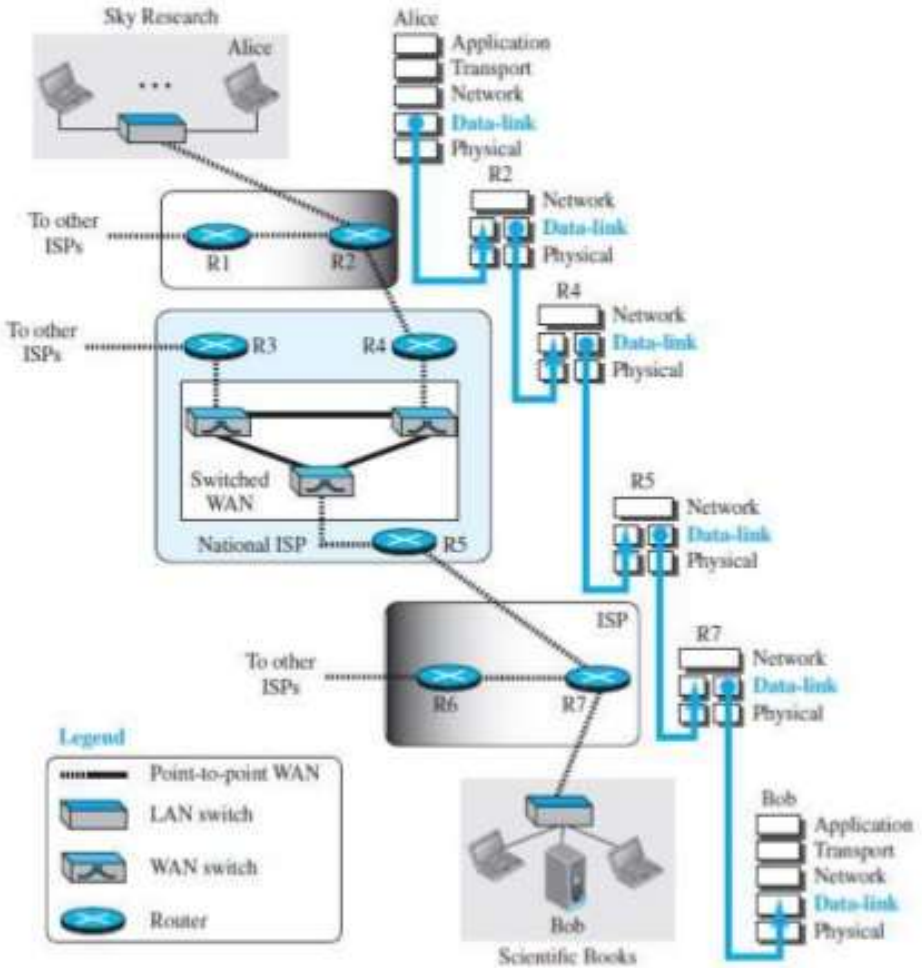
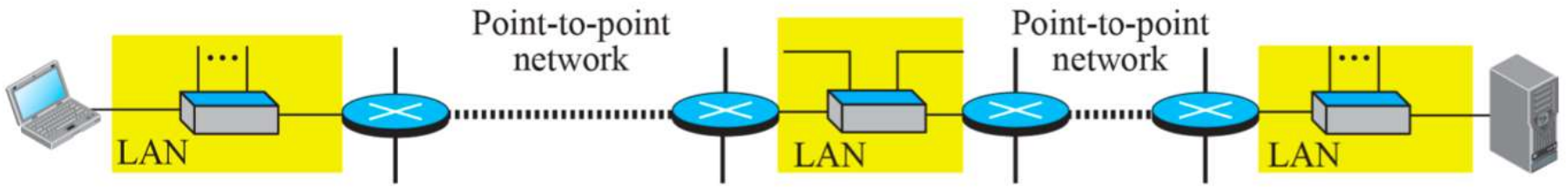
# Introduction
# to
# Data Link Layer

# Communication at the DLL



# Nodes and Links




a. A small part of the Internet


b. Nodes and links

- Communication in DLL is node-to-node
- LANs and WANs are connected using routers.
- The two end hosts and the routers as nodes and the networks in between as links.

# Services Provided to the Network Layer

Transferring data from the network layer on the source machine to the network layer on the destination machine.

The Data link layer transmits the bits to the destination machine so they can be handed over to the network layer there.

# Services Provided to the Network Layer

The data link layer can be designed to offer various services. The services are:

1. Unacknowledged connectionless service.
2. Acknowledged connectionless service.
3. Acknowledged connection-oriented service.

# Services Provided to the Network Layer

1. **Unacknowledged connectionless service.**

   The source machine sends independent frames to the destination machine without having the destination machine acknowledge them.

   Example: ***Ethernet***

   ***No logical connection*** is established beforehand or released afterward.

# Services Provided to the Network Layer

2. **Acknowledged connectionless service**.

When this service is offered, there are still ***no logical connections*** used.

Each frame sent is individually acknowledged.

The sender knows whether a frame has arrived correctly or been lost.

# Services Provided to the Network Layer

**3. Acknowledged connection-oriented service.**

The source and destination machines establish a connection before any data are transferred.

Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is received.

It guarantees that each frame is received exactly once and that all frames are received in the right order.

# Services Provided to the Network Layer

**4. Framing**

The bit stream received by the data link layer is not guaranteed to be error-free.

The number of bits received may be less than, equal to, or more than the number of bits transmitted.

It is up to the data link layer to detect and if necessary, correct errors.

The data link layer breaks the bit stream into discrete frames and computes the checksum for each frame.

When a frame arrives at the destination, the checksum is recomputed.

If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it.

# Services Provided to the Network Layer

## 5. Flow Control

What to do with a sender that systematically wants to transmit frames faster than the receiver can accept them.

This situation can occur when the sender is running on a fast, powerful computer and the receiver is running on a slow, low-end machine.

# Services Provided to the Network Layer

## 5. Flow Control

There are two approaches to prevent this problem:

- **Feedback-based flow control**: the receiver sends back information to the sender giving it permission to send more data, or at least telling the sender how the receiver is doing.
- **Rate-based flow control:** the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

# Services Provided to the Network Layer

## 6. Error Control

The problem here is how to make sure all frames are eventually delivered to the network layer at the destination and in the proper order.

The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line.

The protocol calls for the receiver to send back special control frames bearing positive or negative acknowledgments.

Hardware troubles may cause a frame to vanish completely.

This possibility is dealt with by introducing timers into the data link layer.

# Services Provided to the Network Layer

**6. Congestion Control**

Although the link may be congested due to frames.

Which may result in frame loss.

Most often occur in network layer.

# Two categories of link.

DLL controls how the medium is used.

We can have a **point-to-point link** or a **broadcast link**.

Link is dedicated to two devices.

Link is shared between several pairs of devices.

# Two Sub-layers

Data Link Control (DLC) – Handles Issues with both

Media Access Control (MAC) –  Handles problems with broadcast links

# Two Sub-layers

Data Link Control (DLC) – Handles Issues with both

Media Access Control (MAC) – Handles problems with broadcast links

- IP Address in datagram should not change. (At both end it will create issues)
- Link Address – Physical Address – MAC Address.

- When a frame has been send the datagram will be encapsulated inside a frame.
- Two addresses will be added to the frame header.
- This will change every time the frame moves from one link to another.

# Link layer Addressing

## Three Types of Addresses

- Unicast Address
  - One to One Communication
  - A frame with a unicast address destination is only for one entity in the link.
- Multicast Address
  - One to Many Communication
  - However the jurisdiction is local (Inside the link)
- Broadcast Address
  - Many to Many Communication
  - One to all communication.
  - A frame with one broadcast address is send to all entities in the link.

# Link layer Addressing

## Address Resolution Protocol (ARP)

- Anytime a node has an IP datagram to send to another node in a link.
- It has the IP address of the receiving node.
- The source host knows the IP address of the default router.
- Each router gets the IP address of the next router by using its forwarding table.
- The last router knows the IP address of the destination host.

# Link layer Addressing

## Address Resolution Protocol (ARP)

Its functionality is to translate IP address to physical address.

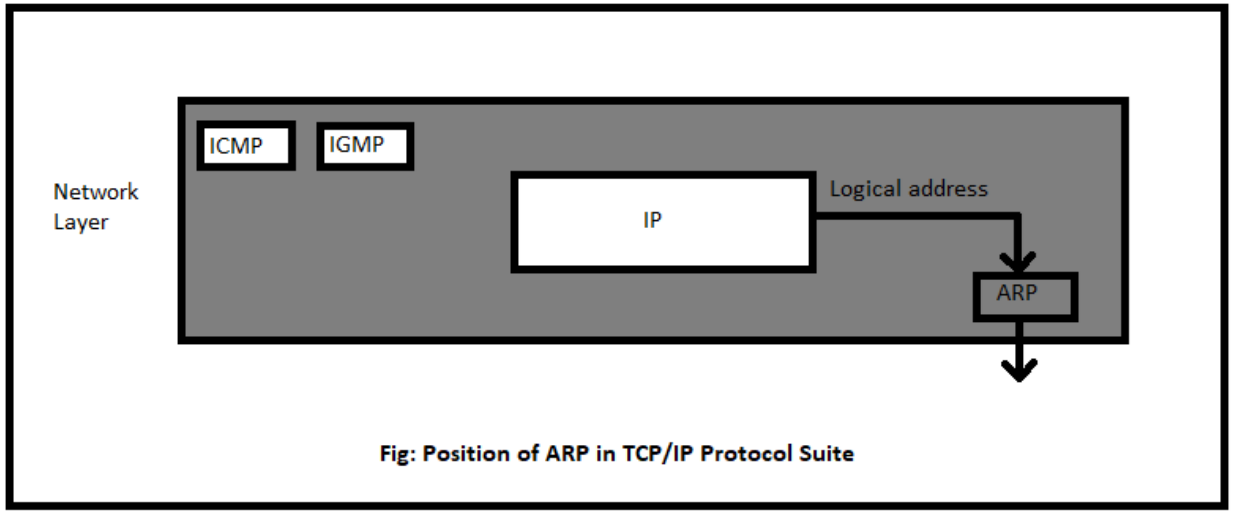


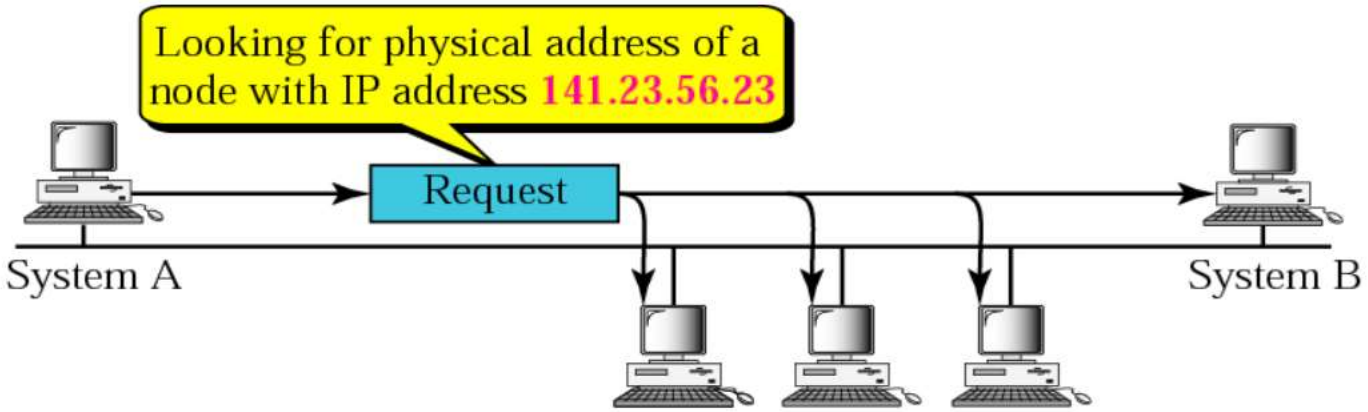

Fig: Position of ARP in TCP/IP Protocol Suite

# ARP

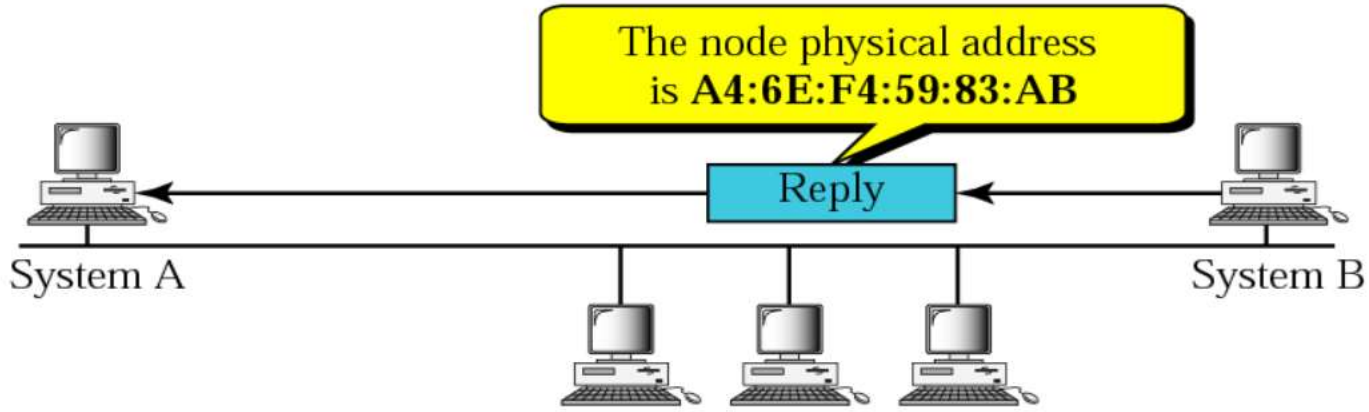
- It belongs to the network layer.
- It maps an IP address to a logical link address.
  - **ARP accepts an IP address from the IP protocol.**
  - **Maps the address to the corresponding link layer address.**
  - **Passes it to the DLL.**

# ARP

## ***ARP OPERATION***




a. ARP request is broadcast




b. ARP reply is unicast

# ARP

The important terms associated with ARP are :

**ARP Cache:** After resolving MAC address, the ARP sends it to the source where it stores in a table for future reference. The subsequent communications can use the MAC address from the table.

**ARP Cache Timeout:** It indicates the time for which the MAC address in the ARP cache can reside.

# ARP

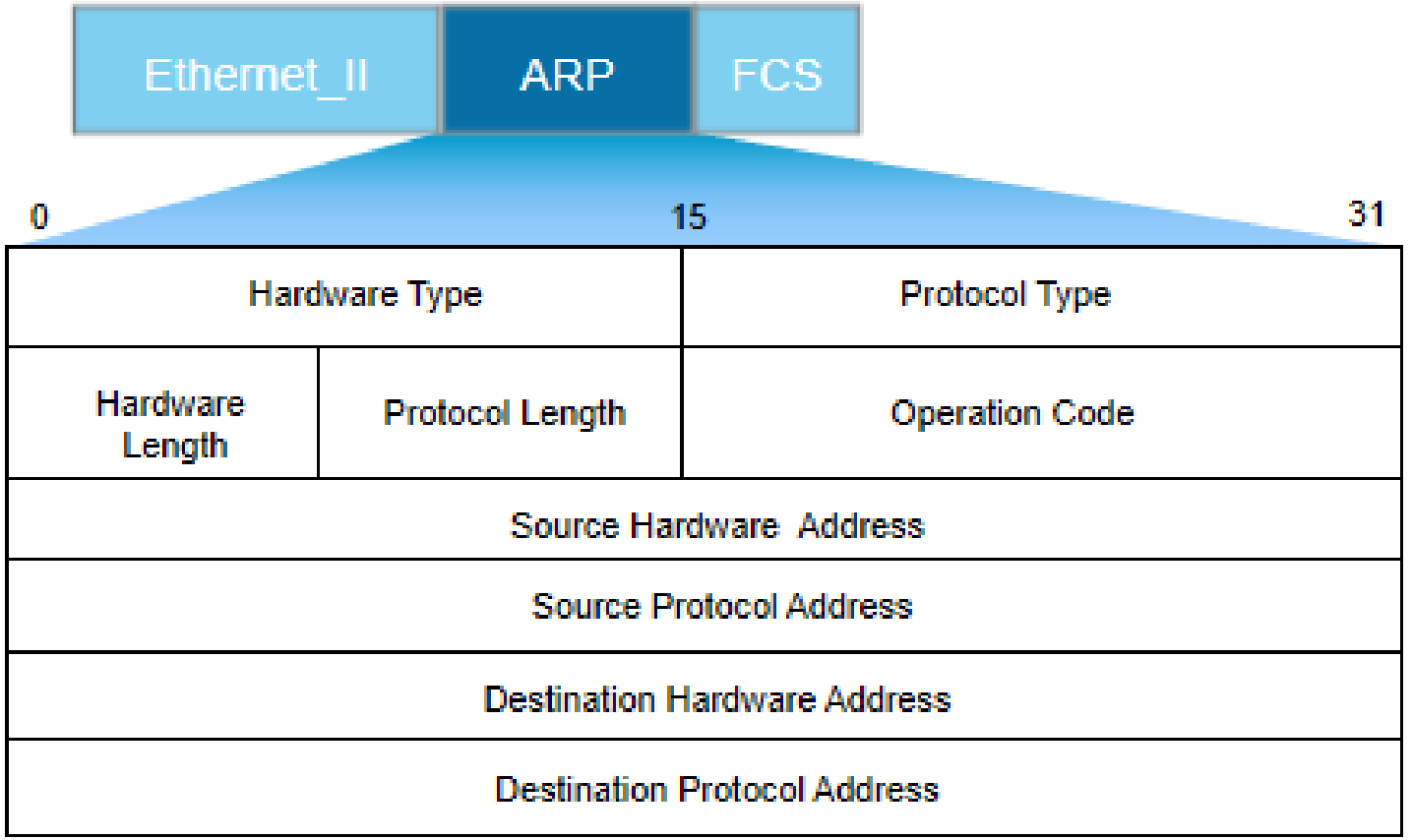
**ARP request:** This is nothing but broadcasting a packet over the network to validate whether we came across destination MAC address or not.

