UNIT – 2

➢ Data-link Layer Introduction

Data Link Layer is second layer of OSI Layered Model. This layer is one of the most complicated layers and has complex functionalities and liabilities. Data link layer hides the details of underlying hardware and represents itself to upper layer as the medium to communicate.

Data link layer works between two hosts which are directly connected in some sense. This direct connection could be point to point or broadcast. Systems on broadcast network are said to be on same link. The work of data link layer tends to get more complex when it is dealing with multiple hosts on single collision domain.

Data link layer is responsible for converting data stream to signals bit by bit and to send that over the underlying hardware. At the receiving end, Data link layer picks up data from hardware which are in the form of electrical signals, assembles them in a recognizable frame format, and hands over to upper layer.

Data link layer has two sub-layers:

• Logical Link Control: It deals with protocols, flow-control, and error control

• Media Access Control: It deals with actual control of media

➢ Functionality of Data-link Layer

Data link layer does many tasks on behalf of upper layer. These are:

• **Framing**

Data-link layer takes packets from Network Layer and encapsulates them into Frames. Then, it sends each frame bit-by-bit on the hardware. At receiver' end, data link layer picks up signals from hardware and assembles them into frames.

• **Addressing**

Data-link layer provides layer-2 hardware addressing mechanism. Hardware address is assumed to be unique on the link. It is encoded into hardware at the time of manufacturing.

• **Synchronization**

When data frames are sent on the link, both machines must be synchronized in order to transfer to take place.

• **Error Control**

Sometimes signals may have encountered problem in transition and the bits are flipped. These errors are detected and attempted to recover actual data bits. It also provides error reporting mechanism to the sender.

• **Flow Control**

Stations on same link may have different speed or capacity. Data-link layer ensures flow control that enables both machines to exchange data on same speed.

• **Multi-Access**

When host on the shared link tries to transfer the data, it has a high probability of collision. Data-link layer provides mechanism such as CSMA/CD to equip capability of accessing a shared media among multiple Systems.

➢ Data Link Layer Error Detection & Correction

There are many reasons such as noise, cross-talk etc., which may help data to get corrupted during transmission. The upper layers work on some generalized view of network architecture and are not aware of actual hardware data processing. Hence, the upper layers expect error-free transmission between the systems. Most of the applications would not function expectedly if they receive erroneous data. Applications such as voice and video may not be that affected and with some errors they may still function well.

Data-link layer uses some error control mechanism to ensure that frames (data bit streams) are transmitted with certain level of accuracy. But to understand how errors is controlled, it is essential to know what types of errors may occur.

❖ Types of Errors

There may be three types of errors:

• **Single bit error**

In a frame, there is only one bit, anywhere though, which is corrupt.

• **Multiple bits error**

Frame is received with more than one bits in corrupted state.

• **Burst error**

Frame contains more than1 consecutive bits corrupted.

Error control mechanism may involve two possible ways:

• Error detection

• Error correction

❖ Error Detection

Errors in the received frames are detected by means of Parity Check and Cyclic Redundancy Check (CRC). In both cases, few extra bits are sent along with actual data to confirm that bits received at other end are same as they were sent. If the counter-check at receiver' end fails, the bits are considered corrupted.

❖ Parity Check

One extra bit is sent along with the original bits to make number of 1s either even in case of even parity, or odd in case of odd parity.

The sender while creating a frame counts the number of 1s in it. For example, if even parity is used and number of 1s is even then one bit with value 0 is added. This way number of 1s remains even.If the number of 1s is odd, to make it even a bit with value 1 is added.

The receiver simply counts the number of 1s in a frame. If the count of 1s is even and even parity is used, the frame is considered to be not-corrupted and is accepted. If the count of 1s is odd and odd parity is used, the frame is still not corrupted.

If a single bit flips in transit, the receiver can detect it by counting the number of 1s. But when more than one bits are error neous, then it is very hard for the receiver to detect the error.

Cyclic Redundancy Check (CRC)

CRC is a different approach to detect if the received frame contains valid data. This technique involves binary division of the data bits being sent. The divisor is generated using polynomials. The sender performs a division operation on the bits being sent and calculates the remainder. Before sending the actual bits, the sender adds the remainder at the end of the actual bits. Actual data bits plus the remainder is called a code word. The sender transmits data bits as code words.

At the other end, the receiver performs division operation on code words using the same CRC divisor. If the remainder contains all zeros the data bits are accepted, otherwise it is considered as there some data corruption occurred in transit.

❖ Error Correction

In the digital world, error correction can be done in two ways:

• Backward Error Correction:- When the receiver detects an error in the data received, it requests back the sender to retransmit the data unit.

• Forward Error Correction:- When the receiver detects some error in the data received, it executes error-correcting code, which helps it to auto-recover and to correct some kinds of errors.

The first one, Backward Error Correction, is simple and can only be efficiently used where retransmitting is not expensive. For example, fiber optics. But in case of wireless transmission retransmitting may cost too much. In the latter case, Forward Error Correction is used.

To correct the error in data frame, the receiver must know exactly which bit in the frame is corrupted. To locate the bit in error, redundant bits are used as parity bits for error detection. For example, we take ASCII words (7 bits data), then there could be 8 kind of information we need: first seven bits to tell us which bit is error and one more bit to tell that there is no error.

For m data bits, r redundant bits are used. r bits can provide 2r combinations of information. In m+r bit codeword, there is possibility that the r bits themselves may get corrupted. So the number of r bits used must inform about m+r bit locations plus no-error information, i.e. m+r+1.

Data-link layer is responsible for implementation of point-to-point flow and error control mechanism.

❖ Flow Control

When a data frame (Layer-2 data) is sent from one host to another over a single medium, it is required that the sender and receiver should work at the same speed. That is, sender sends at a speed on which the receiver can process and accept the data. What if the speed (hardware/software) of the sender or receiver differs? If sender is sending too fast the receiver may be overloaded, (swamped) and data may be lost.

Two types of mechanisms can be deployed to control the flow:

• Stop and Wait

This flow control mechanism forces the sender after transmitting a data frame to stop and wait until the acknowledgement of the data-frame sent is received.

• **Sliding Window**

In this flow control mechanism, both sender and receiver agree on the number of data- frames after which the acknowledgement should be sent. As we learnt, stop and wait flow control mechanism wastes resources, this protocol tries to make use of underlying resources as much as possible.

❖ Error Control

When data-frame is transmitted, there is a probability that data-frame may be lost in the transit or it is received corrupted. In both cases, the receiver does not receive the correct data-frame and sender does not know anything about any loss.In such case, both sender and receiver are equipped with some protocols which helps them to detect transit errors such as loss of data-frame. Hence, either the sender retransmits the data-frame or the receiver may request to resend the previous data-frame.

