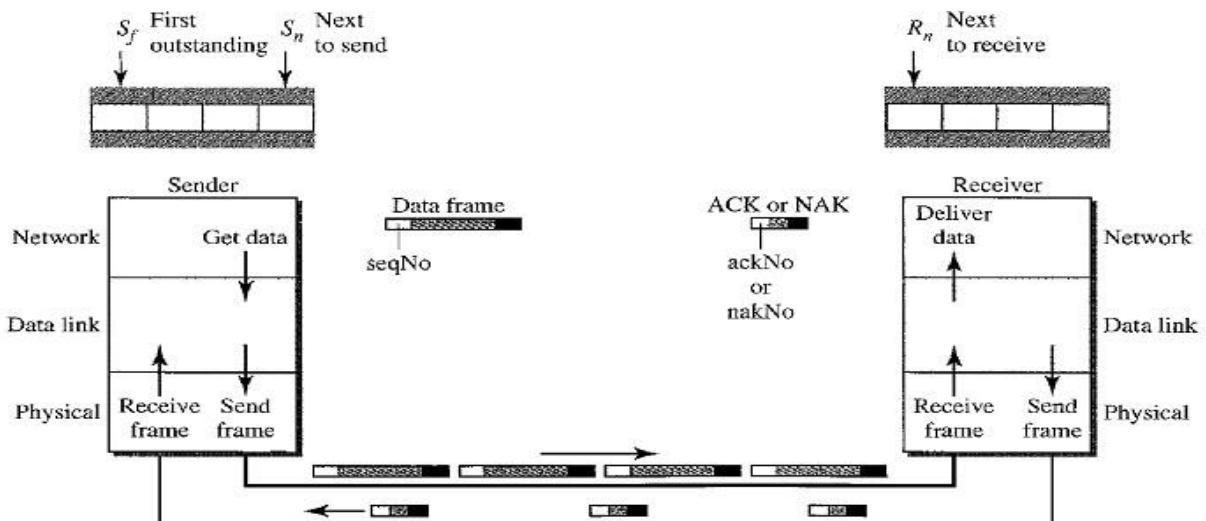
### SELECTIVE REPEAT Automatic Repeat Request

It is more efficient for Noisy links but processing at the receiver is more complex.

#### Design

##### **Window Sizes**

Window size  $2^{m-1}$  means the size of the sender and receiver window is at most one half of  $2^m$  (i.e  $2^m/2$ ).



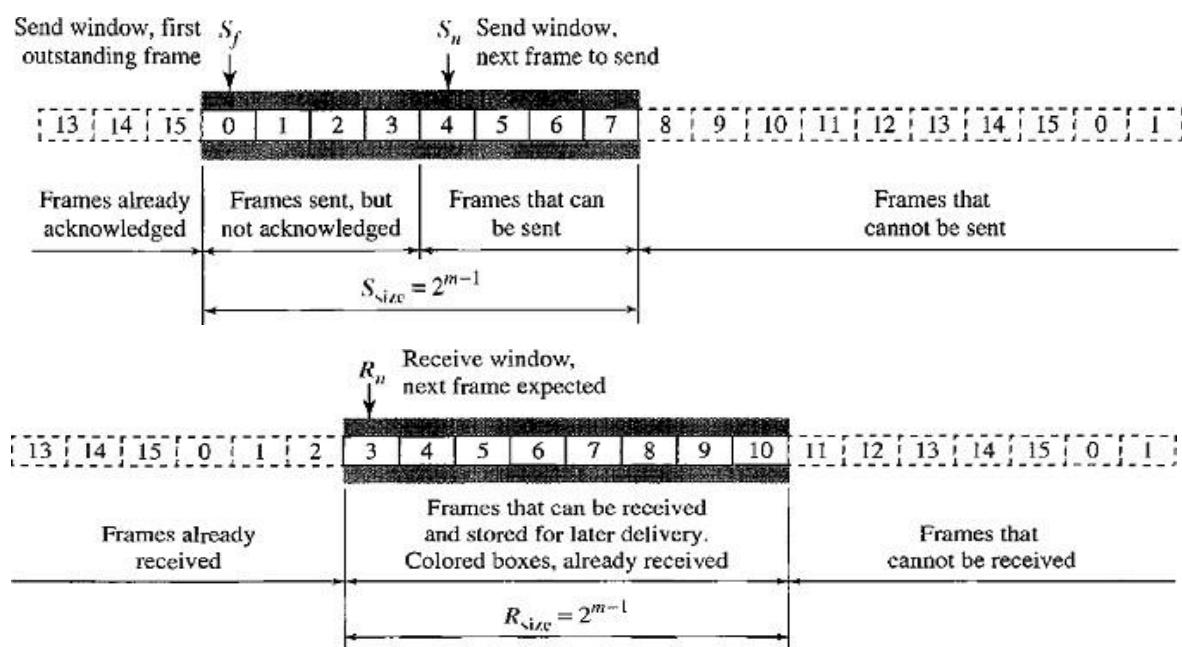
## Windows

It uses two windows:

1. Send Window
2. Receive Window

The size of the send window and receive window is same as  $2^{m-1}$ .

**Example:** If  $m = 4$ , the sequence numbers ranges from 0 to 15, but the size of the window is just 8.



- The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer.
- Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered.
- We need to mention that the receiver never delivers packets out of order to the network layer.
- Those slots inside the receive window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