1. The physical address of the sender.
2. The IP address of the sender.
3. The physical address of the receiver is FF:FF:FF:FF:FF:FF.
4. The IP address of the receiver.

**ARP response/reply:** It is the MAC address response that the source receives from the destination which aids in further communication of the data.

# ARP

## Packet Format



# Data Link Control (DLC)

## Methods of Framing

Byte Count:

This method uses a field in the header to specify the number of bytes in the frame.

When the data link layer at the destination sees the byte count, it knows how many bytes follow and hence where the end of the frame is.

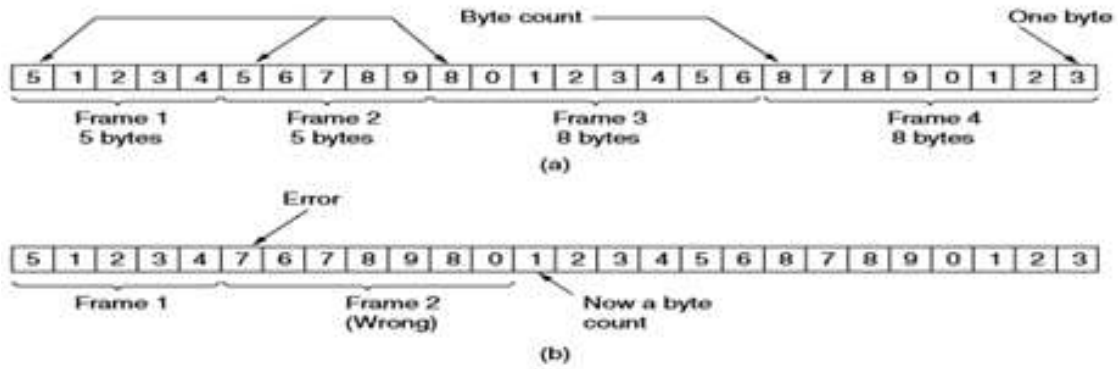



Figure 3-3. A byte stream. (a) Without errors. (b) With one error.

# Data Link Control (DLC)

## Methods of Framing

### Flag bytes with byte stuffing:

Each frame start and end with special bytes.

This byte called the flag byte is used as starting and ending delimiter.

The sender's data link layer inserts a special escape byte (ESC) just before each ***accidental flag byte*** in the data.

The data link layer on the receiving end removes the escape bytes before giving the data to the network layer.

A framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it.

# Data Link Control (DLC)

## Methods of Framing

Flag bytes with byte stuffing:

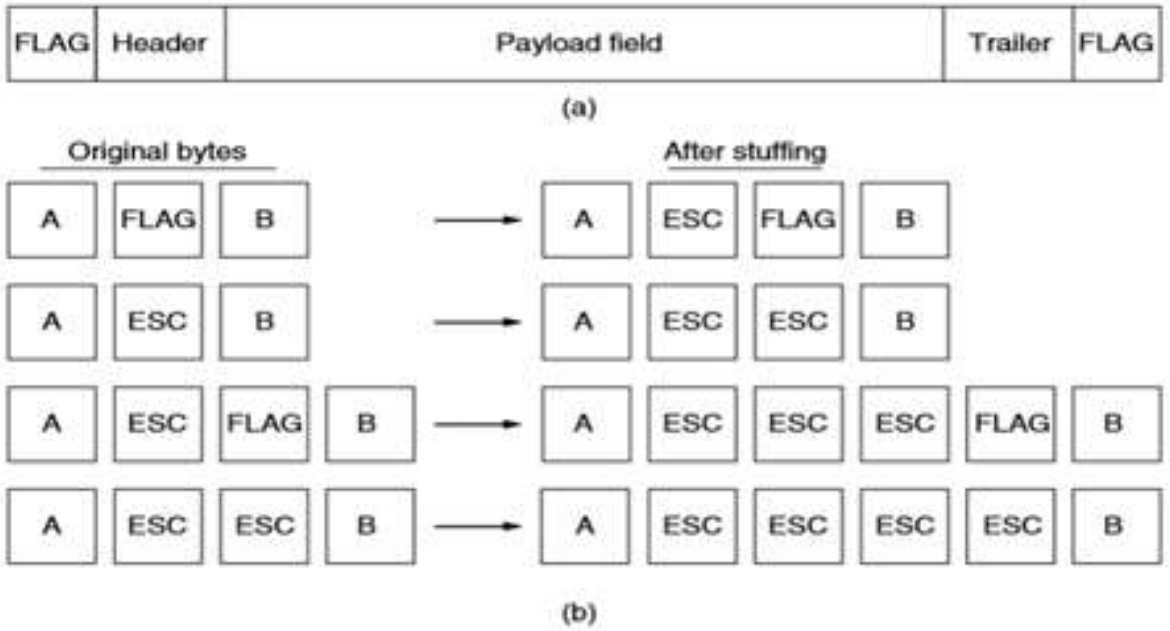



**Figure 3-4.** (a) A frame delimited by flag bytes. (b) Four examples of byte sequences before and after byte stuffing.

# Data Link Control (DLC)

## Methods of Framing

Flag bits with bit stuffing:

- Each frame begins and ends with a special bit pattern (in fact a flag byte).
- Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This is a bit ***stuffing.***
- When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically de-stuffs (deletes) the 0 bit.

# Data Link Control (DLC)

## Methods of Framing

Flag bits with bit stuffing:

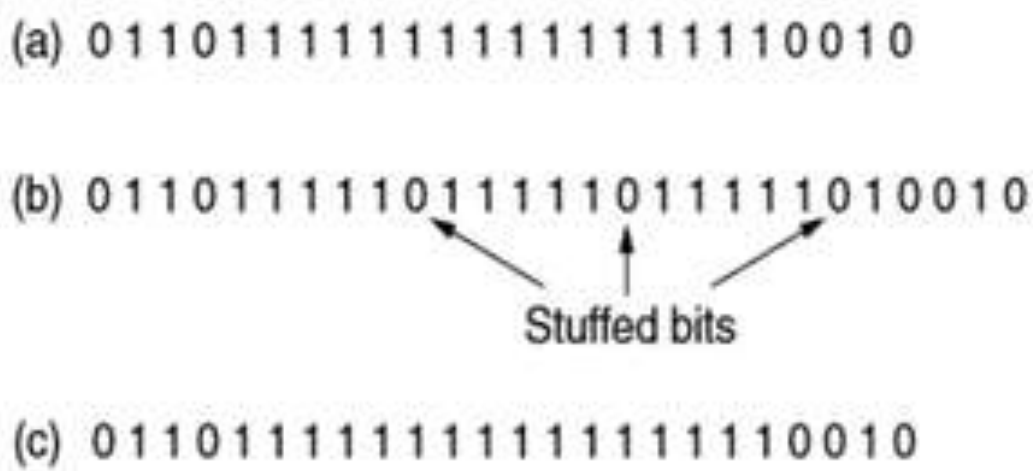



**Figure 3-5.** Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuffing.

# Data Link Layer Protocols

**What happens in the Data Link Layer?**

*The packet passed across the interface to it from the network layer is pure data. When the data link layer accepts a packet, it encapsulates the packet in a frame consisting of an embedded packet, some control information (in the header), and a checksum (in the trailer).*

# Data Link Layer Protocols

**Simplex Protocol** *(Neither flow control nor error control )*

- We assume that the receiver can immediately handle any frame it receives.
- The protocol consists of two distinct procedures, **a sender and a receiver.**
- The sender runs in the data link layer of the source machine, and the receiver runs in the data link layer of the destination machine.
- No ***sequence numbers*** or ***acknowledgment***s are used here, so ***MAX SEQ*** is not needed.
- The only event type possible is ***frame_arrival***.
- The sender is in an infinite while loop just pumping data out onto the line.

# Data Link Layer Protocols

**Some definitions needed in the protocols to follow:**

```
typedef enum {frame arrival} event type;
#include "protocol.h"
void sender1(void)
{
frame s;                                  /* buffer for an outbound frame */
packet buffer;                            /* buffer for an outbound packet */
while (true) {
from network layer(&buffer);              /* go get something to send */
s.info = buffer;                          /* copy it into s for transmission */
to physical layer(&s);                    /* send it on its way */
}
```

# Data Link Layer Protocols

```
void receiver1(void)
{
frame r;
event type event;                    /* filled in by wait, but not used here */
while (true) {
wait for event(&event);              /* only possibility is frame arrival */
from physical layer(&r);             /* go get the inbound frame */
to network layer(&r.info);           /* pass the data to the network layer */
}
```

# Data Link Layer Protocols

**Simplex Protocol** *(Neither flow control nor error control )*

The body of the loop consists of three actions:

- Go fetch a packet from the network layer.
- Construct an outbound frame using the variables.
- Send the frame on its way.
- Only the info field of the frame is used by this protocol.
- The receiver initially waits for something to happen.
- The frame arrives and the procedure waits for event returns, with event set ***to_frame_arrival*** (which is ignored anyway).
- The call to ***from_physical_layer*** removes the newly arrived frame from the hardware buffer.
- Finally, the data portion is passed on to the network layer, and the data link layer settles back to wait for the next frame.

# Data Link Layer Protocols

## Stop and Wait Protocol

Uses both flow and error control.

Here the main problem is to deal with how to prevent the sender from flooding the receiver with frames faster than the latter is able to process.

Protocols in which the sender sends one frame and then waits for an acknowledgment before proceeding are called ***stop-and-wait***.

The sender must wait until an acknowledgment frame arrives before looping back and fetching the next packet from the network layer.

# Data Link Layer Protocols

**Stop and Wait Protocol**

*Sender states:*

**1. Ready state:-**

- It is only waiting for a packet from the network layer.
- If a packet comes, the sender creates a frame, saves a copy of the frame, starts the timer and sends the frame.
- Then moves to the blocking state.

**2. Blocking state:-**

Now three events can occur,

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

# Data Link Layer Protocols

## Stop and Wait Protocol

*Receiver states:*

- Always in the ready state.
- Here two events may occur:
  1. If an error free frame arrives, the message in the frame is delivered to the network layer and the ACK is send.
  2. If a corrupted frame arrives, the frame is discarded.

# Data Link Layer Protocols

The **sequence number** is the byte **number** of the first byte of data in the TCP packet sent (also called a TCP segment).

The **acknowledgement number** is the **sequence number** of the next byte the receiver expects to receive.

Sequence numbers – Data frames (0,1,0,1.....)

Acknowledgment numbers – ACK frames (1,0,1,0.....)

# Data Link Layer Protocols

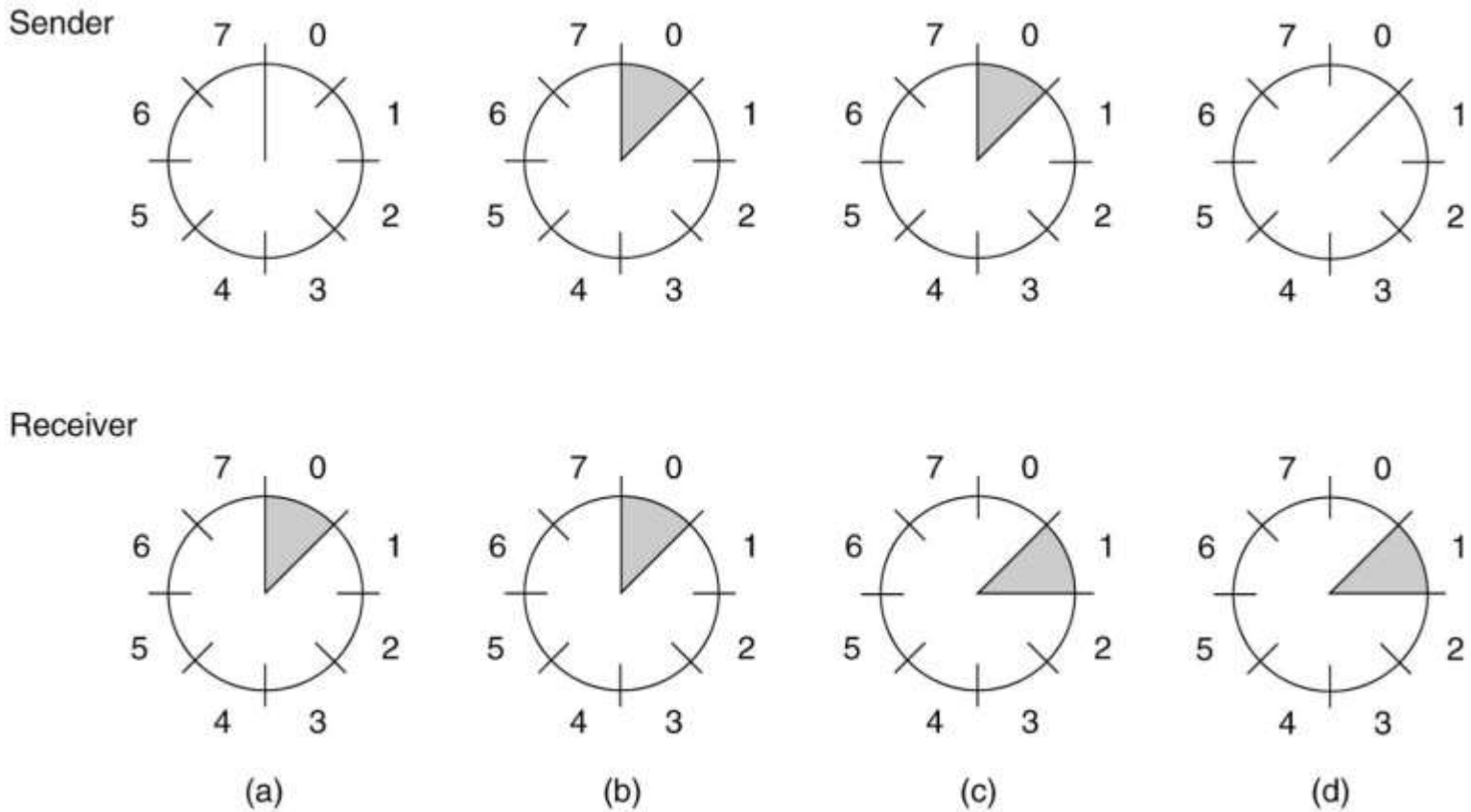
**PIGGYBACKING**

- When a data frame arrives, instead of immediately sending a separate control frame, the receiver restrains itself and waits until the network layer passes the next packet.
- The acknowledgment is attached to the outgoing data frame (using the ack field in the frame header).
- The acknowledgment gets a free ride on the next outgoing data frame.
- The technique of temporarily delaying outgoing acknowledgments so that they can be hooked onto the next outgoing data frame is known as ***piggybacking***.
- The principal advantage of using piggybacking over having distinct acknowledgment frames is a better use of the available channel bandwidth.

# SLIDING WINDOW PROTOCOLS

- The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence numbers corresponding to the frames it is permitted to send.
- These frames are said to fall within the ***sending window***. Similarly, the receiver also maintains a ***receiving window*** corresponding to the set of frames it is permitted to accept.