Requirements for error control mechanism:

• Error detection - The sender and receiver, either both or any, must ascertain that there is some error in the transit.

• Positive ACK - When the receiver receives a correct frame, it should acknowledge it.

• Negative ACK - When the receiver receives a damaged frame or a duplicate frame, it sends a NACK back to the sender and the sender must retransmit the correct frame.

• Retransmission: The sender maintains a clock and sets a timeout period. If an acknowledgement of a data-frame previously transmitted does not arrive before the timeout the sender retransmits the frame, thinking that the frame or it's acknowledgement is lost in transit.

There are three types of techniques available which Data-link layer may deploy to control the errors by Automatic Repeat Requests (ARQ):

• **Stop-and-wait ARQ**

The following transition may occur in Stop-and-Wait ARQ:

• The sender maintains a timeout counter.

• When a frame is sent, the sender starts the timeout counter.

• If acknowledgement of frame comes in time, the sender transmits the next frame in queue.

• If acknowledgement does not come in time, the sender assumes that either the frame or its acknowledgement is lost in transit. Sender retransmits the frame and starts the timeout counter.

• If a negative acknowledgement is received, the sender retransmits the frame.

• Go-Back-N ARQ

Stop and wait ARQ mechanism does not utilize the resources at their best.When the acknowledgement is received, the sender sits idle and does nothing. In Go-Back-N ARQ method, both sender and receiver maintain a window.

The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones. The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame's sequence number.

When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of frames. If sender finds that it has received NACK or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.

• **Selective Repeat ARQ**

In Go-back-N ARQ, it is assumed that the receiver does not have any buffer space for its window size and has to process each frame as it comes. This enforces the sender to retransmit all the frames which are not acknowledged.

In Selective-Repeat ARQ, the receiver while keeping track of sequence numbers, buffers the frames in memory and sends NACK for only frame which is missing or damaged.

The sender in this case, sends only packet for which NACK is received.

➢ Elementary Data Link Protocols

❖ An unrestricted simplex protocol In order to appreciate the step by step development of efficient and complex protocols such as SDLC, HDLC etc., we will begin with a simple but unrealistic protocol. In this protocol:

• Data are transmitted in one direction only

• The transmitting (Tx) and receiving (Rx) hosts are always ready

• Processing time can be ignored

• Infinite buffer space is available

• No errors occur; i.e. no damaged frames and no lost frames (perfect channel)

The protocol consists of two procedures, a sender and receiver as depicted below:

```
/* protocol 1 */
Sender() {
forever {
from_host(buffer); S.info = buffer; sendf(S); } }
Receiver() {
forever {
wait(event); getf(R); to_host(R.info); } }
```

❖ **A simplex stop-and-wait protocol In this protocol we assume that**

• Data are transmitted in one direction only

• No errors occur (perfect channel)

• The receiver can only process the received information at a finite rate

These assumptions imply that the transmitter cannot send frames at a rate faster than the receiver can process them.

The problem here is how to prevent the sender from flooding the receiver.

A general solution to this problem is to have the receiver provide some sort of feedback to the sender. The process could be as follows: The receiver send an acknowledge frame back to the sender telling the sender that the last received frame has been processed and passed to the host; permission to send the next frame is granted. The sender, after having sent a frame, must wait for the acknowledge frame from the receiver before sending another frame. This protocol is known as stop-and-wait.

The protocol is as follows:

```
/* protocol 2 */
Sender() {
forever {
from_host(buffer); S.info = buffer; sendf(S); wait(event); } }
Receiver() {
forever {
wait(event); getf(R); to_host(R.info); sendf(S); } }
```

❖ **A simplex protocol for a noisy channel**

In this protocol the unreal "error free" assumption in protocol 2 is dropped. Frames may be either damaged or lost completely. We assume that transmission errors in the frame are detected by the hardware checksum.

One suggestion is that the sender would send a frame, the receiver would send an ACK frame only if the frame is received correctly. If the frame is in error the receiver simply ignores it; the transmitter would time out and would retransmit it.

One fatal flaw with the above scheme is that if the ACK frame is lost or damaged, duplicate frames are accepted at the receiver without the receiver knowing it.

Imagine a situation where the receiver has just sent an ACK frame back to the sender saying that it correctly received and already passed a frame to its host. However, the ACK frame gets lost completely, the sender times out and retransmits the frame. There is no way for the receiver to tell whether this frame is a retransmitted frame or a new frame, so the receiver accepts this duplicate happily and transfers it to the host. The protocol thus fails in this aspect.

To overcome this problem it is required that the receiver be able to distinguish a frame that it is seeing for the first time from a retransmission. One way to achieve this is to have the sender put a sequence number in the header of each frame it sends. The receiver then can check the sequence number of each arriving frame to see if it is a new frame or a duplicate to be discarded.

The receiver needs to distinguish only 2 possibilities: a new frame or a duplicate; a 1-bit sequence number is sufficient. At any instant the receiver expects a particular sequence number. Any wrong sequence numbered frame arriving at the receiver is rejected as a duplicate. A correctly numbered frame arriving at the receiver is accepted, passed to the host, and the expected sequence number is incremented by 1 (modulo 2).

The protocol is depicted below:

```
/* protocol 3 */

Sender() {

NFTS = 0; /* NFTS = Next Frame To Send */ from_host(buffer); forever {

S.seq = NFTS; S.info = buffer; sendf(S); start_timer(S.seq); wait(event); if(event == frame_arrival) {

from_host(buffer); ++NFTS; /* modulo 2 operation */ }
} }
Receiver() {
FE = 0; /* FE = Frame Expected */ forever {
wait(event); if(event == frame_arrival) {
getf(R); if(R.seq == FE) {
to_host(R.info); ++FE; /* modulo 2 operation */ } sendf(S); /* ACK */ } } }
```

This protocol can handle lost frames by timing out. The timeout interval has to be long enough to prevent premature timeouts which could cause a "deadlock" situation.

➢ Sliding Window Protocols

❖ Piggybacking technique

In most practical situations there is a need for transmitting data in both directions (i.e. between 2 computers). A full duplex circuit is required for the operation.

If protocol 2 or 3 is used in these situations the data frames and ACK (control) frames in the reverse direction have to be interleaved. This method is acceptable but not efficient. An efficient method is to absorb the ACK frame into the header of the data frame going in the same direction. This technique is known as piggybacking.

When a data frame arrives at an IMP (receiver or station), instead of immediately sending a separate ACK frame, the IMP restrains itself and waits until the host passes it the next message. The acknowledgement is then attached to the outgoing data frame using the ACK field in the frame header. In effect, the acknowledgement gets a free ride in the next outgoing data frame.

This technique makes better use of the channel bandwidth. The ACK field costs only a few bits, whereas a separate frame would need a header, the acknowledgement, and a checksum.

