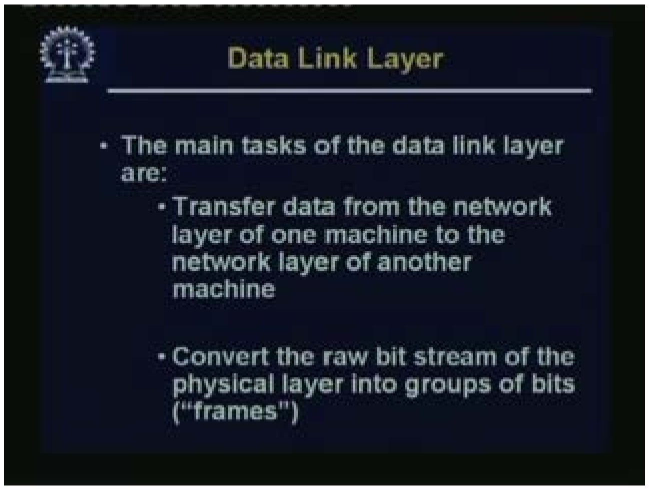
Computer Networks
Prof. Sujoy Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology – Kharagpur

Lecture - 15
Data Link Protocols

Good day. Today we will start our discussion on Data Link layer. As a matter of fact, we have already discussed a part of the data link layer, namely, the MAC sublink layer. We will see how it all fits in. To fit it into the broader picture, if you remember when we discussing the 7-layer OSI protocol, starting from the application layer the bottom most layer was the physical layer. Just above the physical layer we have the data link layer. We will see the different components of data link layer and how they are used, we will discuss different protocols, etc. These are protocols which are used in the data link layer.

(Refer Slide Time: 01:39)

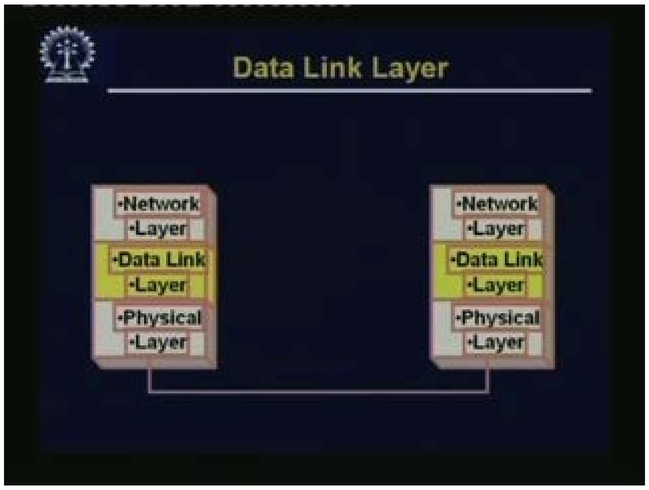


Data Link Layer

- The main tasks of the data link layer are:
    - Transfer data from the network layer of one machine to the network layer of another machine
    - Convert the raw bit stream of the physical layer into groups of bits ("frames")

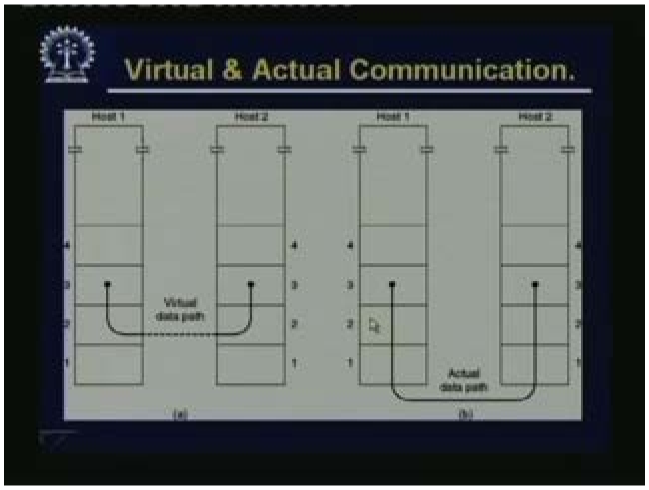
The main task of the data link layer is that it transfers data from the network layer of one machine to the network layer of another machine. This is a part of the services it gives to the upper layer. If you remember, above the data link layer, we have the network layer. The data link layer gives a service to the network layer, and this service is the transfer of data from one network layer to the other, and this in turn uses the physical layer. It converts the raw bit stream of the physical layer into groups of bits or frames.

(Refer Slide Time: 02:28)



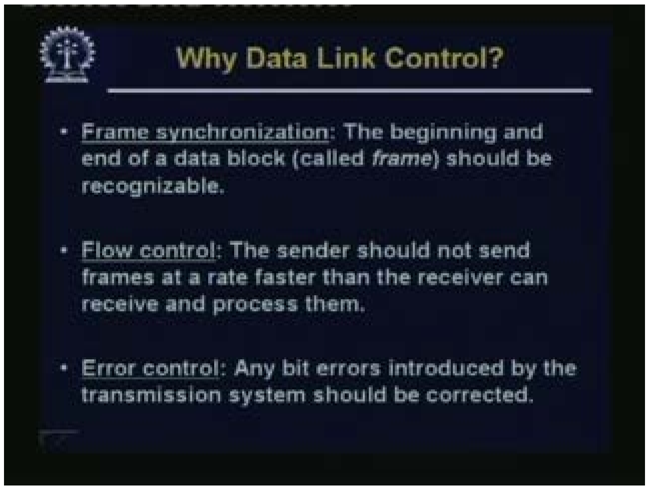
This is how we can look at it: this is one node and this is another node. Above this is there could be other layers; we are not concerned with these at the moment. This gets some data to be sent from the network layer and the data link layer sends it to the next data link layer. Remember that the network layer is concerned with transfer of data, etc., across the network. That means it may take several such hops; but data link layer is concerned with just a single hop. This is how we simplify the problem of multi- hop; this we will leave to the network layer for the single hop. So, multi-hop will constitute several such hops and data link layer will handle transfer of data from one hop to the next hop. This is another way of looking at it.

(Refer Slide Time: 03:34)



There is a virtual data path from layer three to layer three, but actually this goes through the data link layer and this is the actual data path, which goes through the physical layer and some physical transmission medium.

(Refer Slide Time: 03: 52)

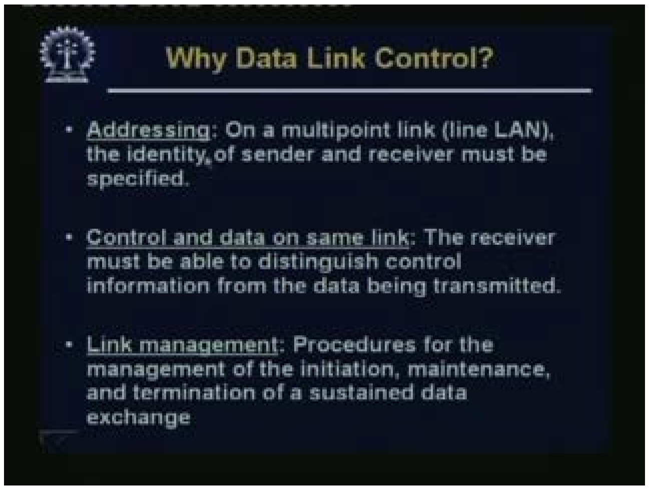


Why Data Link Control?

- Frame synchronization: The beginning and end of a data block (called *frame*) should be recognizable.
- Flow control: The sender should not send frames at a rate faster than the receiver can receive and process them.
- Error control: Any bit errors introduced by the transmission system should be corrected.

Now, why do we require any controls in the main data link? These are the main functions of the data link layer. Actually, all the combinations are not used in all the situations, but these are the general categorizations of data link. The frame synchronization: the beginning and end of a data block, that means, a frame, should be recognizable. That means when you are sending number of bytes from one machine to other, they are formed into blocks. At that time, the other machine should be able to recognize the beginning and end of the block. So you have to organize these bytes into frames. That is the first job of the data link layer. Second job of data link layer that it might or might not do is flow control. The sender should not send frames at a rate faster than the receiver can receive and process them, because the sender may not know everything about the state of the receiver at that particular point of time. If it goes on sending data, the receiver may be forced to drop some of the data. In some cases, this is ignored. But in some cases, it is relevant and the data link layer may do some floor control. The third thing is error control, where any bit errors introduced by the transmission system should be corrected. Whenever you are sending something through a transmission medium there is always a chance of some error, and the data link layer has to take care of this error.

(Refer Slide Time: 06: 24)

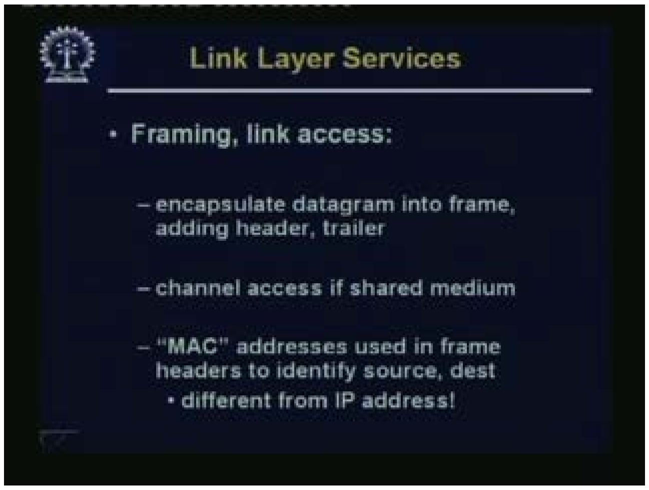


Why Data Link Control?