# SLIDING WINDOW PROTOCOLS



# SLIDING WINDOW PROTOCOLS

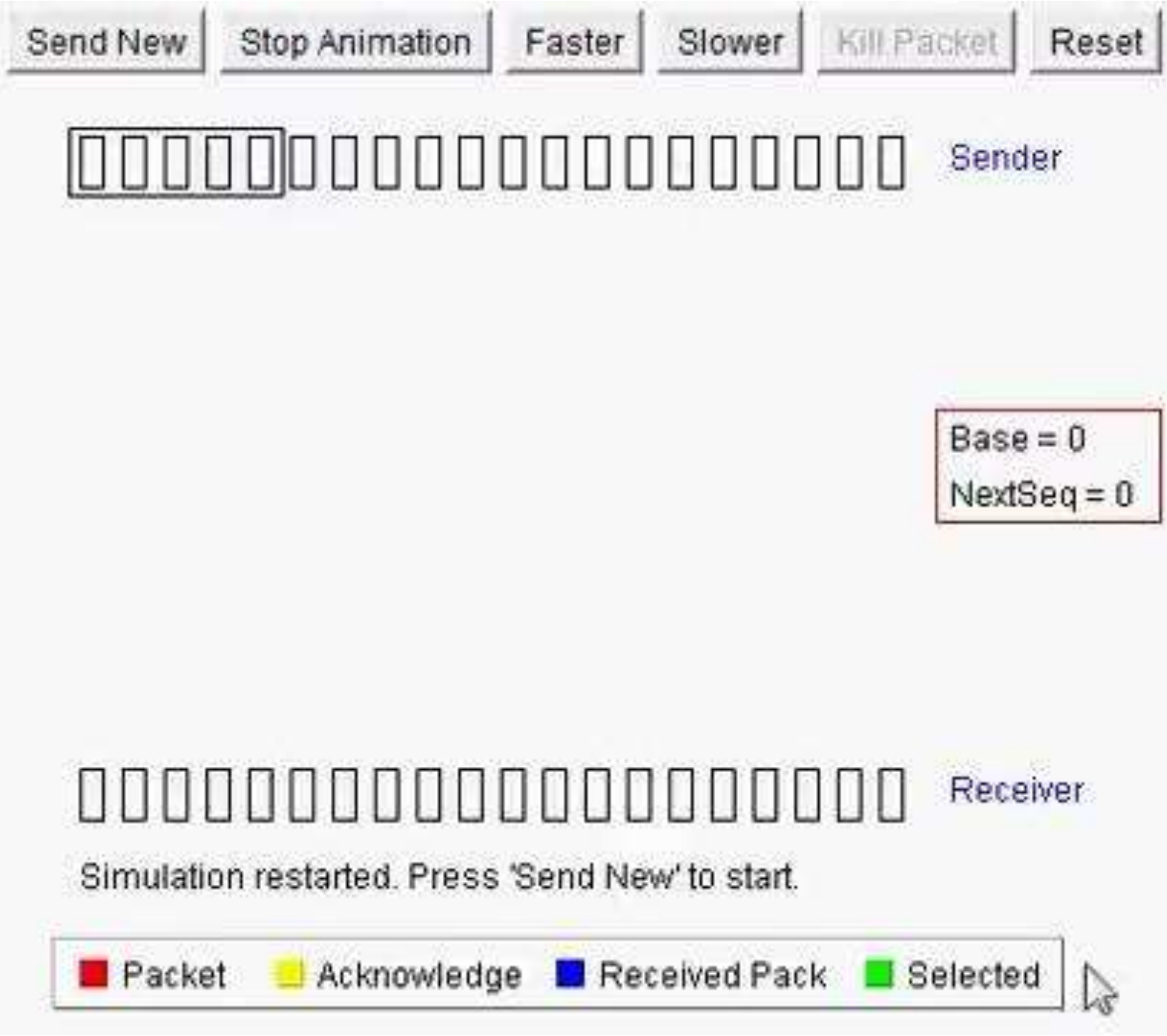
The sequence numbers within the ***sender's window*** represent ***frames that have been sent or can be sent*** but are as ***yet not acknowledged***. Whenever a new packet arrives from the network layer, it is given the next highest sequence number, and the upper edge of the window is advanced by one.

When an ***acknowledgment comes in***, the ***lower edge is advanced by one***.

# SLIDING WINDOW PROTOCOLS

**A Protocol Using Go-Back-N**

- Go-back-n is for the receiver simply to discard all subsequent frames, sending no acknowledgments for the discarded frames.
- If the sender's window fills up before the timer runs out, the pipeline will begin to empty.
- Eventually, the sender will time out and retransmit all unacknowledged frames in order, starting with the damaged or lost ones.
- This approach can waste a lot of bandwidth if the error rate is high.

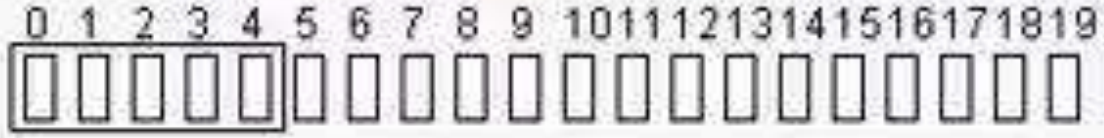

Send New
Stop Animation
Faster
Slower
Kill Packet
Reset
Sender
Base = 0
NextSeq = 0
Receiver
Simulation restarted. Press 'Send New' to start.
Packet
Acknowledge
Received Pack
Selected

# SLIDING WINDOW PROTOCOLS

**A Protocol Using Selective Repeat**

- The selective repeat protocol is to allows the receiver to accept and buffer the frames following a damaged or lost one.
- Both the sender and the receiver maintain a window of outstanding and acceptable sequence numbers.

Send New | Stop Animation | Faster | Slower | Kill Packet/Ack | Reset




Sender (Send Window Size = 5)

base = 0

nextseqnum = 0


Receiver (Receiver Window Size = 5)

Packet | Received | Ack | Ack Received | Selected | Buffered

(S) - Action at Sender (R) - Action at Receiver

# ERROR DETECTION & CORRECTION

- Networks must be able to transfer data from one device to another with acceptable accuracy.
- The system must guarantee that the data received are identical to data to the data transmitted.
- Anytime the data are transmitted from one node to the next, they can get corrupted in passage.
- Many factors can alter one or more bits of a message.
- In DLL, if a frame is corrupted between the two nodes, it needs to be corrected before it continues its journey to other nodes.

# ERROR DETECTION & CORRECTION

*Types of errors:*

- When bits flow from one point to another, they are subject to unpredictable changes because of ***interference.***
- It can change the shape of the signal.

*Single bit error*

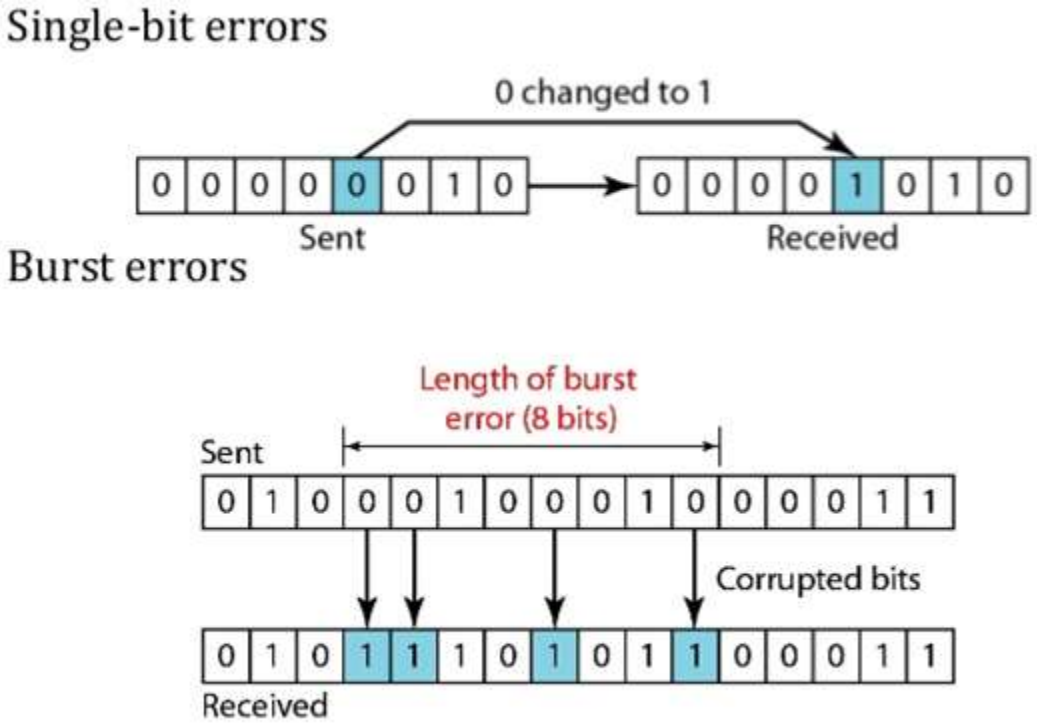
- Only 1 bit of a given data unit is changed from 0 to 1 or from 1 to 0.

*Burst error*

- *Two or more bits in the data unit have changed from 0 to 1 or 1 to 0.*

# ERROR DETECTION & CORRECTION

*Types of errors:*



# ERROR DETECTION & CORRECTION

*Redundancy:*

- The central concept in detecting or correcting errors is redundancy.
- To be able to detect or correct errors, we need to send some extra bits with our data.
- These redundant bits are added by the sender and removed by the receiver.
- Their presence allows the receiver to detect or correct corrupted bits.

# ERROR DETECTION & CORRECTION

## ***Coding***

Redundancy is achieved through various coding schemes.

We can divide coding schemes in to two broad categories:

✓Block Coding

✓Convolution Coding.

# ERROR DETECTION & CORRECTION

## ***Block Coding***

- Here, we divide our message in to blocks, each of ***k*** bits, called ***datawords.***
- We add ***r*** redundant bits to each block to make the length ***n = k + r***.
- The resulting ***n-bit*** words are called ***codewords.***

# ERROR DETECTION & CORRECTION

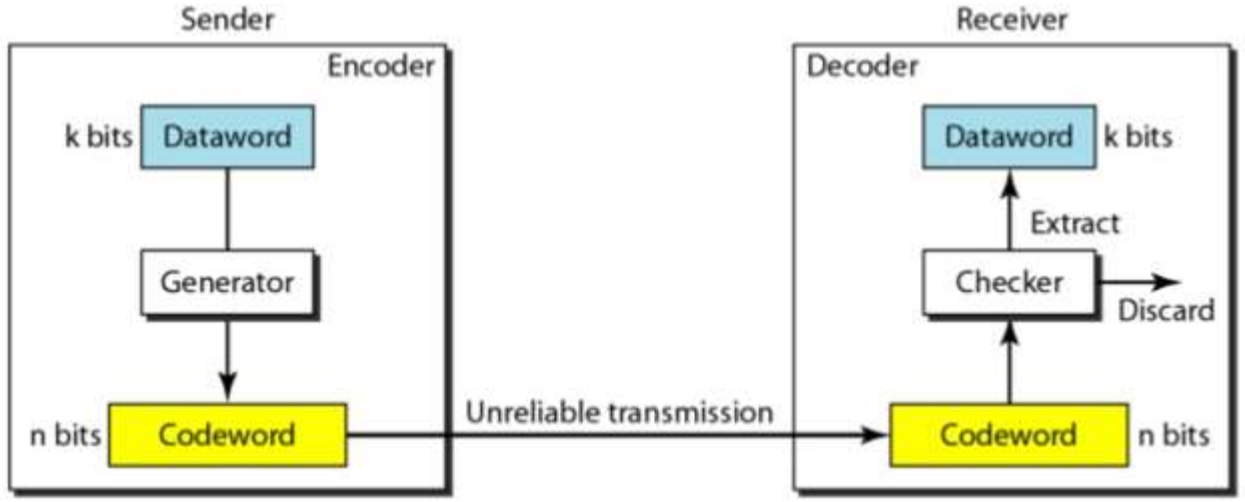
## ***Error Detection***

- If the following two conditions are met, the receiver can detect a change in the original codeword.
  1. The receiver has a list of valid codewords.
  2. The original codeword has changed to an invalid one.
- Each codeword  sent to the receiver may change during transmission.
- If the received codeword is the same as one of the valid codewords, the word is accepted.
- If the codeword is not valid, it is discarded.

# ERROR DETECTION & CORRECTION

## ***Error Detection***

- If the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected.



# ERROR DETECTION & CORRECTION

## *Error Detection*

Example:

Let us assume that k =2 and n = 3.

| Dataword | Codeword | Dataword | Codeword |
|---|---|---|---|
| 00 | 000 | 10 | 101 |
| 01 | 011 | 11 | 110 |

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

- The receiver receives 011. It is a valid Codeword. The receiver extracts the dataword 01 from it.
- The Codeword is corrupted during transmission, and 111 is received. This is not a valid Codeword and is discarded.
- The Codeword is corrupted during transmission, and 000 is received. This is a valid Codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

# ERROR DETECTION & CORRECTION

## *Error Detection*

**An error detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.**

# ERROR DETECTION & CORRECTION

## ***Hamming Distance***

The hamming distance between two words (of the same size) is the number of differences between the corresponding bits.

***It is the number of bits that are corrupted during transmission.***

If the Codeword 00000 is sent and 01101 is received, 3 bits are in error and the hamming distance between the two is,

***d(00000, 01101) = 3.***

***If the Hamming Distance between the sent and the received Codeword is not zero, the Codeword has been corrupted during transmission.***

# ERROR DETECTION & CORRECTION

## ***Hamming Distance***

The easiest way is to apply XOR operation and count the number of 1's from the result.

*Example:*

Find the Hamming Distance between two codewords.

1. d(000, 011) = 2 Because $(000 \oplus 011) = 011$ i.e. two 1's.
2. d(10101, 11110) = 3 Because $(10101 \oplus 11110) = 01011$ i.e. three 1's.

# ERROR DETECTION & CORRECTION

## ***Error Detection***

*Minimum Hamming distance for error detection*

- In a set of codewords, it is the smallest hamming distance between all possible pairs of codewords.

$$d_{min} = s + 1$$

*Where **s** is the errors occurred during transmission.*

If our system is to detect up to ***s*** errors, the minimum distance between the valid codes must be ***(s + 1).***

# ERROR DETECTION & CORRECTION

## ***Error Detection***

*Linear Block Codes*

Almost all block codes used today belong to a subset of block codes called **linear block codes**.

- A linear block code is a code, in which the exclusive OR (addition Modulo-2) of two valid codewords creates another valid codeword.

# ERROR DETECTION & CORRECTION

## ***Error Detection***

*Linear Block Codes*

1. Parity-Check Code:

   The code is a linear block code.

   A k bit dataword is changed to an n bit codeword, where n = k + 1. The extra bit is called a ***Parity Bit.***

   It is selected to make the total number of 1's in the codeword even.