#### Sender site:

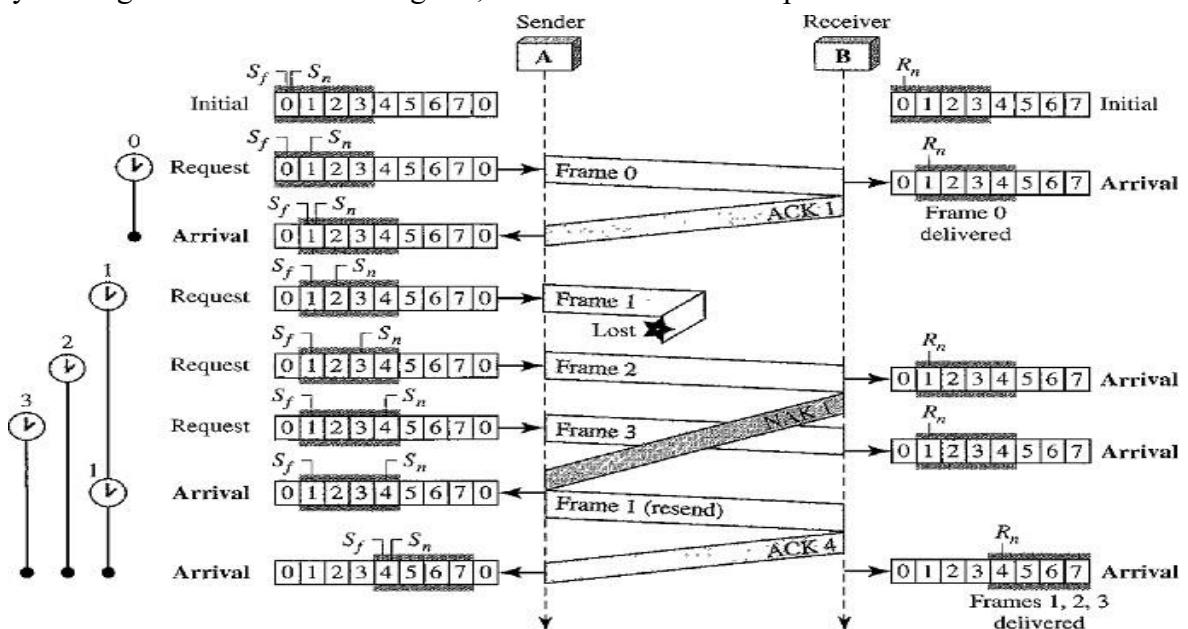
- For every request event a timer is started before sending any frame. The arrival event is more complicated. An ACK or a NAK frame may arrive at the sender site.
- If a valid NAK frame arrives, we just resend the corresponding frame.
- If a valid ACK arrives 3 actions will be done:
  1. Purge the buffers
  2. Stop the corresponding timer
  3. Move the left wall of the window.
- The time-out event is simpler, only the frame which times out is resent.

#### Receiver site:

- Receiver sends ACK and NAK frames to sender.
- If the Receiver receives a corrupted frame and a NAK has not yet been sent, Receiver sends a NAK to tell the sender that we have not received the frame we expected.
- If the frame is not corrupted and the sequence number is in the window, Receiver stores the frame and marks the slot.
- If contiguous frames, starting from  $R_n$  have been marked, Data link layer delivers their data to the network layer and slide the window.

#### Flow Diagram

By looking at the below flow diagram, we can observe below points:



## Timers

- Each frame sent or resent needs a timer and the timers need to be numbered such as 0,1,2,3..etc.
- The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives.
- The timer for frame 1 starts at the second request and restarts when a NAK arrives, and finally stops when the last ACK arrives.
- The other two timers start when the corresponding frames are sent and stop at the last arrival event.

## Receiver Site

- At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer.
- At the second arrival, frame 2 arrives and is stored and marked (colored slot), but it cannot be delivered because frame 1 is missing.
- At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered.
- Only at the last arrival, when finally a copy of frame 1 arrives and frames 1, 2, and 3 be delivered to the network layer.
- There are two conditions for the delivery of frames to the network layer:
  - i. a set of consecutive frames must have arrived.
  - ii. the set starts from the beginning of the window.
- After the first arrival, there was only one frame and it started from the beginning of the window.
- After the last arrival, there are three frames and the first one starts from the beginning of the window.

## Importance of NAK's

- Here a NAK is sent after the second arrival, but not after the third.
- The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames.
- The second NAK would still be NAK1 to inform the sender to resend frame 1 again; this has already been done.
- The first NAK sent is remembered and is not sent again until the frame slides.
- A NAK is sent once for each window position and defines the first slot in the window.

## Importance of ACK's

- There are only two ACKs sent here. The first one acknowledges only the first frame. The second one acknowledges three frames.
- In Selective Repeat, ACKs are sent when data are delivered to the network layer.
- If the data belonging to  $n$  frames are delivered in one shot, only one ACK is sent for all of them.
- In the above figure frame 1,2,3, are sent to Network layer and then ACK4 is sent, to represent that Frame1, Frame2, Frame3 are delivered.

There is a question arises that :

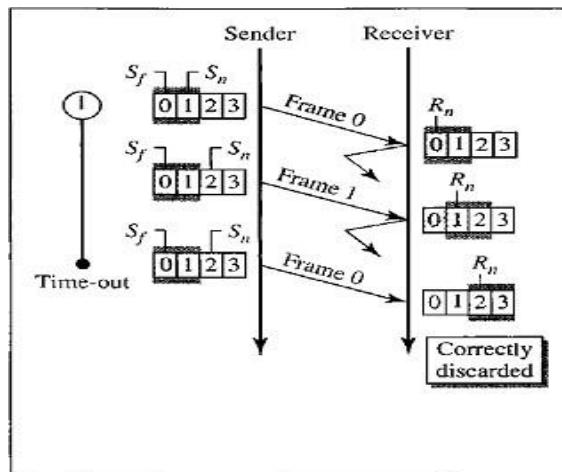
### Why the size of the sender and receiver windows is $2^{m-1}$ ?

Sol: For an example, take  $m = 2$ , which means the size of the window is  $2^m/2$ , or 2.

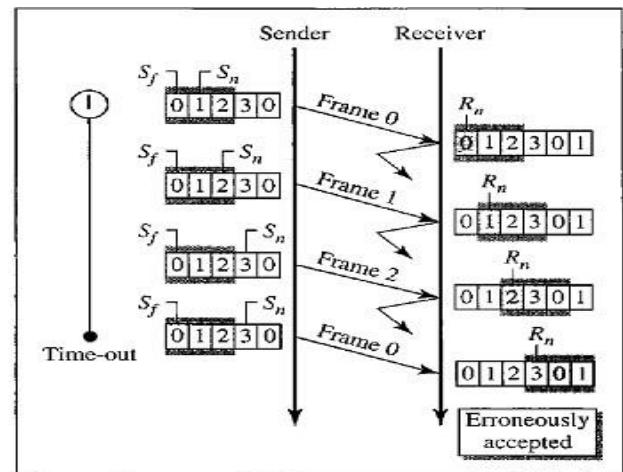
Now we compare window size=2 and window size =3.

#### Window size=2

- If the size of the window is 2 and all acknowledgments are lost, the timer for frame 0 expires and frame 0 is resent.
- The window of the receiver is now expecting frame 2, not frame 0, so this duplicate frame is correctly discarded.



a. Window size =  $2^{m-1}$



b. Window size >  $2^{m-1}$

#### Window Size=3

- When the size of the window is 3 and all acknowledgments are lost, the sender sends a duplicate of frame 0.
- However, this time, the window of the receiver expects to receive frame 0 (0 is part of the window), so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle.
- This is clearly an error.