- Addressing: On a multipoint link (line LAN), the identity of sender and receiver must be specified.
- Control and data on same link: The receiver must be able to distinguish control information from the data being transmitted.
- Link management: Procedures for the management of the initiation, maintenance, and termination of a sustained data exchange

Addressing: on a multipoint link like LAN the identity of sender and receiver must be specified. In such a case, the identity of the receiver would be very important because in many such links, especially in broadcast links, what happens is that the data reaches everybody. The receiving station must know whether or not one of these frames is meant for him. The addressing of receiver is important for that reason. For all the above points, we require some data link layer protocol to run between the two layers. In order for this protocol to run, these two nodes need to have some kind of control or management information. The control and management information will be transmitted on the same medium. Control and data flow on the same link. The receiver must be able to distinguish control information from the data being transmitted. We also require link management – the procedures for the management of the initiation, maintenance and termination of a sustained data exchange.

(Refer Slide Time: 08:08)

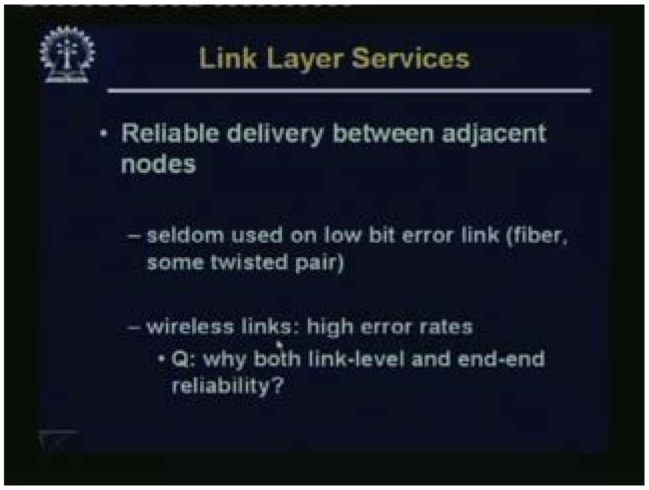


Link Layer Services

- Framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!

These are the different functions of the data link layer. The link layer services are framing and link access. It encapsulates datagram into frame, adding header or trailer. What the header or trailer contains depends on the protocol that you are using on that link; some do not have a trailer, some do not have header; some have both header and trailer. This is the part that is used for the protocol to run. Channel access is also there if it is a shared medium. MAC protocol is a part of it; if it is a shared medium, the medium access control has to be used. MAC addresses are used in frame headers to identify source and destination; it is different from another address we have over the entire network, called the IP address. Right now, we will be concentrating on MAC addresses.

(Refer Slide Time: 09:23)

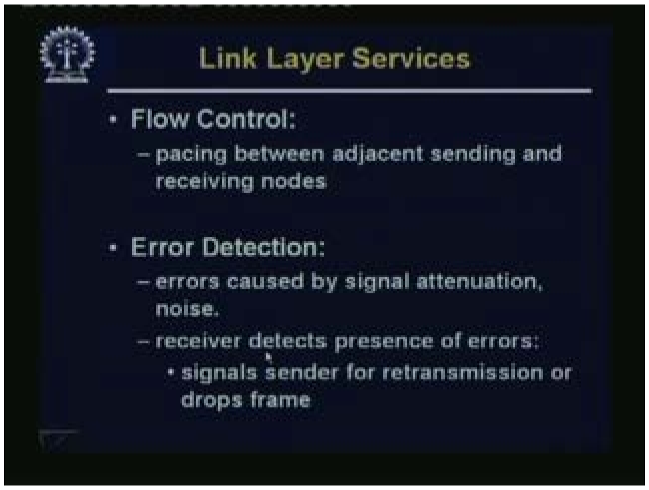


Link Layer Services

- Reliable delivery between adjacent nodes
  - seldom used on low bit error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?

Link layer services offer reliable delivery between adjacent nodes. If the medium is prone to noise or introduction to errors, the data link layer has to address the problem. Apart from this link, you may require resilience at a higher link because a particular node of the network link might fail. So, whatever protocol you might have for the data link might not work, so the chain may break down. This is for making individual links of the chain as good as possible. This is one of its jobs – reliable data delivery. Just as fiber may have low error bit, similarly, some links like wireless links have high error rates. We have just discussed why it gives both link-level and end-end reliability.

(Refer Slide Time: 10:59)

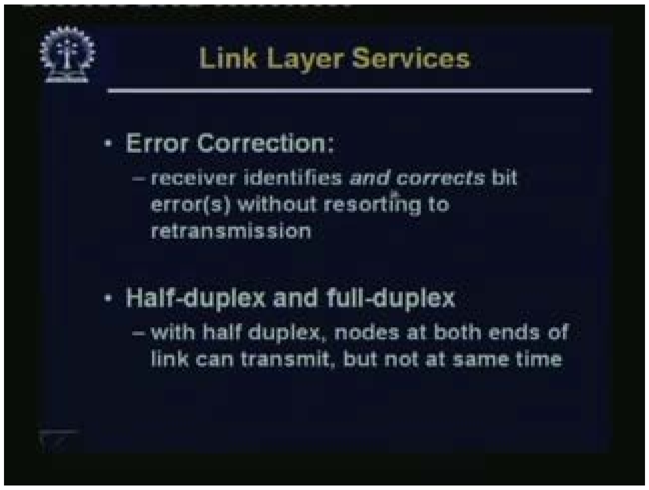


Link Layer Services

- Flow Control:
  - pacing between adjacent sending and receiving nodes
- Error Detection:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame

The flow control is pacing between the adjacent sending and receiving nodes. They should always be in sync, whatever the sending station sends, the receiving station should be able to absorb that much. For that, you may require some explicit controls sometimes. We will look at floor control later on, in a later lecture. The other service is error detection, which includes errors caused by signal attenuation and noise. Receiver detects presence of errors; that is the error detection part. Once you detect that there has been some error, you may ask for some retransmission from the other side so that you may get the correct data, or you might be able to look at that faulty data and correct it locally. The receiver identifies and corrects bit errors without resorting to retransmission. All these would happen on the line, and the line could be simplex, half-duplex or full-duplex. Simplex is not very common, because simplex can go only in one direction.

(Refer Slide Time: 12:16)

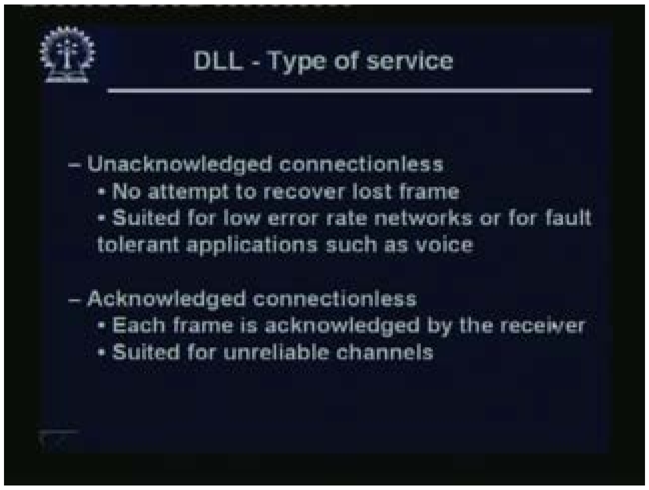


Link Layer Services

- Error Correction:
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- Half-duplex and full-duplex
  - with half duplex, nodes at both ends of link can transmit, but not at same time

In half-duplex nodes at both ends of link can transmit, but not at the same time.

(Refer Slide Time: 12:47)

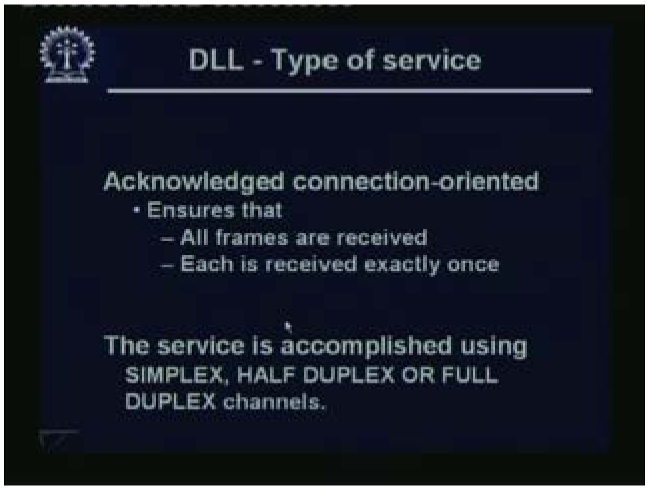


DLL - Type of service

- Unacknowledged connectionless
  - No attempt to recover lost frame
  - Suited for low error rate networks or for fault tolerant applications such as voice
- Acknowledged connectionless
  - Each frame is acknowledged by the receiver
  - Suited for unreliable channels

DLL offers unacknowledged connectionless and acknowledged connectionless services. In unacknowledged connectionless, there is no attempt to recover lost frame and there is no acknowledgement from the other side. It is suited for low error rate networks or for fault tolerant applications such as voice. By voice tolerant application, we mean that even if some of the bits in a digitized voice stream drop, there will be some degradation on the other side. But to the human ear, it is imperceptible. That is why it is fault-tolerant. In acknowledged connectionless service, each frame is acknowledged by the receiver and it is suited for unreliable channels, where acknowledgement is required for special reliability.