An issue arising here is the time period that the IMP waits for a message onto which to piggyback the ACK. Obviously the IMP cannot wait forever and there is no way to tell exactly when the next message is available. For these reasons the waiting period is usually a fixed period. If a new host packet arrives quickly the acknowledgement is piggybacked onto it; otherwise, the IMP just sends a separate ACK frame.

❖ Sliding window When one host sends traffic to another it is desirable that the traffic should arrive in the same sequence as that in which it is dispatched. It is also desirable that a data link should deliver frames in the order sent.

A flexible concept of sequencing is referred to as the sliding window concept and the next three protocols are all sliding window protocols.

In all sliding window protocols, each outgoing frame contains a sequence number SN ranging from 0 to 2^(n -1)(where n is the number of bits reserved for the sequence number field).

At any instant of time the sender maintains a list of consecutive sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the sending window. Similarly, the receiver maintains a receiving window corresponding to frames it is permitted to accept.

The size of the window relates to the available buffers of a receiving or sending node at which frames may be arranged into sequence.

At the receiving node, any frame falling outside the window is discarded. Frames falling within the receiving window are accepted and arranged into sequence. Once sequenced, the frames at the left of the window are delivered to the host and an acknowledgement of the delivered frames is transmitted to their sender. The window is then rotated to the position where the left edge corresponds to the next expected frame, RN.

Whenever a new frame arrives from the host, it is given the next highest sequence number, and the upper edge of the sending window is advanced by one. The sequence numbers within the sender's window represent frames sent but as yet not acknowledged. When an acknowledgement comes in, it gives the position of the receiving left window edge which indicates what frame the receiver expects to receive next. The sender then rotates its window to this position, thus making buffers available for continuous transmission.

❖ A one bit sliding window protocol: protocol 4

The sliding window protocol with a maximum window size 1 uses stop-and-wait since the sender transmits a frame and waits for its acknowledgement before sending the next one. /* protocol 4 */

Send_and_receive()

```
{
NFTS = 0; FE = 0; from_host(buffer); S.info = buffer; S.seq = NFTS; S.ack = 1-FE; sendf(S); start_timer(S.seq);
forever {
wait(event); if(event == frame_arrival) {
getf(R); if(R.seq == FE) {
to_host(R.info); ++FE; } if(R.ack == NFTS) {
from_host(buffer); ++NFTS; } } S.info = buffer; S.seq = NFTS; S.ack = 1-FE; sendf(S); start_timer(S.seq); } }
```

❖ Pipelining

In many situations the long round-trip time can have important implications for the efficiency of the bandwidth utilisation.

As an example, consider a satellite channel with a 500ms round-trip propagation delay. At time t~~=0 the sender starts sending the first frame. Not until at least t~>=~500 ms has the acknowledgement arrived back at the sender. This means that the sender was blocked most of the time causing a reduction in efficiency. As another example, if the link is operated in a two-way alternating mode (half-duplex), the line might have to be "turned around" for each frame in order to receive an acknowledgement. This acknowledgement delay could severely impact the effective data transfer rate.

The effects of these problems can be overcome by allowing the sender to transmit multiple contiguous frames (say up to w frames) before it receives an acknowledgement. This technique is known as pipelining.

In the satellite example, with a channel capacity of 50kbps and 1000-bit frames, by the time the sender has finished sending 26 frames, t~=~520 ms, the acknowledgement for frame 0 will have just arrived, allowing the sender to continue sending frames. At all times, 25 or 26 unacknowledged frames will be outstanding, and the sender's window size needs to be at least 26.

Pipelining frames over an unreliable communication channel raises some serious issues. What happens if a frame in the middle of a long stream is damaged or lost? What should the receiver do with all the correct frames following the bad one?

The are two basic Automatic Request for Repeat (ARQ) methods for dealing with errors in the presence of pipelining.

One method, the normal mode of ARQ is called Go-back-N. If the receiver detects any error in frame N, it signals the sender and then discards any subsequent frame.

The sender, which may currently be sending frame N+X when the error signal is detected, initiates retransmission of frame N and all subsequent frames.

The other method is called selective reject. In this method the receiver stores all the correct frames following the bad one. When the sender finally notices what was wrong, it just retransmits the one bad frame, not all its successors.

❖ Protocol 5: Pipelining, Multiple outstanding frames (MaxSeq)

In this protocol, the sender may transmit up to MaxSeq frames without waiting for an acknowledgement. In addition, unlike the previous protocols, the host is not assumed to have a new message all the time. Instead, the host causes host ready events when there is a message to send.

This protocol employs the Go-back-N technique. In the example below, the window size of the receiver is equal to 1, and a maximum of MaxSeq frames may be outstanding at any instant.

```
/* protocol 5 */

send_data(frame_number) {

S.info = buffer[frame_number]; S.seq = frame_number; S.ack = (FE+MaxSeq) % (MaxSeq+1); sendf(S);
start_timer(frame_number); }

send_receive() {

enable_host(); NFTS = 0; Ack_expected = 0; Frame_expected = 0; nbuffered = 0; forever
{
wait(event); switch(event) { case host_ready:
from_host(buffer[NFTS]); ++nbuffered; send_data(NFTS); ++NFTS; break; case frame_arrival:
getf(R); if(R.seq == Frame_expected) {
to_host(R.info); ++Frame_expected; } if( (Ack_expected <= R.ack && R.ack < NFTS)
||(NFTS < Ack_expected && Ack_expected <= R.ack) ||(R.ack < NFTS &&NFTS < Ack_expected)) {
--nbuffered; stop_timer(Ack_expected); ++Ack_expected; } break; case checksum_error:
/* just ignore the bad frame */ break; case timeout:
NFTS = Ack_expected; i = 0; do {
send_data(NFTS); ++NFTS; ++i; } while(i<=nbuffered); break; } if(nbuffered < MaxSeq) enable_host(); else
                        disable_host(); } }
```

❖ Protocol 6

This protocol employs the selective reject technique. The protocol does not discard good frames because an earlier frame was damaged or lost provided that these good frames fall within the receiving window.

Associated with each outstanding frame is a timer. When the timer goes off, (or when the transmitter is notified of any error), only that one frame is retransmitted, not all the outstanding frames, as in protocol 5.

In this protocol, the receiver's window size is fixed, the maximum of which is (MaxSeq+1)/2 . The maximum number is thus chosen to ensure that there will not be an overlapping between the new window and the previous window. The overlapping of windows means that the receiver would not be able to differentiate between a new frame and a retransmitted frame.

The receiver has a buffer reserved for each sequence number within its window. Whenever a frame arrives, its sequence number is checked to see if it falls within the receiver's window. If so, and if it has not already been received, it is accepted and stored regardless of whether or not it is the next frame expected by the host. Frames passed to the host must always be in order.