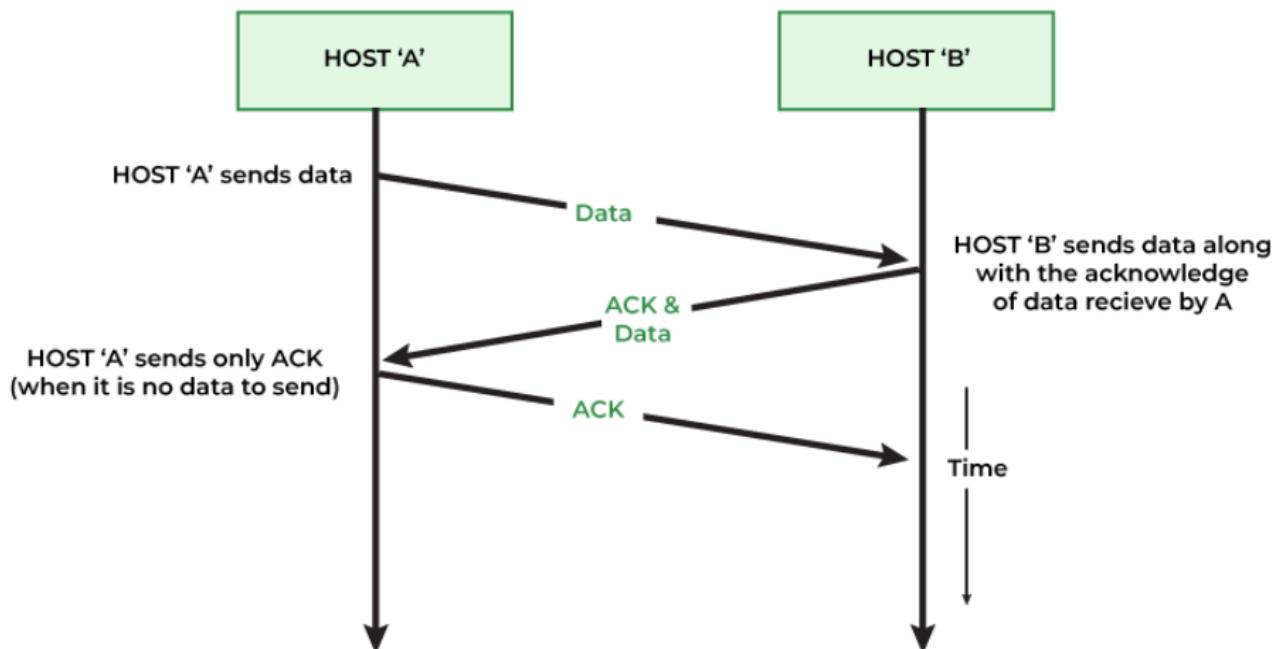
### PIGGYBACKING

- The protocols Stop-and-wait ARQ, Go-Back-N ARQ, Selective Repeat ARQ are all unidirectional. That means data frames flow in only one direction although control information such as ACK and NAK frames can travel in the other direction.
- In real life, data frames are normally flowing in both directions: from node A to node B and from node B to node A. This means that the control information also needs to flow in both directions. This technique called **piggybacking**.
- **Piggybacking** is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information from B to A. Similarly when a frame is carrying data from B to A, it can also carry control information frames from A to B.

Piggybacking is the technique of delaying outgoing acknowledgment temporarily and

attaching it to the next data packet. When a data frame arrives, the receiver waits and does not send the control frame (acknowledgment) back immediately. The receiver waits until its network layer moves to the next data packet. Acknowledgment is associated with this outgoing data frame. Thus the acknowledgment travels along with the next data frame.

## Working of Piggybacking



## **MULTIPLE ACCESS RESOLUTION PROTOCOLS**

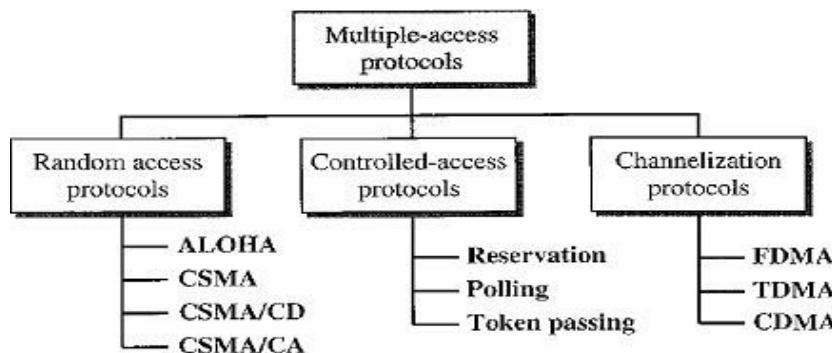
The Data Link Layer can be divided into 2 sublayers:

1. Data Link Control

2. Multiple Access Resolution

- The Multiple access resolution layers is responsible for resolving access to the shared media.
- Multiple Access Resolution layer is needed only when the channel is not dedicated. If the channel is dedicated then there is no need for multiple access sublayer.
- When nodes or stations are connected and use a common link, called a **Multipoint link** or **Broadcast link**, we need a multiple-access protocol to coordinate access to the link.
- The protocols have been devised to handle access to a shared link are called **Multiple Access Resolution Protocols**.

These protocols are categorized as below:



### **Random Access Protocols**

Random Access method is also called Contention method.

- There is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called **Random Access methods**.
- There are no rules to specify which station should send next. Stations compete with one another to access the medium. That is why these methods are also called **Contention Methods**.

At each instance, a station that has data to send, uses a procedure defined by the protocol to make a decision on whether to send the data or not to send the data. This decision depends on the state of the medium (idle or busy).

In this method:

- No station is superior to another station.
- No station is assigned the control over another station.
- No station permits or does not permit another station to send.

In this method, each station has the right to the medium without being controlled by any other station. If more than one station tries to send, there is an **Access Conflict-Collision** and the frames will be either destroyed or modified during the collision.

In order to avoid these collisions 4 protocols are implemented, they are:

- ALOHA
- CSMA
- CSMA/CD
- CSMA/CA

### **ALOHA**

ALOHA was developed at the University of Hawaii in early **1970**. It was designed for a Wireless radio LAN, but it can be used on any shared medium. Due to shared medium there are potential collisions in this arrangement.