(Refer Slide Time: 14:17)



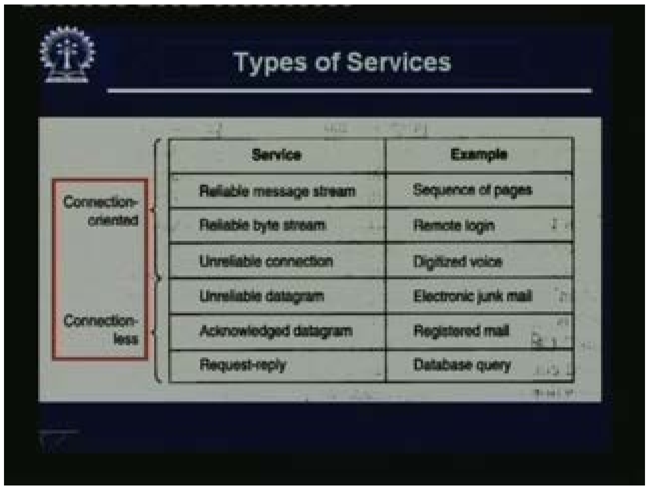
DLL - Type of service

Acknowledged connection-oriented
- Ensures that
  - All frames are received
  - Each is received exactly once

The service is accomplished using SIMPLEX, HALF DUPLEX OR FULL DUPLEX channels.

Acknowledged connection-oriented service ensures that all frames are received and each is received exactly once and these services are accomplished using simplex not the usual, but half-duplex or full-duplex channels.

(Refer Slide Time: 14:32)



Types of Services

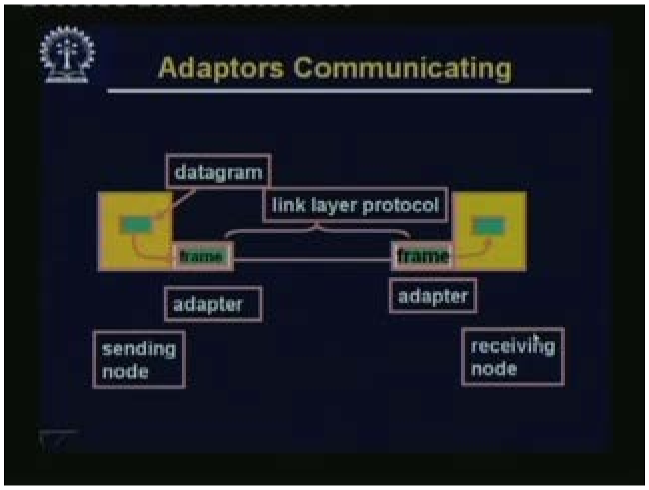
Connection-oriented

Connection-less

Service	Example
Reliable message stream	Sequence of pages
Reliable byte stream	Remote login
Unreliable connection	Digitized voice
Unreliable datagram	Electronic junk mail
Acknowledged datagram	Registered mail
Request-reply	Database query

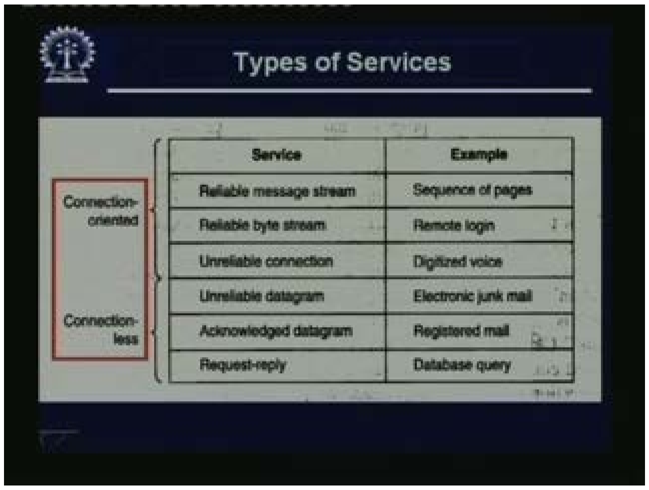
These are some examples. It is a reliable message stream. It may be connection-oriented service or connectionless service. It may be a reliable message stream (sequence of pages) or reliable byte stream (reliable login): in the latter it is coming byte by byte and in the former, it is page by page. An example of unreliable connection is digitized voice; unreliable datagram (electronic junk mail) is connectionless service.

(Refer Slide Time: 14:37)



The data layer link exists in the network interface card. If you have a PC that is connected to a network, it is probably connected through an Ethernet card, or Network Interface Card (NIC) or something like that. The card has a socket where the Ethernet cable will be plugged in. So, there is a network adaptor. The adaptors implement most of these data link functions, so it is the adaptors that are communicating. The datagram from a higher layer is made into a frame by the adaptor and it is then sent, following the link layer protocol, to the other node. The adaptors communicating have two nodes: one sending node and one receiving node.

(Refer Slide Time: 14:39)

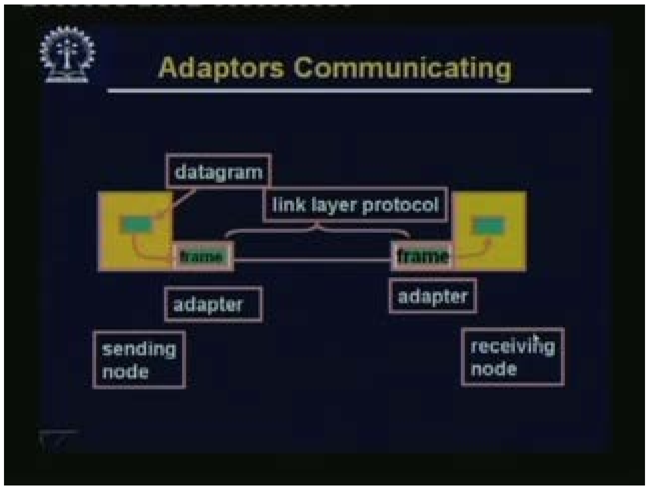


Types of Services

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connection-less	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Registered mail
	Request-reply	Database query

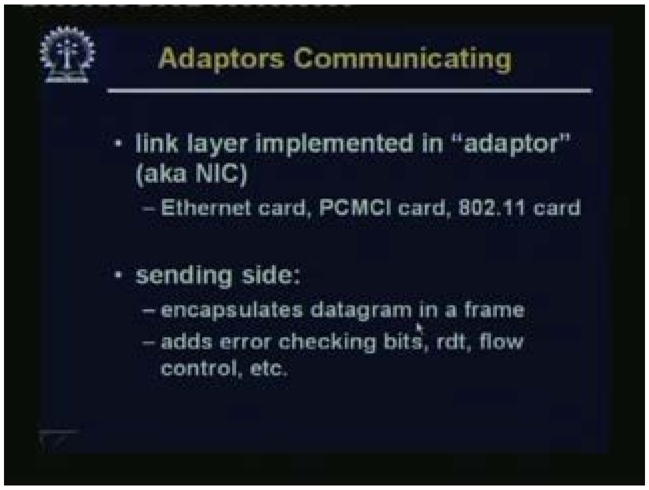
The link layer in adaptor is also called NIC, examples of which are the Ethernet card, PCMCI card or the 802.11 card. On the sending side, the adaptor encapsulates the datagram in a frame. It adds error-checking bits, rdt, flow control, etc. On the receiving side, it looks for errors, rdt, flow control, etc., extracts datagram and passes to the receiving node. The adaptor is semi-autonomous and communicates directly with link and physical layers. The adaptor has some hardware and some in-built software.

(Refer Slide Time: 15:15)



Adaptors Communicating

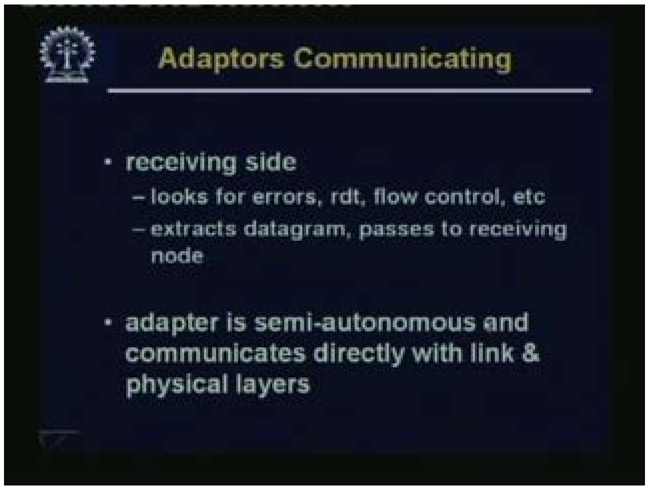
(Refer Slide Time: 16:49)



Adaptors Communicating

- link layer implemented in "adaptor" (aka NIC)
  - Ethernet card, PCMCI card, 802.11 card
- sending side:
  - encapsulates datagram in a frame
  - adds error checking bits, rdt, flow control, etc.