# ERROR DETECTION & CORRECTION

## ***Error Detection***

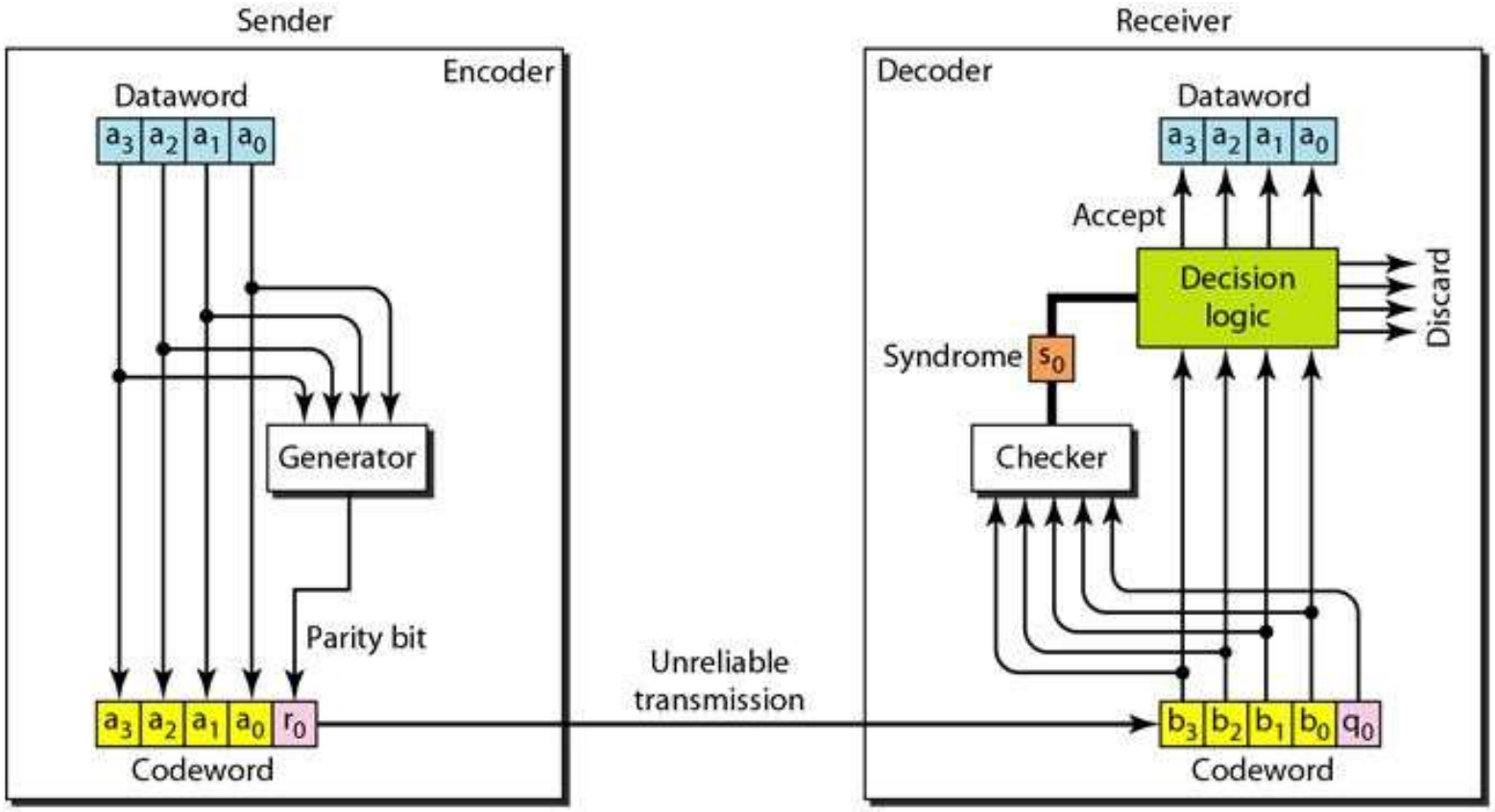
*Linear Block Codes*

1. Parity-Check Code:

✓***Calculation is done in modular arithmetic.***

| Decimal value | Data Block | Parity bit | Code word |
|---|---|---|---|
| 0 | 0000 | 0 | 00000 |
| 1 | 0001 | 1 | 00011 |
| 2 | 0010 | 1 | 00101 |
| 3 | 0011 | 0 | 00110 |
| 4 | 0100 | 1 | 01001 |
| 5 | 0101 | 0 | 01010 |
| 6 | 0110 | 0 | 01100 |
| 7 | 0111 | 1 | 01111 |
| 8 | 1000 | 1 | 10001 |
| 9 | 1001 | 0 | 10010 |
| 10 | 1010 | 0 | 10100 |
| 11 | 1011 | 1 | 10111 |
| 12 | 1100 | 0 | 11000 |
| 13 | 1101 | 1 | 11011 |
| 14 | 1110 | 1 | 11101 |
| 15 | 1111 | 0 | 11110 |

# ERROR DETECTION & CORRECTION



# ERROR DETECTION & CORRECTION

## ***Cyclic Codes***

- Special linear block codes with one extra property.
- If the codeword is cyclically shifted (rotated), the result is another codeword.

# ERROR DETECTION & CORRECTION

## *Cyclic Codes*

### Cyclic Redundancy Check

We can create cyclic codes *to correct errors*.

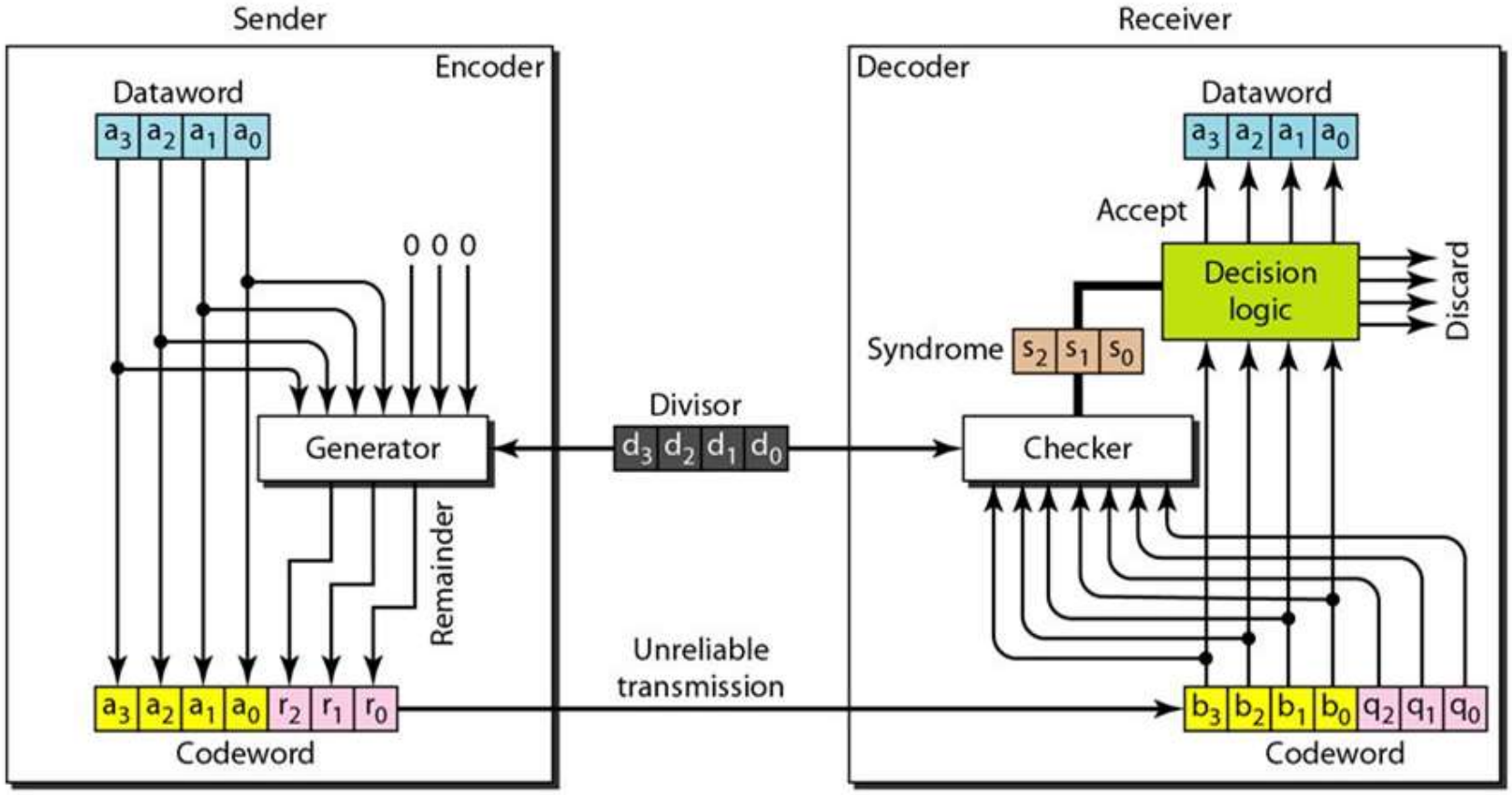
A subset of cyclic codes are called CRC, used in LANs and WANs.

| *Dataword* | *Codeword* | *Dataword* | *Codeword* |
|---|---|---|---|
| 0000 | 0000000 | 1000 | 1000101 |
| 0001 | 0001011 | 1001 | 1001110 |
| 0010 | 0010110 | 1010 | 1010011 |
| 0011 | 0011101 | 1011 | 1011000 |
| 0100 | 0100111 | 1100 | 1100010 |
| 0101 | 0101100 | 1101 | 1101001 |
| 0110 | 0110001 | 1110 | 1110100 |
| 0111 | 0111010 | 1111 | 1111111 |

# ERROR DETECTION & CORRECTION

## *Cyclic Codes*

### Cyclic Redundancy Check



# ERROR DETECTION & CORRECTION

## *Cyclic Codes*

### Cyclic Redundancy Check

- Encoder:-
  - It has k datawords; n codewords;
  - The size of the dataword is augmented by adding n – k 0's to the RHS of the word.
  - The n bit result is fed in to the generator.
  - The generator uses a divisor of size n – k + 1, predefined and agreed upon.
  - It divides the dataword by the divisor(Modulo 2).
  - The quotient of the division is discarded.
  - The remainder is appended to the dataword to create the codeword.

# ERROR DETECTION & CORRECTION

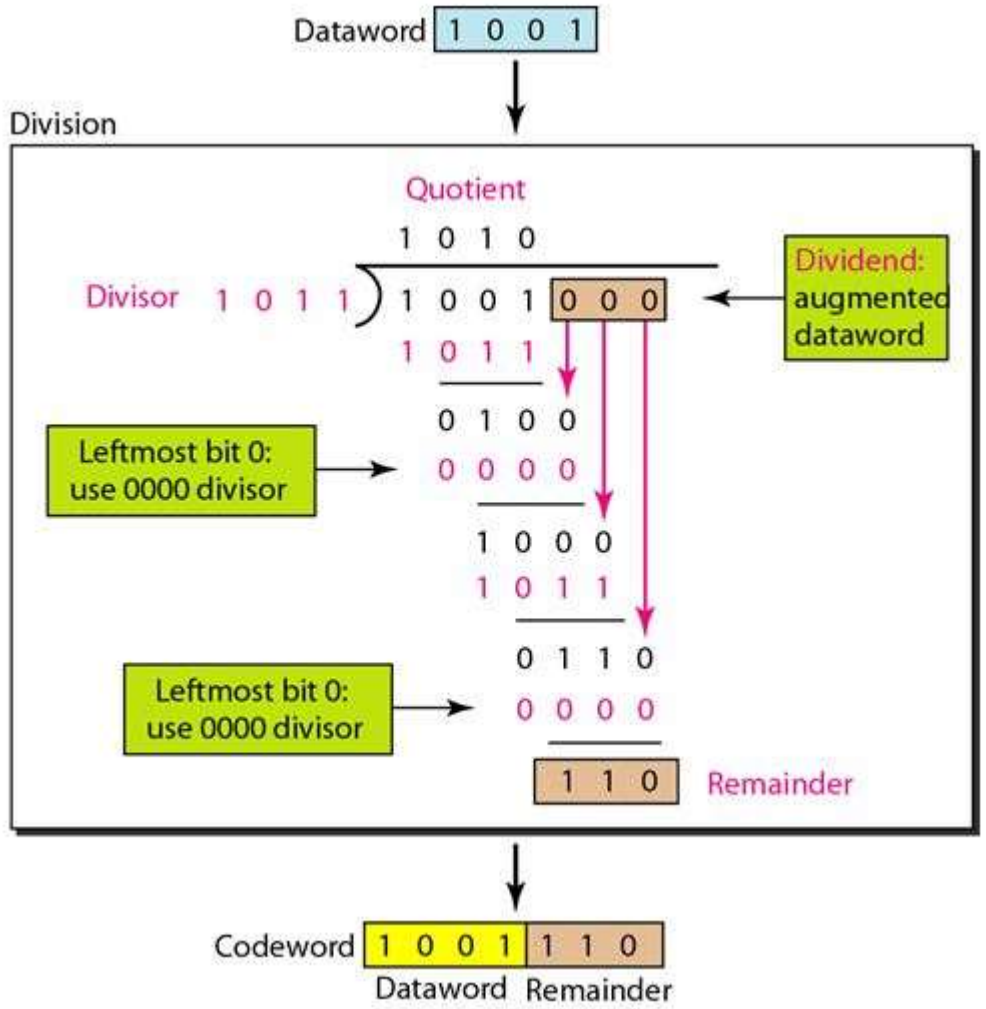
## *Cyclic Codes*

### Cyclic Redundancy Check

- Decoder:-
  - It receives the codeword.
  - A copy of all n bits is fed to the checker, which is a replica of the generator.
  - The remainder produced by the checker is a syndrome of n – k bits, which is fed to the decision logic analyzer.
  - It has a simple function; If the syndrome bits are all 0’s, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error).
  - Otherwise, the 4 bits are discarded (error).

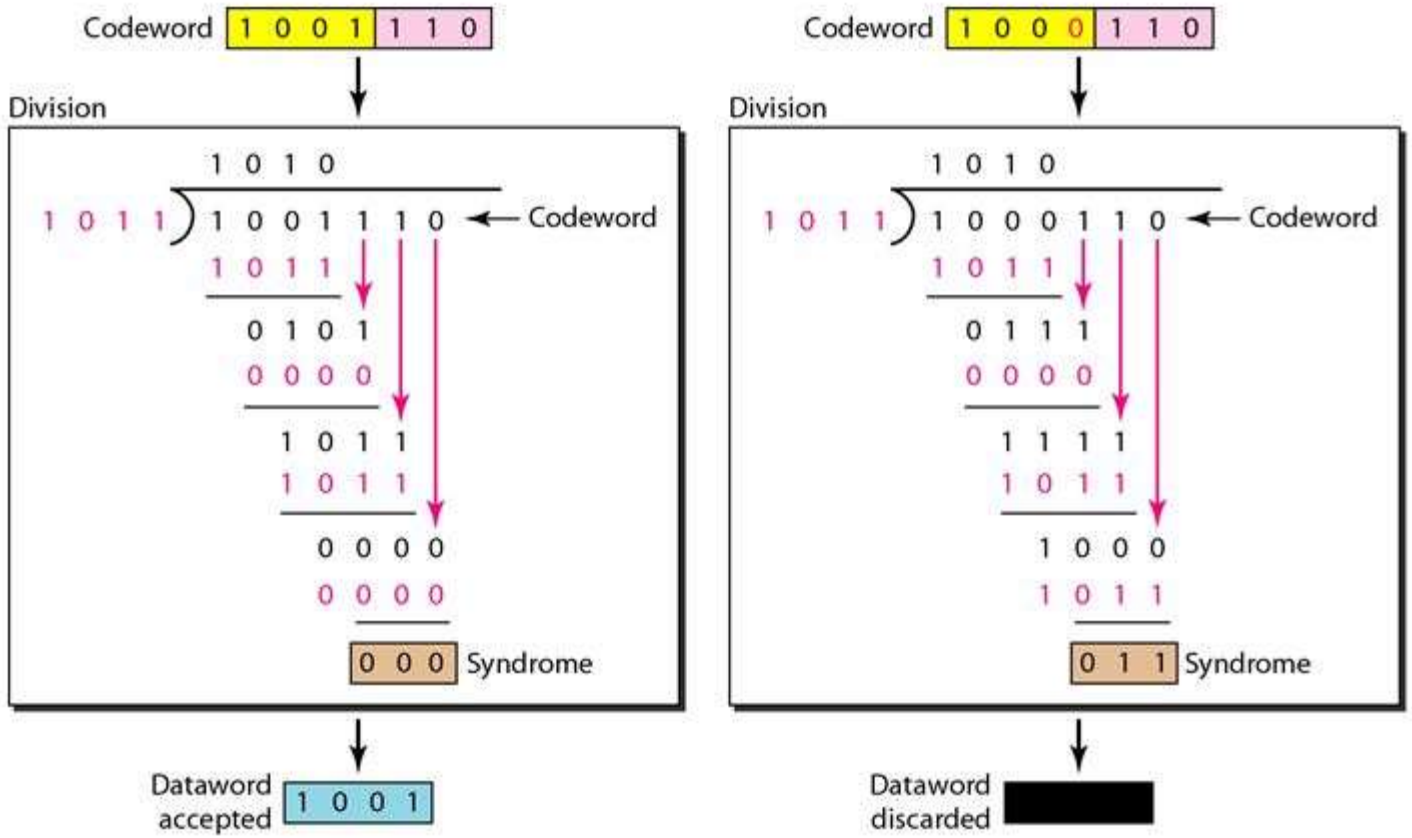
# ERROR DETECTION & CORRECTION

## Cyclic Redundancy Check : Encoder



# ERROR DETECTION & CORRECTION

## Cyclic Redundancy Check : Decoder



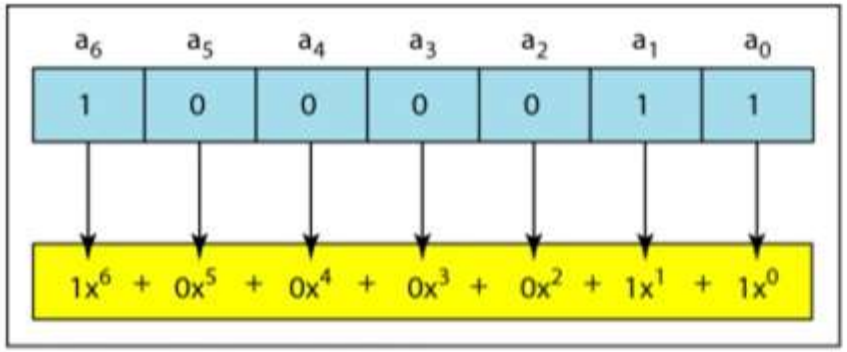
# ERROR DETECTION & CORRECTION

***Polynomials***

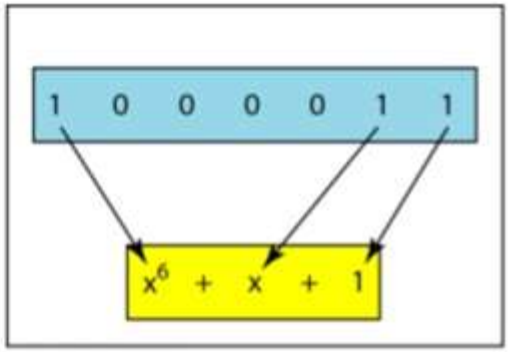
A better way to understand Cyclic codes.