Protocol 6 is more efficient than protocol 5 in that:

• Protocol 6 employs an auxiliary timer to prevent delay in piggybacking. If no reverse traffic has presented itself before the timer goes off, a separate acknowledgement is sent.

• Whenever the receiver has reason to suspect that an error has occured it sends a negative acknowledgement (NAK) frame back to the sender. Such a frame is a request for retransmission of the frame specified in the NAK.

➢ Example of Data Link Protocols (HDLC and PPP)

❖ HDLC

Introduction In this subsection we describe the International Standards Organisation data link protocol HDLC (High-level Data Link Control). CCITT Recommendation X.25 level 2 is one of the permissible options of HDLC. All these bit-oriented protocols grew out from the original IBM SDLC (Synchronous Data Link Control). All these protocols are based on the same protocols. They differ only in minor ways (see the appropriate protocol definition).

HDLC is a discipline for the management of information transfer over a data communication channel.

HDLC has a basic structure that governs the function and the use of control procedures. The basic structure includes:

• The definitions of primary and secondary station responsibilities.

• The design of information grouping for control and checking.

• The design of the format for transfer of information and control data.

❖ **Primary and secondary stations**

A data link involves two or more participating stations. For control purposes one station on the link is designated a primary station, the others are secondary stations. The primary station is responsible for the organisation of data flow and for the link level error recovery procedures.

A frame sent from a primary station is referred to as a command frame. A frame from a secondary to a primary is referred to as a response frame. Normally when a command is sent, a response or a string of responses is expected in reply.

On a point-to-point link either station could be the primary. On multidrop links and where polling is employed, the station which polls the other is the primary. A station may have several links connected to it. In some configurations it may be assigned as a primary for some links and a secondary for others. The station assignments are normally made when a system is designed.

❖ Frame structure In HDLC the input data string to be transmitted over the link is broken up into data frames which are then transmitted sequentially. The input data string could be command, response or information.

All frames start and end with the flag byte 01111110. There is a 3-byte header at the start of the frame and a 3-byte trailer at the end. Between the header and the trailer any number of bits can be carried (bit-oriented).

The header FLAG

The 8 bit pattern "01111110" which signals the beginning and end of an HDLC frame. If a piece of data within the frame to be transmitted contains a series of 5 or more 1's, the transmitting station must insert a 0 to distinguish this set of 1's in the data from the flags at the beginning and end of the frame. This technique of inserting bits is called bit- stuffing. These bits are detected and removed upon receipt. If a pattern of five 1's is followed by a 1 and then a 0, it marks the end of the frame. If a pattern of five ones is followed by two more 1's, it is a signal from the transmitting station to abort.

ADDRESS

The address field always contains the address of a secondary station. When the primary station transmits, it is the receiving secondary station's address, if it is a secondary responding, it is it's own address. This field is only populated for Unbalanced connections, it is otherwise empty for point-to-point (Balanced) links.

CONTROL

The format of the control field varies with the data it contains. There are three categories of HDLC frames:

- Information (I-frame) - Carries data
- Supervisory (S-frame) - Carries commands and responses
- Unnumbered (U-frame) - Carries additional command sequences

The trailer

FRAME CHECK SEQUENCE (CRC)

The FCS field contains a 16-bit Cyclic Redundancy Check (CRC) error detecting code.

FLAG

This flag indicates the end of the frame. When this flag is detected the receiver examines the preceding two bytes and executes its error detection algorithm.

❖ Modes of operation For a secondary station two operational modes are defined: Normal response mode and Asynchronous response mode.

Normal response mode (NRM) In this mode a secondary station can transmit only in response to a command frame from the primary station. The response transmission may consist of one or more frames, the last of which must be explicitly indicated (using the Poll/Final bit). The secondary then cannot transmit again until it receives another command to do so.

Asynchronous response mode (ARM) In this mode the secondary station may initiate transmission of a frame or group of frames without receiving explicit permission from the primary. The frames may contain data or control information. The secondary station is responsible for time-out and retransmission if necessary.

ARM is necessary for the control of loops of stations or multi-drop lines with hub polling. In these configurations a secondary may receive a "go-ahead" message from another secondary and transmit in response to it.

Unfortunately, the above two modes are unbalanced modes in which one end of the link is the master and the other end is the slave. To make the protocol more suitable when the two stations are equals (e.g. the nodes of mesh structured computer networks), HDLC has an additional mode: Asynchronous Balanced Mode (ABM).

Asynchronous balanced mode (ABM) In this mode the stations have identical protocols. They both can initiate transmission at any time. They can send both commands and responses.

➢ PPP (Point-to-Point Protocol)

🗣 PPP (Point-to-Point Protocol) is a protocol for communication between two computers using a serial interface, typically a personal computer connected by phone line to a server. For example, your Internet server provider may provide you with a PPP connection so that the provider's server can respond to your requests, pass them on to the Internet, and forward your requested Internet responses back to you. PPP uses the Internet protocol (IP) (and is designed to handle others). It is sometimes considered a member of the TCP/IP suite of protocols. Relative to the Open Systems Interconnection (OSI) reference model, PPP provides layer 2 (data-link layer) service. Essentially, it packages your computer's TCP/IP packets and forwards them to the server where they can actually be put on the Internet.

🗣 PPP is a full-duplex protocol that can be used on various physical media, including

twisted pair or fiber optic lines or satellite transmission. It uses a variation of High Speed Data Link Control (HDLC) for packet encapsulation.

🗣 PPP is usually preferred over the earlier de facto standard Serial Line Internet Protocol

(SLIP) because it can handle synchronous as well as asynchronous communication. PPP can share a line with other users and it has error detection that SLIP lacks. Where a choice is possible, PPP is preferred.

➢ The Medium Access Control Sublayer

This section deals with broadcast networks and their protocols. The basic idea behind broadcast networks is how to determine who gets to use the channel when many users want to transmit over it. The protocols used to determine who goes next on a multi access channel belong to a sublayer of the data link layer called MAC.

The MAC sub-layer interacts with the physical layer and is primarily responsible for framing/de-framing and collision resolution.

Framing/De-Framing and interaction with PHY: On the sending side, the MAC sub-layer is responsible for creation of frames from network layer packets, by adding the frame header and the frame trailer. While the frame header consists of layer2 addresses (known as MAC address) and a few other fields for control purposes, the frame trailer consists of the CRC/checksum of the whole frame. After creating a frame, the MAC layer is responsible for interacting with the physical layer processor (PHY) to transmit the frame.

On the receiving side, the MAC sub-layer receives frames from the PHY and is responsible for accepting each frame, by examining the frame header. It is also responsible for verifying the checksum to conclude whether the frame has come uncorrupted through the link without bit errors. Since checksum computation and verification are computer intensive tasks, the framing/de-framing functionality is done by dedicated piece of hardware (e.g. NIC card on PCs).