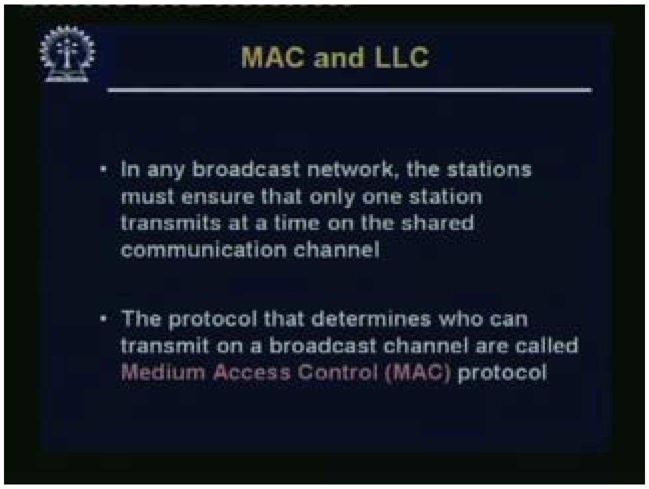
(Refer Slide Time: 17:20)



Adaptors Communicating

- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to receiving node
- adapter is semi-autonomous and communicates directly with link & physical layers

(Refer Slide Time: 17:59)

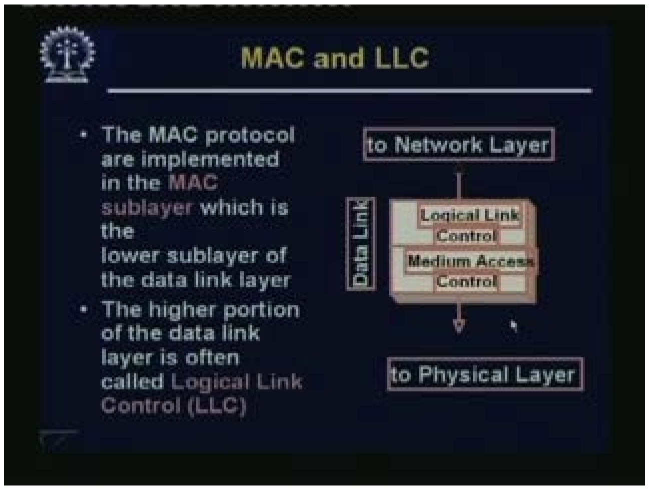


MAC and LLC

- In any broadcast network, the stations must ensure that only one station transmits at a time on the shared communication channel
- The protocol that determines who can transmit on a broadcast channel are called Medium Access Control (MAC) protocol

The data link layer is divided into two parts. One is the MAC and the other is the LLC. So in any broadcast network, the stations must ensure that only one station transmits at a time on the shared communication channel. That is the MAC part of it. The protocol that determines who can transmit on a broadcast channel is called Medium Access Control (MAC) protocol. We have seen a number of MAC protocols already.

(Refer Slide Time: 18:43)

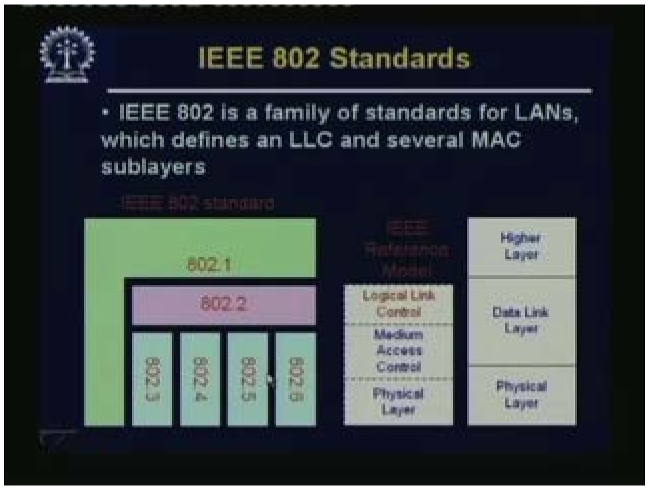


MAC and LLC

- The MAC protocol are implemented in the MAC sublayer which is the lower sublayer of the data link layer
- The higher portion of the data link layer is often called Logical Link Control (LLC)

The data link layer is divided into two sublayers. Above the data link layer, you see two network layers and below it is the physical layer. The data link layer itself is divided into two parts: the medium access control part, which is closer to the physical layer, and the logical link control. The MAC protocols are implemented in the MAC sublayer, which is the lower sublayer of the data link layer. The higher portion of the data link layer is often called logical link control or LLC.

(Refer Slide Time: 19:27)

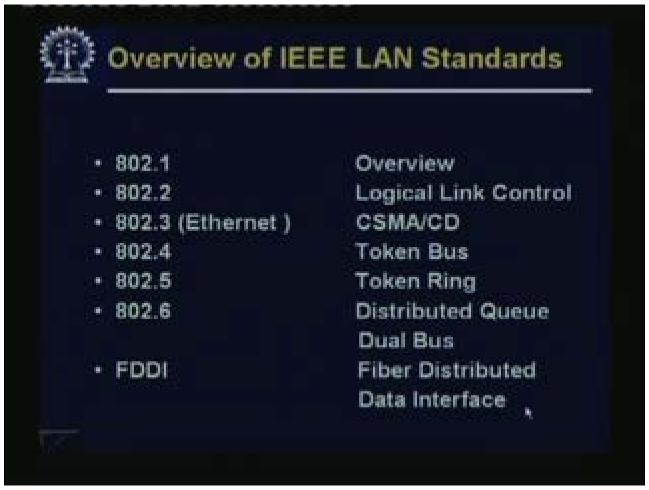


IEEE 802 Standards

- IEEE 802 is a family of standards for LANs, which defines an LLC and several MAC sublayers

This is the broad picture: we have been referring to some numbers like 802.1, 2, 3, 4 etc. So IEEE 802 is a family of standards for LANs, which define an LLC and several MAC sublayers. So 802 encompass all these. This 802.2 is the LLC part and below the 802.2 there are various kinds of medium access controls. There are 802.3, 4, 5, 6 and then 10, 11, 12 etc., and now we have 15, 16, etc. Above the data link layer, there is a higher layer and below that, there is the physical layer.

(Refer Slide Time: 20:21)



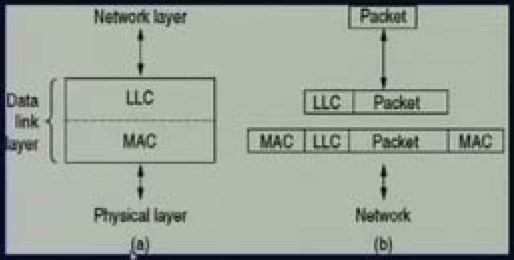
Overview of IEEE LAN Standards

• 802.1	Overview
• 802.2	Logical Link Control
• 802.3 (Ethernet)	CSMA/CD
• 802.4	Token Bus
• 802.5	Token Ring
• 802.6	Distributed Queue Dual Bus
• FDDI	Fiber Distributed Data Interface

802.1 gives you an overview; 802.2 is the LLC we will be talking about today. 802.3 is the famous Ethernet; CSMA/CD is the kind of MAC protocol that it uses. 802.4 is the token bus, which we have already seen. 802.5 is the token ring. 802.6 is the distributed queue dual bus. FDDI is the fiber distributed data interface and there are others. As new protocols come up, they keep adding to this list.

(Refer Slide Time: 21:06)

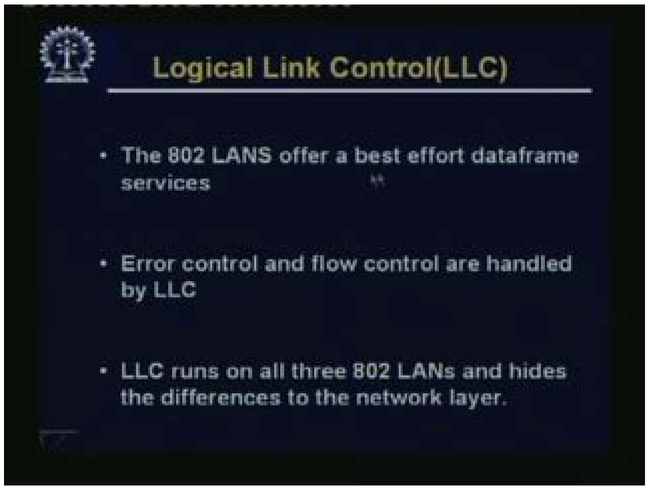
IEEE 802.2: Logical Link Control



(a) Position of LLC. (b) Protocol formats.

LLC, whatever it does, it requires some headers. So when the packet is coming from the network layer, the LLC header is added to the packet. So it will reach the LLC sublayer on the other side. Then it comes to the MAC sublayer and MAC sublayer will add its header and it may add some trailer also. The MAC, LLC, and original packet may constitute one frame and it is pushed on to the physical layer in the network.

(Refer Slide Time: 22:01)

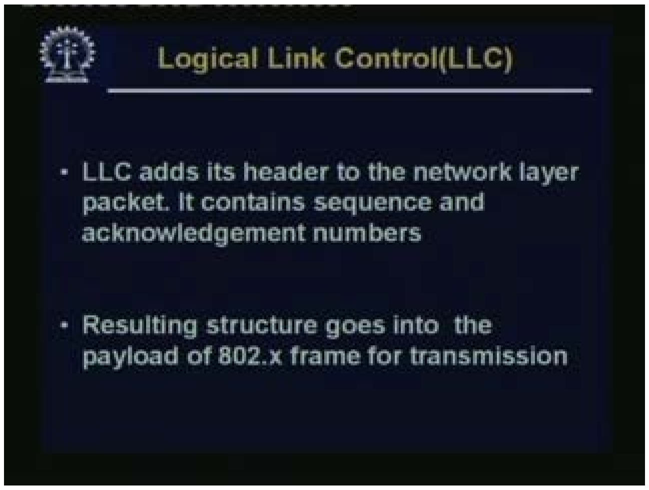


Logical Link Control(LLC)

- The 802 LANS offer a best effort dataframe services
- Error control and flow control are handled by LLC
- LLC runs on all three 802 LANs and hides the differences to the network layer.