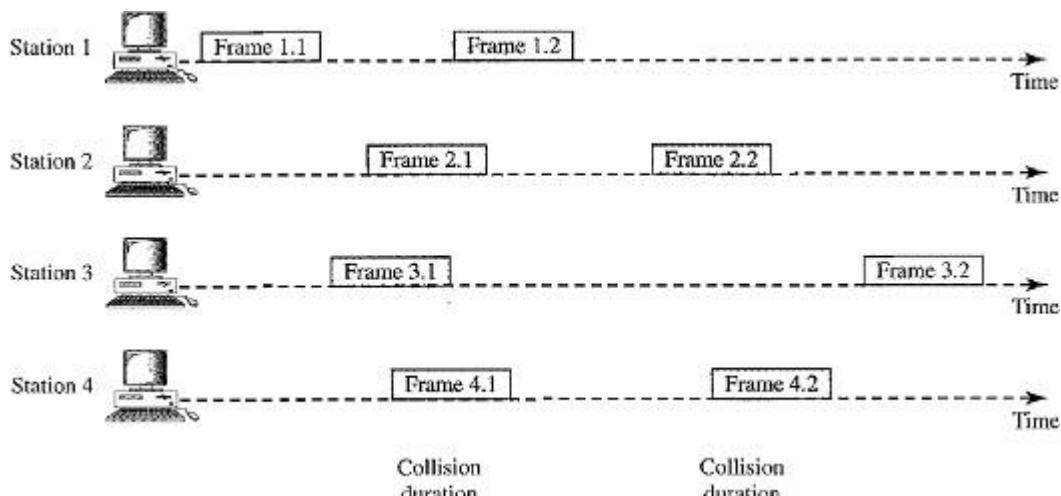
There are two types of methods in ALOHA

- i. Pure ALOHA
- ii. Slotted ALOHA

### **Pure ALOHA**

The original ALOHA or Pure ALOHA is a simple protocol.

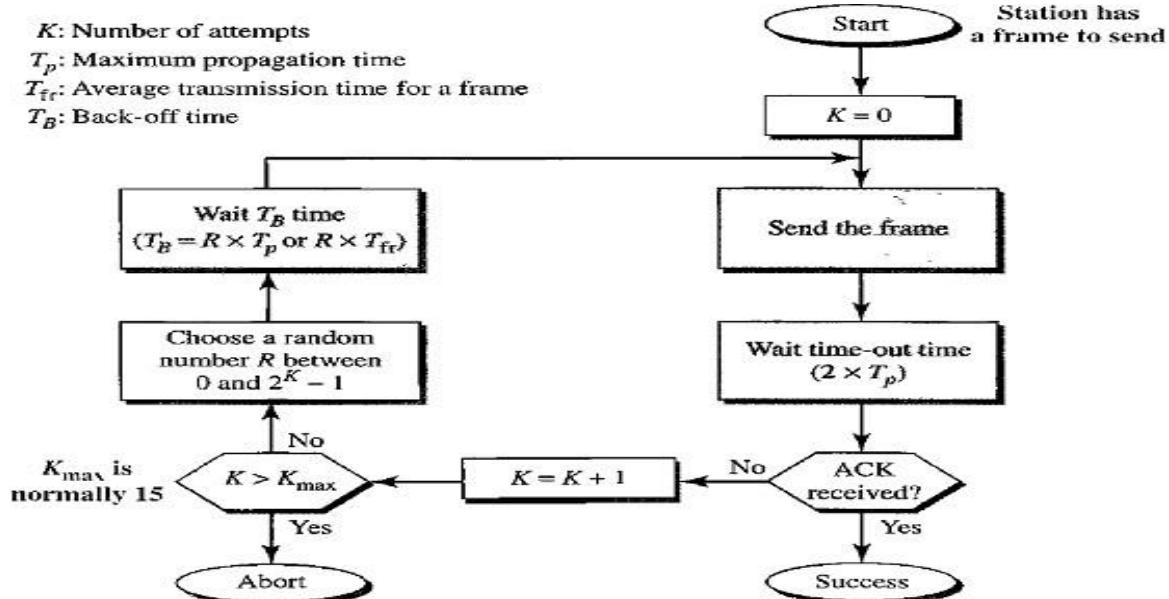
The idea is that each station sends a frame whenever it has a frame to send, there is only one channel to share, and there is the possibility of collision between frames from different stations.



- In the above figure, there are four stations each sending two frames and shares the same channel. Some of these frames collide because multiple frames are in contention for the shared channel.
- By observing the above figure only 2 frames frame 1.1 and frame 3.2 can be delivered at receiver, and the remaining frames collide with each other and they are lost or discarded at the receiver side.
- The pure ALOHA protocol relies on Acknowledgments (ACK) from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment.
- If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the 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. The randomness will help avoid more collisions; this time is called the Back-Off Time  $T_B$ .
- Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts  $K_{\max}$  a station must give up and try later.

The procedure for pure ALOHA is given in the figure:



- The time-out period is equal to the maximum possible round-trip propagation delay, which is twice the amount of time required to send a frame between the two most widely separated stations ( $2 \times T_p$ ).
- The back-off time  $T_B$  is a random value that normally depends on  $K$  (the number of attempted unsuccessful transmissions).
- For each retransmission, a multiplier in the range 0 to  $2^K - 1$  is randomly chosen and multiplied by  $T_p$  (maximum propagation time) or  $T_{fr}$  (Frame transmission time or the average time required to send out a frame) to find  $T_B$ .
- Note that in this procedure, the range of the random numbers increases after each collision. The value of  $K_{\max}$  is usually chosen as **15**.

### Vulnerable time

Vulnerable time is the length of time, in which there is a possibility of collision.

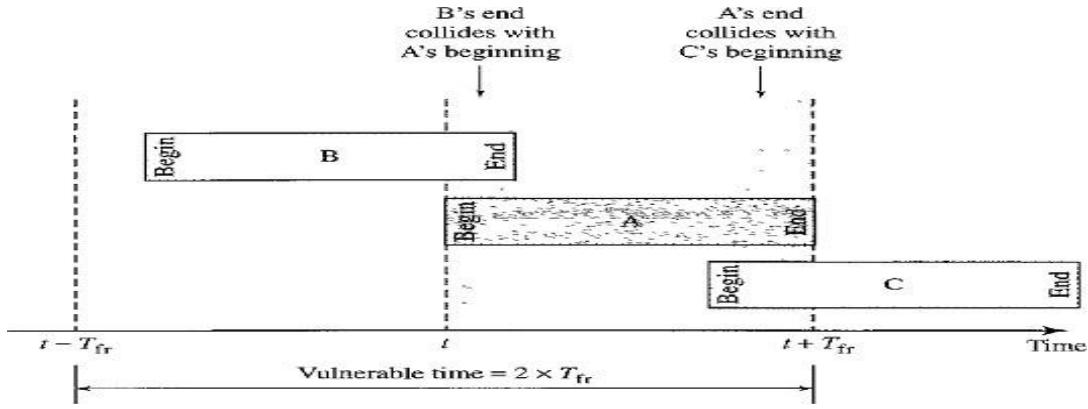
Let us assume that the stations send fixed-length frames with each frame taking  $T_{fr}$ 's to send.

Station A sends a frame at time  $t$ .

Now station B has already sent a frame between  $t - T_{fr}$  and  $t$ .