Collision Resolution : On shared or broadcast links, where multiple end nodes are connected to the same link, there has to be a collision resolution protocol running on each node, so that the link is used cooperatively. The MAC sub-layer is responsible for this task and it is the MAC sub-block that implements standard collision resolution protocols like CSMA/CD, CSMA etc. For half-duplex links, it is the MAC sub-layer that makes sure that a node sends data on the link only during its turn. For full-duplex point-to-point links, the collision resolution functionality of MAC sub-layer is not required.

In end nodes and in intermediate devices like L2 switches and Routers, the LLC functionality is implemented in network device driver software that is part of the Operating System and the MAC functionality is implemented in dedicated piece of hardware.

❖ The Channel Allocation Problem

The channel allocation problem is "How to allocate a single broadcast channel among competing users". There are two schemes that can be applied to allocate a single channel among competing users and they are:

1. Static Channel Allocation 2. Dynamic Channel Allocation

Static Channel Allocation

Static channel allocation can be done by using the static multiplexing method; FDM and TDM (Frequency/Time Division Multiplexing). FDM is used in Radio or TV broadcasting whereas TDM is POTS (Plain Old Telephone System). These channel allocation techniques waste bandwidth as they cannot cope with the dynamic environment.

Dynamic Channel Allocation

Dynamic Channel Allocation is done by using Pure/Slotted ALOHA Protocol or Carrier Sense Multiple Access (CSMA) Protocols. It is more efficient than static channel allocation as it uses collision free protocols and doesn't waste bandwidth.

❖ Channel Allocation Methods

Pure ALOHA

One of the newly discovered algorithms-protocols for allocating a multiple access channel is ALOHA. The idea is simple. Users transmit whenever they have data to be sent. Frames are destroyed when collision occurs. When a sender detects a collision waits for a random amount of time and retransmits the frame. With this method the best theoretical throughput and channel utilization we can have is 18%. Term throughput means the amount of work that a computer can do in a given time period

Slotted ALOHA

In slotted ALOHA time is divided into discrete intervals, each corresponding to one frame. A computer is not permitted to send whenever it has data to send. Instead it is required to wait for the next available slot. The best it can be achieved is 37% of slots empty, 37% success and 26% collision.

➢ Multiple Access Protocol

we noted that there are two types of network links: point-to-point links, and broadcast links. A point-to-point link consists of a single sender on one end of the link, and a single receiver at the other end of the link. Many link-layer protocols have been designed for point-to-point links; PPP (the point-to-point protocol) and HDLC are two such protocols. The second type of link, a broadcast link, can have multiple sending and receiving nodes all connected to the same, single, shared broadcast channel. The term "broadcast" is used here because when any one node transmits a frame, the channel broadcasts the frame and each of the other nodes receives a copy. Ethernet is probably the most widely deployed broadcast link technology.

In order to ensure that the broadcast channel performs useful work when multiple nodes are active, it is necessary to somehow coordinate the transmissions of the active nodes. This coordination job is the responsibility of the multiple access protocol. we can classify just about any multiple access protocol as belonging to one of three categories: channel partitioning protocols, random access protocols, and taking-turns protocols.

Channel Partitioning Protocols

Time Division Multiplexing (TDM) and Frequency Division Multiplexing (FDM) are two techniques that can be used to partition a broadcast channel's bandwidth among all nodes sharing that channel. As an example, suppose the channel supports N nodes and that the transmission rate of the channel is R bps. TDM divides time into time frames (not to be confused the unit of data, the frame, at the data link layer) and further divides each time frame into N time slots. Each slot time is then assigned to one of the N nodes. Whenever a node has a frame to send, it transmits the frame's bits during its assigned time slot in the revolving TDM frame. Typically, frame sizes are chosen so that a single frame can be transmitting during a slot time. Figure 5.3-3 shows a simple four-node TDM example. Returning to our cocktail party analogy, a TDM-regulated cocktail party would allow one partygoer to speak for a fixed period of time, and then allow another partygoer to speak for the same amount of time, and so on. Once everyone has had their chance to talk, the pattern repeats.

TDM is appealing as it eliminates collisions and is perfectly fair: each node gets a dedicated transmission rate of R/N bps during each slot time. However, it has two major drawbacks. First, a node is limited to this rate of R/N bps over a slot's time even when it is the only node with frames to send. A second drawback is that a node must always wait for its turn in the transmission sequence - again, even when it is the only node with a frame to send. Imagine the partygoer who is the only one with anything to say (and imagine that this is the even rarer circumstance where everyone at the party wants to hear what that one person has to say). Clearly, TDM would be a poor choice for a multiple access protocol for this particular party.

While TDM shares the broadcast channel in time, FDM divides the R bps channel into different frequencies (each with a bandwidth of R/N) and assigns each frequency to one of the N nodes. FDM thus creates N "smaller" channels of R/N bps out of the single, "larger" R bps channel. FDM shares both the advantages and drawbacks of TDM. It avoids collisions and divides the bandwidth fairly among the N nodes. However, FDM also shares a principal disadvantage with TDM - a node is limited to a bandwidth of R/N, even when it is the only node with frames to send.

A third channel partitioning protocol is Code Division Multiple Access (CDMA). While TDM and FDM assign times slots and frequencies, respectively, to the nodes, CDMA assigns a different code to each node. Each node then uses its unique code to encode the data bits it sends, as discussed below. We'll see that CDMA allows different nodes to transmit simultaneously and yet have their respective receivers correctly receive a sender's encoded data bits (assuming the receiver knows the sender's code) in spite of "interfering" transmissions by other nodes. CDMA has been used in military systems for some time (due its anti jamming properties) and is now beginning to find widespread civilian use, particularly for use in wireless multiple access channels.

In a CDMA protocol, each bit being sent by the sender is encoded by multiplying the bit by a signal (the code) that changes at a much faster rate (known as the chipping rate) than the original sequence of data bits.

The world is far from ideal, however, and as noted above, CDMA must work in the presence of interfering senders that are encoding and transmitting their data using a different assigned code. But how can a CDMA receiver recover a sender's original data bits when those data bits are being tangled with bits being transmitted by other senders? CDMA works under the assumption that the interfering transmitted bit signals are additive, e.g., that if three senders send a 1 value, and a fourth sender sends a -1 value during the same mini-slot, then the received signal at all receivers during hat mini-slot is a 2 (since $1+ 1 + 1 - 1 = 2$).

Our discussion here of CDMA is necessarily brief and a number of difficult issues must be addressed in practice. First, in order for the CDMA receivers to be able to extract out a particular sender's signal, the CDMA codes must be carefully chosen. Secondly, our discussion has assumed that the received signal strengths from various senders at a receiver are the same; this can be difficult to achieve in practice. There is a considerable body of literature addressing these and other issues related to CDMA.