The 802 LANs offer the best effort data frame services. Error control and flow control are handled by LLC. LLC runs on all the three 802 LANs and hides the differences to the network layer.

(Refer Slide Time: 23:04)

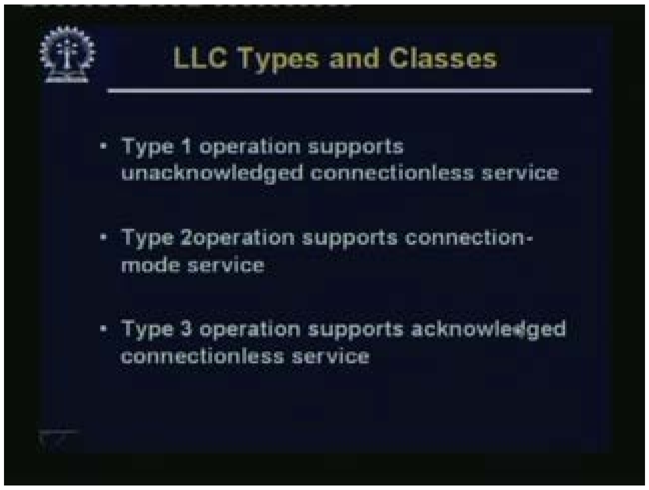


Logical Link Control(LLC)

- LLC adds its header to the network layer packet. It contains sequence and acknowledgement numbers
- Resulting structure goes into the payload of 802.x frame for transmission

The different physical layer protocols are transparent to the network. All that the data link network knows is that this network layer will provide a reliable service for sending the data from this node to the next. LLC adds its header to the network layer packet. It contains sequence and acknowledgment numbers. Resulting structure goes into the payload of 802.x frame for transmission.

(Refer Slide Time: 23:18)

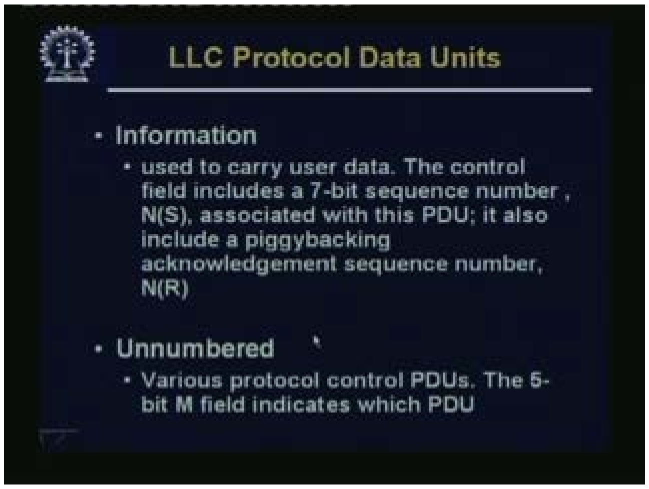


LLC Types and Classes

- Type 1 operation supports unacknowledged connectionless service
- Type 2operation supports connection-mode service
- Type 3 operation supports acknowledged connectionless service

LLC operations are sometimes divided in this fashion. Type 1 operation supports un acknowledgement connectionless service. Type 2 operation supports connection mode service. Type 3 operation supports acknowledged connectionless service.

(Refer Slide Time: 23:47)

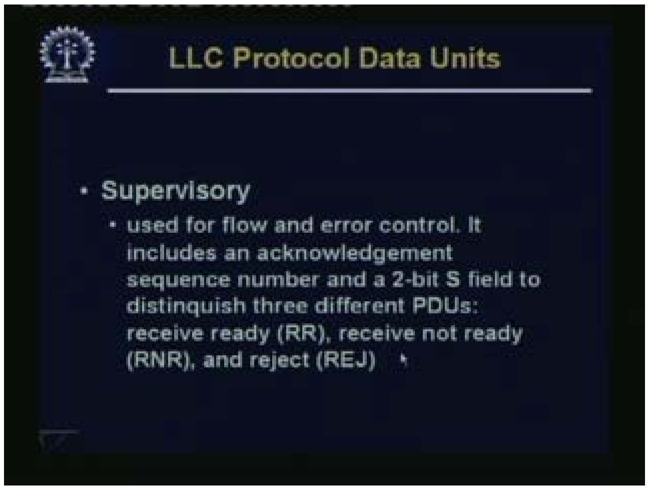


LLC Protocol Data Units

- Information
  - used to carry user data. The control field includes a 7-bit sequence number , N(S), associated with this PDU; it also include a piggybacking acknowledgement sequence number, N(R)
- Unnumbered
  - Various protocol control PDUs. The 5-bit M field indicates which PDU

The LLC has protocol data units or PDU, which carries user information. The control field includes a 7-bit sequence number N(S), associated with this PDU. It also includes a piggybacking acknowledgment sequence number N(R). Unnumbered various protocol control PDUs. These five bit M fields indicates a what kind of PDU it is.

(Refer Slide Time: 24:32)

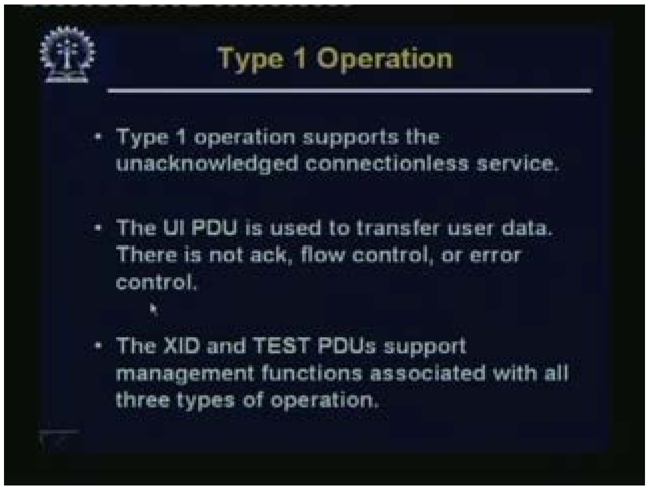


LLC Protocol Data Units

- Supervisory
  - used for flow and error control. It includes an acknowledgement sequence number and a 2-bit S field to distinquish three different PDUs: receive ready (RR), receive not ready (RNR), and reject (REJ)

There are some supervisory PDUs used for flow and error control. It includes an acknowledgment and sequence number and a 2-bit S field to distinguish three different PDUs: receive ready (RR), receive not ready (RNR) and reject (REJ).

(Refer Slide Time: 24:53)

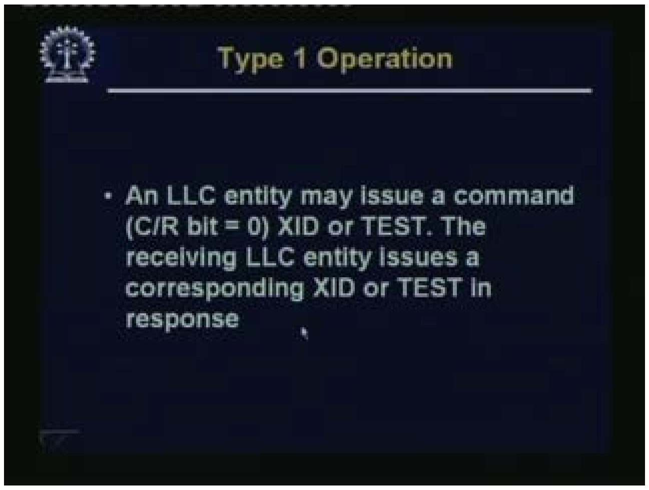


Type 1 Operation

- Type 1 operation supports the unacknowledged connectionless service.
- The UI PDU is used to transfer user data. There is not ack, flow control, or error control.
- The XID and TEST PDUs support management functions associated with all three types of operation.

The type 1 operation supports the unacknowledged connectionless service. The UI PDU is used to transfer user data. There is not acknowledgement, flow control or error control. The XID and TEST PDUs support management functions associated with all the three types of operation.

(Refer Slide Time: 25:20)

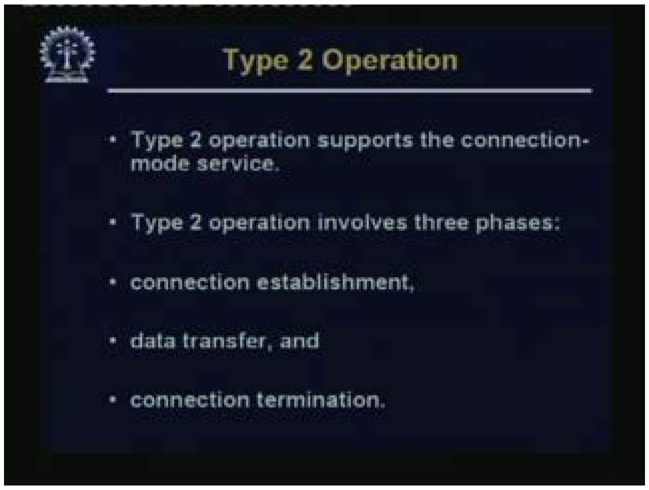


Type 1 Operation

- An LLC entity may issue a command (C/R bit = 0) XID or TEST. The receiving LLC entity issues a corresponding XID or TEST in response

An LLC entity may issue a command XID or TEST. The receiving LLC entity issues a corresponding XID or TEST in response.