This leads to a collision between the frames from station A and station B.

The end of B's frame collides with the beginning of A's frame.



Suppose that station C sends a frame between  $t$  and  $t + T_{fr}$ .

Here, there is a collision between frames from station A and station C. The beginning of C's frame collides with the end of A's frame.

The vulnerable time, during which a collision may occur in pure ALOHA, is 2 times the frame transmission time.

$$\text{Pure ALOHA vulnerable time} = 2 \times T_{fr}$$

### Throughput

- The throughput for pure ALOHA is  $S = G \times e^{-2G}$ .
- The maximum throughput  $S_{max} = 0.184$  when  $G = (1/2)$ .
- (i.e.) one frame is generated during two frame transmission times, then 18.4 percent of these frames reach their destination successfully.

### Slotted ALOHA

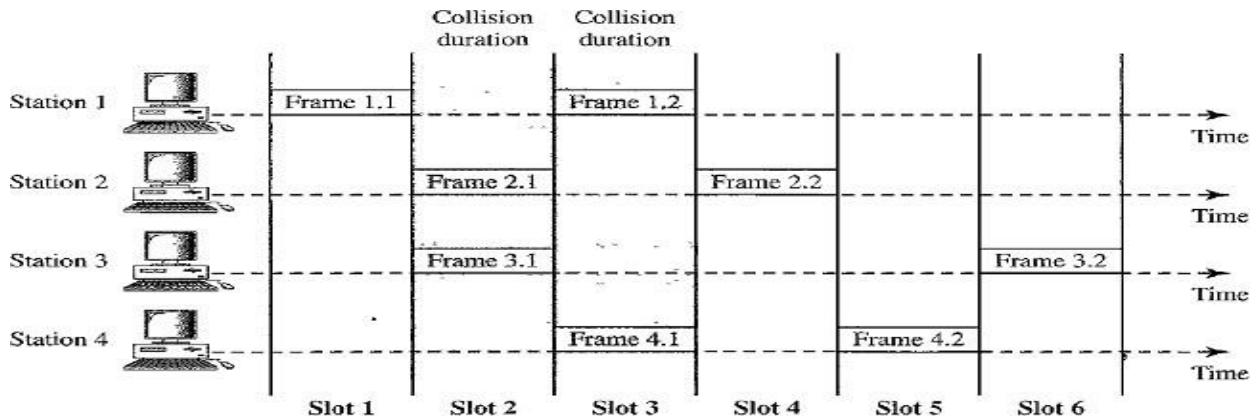
Pure ALOHA has a vulnerable time of  $2 \times T_{fr}$ . This is so because there is no rule that defines when the station can send.

- A station may send soon after another station has started or soon before another station has finished.
- Slotted ALOHA was invented to improve the efficiency of pure ALOHA.
- In slotted ALOHA we divide the time into slots of  $T_{fr}$ 's and force the station to send only at the beginning of the time slot.
- A station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot.
- This means that the station which started at the beginning of this slot has already finished sending its frame.
- There is still the possibility of collision if two stations try to send at the beginning of the same time slot.

i.e. the vulnerable time for slotted ALOHA is one-half that of pure ALOHA.

$$\text{Slotted ALOHA vulnerable time} = T_{fr}.$$

Below figure shows an example of frame collisions in slotted ALOHA.



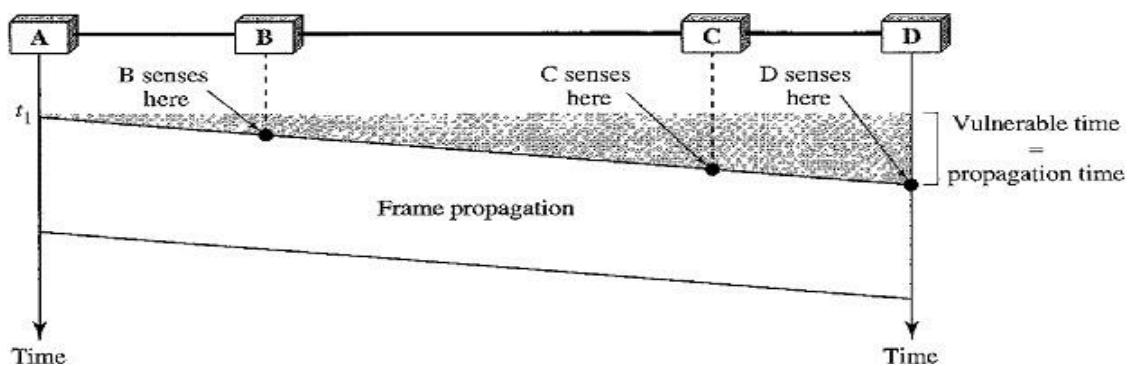
### Throughput

- The throughput for slotted ALOHA is  $S = G \times e^{-G}$ .
- The maximum throughput  $S_{max} = 0.368$  when  $G = 1$ .
- If a frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully.

### Carrier Sense Multiple Access (CSMA)

CSMA was developed to minimize the chance of collision and improve the performance, but it cannot eliminate the collision.

- CSMA is based on the principle of “sense before transmit” or “listen before talk”.
- That means, CSMA requires that each station first listen to the medium or check the state of the medium before sending. Stations are connected to a shared channel.
- Collision occurs due to propagation delay. (i.e.) when a station sends a frame, it still takes time for the first bit to reach every station and for every station to sense it.
- At time  $t_1$ , station B senses the medium and finds it idle, so it sends a frame.
- At time  $t_2$  ( $t_2 > t_1$ ) station C senses the medium and finds it idle because at this time, the first bits from station B have not reached station C.
- Station C also sends a frame. The two signals collide and both frames are destroyed.



### Vulnerable Time of CSMA

$$\text{Vulnerable Time} = \text{Propagation Time}$$

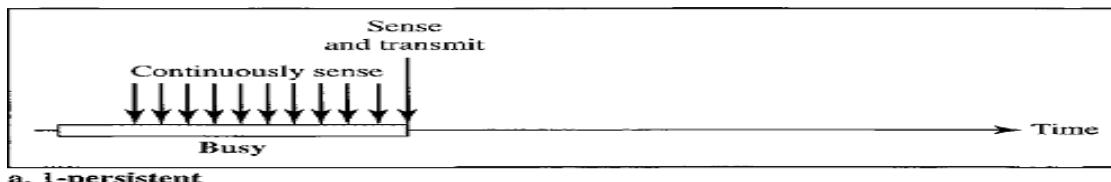
- It is the time needed for a signal to propagate from one end of the medium to the other.
- The leftmost station A sends a frame at time  $t_1$  which reaches the rightmost station D at time  $t_1 + T_p$ .