Random Access Protocols

The second broad class of multiple access protocols are so-called random access protocols. In a random access protocol, a transmitting node always transmits at the full rate of the channel, namely, R bps. When there is a collision, each node involved in the collision repeatedly retransmit its frame until the frame gets through without a collision. But when a node experiences a collision, it doesn't necessarily retransmit the frame right away. Instead it waits a random delay before retransmitting the frame. Each node involved in a collision chooses independent random delays. Because after a collision the random delays are independently chosen, it is possible that one of the nodes will pick a delay that is sufficiently less than the delays of the other colliding nodes, and will therefore be able to "sneak" its frame into the channel without a collision.

Slotted ALOHA

Let's begin our study of random access protocols with one of the most simple random access protocols, the so-called slotted ALOHA protocol. In our description of slotted ALOHA, we assume the following:

• All frames consist of exactly L bits.

• Time is divided into slots of size L/R seconds (i.e., a slot equals the time to transmit one frame).

• Nodes start to transmit frames only at the beginnings of slots.

• The nodes are synchronized so that each node knows when the slots begin.

• If two or more frames collide in a slot, then all the nodes detect the collision event before the slot ends.

Let p be a probability, that is, a number between 0 and 1. The operation of slotted ALOHA in each node is simple::

• When the node has a fresh frame to send, it waits until the beginning of the next slot and transmits the entire frame in the slot.

• If there isn't a collision, the node won't consider retransmitting the frame. (The node can prepare a new frame for transmission, if it has one.)

• If there is a collision, the node detects the collision before the end of the slot. The node retransmits its frame in each subsequent slot with probability p until the frame is transmitted without a collision.

By retransmitting with probability p, we mean that the node effectively tosses a biased coin; the event heads corresponds to retransmit, which occurs with probability p. The event tails corresponds to "skip the slot and toss the coin again in the next slot"; this occurs with probability (1−p). Each of the nodes involved in the collision toss their coins independently.

Slotted ALOHA would appear to have many advantages. Unlike channel partitioning, slotted ALOHA allows a single active node (i.e., a node with a frame to send) to continuously transmit frames at the full rate of the channel. Slotted ALOHA is also highly decentralized, as each node detects collisions and independently decides when to retransmit. (Slotted ALOHA does, however, require the slots to be synchronized in the nodes; we'll shortly discuss an unslotted version of the ALOHA protocol, as well as CSMA protocols; noe of which require such synchronization and are therefore fully decentralized.) Slotted ALOHA is also an extremely simple protocol.

Slotted ALOHA also works great when there is only one active node, but how efficient is it when there are multiple active nodes? There are two possible efficiency concerns here. First, as shown in Figure, when there are multiple active nodes, a certain fraction of the slots will have collisions and will therefore be "wasted." The second concern is that another fraction of the slots will be empty because all active nodes refrain from transmitting as a result of the probabilistic transmission policy. The only "unwasted" slots will be those in which exactly one node transmits. A slot in which exactly one node transmits is said to be a successful slot. The efficiency of a slotted multiple access protocol is defined to be the long-run fraction of successful slots when there are a large number of active nodes, with each node having a large number of frames to send. Note that if no form of access control were used, and each node were to immediately retransmits after each collision, the efficiency would be zero. Slotted ALOHA clearly increases the efficiency beyond zero, but by how much?

Figure: Nodes 1, 2 and 3 collide in the first slot. Node 2 finally succeeds in the fourth slot, node 1 in the eighth slot, and node 3 in the ninth slot. The notation C, E and S represent "collision slot", "empty slot" and "successful slot", respectively

We now proceed to outline the derivation of the maximum efficiency of slotted ALOHA. To keep this derivation simple, let's modify the protocol a little and assume that each node attempts to transmit a frame in each slot with probability p. (That is, we assume that each node always has a frame to send and that the node transmits with probability p for a fresh frame as well as for a frame that has already suffered a collision.) Suppose first there are N nodes. Then the the probability that a given slot is a successful slot is the probability that one of the nodes transmits and that the remaining $N-1$ nodes do not transmit. The probability that a given node transmits is p; the probability that the remaining nodes do not transmit is $(1-p)N-1$. Therefore the probability a given node has a success is $p(1-p)N-1$. Because there are N nodes, the probability that an arbitrary node has a success is $Np(1-p)N-1$.

Thus, when there are N active nodes, the efficiency of slotted ALOHA is Np(1−p)N−1 . To obtain the maximum efficiency for N active nodes, we have to find the p* that maximizes this expression. (See the homework problems for a general outline of this derivation.) And to obtain the maximum efficiency for a large number of active nodes, we take the limit of Np*(1−p*)N−1 as N approaches infinity. (Again, see homework problems.) After performing these calculations, we'll find that the maximum efficiency of the protocol is given by 1/e = .37. That is, when a large number of nodes have many frames to transmit, then (at best) only 37% of the slots do useful work. Thus the effective transmission rate of the channel is not R bps but only .37 R bps! A similar analysis also shows that 37% of the slots go empty and 26% of slots have collisions. Imagine the poor network administrator who has purchased a 100 Mbps slotted ALOHA system, expecting to be able to use the network to transmit data among a large number of users at an aggregate rate of, say, 80 Mbps! Although the channel is capable of transmitting a given frame at the full channel rate of 100Mbps, in the long term, the successful throughput of this channel will be less that 37 Mbps.

ALOHA

The slotted ALOHA protocol required that all nodes synchronize their transmissions to start at the beginning of a slot. The first ALOHA protocol was actually an unslotted, fully decentralized, protocol. In so-called pure ALOHA, when a frame first arrives (i.e., a network layer datagram is passed down from the network layer at the sending node), the node immediately transmits the frame in its entirely into the broadcast channel. If a transmitted frame experiences a collision with one or more other transmissions, the node will then immediately (after completely transmitting its collided frame) retransmit the frame with probability p. Otherwise, the node waits for a frame transmission time. After this wait, it then transmits the frame with probability p, or waits (remaining idle) for another frame time with probability 1−p.

Figure: Interfering transmissions in pure Aloha

To determine the maximum efficiency of pure ALOHA, we focus on an individual node. We'll make the same assumptions as in our slotted ALOHA analysis and take the frame transmission time to be the unit of time At any given time, the probability that a node is transmitting a frame is p. Suppose this frame begins transmission at time t

0

. As shown in Figure, in order for this

frame to be successfully transmitted, no other nodes can begin their transmission in the interval of time $[t_0 - 1, t_0]$. Such a transmission would overlap with the beginning of the transmission of node i's frame. The probability that all other nodes do not begin a transmission in this interval is (1−p)N−1. Similarly, no other node can begin a transmission while node i is transmitting, as such a transmission would overlap with the latter part of node i's transmission. The probability that all other nodes do not begin a transmission in this interval is also (1−p)N−1. Thus, the probability that a given node has a successful transmission is p(1−p)2(N−1). By taking limits as in the slotted ALOHA case, we find that the maximum efficiency of the pure ALOHA protocol is only 1/(2e) – exactly half that of slotted ALOHA. This then is the price to be paid for a fully decentralized ALOHA protocol.