(Refer Slide Time: 26:00)

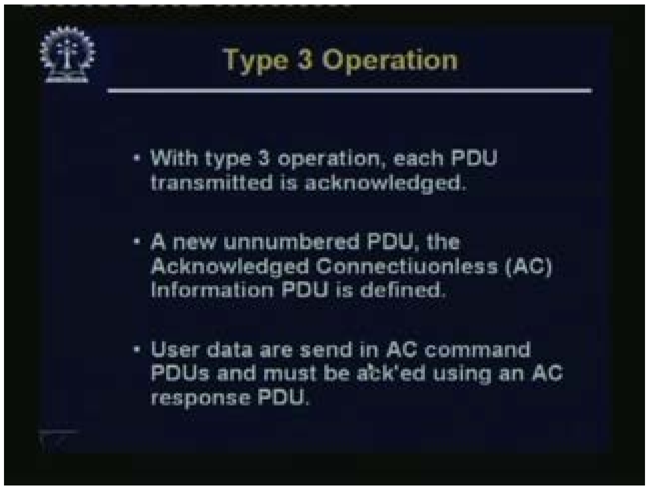


Type 2 Operation

- Type 2 operation supports the connection-mode service.
- Type 2 operation involves three phases:
- connection establishment,
- data transfer, and
- connection termination.

Type 2 operations involve three phases: connection establishment, data transfer and connection termination.

(Refer Slide Time: 26:09)

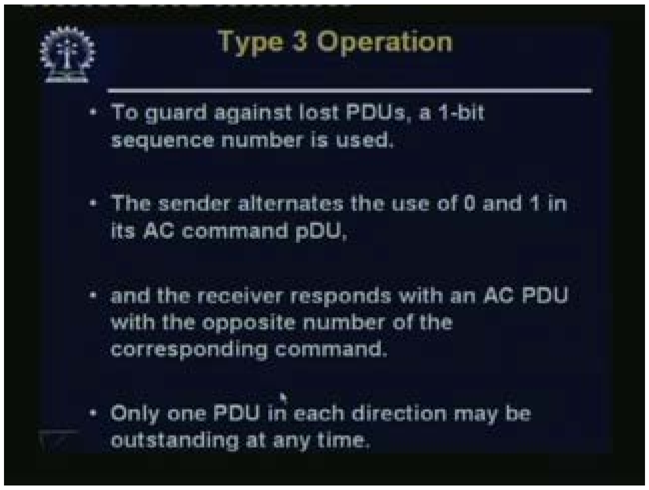


Type 3 Operation

- With type 3 operation, each PDU transmitted is acknowledged.
- A new unnumbered PDU, the Acknowledged Connectiuonless (AC) Information PDU is defined.
- User data are send in AC command PDUs and must be ack'ed using an AC response PDU.

With type 3 operation, each PDU transmitted is acknowledged. A new unnumbered PDU, the acknowledged connectionless (AC) information PDU, is defined. User data are sent in AC command PDUs and must be acknowledged using an AC response PDU.

(Refer Slide Time: 26:46)

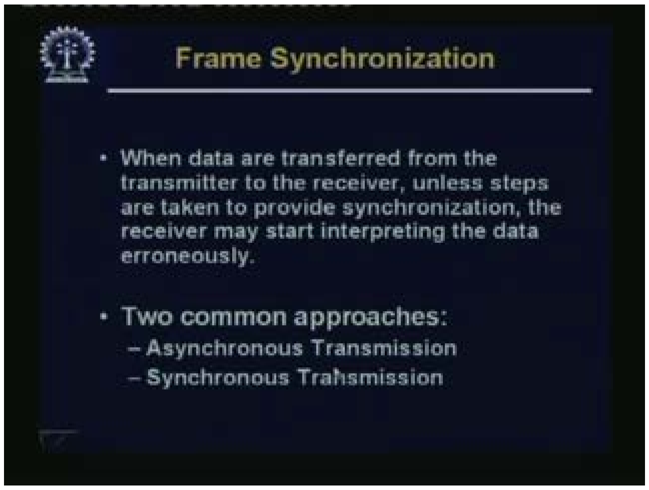


Type 3 Operation

- To guard against lost PDUs, a 1-bit sequence number is used.
- The sender alternates the use of 0 and 1 in its AC command pDU,
- and the receiver responds with an AC PDU with the opposite number of the corresponding command.
- Only one PDU in each direction may be outstanding at any time.

To guard against lost PDUs a 1-bit sequence number is used. The sender alternates the use of 0 and 1 in this 1 bit. The receiver responds with an AC PDU with opposite number of the corresponding command. Only one PDU in each direction may be outstanding at any time.

(Refer Slide Time: 27:46)

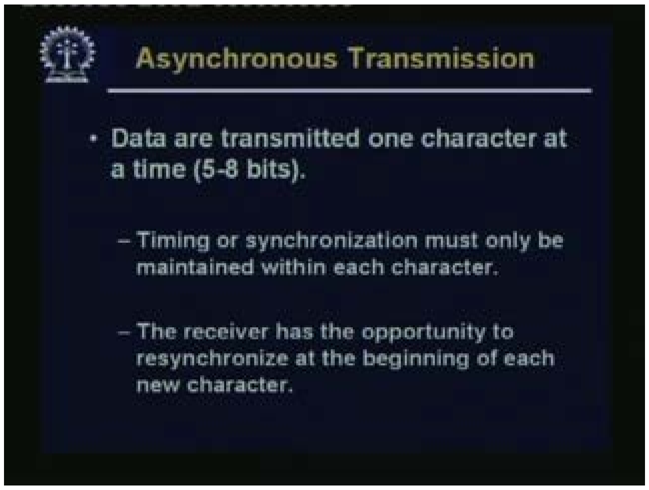


Frame Synchronization

- When data are transferred from the transmitter to the receiver, unless steps are taken to provide synchronization, the receiver may start interpreting the data erroneously.
- Two common approaches:
  - Asynchronous Transmission
  - Synchronous Transmission

Frame Synchronization: Two sides must be able to synchronize their movements. This synchronization is of two types. Suppose you are sending data 1 byte at a time, or you are sending blocks, each block containing a number of bytes. Synchronization of frames is necessary for this. When data are transferred from the transmitted to the receiver unless steps are taken to provide synchronization the receiver may start interpreting the data erroneously. Suppose you have taken the second byte as the first byte, you will never be able to know that it is not the first byte. There are two common approaches: Asynchronous Transmission and Synchronous Transmission.

(Refer Slide Time: 29:17)

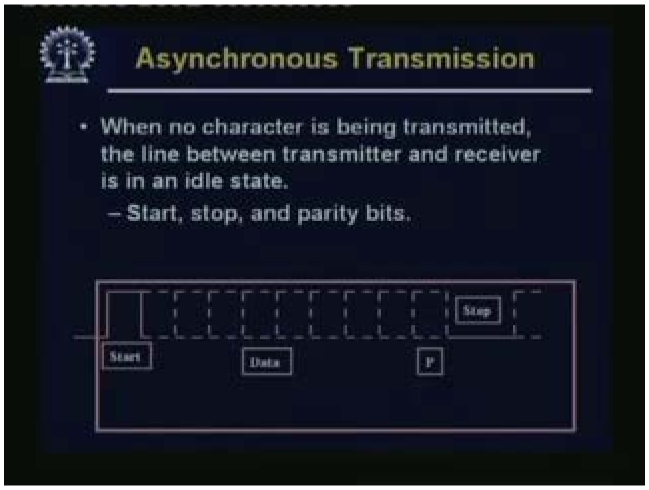


Asynchronous Transmission

- Data are transmitted one character at a time (5-8 bits).
  - Timing or synchronization must only be maintained within each character.
  - The receiver has the opportunity to resynchronize at the beginning of each new character.

In asynchronous transmission, data are transmitted one character at a time. Timing of synchronization must only be maintained within each character. The receiver has the opportunity to resynchronize at the beginning of each new character. If you use encoding, you can use the transition time for synchronization between the sending and receiving nodes.

(Refer Slide Time: 30:11)

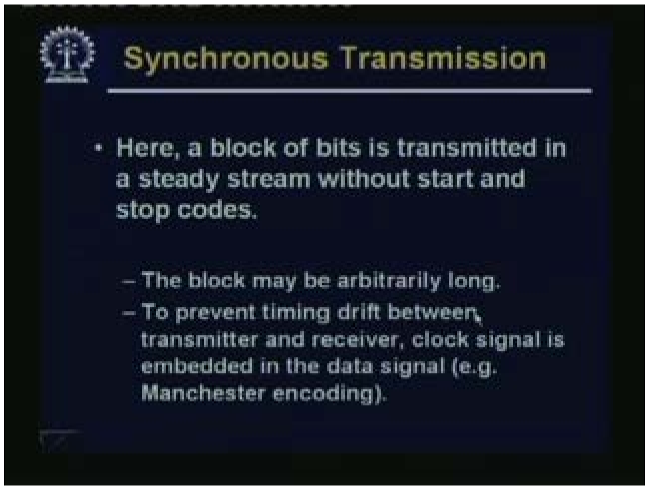


Asynchronous Transmission

- When no character is being transmitted, the line between transmitter and receiver is in an idle state.
  - Start, stop, and parity bits.