A pattern of 0's and 1's can be represented as polynomial with coefficient of 0 and 1.

- The power of each term shows the position of the bit.
- The coefficient shows the value of the bit.



a. Binary pattern and polynomial



b. Short form

# ERROR DETECTION & CORRECTION

***Polynomials***

Degree of a polynomial : Highest power in the polynomial.

*Note: The degree of a polynomial is 1 less than the number of bits in the pattern.*

- Addition/Subtraction:-

  Example:

  - $(x^5 + x^4 + x^2) + (x^6 + x^4 + x^2) = x^6 + x^5$

- Multiplication and Division:-

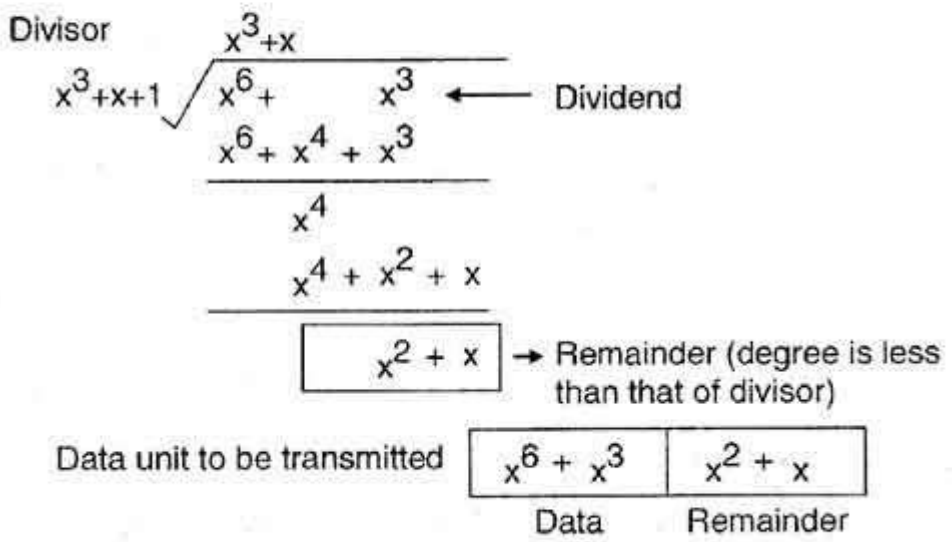
  Example:

  - $x^3 * x^4 = x^7$ & $x^5/x^2 = x^3$

# ERROR DETECTION & CORRECTION

## CRC Division using Polynomials

| | | |
|---|---|---|
| Data | 1001 | $x^3 + 1$ |
| Division (polynomial generator) | 1011 | $x^3 + x + 1$ |




CRC division using polynomial

# MEDIA ACCESS CONTROL (MAC)

## THE CHANNEL ALLOCATION PROBLEM

- **Continuous or Slotted Time.**

  Time may be assumed continuous.

  In this case frame transmission can begin at any instant.

  Alternatively, time may be slotted or divided into discrete intervals (called slots).

  Frame transmissions must then begin at the start of a slot.

  A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful transmission, or a collision, respectively.

# MEDIA ACCESS CONTROL (MAC)

## THE CHANNEL ALLOCATION PROBLEM

- **Carrier Sense or No Carrier Sense**.

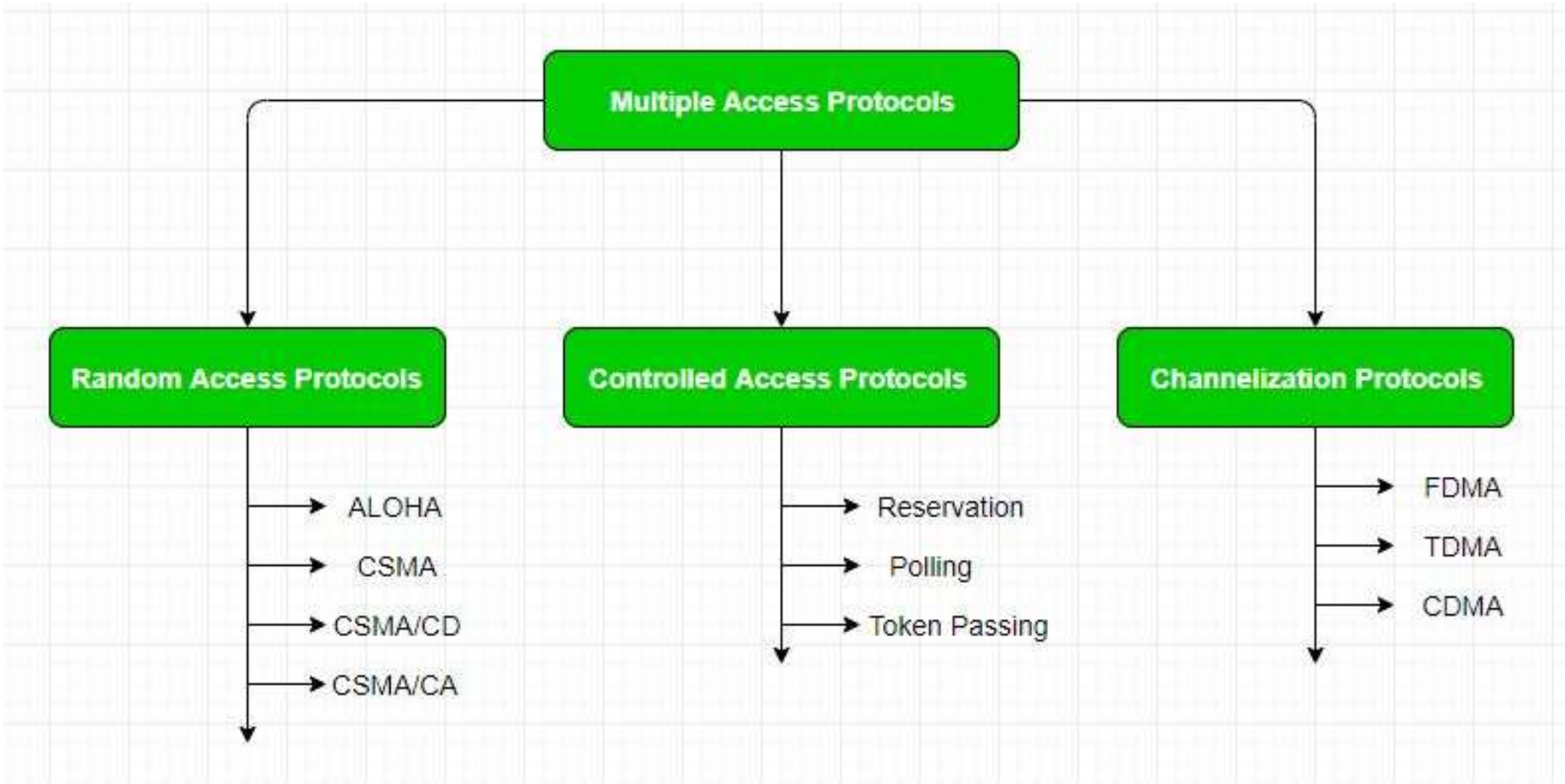
  With the carrier sense assumption, stations can tell if the channel is in use before trying to use it.

  No station will attempt to use the channel while it is sensed as busy.

  If there is no carrier sense, stations cannot sense the channel before trying to use it. They just go ahead and transmit.

  Only later can they determine whether the transmission was successful.

# MEDIA ACCESS CONTROL (MAC)



# Random Access Protocols

- ✓There is no scheduled time for a station to transmit.
- ✓Transmission is random among the stations.
- ✓This is why its called **Random Access.**
- ✓No rules 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.

# Random Access Protocols

If more than one station tries to send, there is an access conflict – ***COLLISION***

The frames will be either destroyed or modified.

To avoid access conflict:

- When can the station access the medium?
- What can the medium do if it is busy?
- How can the medium determine the success or the failure of the transmission?
- What can station do if there is an access conflict?

# Random Access Protocols

**ALOHA**

- The earliest random access method.
- The medium is shared between the stations.
- When one station sends data, another station may attempt to do so at the same time.
- There may be collision and become garbled.

# Random Access Protocols

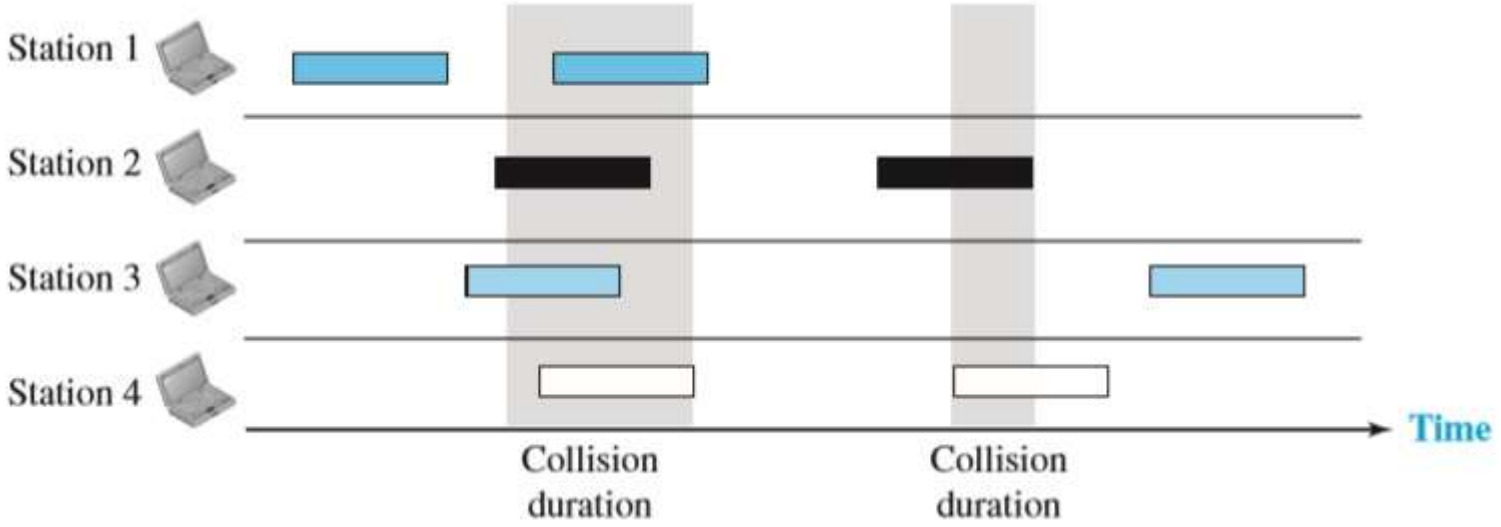
## Pure ALOHA

- Original ALOHA is called as pure ALOHA.
- Each station sends a frame whenever it has a frame to send.
- There is only one channel to share.
- There is a probability of collision between frames from different stations.

# Random Access Protocols

## Pure ALOHA

- Here each station sends two frames.
- Total eight frames on the medium.
- Some will collide because multiple frames are in contention for the shared channel.
- Only two frames got successfully transferred.



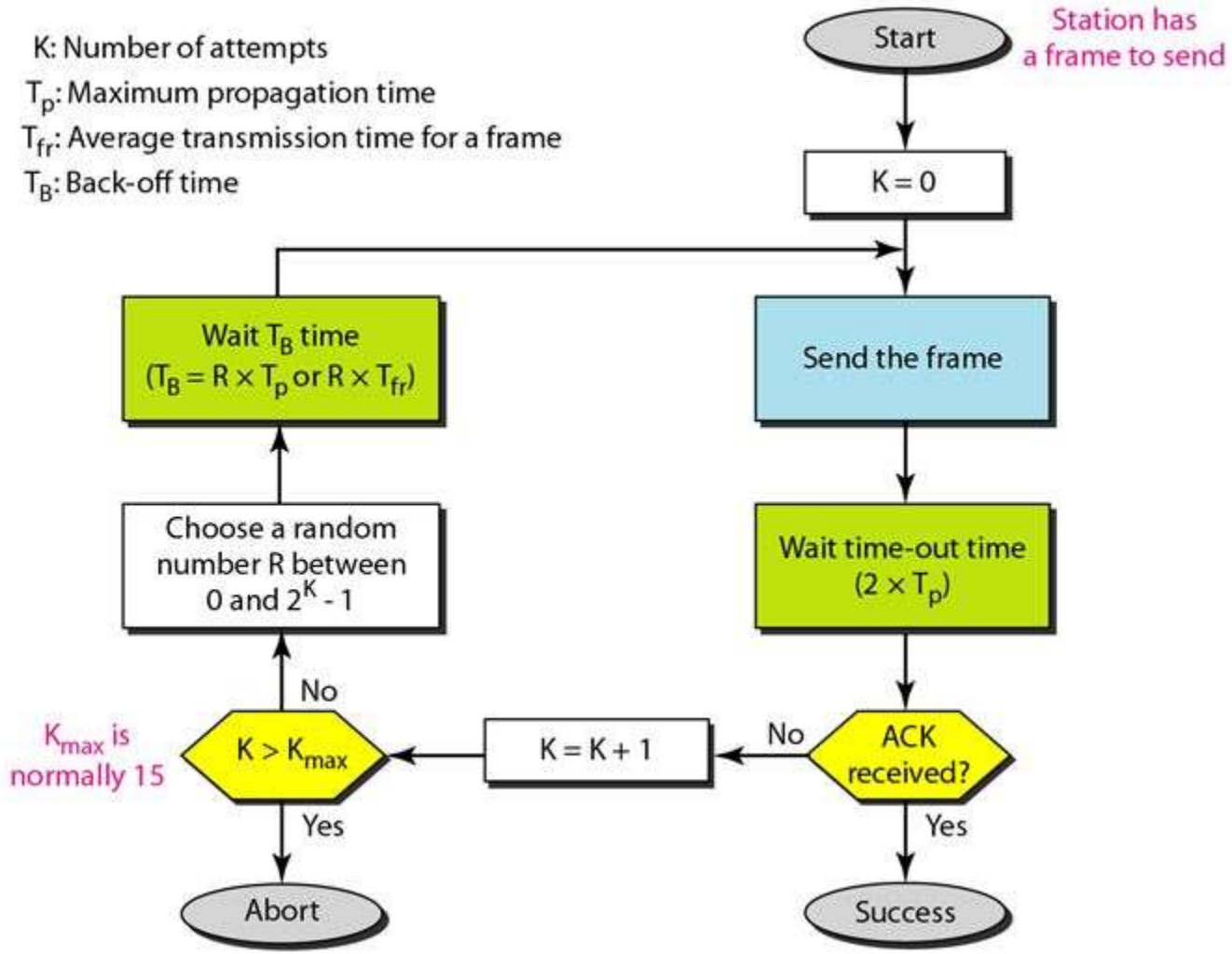
# Random Access Protocols

## Pure ALOHA

- Need to resend the frames that have been destroyed during transmission.
- The protocol relies on acknowledgments from the receiver.
- 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.
- When the time-out period passes, each station waits a random amount of time before resending its frame. ( ***backoff time*** $T_B$ )
- This will help to avoid more collisions.
- After a maximum number of retransmission attempts $K_{max}$ a station must give up and try later.