CSMA – Carrier Sense Multiple Access

In both slotted and pure ALOHA, a node's decision to transmit is made independently of the activity of the other nodes attached to the broadcast channel. In particular, a node neither pays attention to whether another node happens to be transmitting when it begins to transmit, nor stops transmitting if another node begins to interfere with its transmission. In our cocktail party analogy, ALOHA protocols are quite like a boorish partygoer who continues to chatter away regardless of whether other people are talking. As humans, we have human protocols that allow allows us to not only behave with more civility, but also to decrease the amount of time spent "colliding" with each other in conversation and consequently increasing the amount of amount of data we exchange in our conversations. Specifically, there are two important rules for polite human conversation:

• Listen before speaking. If someone else is speaking, wait until they are done. In the networking world, this is termed carrier sensing - a node listens to the channel before transmitting. If a frame from another node is currently being transmitted into the channel, a node then waits ("backs off") a random amount of time and then again senses the channel. If the channel is sensed to be idle, the node then begins frame transmission. Otherwise, the node waits another random amount of time and repeats this process.

• If someone else begins talking at the same time, stop talking. In the networking world, this is termed collision detection - a transmitting node listens to the channel while it is transmitting. If it detects that another node is transmitting an interfering frame, it stops transmitting and uses some protocol to determine when it should next attempt to transmit.

These two rules are embodied in the family of CSMA (Carrier Sense Multiple Access) and

CSMA/CD (CSMA with Collision Detection) protocols. Many variations on CSMA and CSMA/CD have been proposed, with the differences being primarily in the manner in which nodes perform backoff. The reader can consult these references for the details of these protocols.

The first question that one might ask about CSMA is that if all nodes perform carrier sensing, why do collisions occur in the first place? After all, a node will refrain from transmitting whenever it senses that another node is transmitting. The answer to the question can best be illustrated using space-time diagrams. Figure shows a space-time diagram of four nodes (A, B, C, D) attached to an linear broadcast bus. The horizontal axis shows the position of each node in space; the y-axis represents time.

, At time t_0 node B senses the channel is idle, as no other nodes are currently transmitting. Node B thus begins transmitting, with its bits propagating in both directions along the broadcast medium. The downward propagation of B's bits in Figure with increasing time indicates that a non-zero amount of time is needed for B's bits to actually propagate (albeit at near the speed-of-light) along the broadcast medium. At time t_1), node D has a frame to send. Although node B is currently transmitting at time t_1 ($t_1 > t_0$, the bits being transmitted by B have yet to reach D, and thus D senses the channel idle at t_1 . In accordance with the CSMA protocol, D thus begins transmitting its frame. A short time later, B's transmission begins to interfere with D's transmission at D. From Figure, it is evident that the end-to-end channel propagation delay of a broadcast channel - the time it takes for a signal to propagate from one of the the channel to another - will play a crucial role in determining its performance. The longer this propagation delay, the larger the chance that a carrier-sensing node is not yet able to sense a transmission that has already begun at another node in the network.

Figure: Space-time diagram of two CSMA nodes with colliding transmissions

In Figure, nodes do not perform collision detection; both B and D continue to transmit their frames in their entirety even though a collision has occurred. When a node performs collision detection it will cease transmission as soon as it detects a collision. Figure shows the same scenario as in Figure, except that the two nodes each abort their transmission a short time after detecting a collision. Clearly, adding collision detection to a multiple access protocol will help protocol performance by not transmitting a useless, damaged (by interference with a frame from another node) frame in its entirety.

Figure: CSMA with collision detection.

Taking-Turns Protocols

Recall that two desirable properties of a multiple access protocol are (i) when only one node is active, the active node has a throughput of R bps, and (ii) when M nodes are active, then each active node has a throughput of nearly R/M bps. The ALOHA and CSMA protocols have this first property but not the second. This has motivated researchers to create another class of protocols -- the taking-turns protocols. As with random-access protocols, there are dozens of taking-turns protocols, and each one of these protocols has many variations. We'll discuss two of the more important protocols here. The first one is the polling protocol. The polling protocol requires one of the nodes to be designated as a "master node" (or requires the introduction of a new node serving as the master). The master node polls each of the nodes in a round-robin fashion. In particular, the master node first sends a message to node 1, saying that it can transmit up to some maximum number of frames. After node 1 transmits some frames (from zero up to the maximum number), the master node tells node 2 it can transmit up to the maximum number of frames. (The master node can determine when a node has finished sending its frames by observing the lack of a signal on the channel.) The procedure continues in this manner, with the master node polling each of the nodes in a cyclic manner.

The polling protocol eliminates the collisions and the empty slots that plague the random access protocols. This allows it to have a much higher efficiency. But it also has a few drawbacks. The first drawback is that the protocol introduces a polling delay, the amount of time required to notify a node that it can transmit. If, for example, only one node is active, then the node will transmit at a rate less than R bps, as the master node must poll each of the inactive nodes in turn, each time the active node sends its maximum number of frames. The second drawback, which is potentially more serious, is that if the master node fails, the entire channel becomes inoperative.

The second taking-turn protocol is the token-passing protocol. In this protocol there is no master node. A small, special-purpose frame known as a token is exchanged among the nodes in some fixed order. For example, node 1 might always send the token to node 2, node 2 might always send the token to node 3, node N might always send the token to node 1. When a node receives a token, it holds onto the token only if it has some frames to transmit; otherwise, it immediately forwards the token to the next node. If a node does have frames to transmit when it receives the token, it sends up to a maximum number of frames and then forwards the token to the next node. Token passing is decentralized and has a high efficiency. But it has its problems as well. For example, the failure of one node can crash the entire channel. Or if a node accidentally neglects to release the token, then some recovery procedure must be invoked to get the token back in circulation? Over the years many token-passing products have been developed, and each one had to address these as well as other sticky issues.

➢ What is IEEE 802.3 Protocol

Ethernet : IEEE 802.3 Local Area Network (LAN) Protocols : Ethernet protocols refer to the family of local-area network (LAN) technology covered by the IEEE 802.3. It is working examplc of the more general carrier sense multiple access with collision detect (CSMA/CD). In the Ethernet Computer Network standard, there are two modes of operation: half-duplex and full-duplex modes. In the half duplex mode, data are transmitted using the popular Carrier-SenseMultiple Access/Collision Detection (CSMA/CD) protocol on as hared medium.