So when no character is being transmitted the line between transmitter and receiver is in idle state; so there must be some start and some stop. There is a start after which we can introduce some parity bits – we will see later how parity bits are introduced – and this is the stop.

(Refer Slide Time: 30:42)

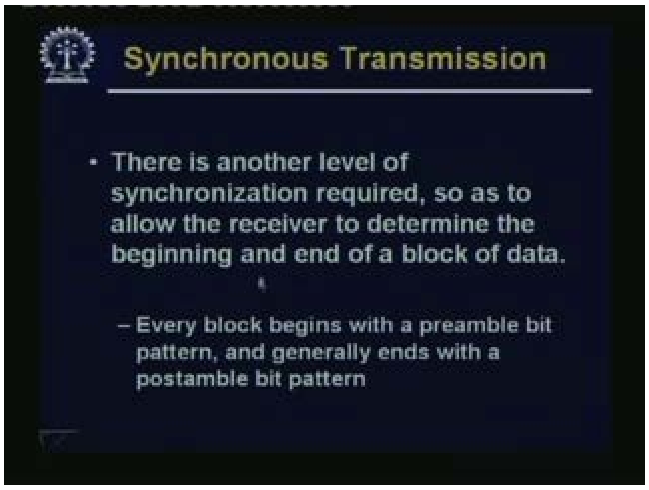


Synchronous Transmission

- Here, a block of bits is transmitted in a steady stream without start and stop codes.
  - The block may be arbitrarily long.
  - To prevent timing drift between transmitter and receiver, clock signal is embedded in the data signal (e.g. Manchester encoding).

In synchronous transmission, a block of bits is transmitted in a steady stream without start and stop codes. Actually asynchronous transmission is not as efficient. The block may be arbitrarily long. To prevent timing drift between the transmitter and receiver, clock signal is embedded in the data signal. (E.g. Manchester encoding)

(Refer Slide Time: 31:47)

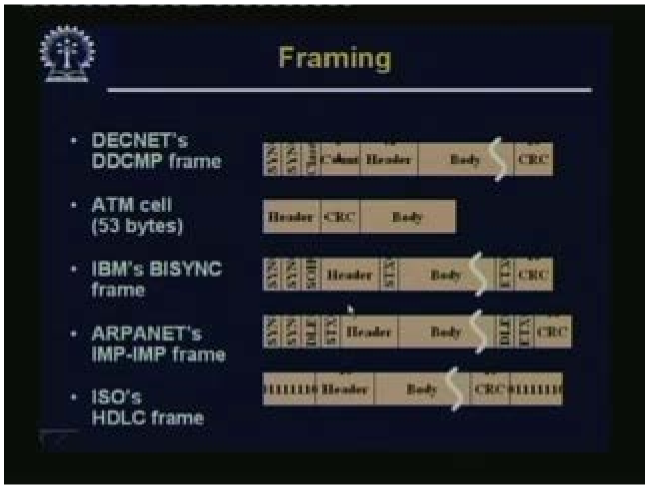


Synchronous Transmission

- There is another level of synchronization required, so as to allow the receiver to determine the beginning and end of a block of data.
  - Every block begins with a preamble bit pattern, and generally ends with a postamble bit pattern

Apart from this synchronization of clock for the bit, you require another level of synchronization, so as to allow the receiver to determine the beginning and end of a block of data. Every block begins with a preamble bit pattern, and generally ends with a postamble bit pattern. The kind of preamble and postamble that are used is directly related to the kind of protocol that is used.

(Refer Slide Time: 32:36)

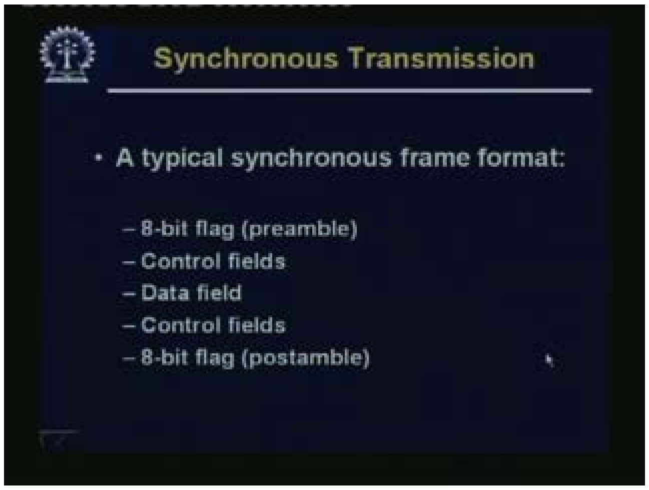


Framing

DECNET's DDCMP frame
SYN SYN Class Count Header Body CRC
ATM cell (53 bytes)
Header CRC Body
IBM's BISYNC frame
SYN SYN SOH Header STX Body ETX CRC
ARPANET's IMP-IMP frame
SYN SYN DLE STX Header Body DLE ETX CRC
ISO's HDLC frame
01111110 Header Body CRC 01111110

There are many kinds of framing available. You can observe that only the body is coming from the higher layer; the rest of it is being added in this layer. For example, DECNET's DDCMP frame has SYNs, header, body and many other bits are there. The ATM cell has only the header, CRC and the body. IBMs have a BISYNC frame, header, body and bits. The ARPANET's IMP-IMP frame has SYN, header and body bits. The ISO's HDLC frame looks like this: header pattern, body and CRC, and again some specific pattern.

(Refer Slide Time: 33:56)

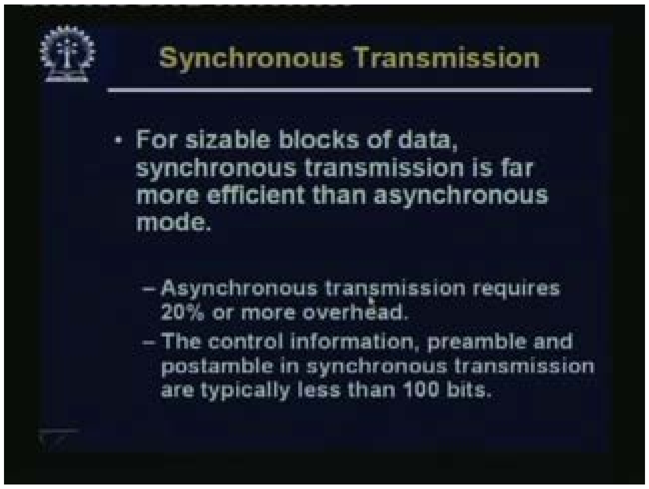


Synchronous Transmission

- A typical synchronous frame format:
  - 8-bit flag (preamble)
  - Control fields
  - Data field
  - Control fields
  - 8-bit flag (postamble)

A typical synchronous frame format would have an 8-bit flag, which would be the preamble, control fields, data field and some more control fields and an 8-bit flag (post amble).

(Refer Slide Time: 34:16)

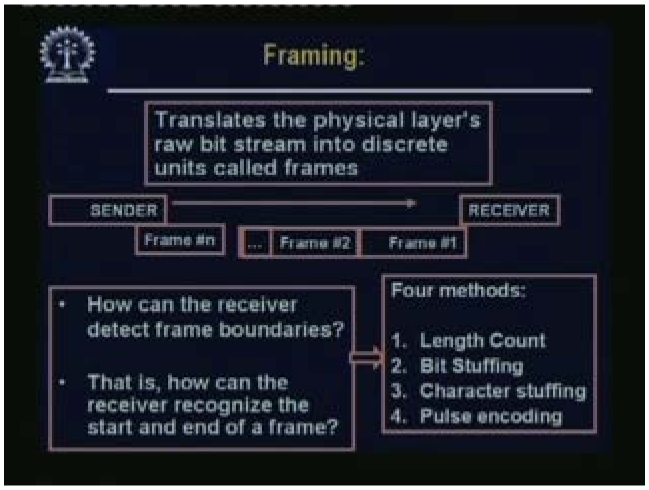


Synchronous Transmission

- For sizable blocks of data, synchronous transmission is far more efficient than asynchronous mode.
  - Asynchronous transmission requires 20% or more overhead.
  - The control information, preamble and postamble in synchronous transmission are typically less than 100 bits.

For sizeable blocks of data, synchronous transmission is far more efficient than asynchronous mode. Asynchronous transmission requires 20% or more of overhead. The control information preamble and postamble in synchronous transmission are typically less than 100 bits. The overhead is low here that is why the efficiency of synchronous transmission is very high. So when you are sending large amount of data, you go for synchronous transmission, because it is efficient.