## Persistence Methods

Persistence methods define what action will be taken by a station if the channel is busy or idle. There are 3 methods in persistence method:

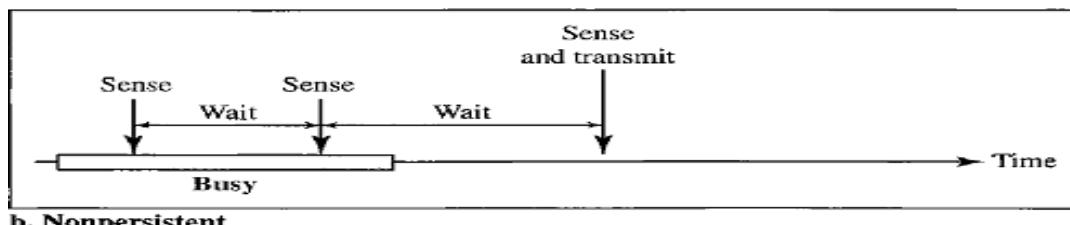
### 1-persistent method

- 1-persistent method is simple and straightforward.
- In this method, after the station finds the line idle, it sends its frame immediately (with probability 1).
- This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately.



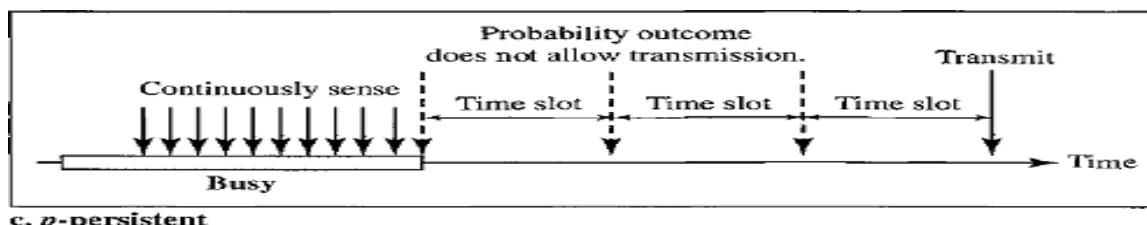
### Non-persistent Method

- In the non-persistent method, a station that has a frame to send senses the line. If the line is idle, it sends immediately.
- If the line is not idle, it waits a random amount of time and then senses the line again.
- The non-persistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously.
- However, this method reduces the efficiency of the network because the medium remains idle when there may be stations with frames to send.



### p-Persistent Method

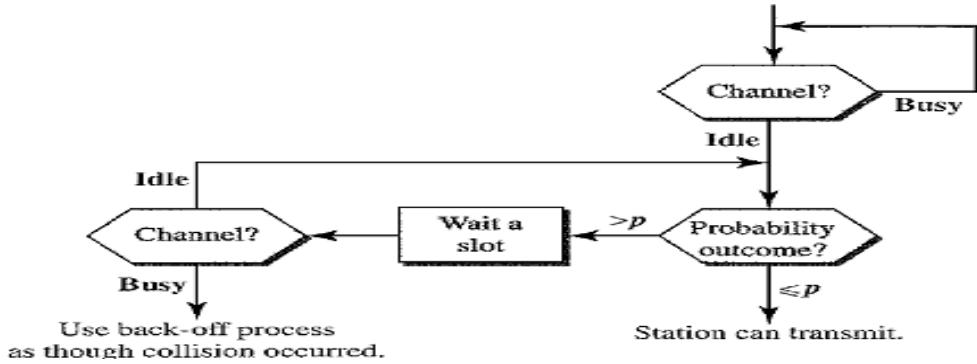
- The p-persistent method is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time.
- It reduces the chance of collision and improves efficiency.



In this method, after the station finds the line idle it follows these steps:

- With probability  $p$ , the station sends its frame.
- With probability  $q = 1 - p$ , the station waits for the beginning of the next time slot and checks the line again.
- If the line is idle, it goes to step 1.

- If the line is busy, it acts as though a collision has occurred and uses the backoff procedure.

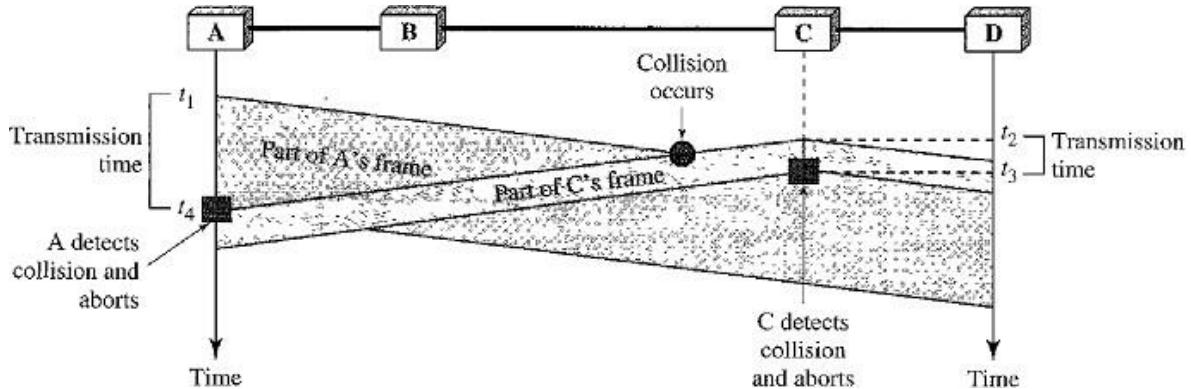


### Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

CSMA method does not specify the procedure following a collision but CSMA/CD 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 it is successful the station is finished.
- If it is not successful and there is a collision, the frame is sent again.



- At time  $t_1$ , station A has executed its persistence procedure and starts sending the bits of its frame.
- At time  $t_2$ , station C has not yet sensed the first bit sent by A.
- Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right.
- The collision occurs sometime after time  $t_2$ . Station C detects a collision at time  $t_3$  when it receives the first bit of A's frame. Station C immediately aborts transmission.
- Station A detects collision at time  $t_4$  when it receives the first bit of C's frame; it also immediately aborts transmission.
- A transmits for the duration  $t_4 - t_1$ ; C transmits for the duration  $t_3 - t_2$ .
- At time  $t_4$ , the transmission of A's frame is aborted; at time  $t_3$ , the transmission of C's frame is aborted. Both are incomplete.
- This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection.