# Random Access Protocols

## Pure ALOHA



# Random Access Protocols

## Pure ALOHA

*Vulnerable Time*

The length of time in which there is a possibility of collision.

Assume that the stations send fixed-length frames with each frame taking $T_{fr}$ seconds to send.

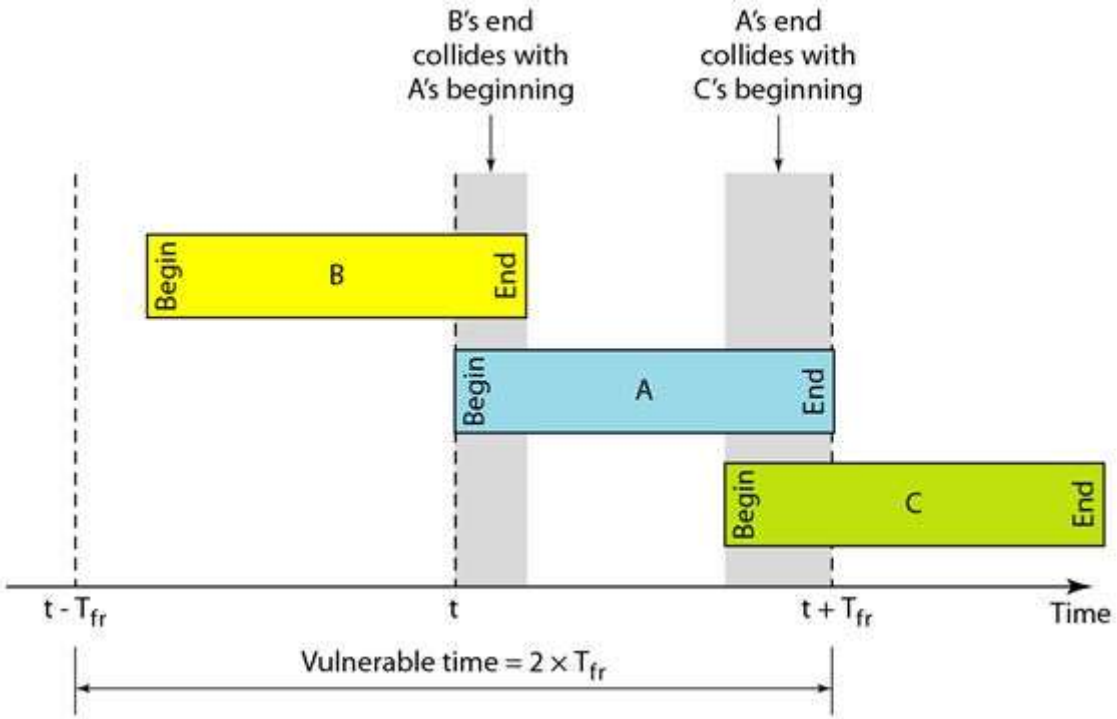
# Random Access Protocols

## Pure ALOHA

*Vulnerable Time*

Station A starts to send a frame at time ***t***. Now imagine station B has started to end its frame after $t - T_{fr}$ .

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



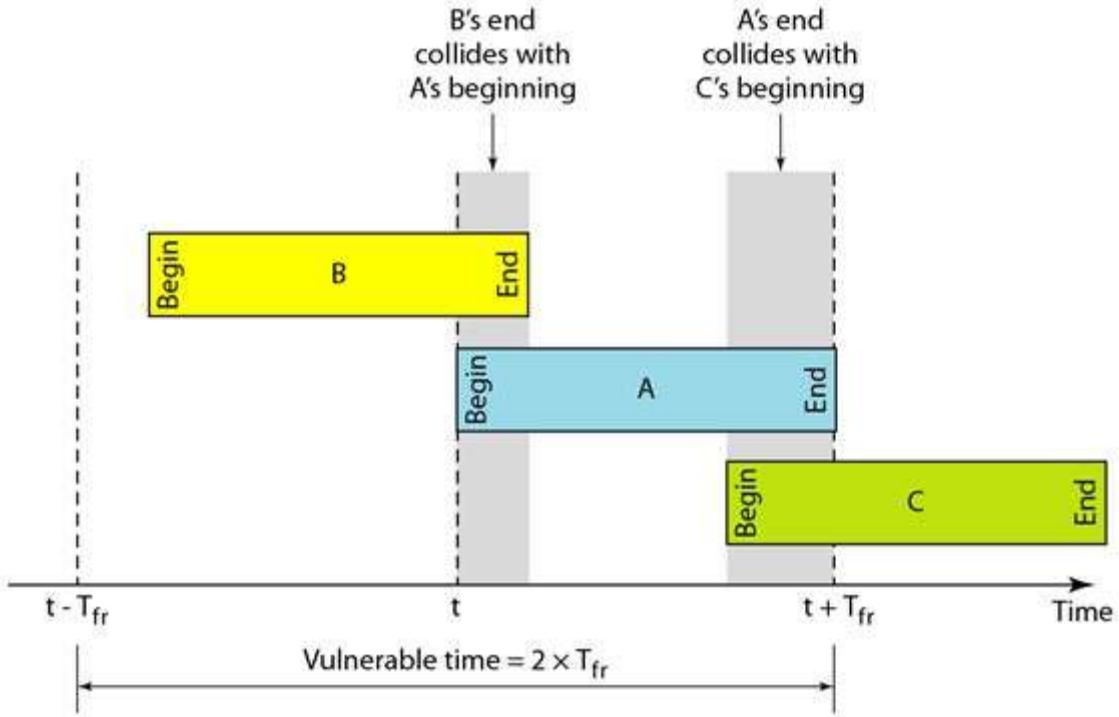
# Random Access Protocols

## Pure ALOHA

*Vulnerable Time*

Suppose that station C starts to send a frame before time $t + T_{fr}$ .

There is also a collision between frames from A and C.



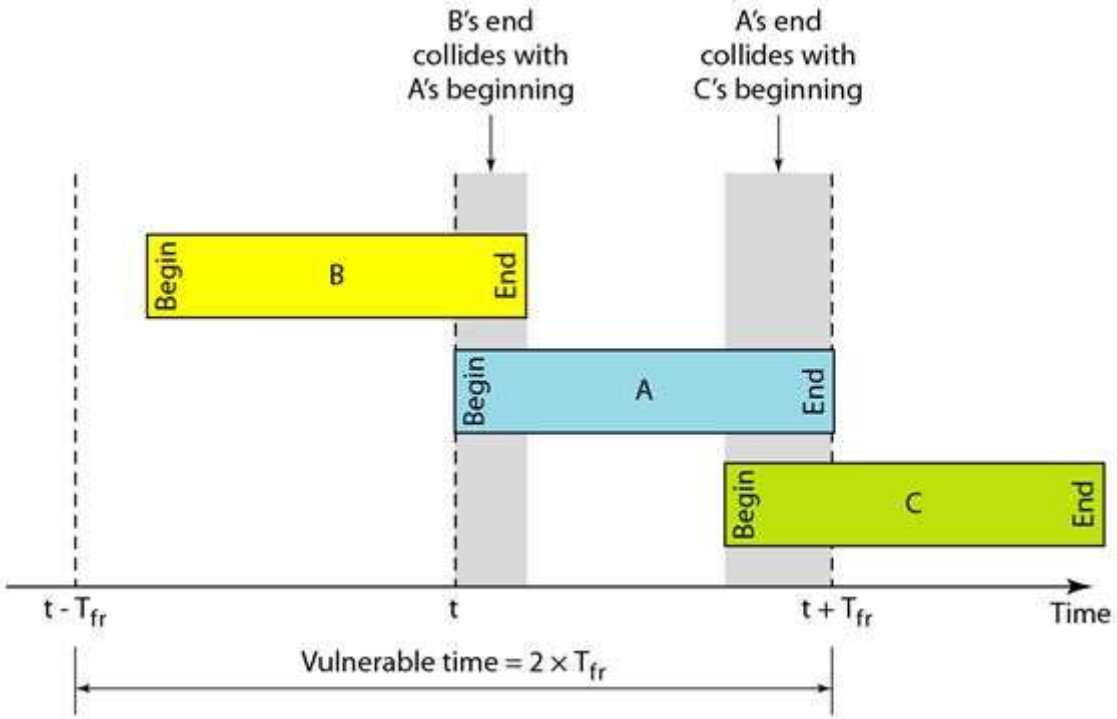
# Random Access Protocols

## Pure ALOHA

*Vulnerable Time*

Vulnerable time during which a collision may occur in pure ALOHA is two times the frame transmission time.

***Pure ALOHA vulnerable time = 2 x $T_{fr}$***



# Random Access Protocols

## Pure ALOHA

*Throughput*

Assume G – Average number of frames generated during one frame transmission time.

Average number of successfully transmitted frames is,

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

The maximum throughput $S_{max} = 0.184$, for $G = ½$

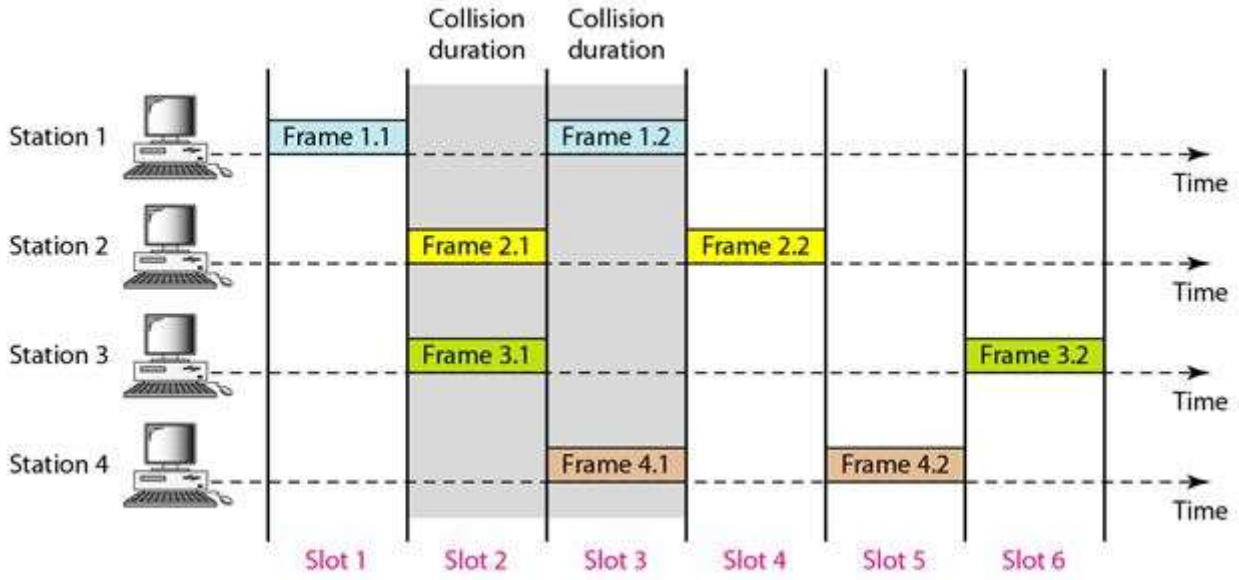
# Random Access Protocols

## Slotted ALOHA

In pure ALOHA there was no rule for WHEN THE STATION CAN SEND.

Slotted ALOHA was invented to improve the quality of Pure ALOHA.

Here, we divide the time into slots of $T_{fr}$ seconds and force the station to send at the beginning of the slot.



# Random Access Protocols

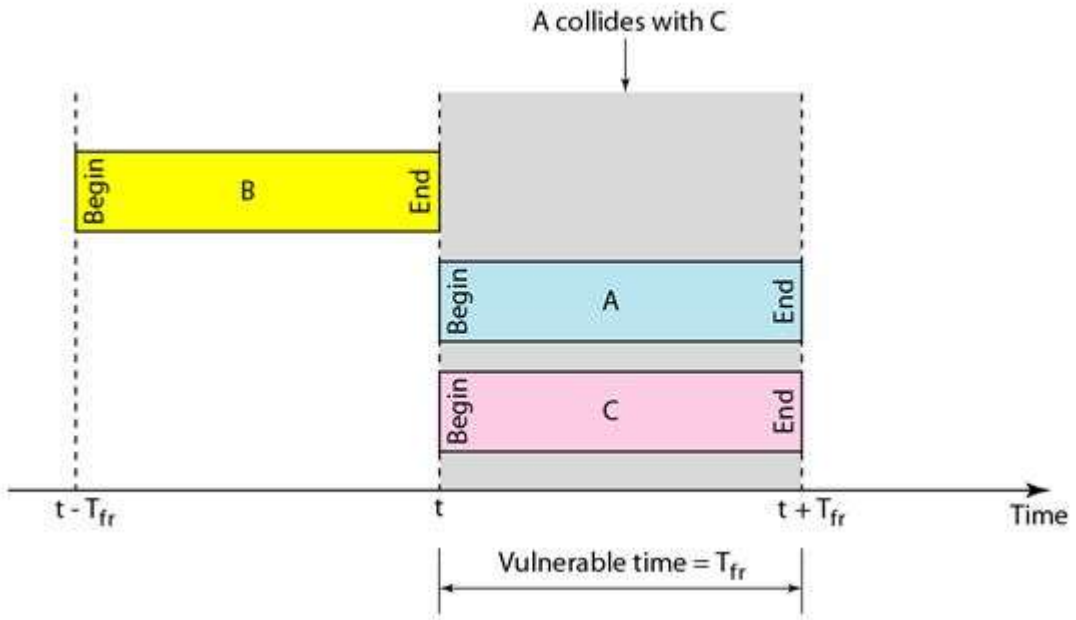
## Slotted ALOHA

Station was only allowed to send at the beginning of the synchronised time slot.

If station misses this moment, it can only send at the beginning of next slot.

Problem comes, when two stations send at the beginning of the same time slot.

However the ' equal to $T_{fr}$



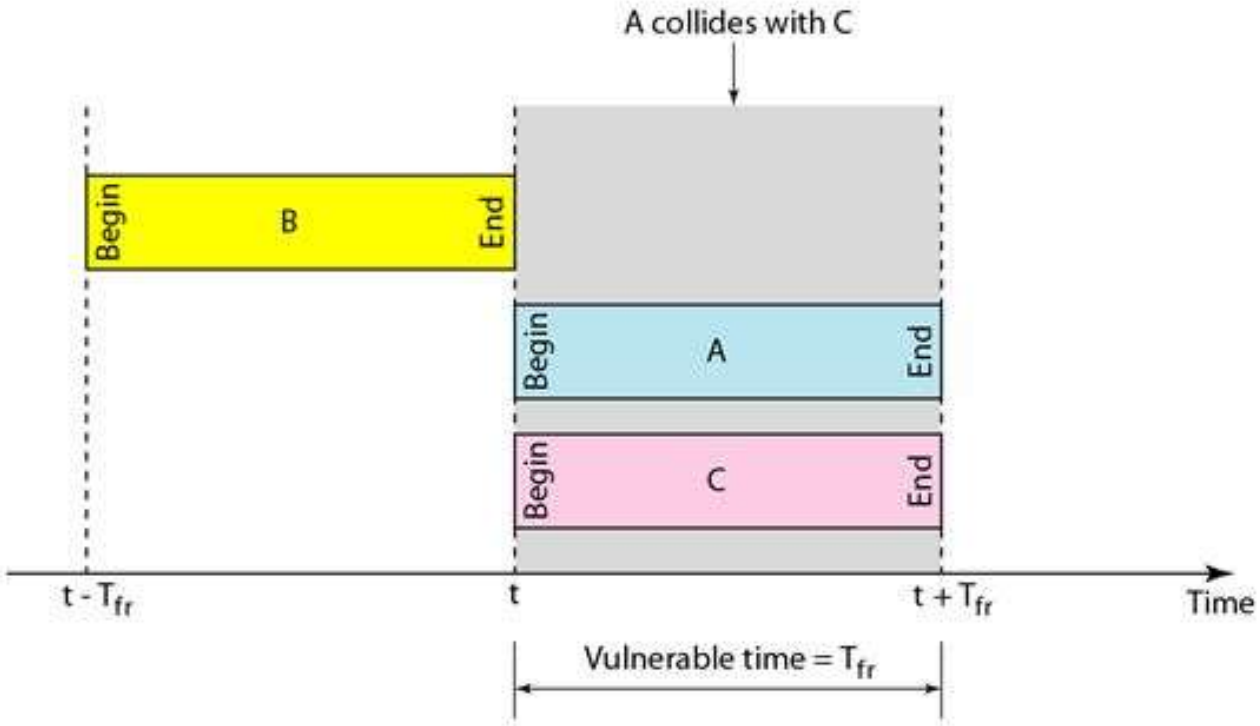
# Random Access Protocols

## Slotted ALOHA

***Vulnerable Time = $T_{fr}$***

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

The maximum throughput $S_{max}$ = 0.368, for G = 1.