The main disadvantages of the half-duplex are the efficiency and distance limitation, in which the link distance islimited by the minimum MAC frame size. This restriction reducesthe efficiency drastically for high-rate transmission. Therefore, thecarrier extension technique is used to ensure the minimum framesize of 512 bytes in Gigabit Ethernet to achieve a reasonable linkdistance.Four data rates are currently defined for operation over opticalfiber and twisted-pair cables :

• 10 Mbps - 10Base-T Ethernet (IEEE 802.3)

• 100 Mbps - Fast Ethernet (IEEE 802.3u)

• 1000 Mbps - Gigabit Ethernet (IEEE 802.3z)

• 10-Gigabit - 10 Gbps Ethernet (IEEE 802.3ae).

The Ethernet is a multi-access network, meaning that a set of nodes send and receive frames over a shared link you can, therefore, think of an Ethernet or being like a bus that has multiple stations plugged into it. The "carrier sense" in CSMA/CD means that all the nodes can distinguish between an idle and a busy link and "collision detect" means that a node listens as it transmits and can therefore detect when a frame it is transmitting has collided with a frame transmitted by another node.

The Ethernet has its root in an early packet radio network, called ALOHA, like, the ALOHA, the problem faced by the Ethernet is how to mediate access to a shared medium fairly and efficiently. In ALOHA, the medium was the atmosphere, while in Ethernet the medium is coax cable.

In the earliest days, 10-Mbps Ethernet war used, but now it has been extended to include a 100-Mbps version called Fast Ethernet and a 1000-Mbps version called Gigabit Ethernet. The Ethernet System consists of three basic elements

(1) The physical medium used to carry Ethernet signals between computers,

(2) a set of medium access control rules embedded in each Ethernet interface that allow multiple computers to fairly arbitrate access to the shared Ethernet channel, and

(3) an Ethernet frame that consists of a standardized set of bits used to carry data over the system.

As with all IEEE 802 protocols, the ISO data link layer is divided into two IEEE 802 sub-layers, the Media Access Control (MAC) sub-layer and the MAC-client sub-layer. The IEEE 802.3 physical layer corresponds to the ISO physical layer.

Each Ethernet-equipped computer operates independently of all other stations on the network: there is no central controller. All stations attached to an Ethernet are connected to a shared signaling system, also called the medium. To send data a station first listens to the channel, and when the channel is idle the station transmits its data in the form of an Ethernet frame, or packet.

After each frame transmission, all stations on the network must contend equally for the next frame transmission opportunity. Access to the shared channel is determined by the medium access control (MAC) mechanism embedded in the Ethernet interface located in each station. The medium access control mechanism is based on a system called Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

As each Ethernet frame is sent onto the shared signal channel, all Ethernet interfaces look at the destination address. If the destination address of the frame matches with the interface address, the frame will be read entirely and be delivered to the networking software running on that computer. All other network interfaces will stop reading the frame when they discover that the destination address does not match their own address.

➢ IEEE 802.11: WIRELESS LAN

We are talking of Wireless LAN (WLAN) that is to say, "Wireless LAN", not to be confused with WAN course. Also referred Radio LAN (WLAN) if the communication medium is the radio (not light infrared for example).The stations of the wireless network can communicate directly with each other, we called Ad Hoc network type, or via relay terminals called APs (Access Points, PA) then it is an infrastructure network. the second type is by far the most common in business.

There are two types of wireless networks:

• Type networks Ad Hoc, where stations communicate directly;

• Infrastructure type networks where stations communicate through access points.

To communicate, each station must of course be equipped with an adapter WiFi and a radio antenna (often integrated into the adapter). More and more computer equipment come with a built-in WiFi adapter. Except not the case, you must buy one and connect it to the station. The connection is very varied: there are WiFi USB adapters, PCMCIA, PCI, etc.

There are several variations of WiFi. In short, 802.11b and 802.11g are compatible them and both operate with the radio waves of a frequency of 2.4 GHz. The 802.11b reached a speed of 11 Mb / s and 802.11g rises to 54 Mb / s. The 802.11a is not compatible with 802.11b and 802.11g, because it works with the waves a radio frequency of 5 GHz. It can reach 54 Mb / s. The 802.11n allows to achieve a real flow rate greater than 100 Mb / s. It is capable of operating at 2.4 GHz or 5 GHz and is compatible with the 802.11b / g and 802.11a. Unfortunately, Most 802.11n equipment available today use only tape 2.4 GHz (and are therefore not compatible with the 802.11a).

Today the WiFi version of the most used is far 802.11g. It should be rapidly overtaken by 802.11n.

The fact that WiFi is originally designed to perform WLAN does not prevent not also be used in other contexts. For example, a myriad of products, such as electronic organizers (PDAs) or Personal Data Assistant (PDAs), printers, computer monitors, VCRs or even Hi-Fi, are now equipped with WiFi connections allowing them to be linked together without any wire. In this case, the WLAN is used to achieve a WPAN. Conversely, many local authorities do not have access to top speed (ADSL is not available everywhere) are turning to WiFi to cover a town or towns with the same wireless network. This can be called Wireless MAN (WMAN).

Finally, companies are deploying wireless networks, called hotspots1 that allow anyone to connect to the Internet wirelessly slightly across the US and around the world. So one sees now what might be called WWAN (Wireless Wide Area Networks) based on WiFi technology (WiFi technology itself, however, carries data over short distances).

❖ 802.11 Architecture

The 802.11architecture defines two types of services and three different types of stations

802.11 Services

The two types of services are

1. Basic services set (BSS)
2. Extended Service Set (ESS)

1. Basic Services Set (BSS)

• The basic services set contain stationary or mobile wireless stations and a central base station called access point (AP).

• The use of access point is optional.

• If the access point is not present, it is known as stand-alone network. Such a BSS cannot send data to other BSSs. This type of architecture is known as ad-hoc architecture.

• The BSS in which an access point is present is known as an infrastructure network.

2. Extend Service Set (ESS)

• An extended service set is created by joining two or more basic service sets (BSS) having access points (APs).

• These extended networks are created by joining the access points of basic services sets through a wired LAN known as distribution system.

• The distribution system can be any IEET LAN.

• There are two types of stations in ESS:

(i) Mobile stations: These are normal stations inside a BSS.

(ii) Stationary stations: These are AP stations that are part of a wired LAN.

• Communication between two stations in two different BSS usually occurs via two APs.

• A mobile station can belong to more than one BSS at the same time.

802.11 Station Types

IEEE 802.11 defines three types of stations on the basis of their mobility in wireless LAN. These are:

1. No-transition Mobility

2. BSS-transition Mobility

3. ESS-transition Mobility

1. No-transition .Mobility: These types of stations are either stationary i.e. immovable or move only inside a BSS.

2. BSS-transition mobility: These types of stations can move from one BSS to another but the movement is limited inside an ESS.

3. ESS-transition mobility: These types of stations can move from one ESS to another. The communication mayor may not be continuous when a station moves from one ESS to another ESS.