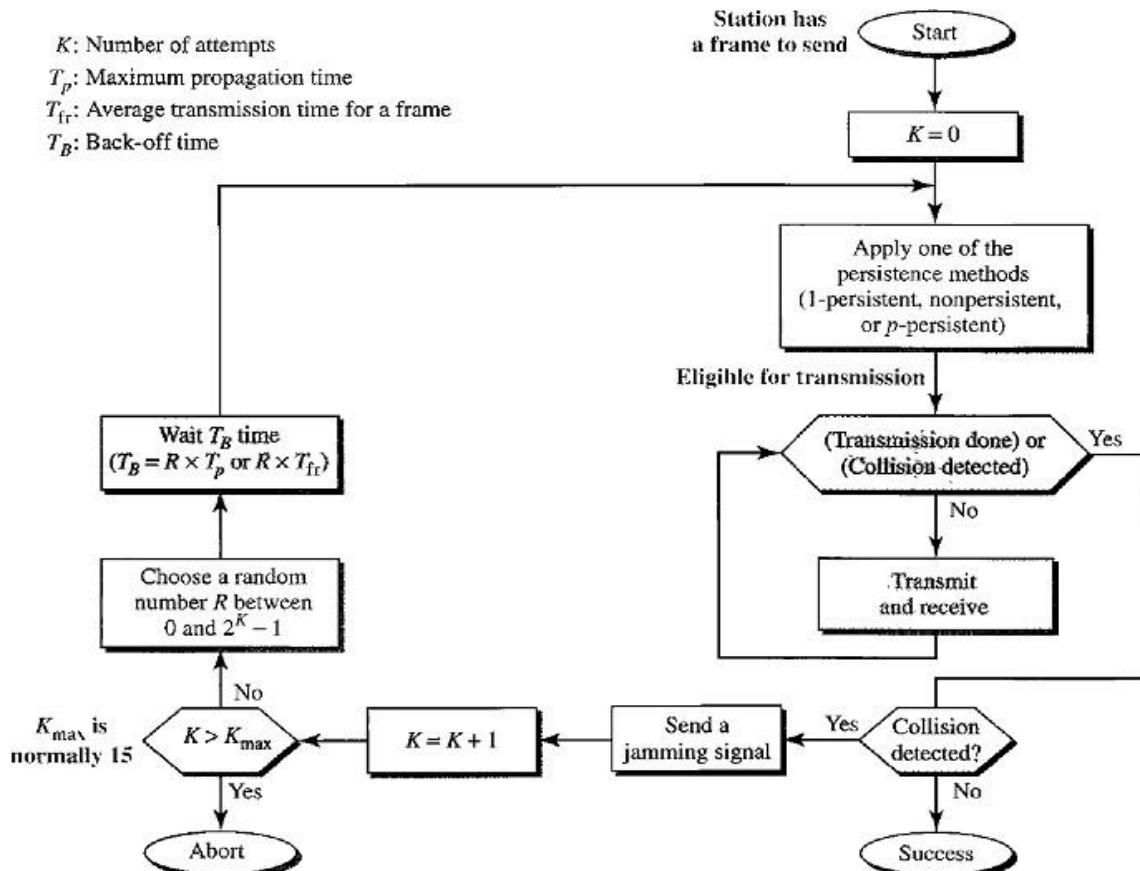
## Minimum Frame Size (2Tp)

If the two stations involved in a collision are the maximum distance apart, the signal from the first station takes  $T_p$  time to reach the second station and the effect of the collision takes another  $T_p$  time to reach the first station.

Therefore, the frame transmission time  $T_{fr}$  must be at least two times the maximum propagation time  $T_p$ . So the first station must still be transmitting after  $2T_p$ .

### Procedure

- We need to sense the channel before we start sending the frame by using one of the persistence processes.
- In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In CSMA/CD, transmission and collision detection is a continuous process.
- We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously using two different ports.
- We use a loop to show that transmission is a continuous process.
- We constantly monitor in order to detect one of two conditions: either transmission is finished or a collision is detected. Either event stops transmission.
- When we come out of the loop, if a collision has not been detected, it means that transmission is complete; the entire frame is transmitted. Otherwise, a collision has occurred.
- Here we send a short jamming signal that enforces the collision in case other stations have not yet sensed the collision.



## **Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)**

Wireless networks cannot detect collisions, hence we need to avoid collision.

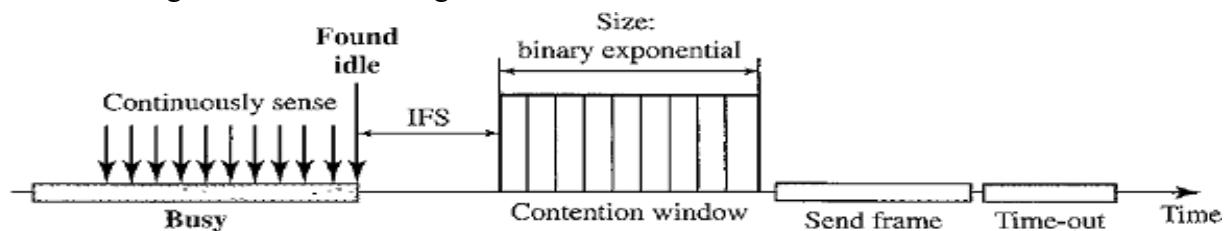
### **Need for CSMA/CA**

- In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver. The detected energy almost doubles when there is a collision.
- In a wireless network much of the **Sent Energy** is lost in transmission. The received signal has very little energy. Therefore a collision may add only 5 to 10 percent additional energy. This is not useful for effective collision detection.
- Hence in a wireless networks we need to avoid collisions instead of detecting collisions. In order to avoid collisions CSMA/CA was introduced for wireless networks.

CSMA/CA uses three strategies to avoid collisions:

- InterFrame Space
- Contention Window
- Acknowledgements

The below figure shows the timing in CSMA/CA:



### **Interframe Space (IFS)**

- Collisions are avoided by deferring transmission 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 may have already started transmitting.
- The 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.
- If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time equal to the contention time.
- The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned a shorter IFS has a higher priority.

### **Contention Window**

- The contention window is an amount of time divided into slots.
- A station that is ready to send chooses a random number of slots as its wait time.
- The number of slots in the window changes according to the binary exponential back-off strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time.