# Random Access Protocols

**CSMA (Carrier Sense Multiple Access)**

To minimise the chance of collision and therefore increase the performance, CSMA is developed.

The chance of collision can be reduced if a station senses the medium before trying to use it.

Each station first listen to the medium (or check the state of the medium)

# Random Access Protocols

**CSMA (Carrier Sense Multiple Access)**

**“Sense before transmit” or “Listen before talk”**

CSMA can reduce the possibility of collision, but it cannot eliminate it.

Stations are connected to a shared channel.

Collision may occur due to propagation delay.

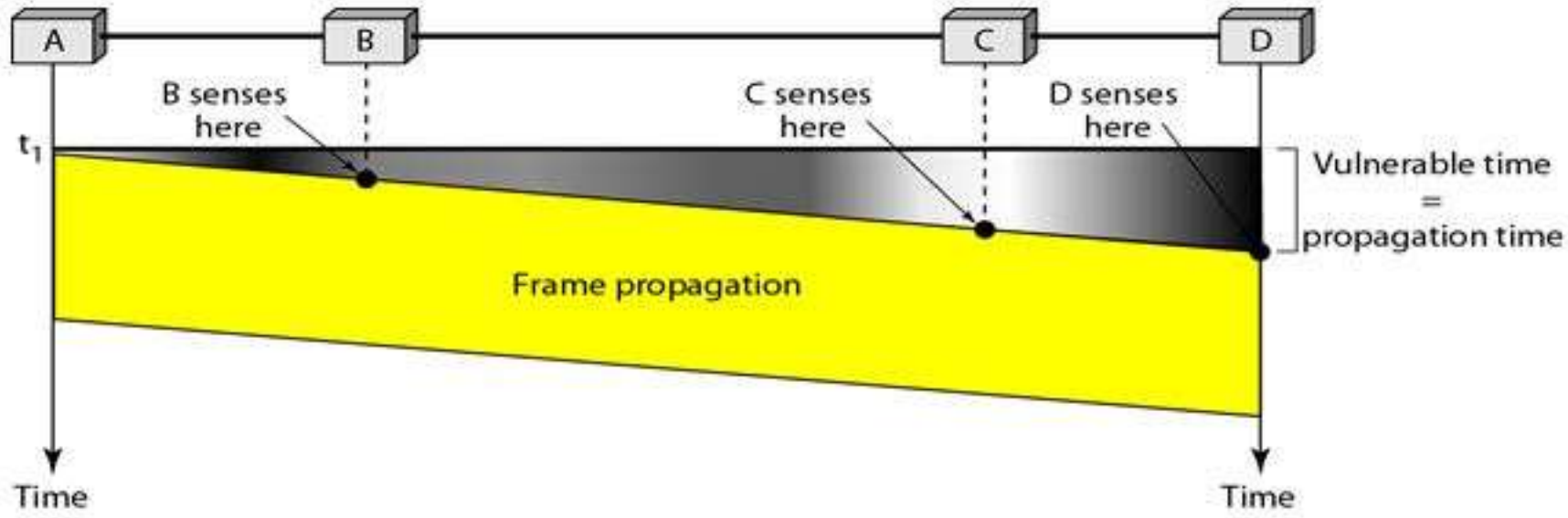
# Random Access Protocols

**CSMA (Carrier Sense Multiple Access)**

*Vulnerable Time:*

The leftmost station sends a frame at time $t_1$ which reaches the rightmost station at time $t_1 + T_p$ .

Gray area shows the vulnerable area in time and space.



# Random Access Protocols

**CSMA (Carrier Sense Multiple Access)**

***Persistence methods***

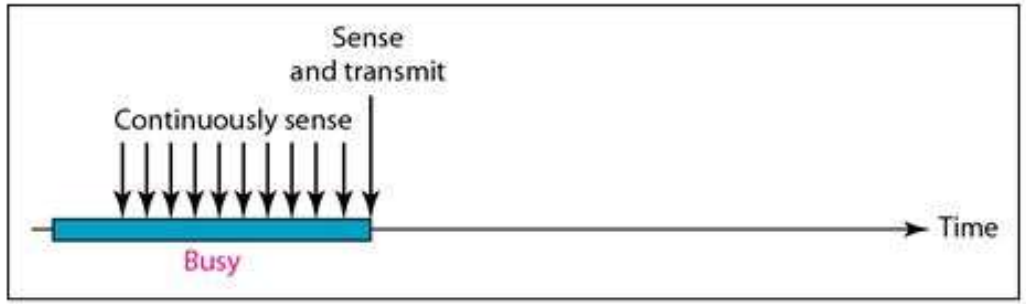
What should the station do, If the channel is busy?

What should the station do, If the channel is idle?

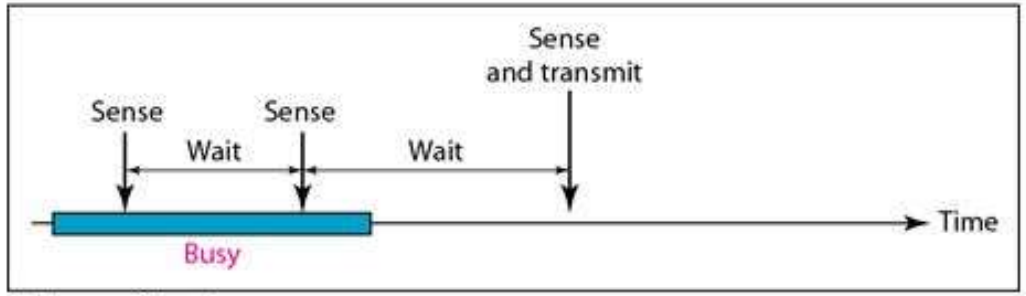
There are 3 methods,

1. 1-persistent CSMA
2. Non-persistent CSMA
3. p-persistent CSMA

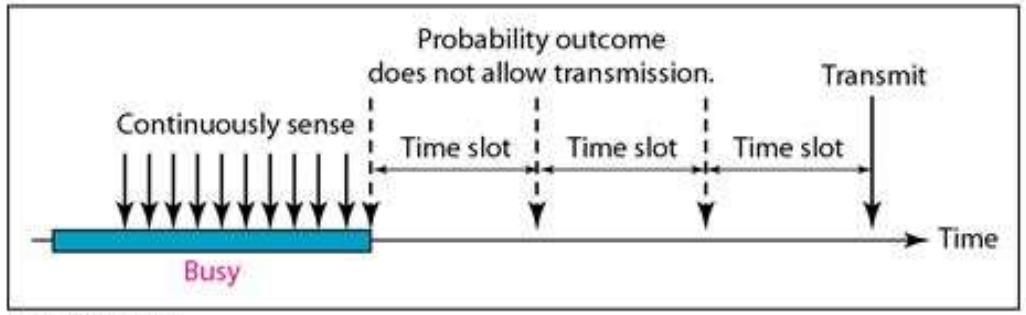
# ***Persistence methods***




a. 1-persistent




b. Nonpersistent




c. p-persistent

# Random Access Protocols

**CSMA (Carrier Sense Multiple Access)**

***Persistence methods***

1-persistent CSMA:

- Simple and straight forward
- After the station finds the medium idle, it sends its frame immediately.
- Highest chance of collision.
- Every other station will do the same.

# Random Access Protocols

**CSMA (Carrier Sense Multiple Access)**

***Persistence methods***

non-persistent CSMA:

- Senses the line first.
- If idle, sends immediately.
- If not idle, it waits a random amount of time, then senses the line again.
- Two or more stations will not wait the same amount of time, or it is very unlikely.

# Random Access Protocols

**CSMA (Carrier Sense Multiple Access)**

***Persistence methods***

p-persistent CSMA:

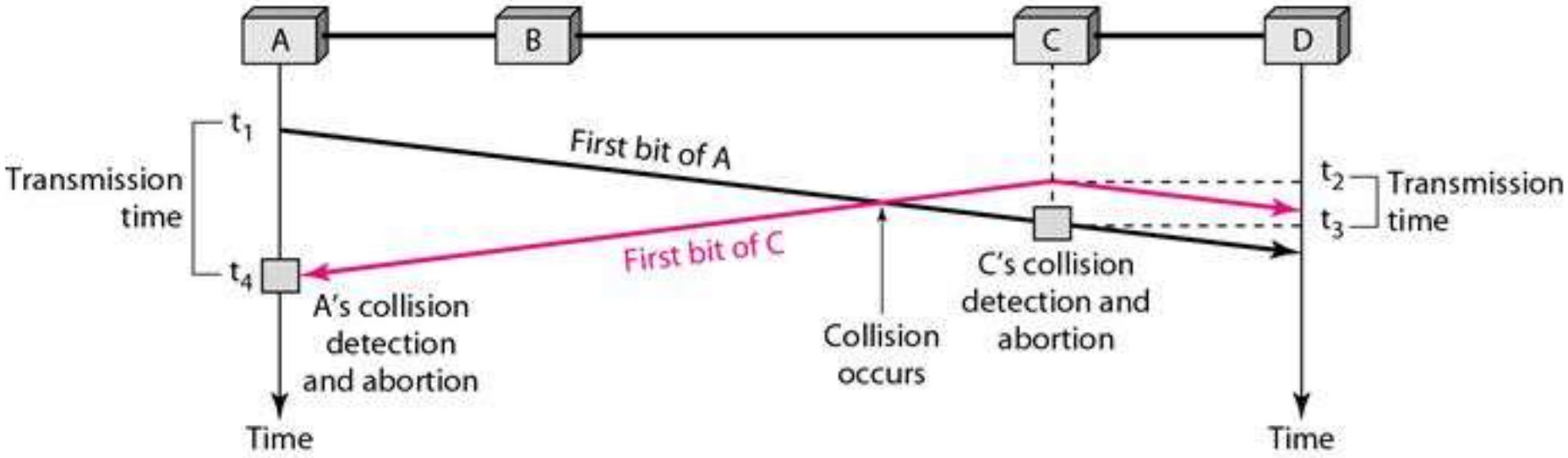
- If the channel has time slots with a slot duration, equal to or greater than the maximum propagation time.
- After the station finds the line idle:
  - The station sends its frame with probability p.
  - With probability ***q = 1 – p ,*** the station waits for the beginning of the next time slot and 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 back-off procedure.

# Random Access Protocols

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

It arguments the algorithm to handle the collision.

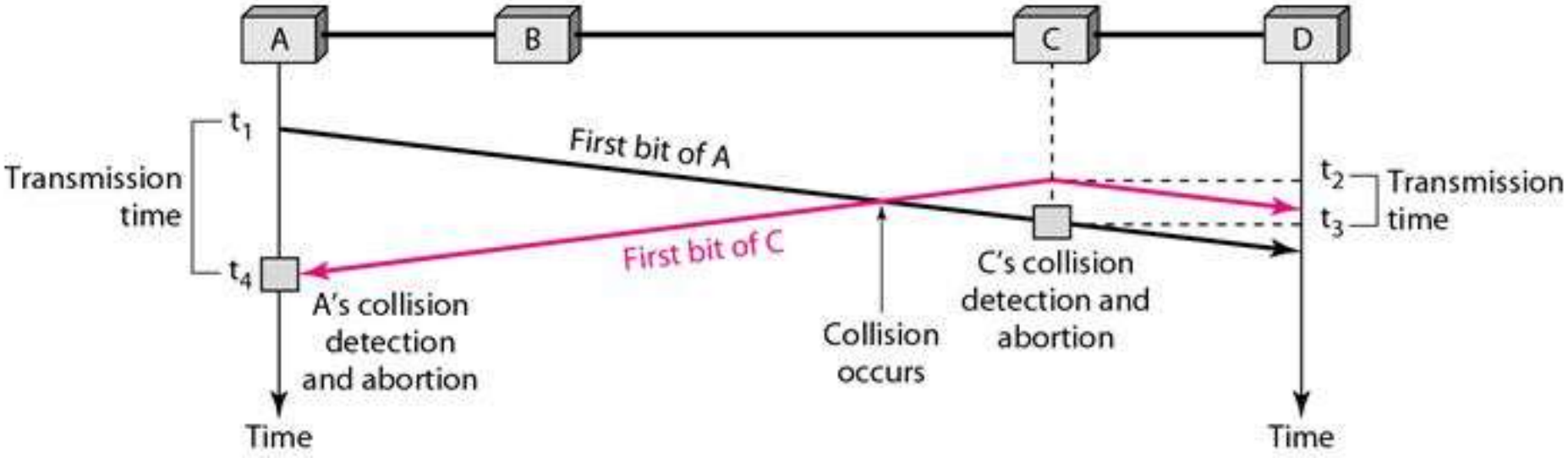
1. At time $t_1$ station A has executed its persistence procedure and starts sending the bits of its frame.
2. At time $t_2$ Station C has not yet sensed the first bit sent by A.



# Random Access Protocols

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

3. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and the right.
4. The collision occur some time after $t_2$ .
5. Station C detects a collision at time $t_3$ when it receives the first bit of A's frame



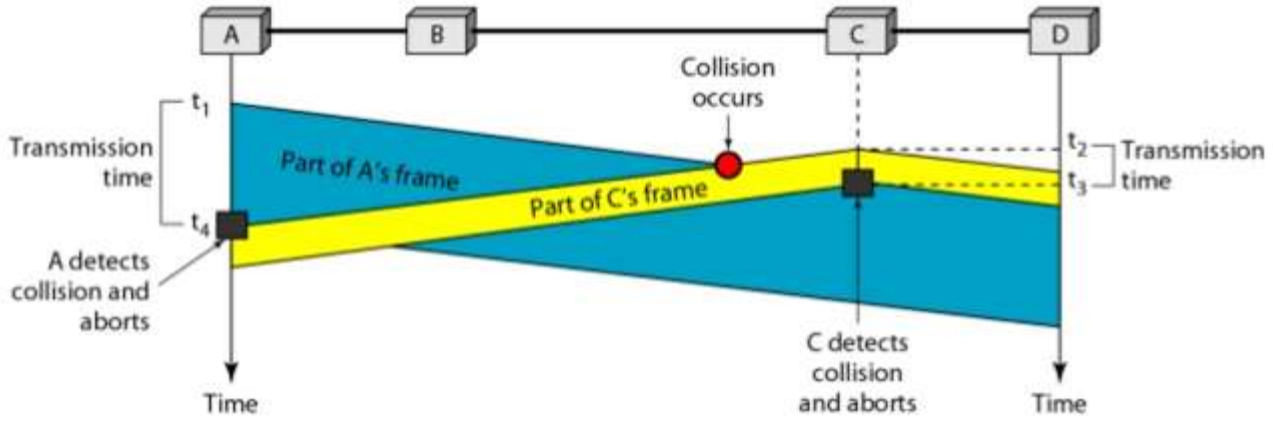
# Random Access Protocols

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

6. Station C immediately aborts transmission.
7. Station A detects collision at time $t_4$ when it receives the first bit of C's frame. It also immediately aborts transmission.

Here, A transmits for the duration $t_4 - t_1$ .

C transmits for the duration $t_3 - t_2$ .



# Random Access Protocols

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

Invented for wireless networks.

Three strategies for avoiding collision:

1. The inter-frame space.
2. The contention window.
3. Acknowledgments.

# Random Access Protocols

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

***Inter-frame Space (IFS):-***

When an idle channel is found, the station does not send immediately.

It waits for a period of time called the inter-frame space (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.***

After waiting an IFS time, if the channel is still idle, the station can send.

# Random Access Protocols

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

***Contention window:-***

It 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 slots changes according to the binary exponential back off algorithm.

***It is set to one slot at the first time, and then doubles each time the station cannot detect an idle channel after the IFS time***.