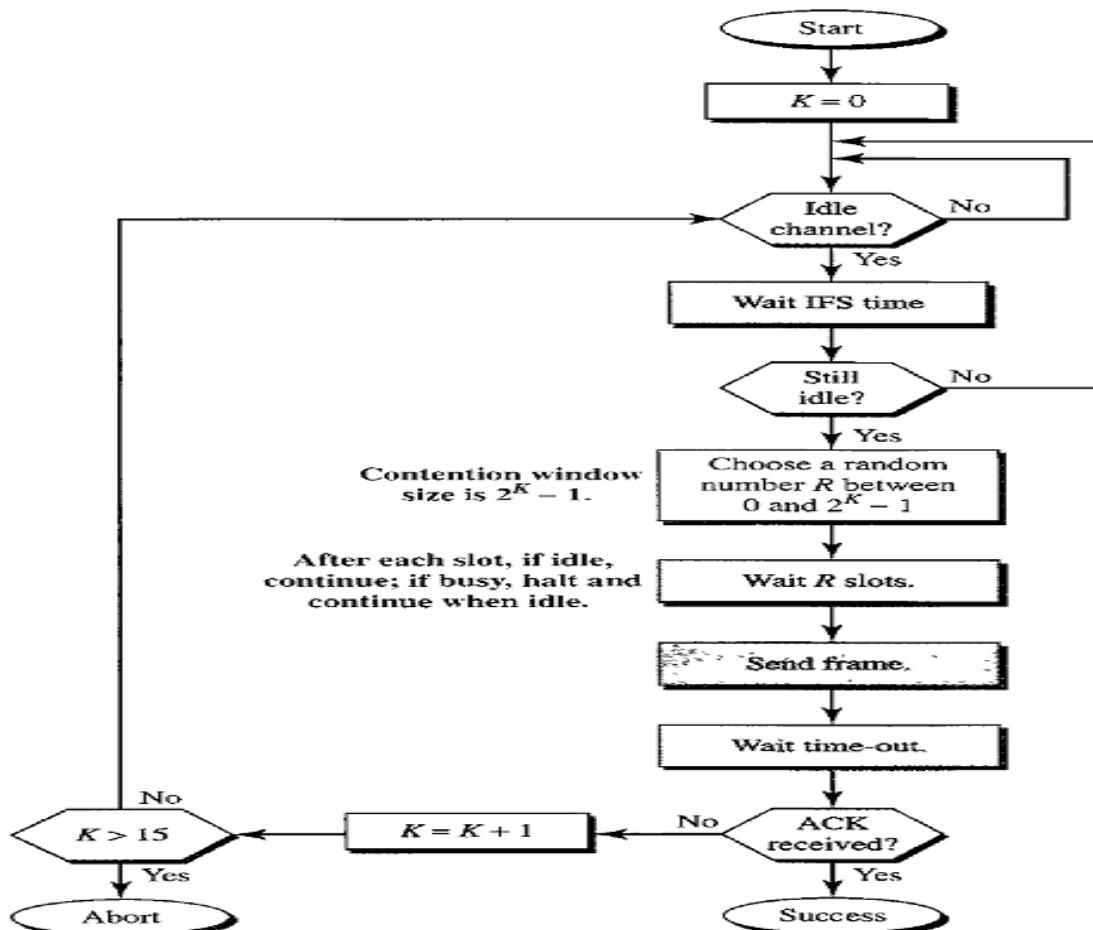
- A random outcome defines the number of slots taken by the waiting station.
- In contention window, the station needs to sense the channel after each time slot.
- In CSMA/CA, if the station finds the channel busy, it does not restart the timer of the contention window instead it stops the timer and restarts it when the channel becomes idle. This gives priority to the station with the longest waiting time.

### Acknowledgment

- The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.
- Acknowledgement is needed even though we take all the precautions, there still may be a collision resulting in destroyed data and the data may be corrupted during the transmission.

### Procedure

- The channel needs to be sensed before and after the IFS.
- The channel also needs to be sensed during the contention time.
- For each time slot of the contention window, the channel is sensed.
- If it is found idle, the timer continues.
- If the channel is found busy, the timer is stopped and continues after the timer becomes idle again.



## **WIRED LANS: ETHERNET (802.3)**

Local Area Network (LAN) is a computer network that is designed for limited geographic area such as building or campus. LAN is a shared resource.

LAN technologies: Ethernet, Token Ring, Token Bus, FDD, ATM LAN.

## **STANDARD ETHERNET (IEEE 802.3)**

The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations:

- Standard Ethernet (1 Mbps)
- Fast Ethernet (100 Mbps)
- Gigabit Ethernet (1 Gbps)
- Ten-Gigabit Ethernet (10 Gbps)

### **(1) Standard Ethernet (1 Mbps)**

#### **MAC Sub-layer**

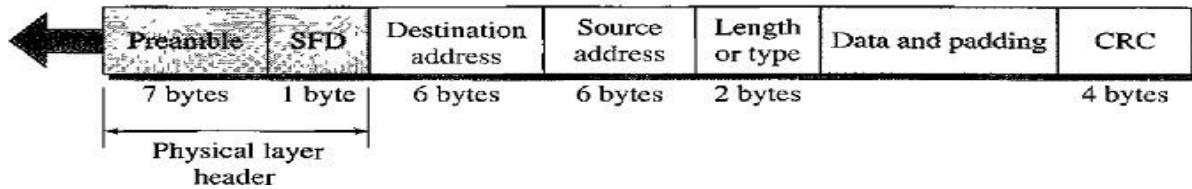
In Standard Ethernet, the MAC sublayer governs the operation of the access method. It also frames data received from the upper layer and passes them to the physical layer.

#### **Frame Format**

The Ethernet frame contains **seven** fields: Preamble, SFD, DA, SA, Length or Type of protocol data unit (PDU), Upper-layer data, and CRC.

**Preamble:** 56 bits of alternating 1s and 0s.

**SFD:** Start frame delimiter, flag (10101011)



#### **Preamble**

- The first field of the 802.3 frame contains 7 bytes (56 bits) of alternating 0's and 1's that alerts the receiving system to the coming frame and enables it to synchronize its input timing.
- The pattern provides only an alert and a timing pulse.
- The 56-bit pattern allows the stations to miss some bits at the beginning of the frame.

Note: The preamble is actually added at the physical layer and is not (formally) part of the frame.

### **Start Frame Delimiter (SFD)**

- The second field (1 byte: 10101011) signals the beginning of the frame.
- The SFD warns the stations that “This is the last chance for synchronization”.
- The last 2 bits is 11 and alerts the receiver that the next field is the destination address.

### **Destination address (DA)**

- The DA field is 6 bytes and contains the physical address of the destination station (i.e.) stations to receive the packet.

### **Source address (SA)**

- The SA field is also 6 bytes and contains the physical address of the sender of the packet.

### **Length or Type**

- This field is defined as a type field or length field. Both uses are common today.
- The original Ethernet used this field as the **Type field** to define the upper-layer protocol using the MAC frame.
- The IEEE standard used it as the length field to define the number of bytes in the data field.

### **Data**

- This field carries data encapsulated from the upper-layer protocols.
- It is a minimum of 46 bytes and a maximum of 1500 bytes.

### **CRC**

- The last field contains Error Detection information. The Length of this field is 4 byte (32-bit) hence a CRC-32 is used.