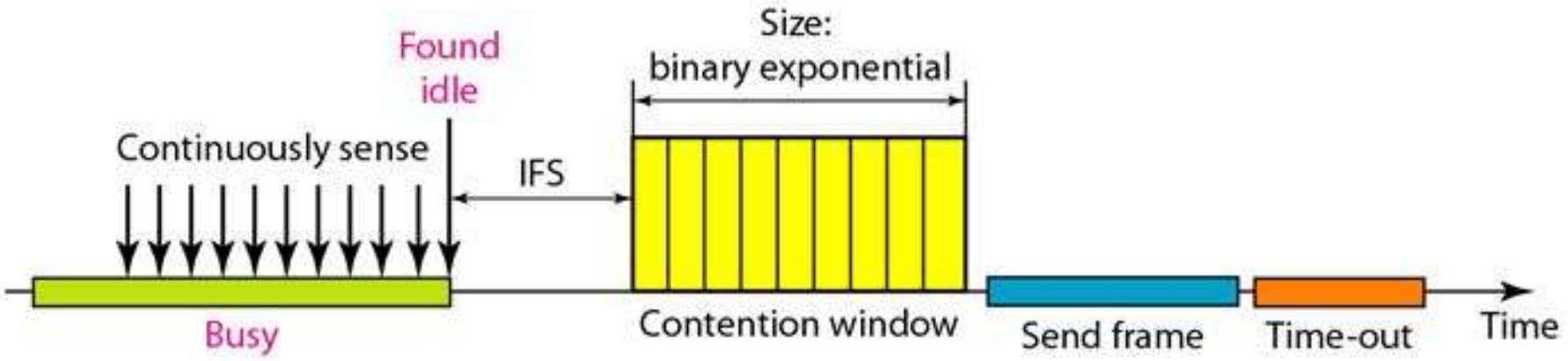
# Random Access Protocols

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

***Acknowledgment:-***

With all the precautions, still there are chances for collision.

So with the above two, positive acknowledgment and the time-out timer can guarantee that a frame has received successfully.



# Controlled Access

The stations consults one another to find which station has the right to send.

A station cannot send unless it has been authorised by other stations.

***Reservation:-***

- A station needs to make a reservation before sending data.
- Time is divided into intervals.
- In each interval, a reservation frame precedes the data frames sent in that interval.
- If there are N stations in the system, there are exactly N number of reservation mini slots in the reservation frame.
- Each mini slot belongs to a station.

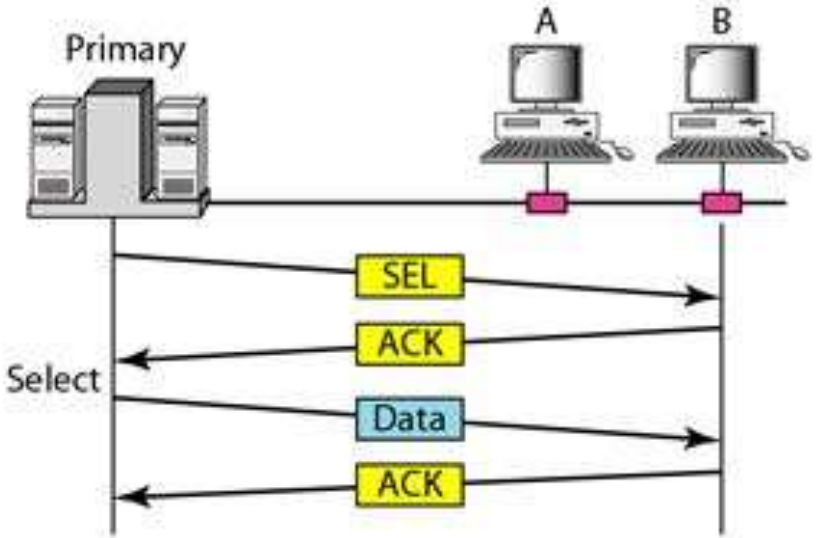
# Controlled Access

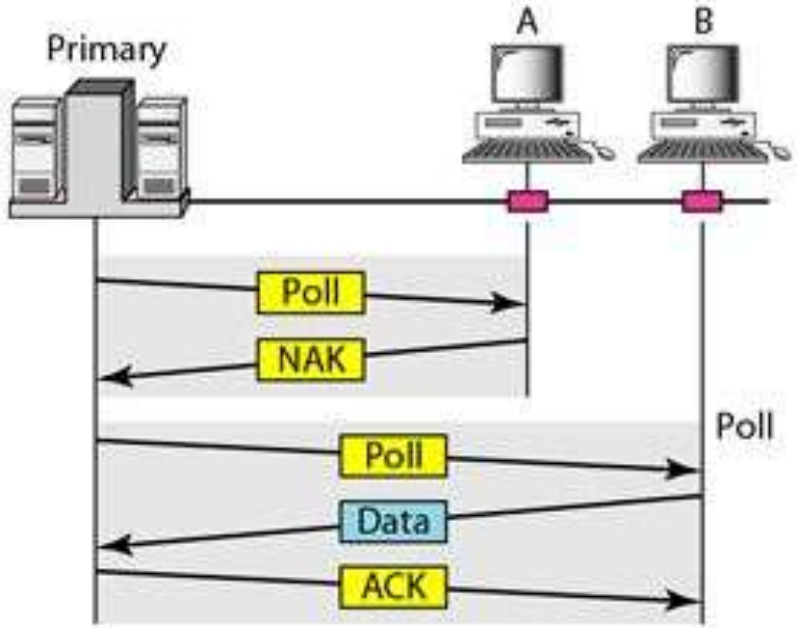
***Polling:-***

- One device is designated as a primary station and the other devices are secondary stations.
- The primary device controls the link.
- The secondary device follow its instructions.
- Primary device will decide which device is allowed to use the channel at a given time.
- Primary device : Initiator of a session.
- This method uses poll and select functions to prevent collisions.

# Controlled Access

***Polling:-***



# Controlled Access

***Token Passing:-***

- Stations in a network are organized in a logical ring.
- For each station, there is a predecessor and successor.
- A special packet called token circulates through the ring.

***The possession of the token gives the station right to access the channel and send its data.***

# Channelization

Also called as *Channel partition*.

It is a multiple access method in which the available *bandwidth of a link is shared in time, frequency or through code*, among different stations.

Three channelization protocols:

- FDMA
- TDMA
- CDMA

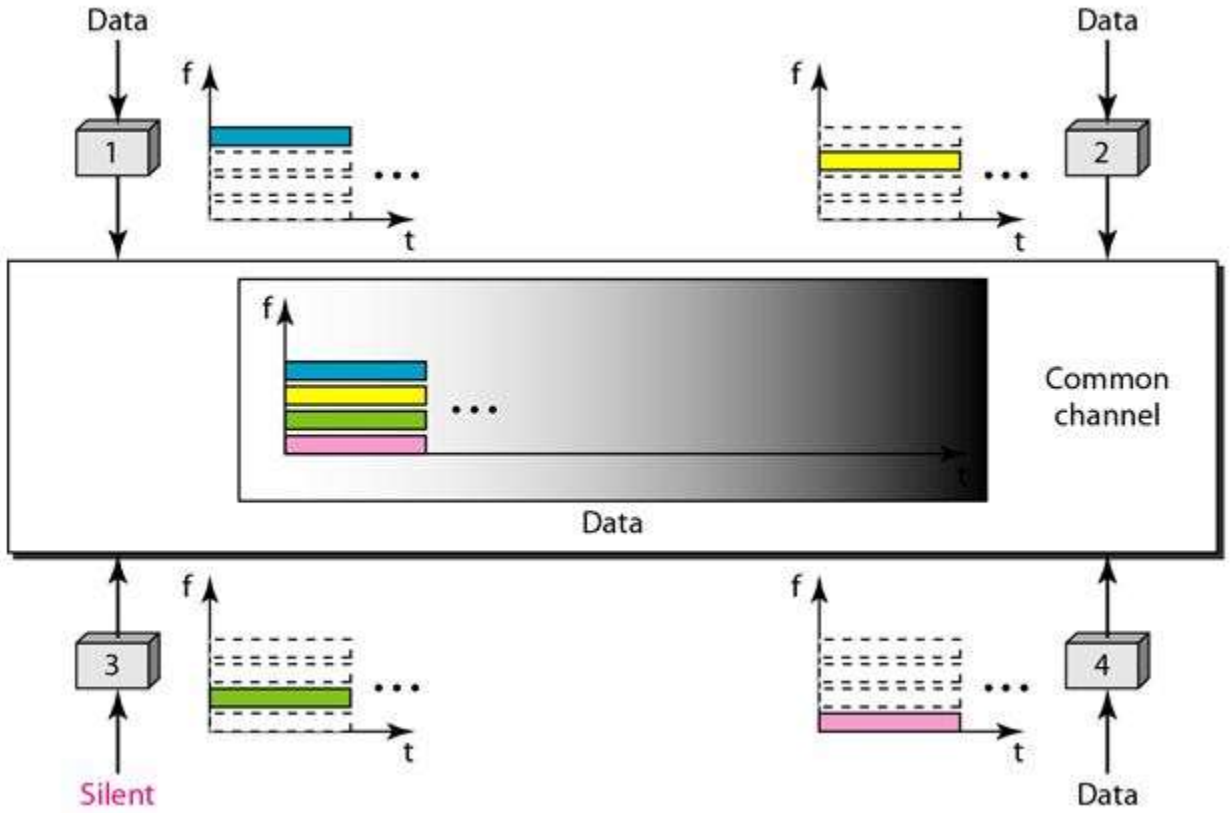
# Channelization

## FDMA (Frequency Division Multiple Access)

- Here, the available bandwidth is divided into *frequency bands*.
- Each station is allocated  a band to send data.
- Each band is reserved for a specific station, and it belongs to that station all the time.
- Each station also uses a *bandpass filter* to confine the transmitter frequencies.
- To prevent station interferences, the allocated bands are separated from one another by small *guard bands*.

# Channelization

## FDMA (Frequency Division Multiple Access)



# Channelization

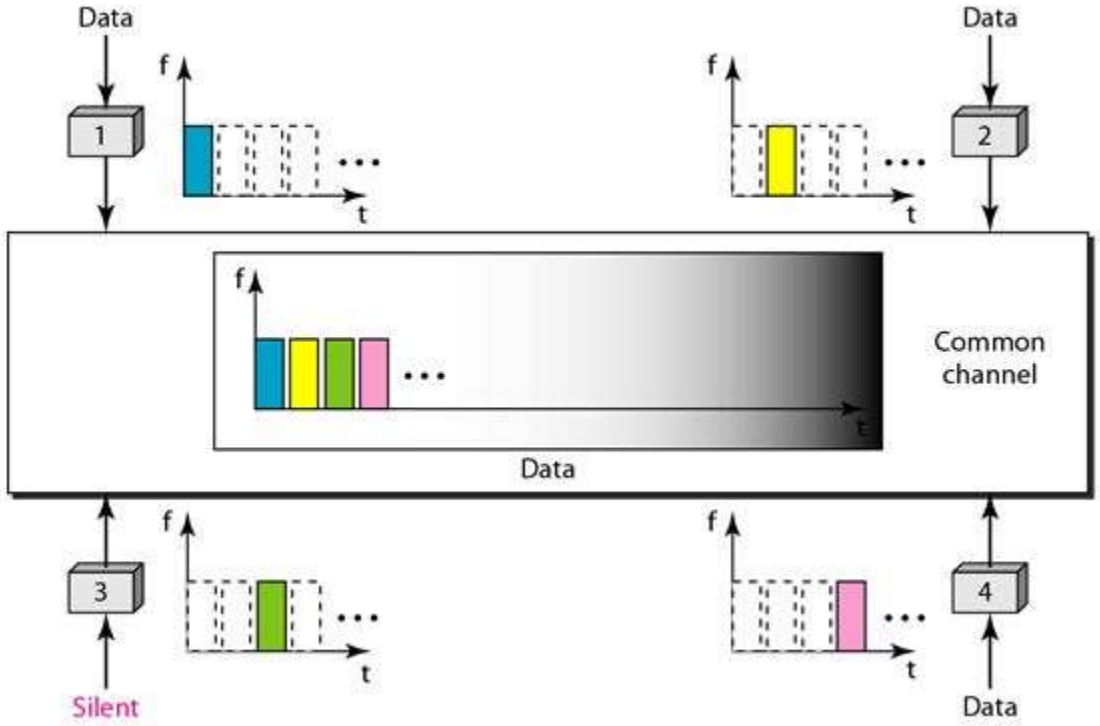
## FDMA (Frequency Division Multiple Access)

- FDMA and FDM conceptually seem similar, there are differences between them.
- FDM is a physical layer technique that *combines the loads from low bandwidth channels* and *transmits them by using a high bandwidth channel*.
- The MUX modulates the signals, combines them and creates a bandpass signal.
- While FDMA, is an access method in the DLL.
- The DLL in each station tells its physical layer to make a bandpass signal from the data passed to it.
- There is no physical MUX at the physical layer.

# Channelization

## TDMA (Time Division Multiple Access)

- 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.



# Channelization

**TDMA (Time Division Multiple Access)**

- Each station needs to know the beginning of its slot and the location of its slot.
- This may be difficult because of propagation delays introduced in the system, if the stations are spread over a large area.
- To compensate delays, we have *guard times*.

# Channelization

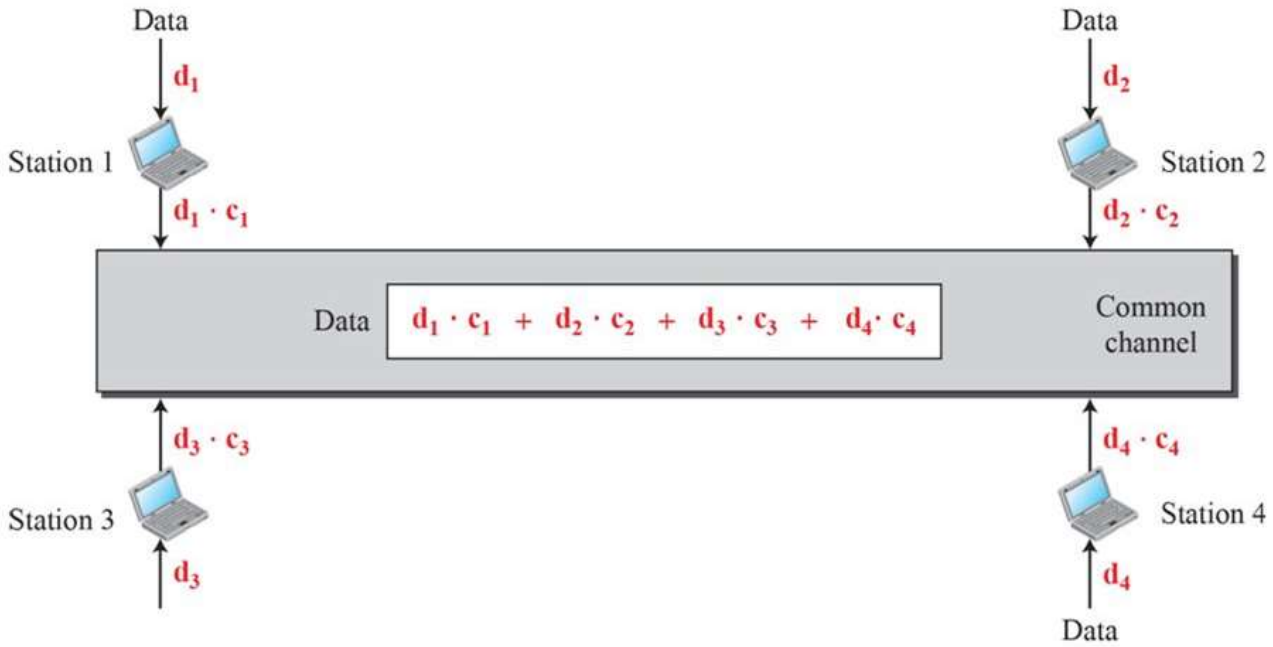
**CDMA (Code Division Multiple Access)**

- CDMA differs from FDMA, in that only one channel occupies the entire bandwidth of the link.
- CDMA differs from TDMA, in that all stations can send the data simultaneously; *there is no time sharing.*
- It means communication with different codes.

*Example: Inside the same room, communication takes place in different languages.*

# Channelization

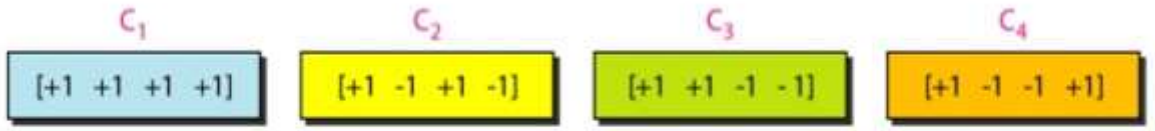
## CDMA (Code Division Multiple Access)



# Channelization

## CDMA (Code Division Multiple Access)

- CDMA is based on coding theory.
- Each station is assigned with a code, which is a sequence of numbers called *chips.*



# Channelization

## CDMA (Code Division Multiple Access)

*Data Representation:-*

- We follow these rules for encoding.
- If a station needs to send a 0 bit, it encodes it as -1;
- If it needs to send a 1 bit. It encodes it as +1.
- When a station is idle, it sends no signal, which is interpreted as a 0.