(Refer Slide Time: 34:54)



Framing:

Translates the physical layer's raw bit stream into discrete units called frames

SENDER
RECEIVER
Frame #n
...
Frame #2
Frame #1

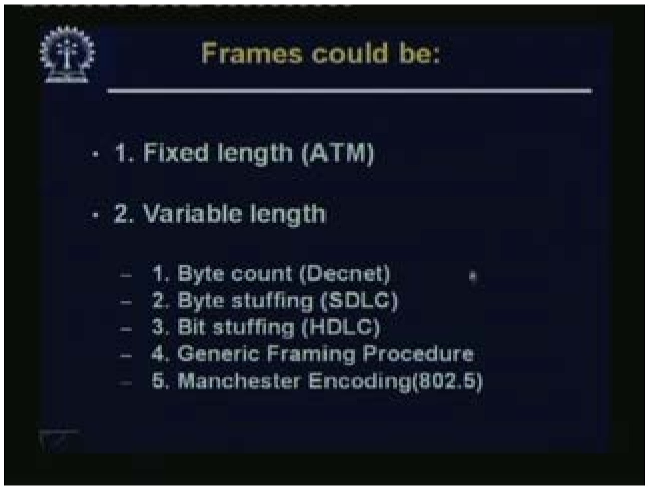
- How can the receiver detect frame boundaries?
- That is, how can the receiver recognize the start and end of a frame?

Four methods:

1. Length Count
2. Bit Stuffing
3. Character stuffing
4. Pulse encoding

Framing translates the physical layer's raw bit stream into discrete units called frames. The sender sends the message, which is transmitted in the form of frames. N frames are on transit and they are going from the sender to the receiver. Now how can the receiver detect frame boundaries? That is, how can the receiver recognize the start and end of a frame? This is done by four methods: length count, bit stuffing, character or bit stuffing and pulse encoding. We will look at some of these now.

(Refer Slide Time: 35:35)

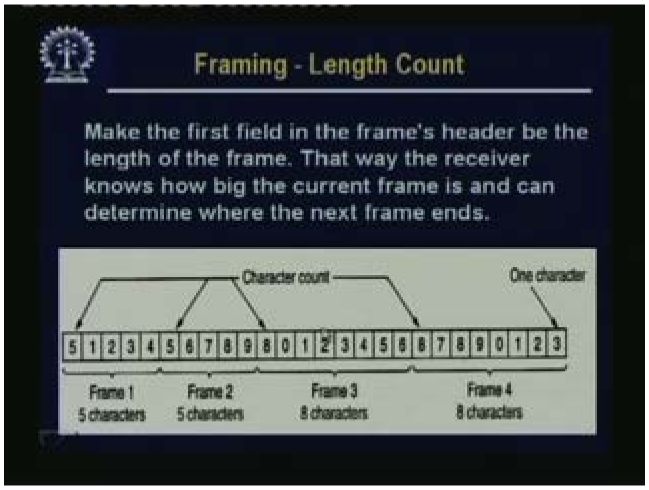


Frames could be:

- 1. Fixed length (ATM)
- 2. Variable length
  - 1. Byte count (Decnet)
  - 2. Byte stuffing (SDLC)
  - 3. Bit stuffing (HDLC)
  - 4. Generic Framing Procedure
  - 5. Manchester Encoding(802.5)

Frames could be of fixed length, like ATM. When it is ATM, you know that it is of 53-byte kind of regularity. They could be of variable length also, in which case we use the byte count, byte stuffing, bit stuffing, generic framing procedure and Manchester encoding. ATM is a kind of fixed length frame. Variable lengths are byte count (DECNET), byte stuffing (SDLC), bit stuffing (HDLC), generic framing procedure, and Manchester encoding (802.5).

(Refer Slide Time: 36:05)



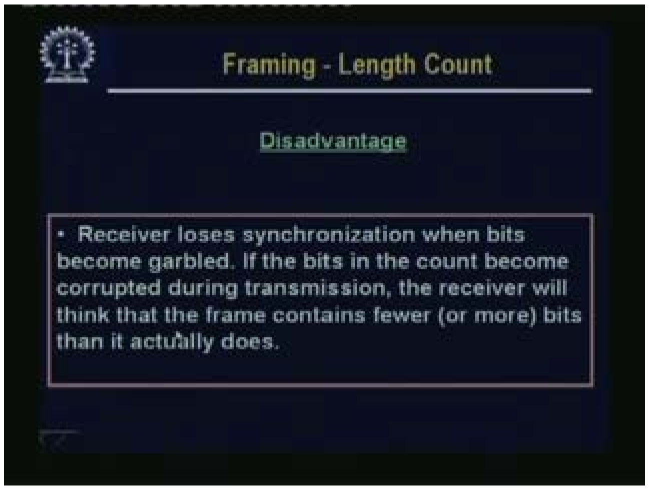
Framing - Length Count

Make the first field in the frame's header be the length of the frame. That way the receiver knows how big the current frame is and can determine where the next frame ends.

Now in framing, we make the first field in the frame's header as the length of the frame. That way the receiver knows how big the current frame is and can determine when the next frame ends.

From the above slide, we can see that frame 1 contains 5 characters, frame 2 contains 5 characters, frame 3 contains 8 characters and frame 4 contains 8 characters.

(Refer Slide Time: 36:54)



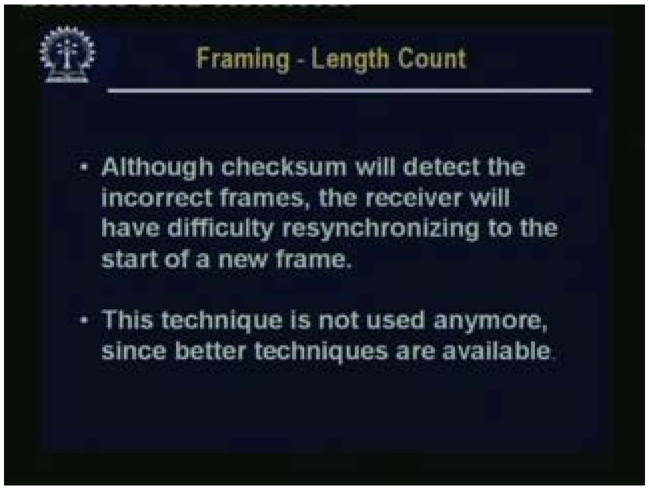
Framing - Length Count

Disadvantage

- Receiver loses synchronization when bits become garbled. If the bits in the count become corrupted during transmission, the receiver will think that the frame contains fewer (or more) bits than it actually does.

Here the disadvantage is that the receiver loses synchronization, when bits become garbled. If the bits in the count become corrupted during transmission, the receiver will think that the frame contains fewer (or more) bits than it actually does.

(Refer Slide Time: 37:56)

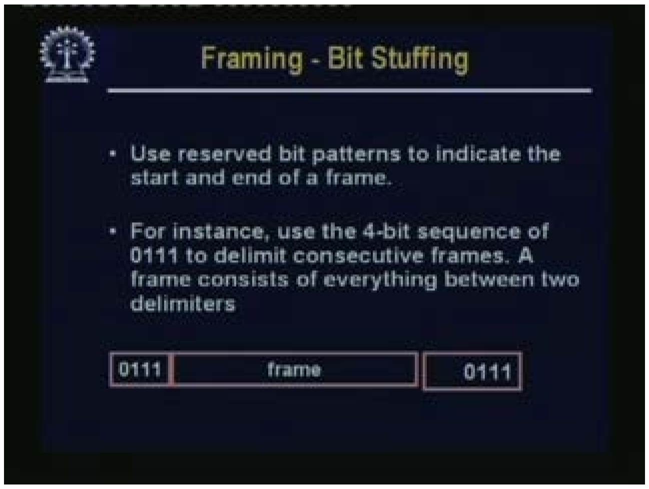


Framing - Length Count

- Although checksum will detect the incorrect frames, the receiver will have difficulty resynchronizing to the start of a new frame.
- This technique is not used anymore, since better techniques are available.

Checksum will detect the incorrect frames; the receiver will have difficulty resynchronizing to the start of a new frame. This technique is not used anymore, since better techniques are available.

(Refer Slide Time: 38:18)



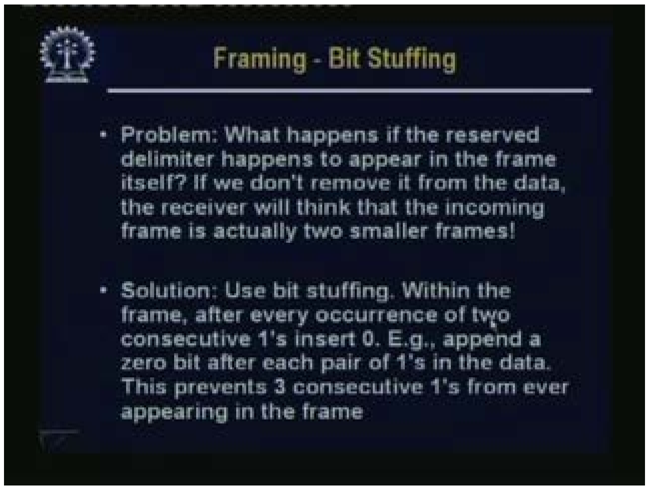
Framing - Bit Stuffing

- Use reserved bit patterns to indicate the start and end of a frame.
- For instance, use the 4-bit sequence of 0111 to delimit consecutive frames. A frame consists of everything between two delimiters

0111
frame
0111

One of the better techniques is known as bit stuffing. Use reserved bit patterns to indicate the start and end of a frame. For instance, use the 4-bit sequence of 0111 to delimit consecutive frames. A frame consists of everything between two delimiters. So you have this one delimiter on one side, 0111, and then the frame and then the 011. So as soon as you get 011, you know that the frame is starting and as soon as you get another 011, you know that the frame has ended. So this way we can know the beginning or the end of the frame.

(Refer Slide Time: 39:05)

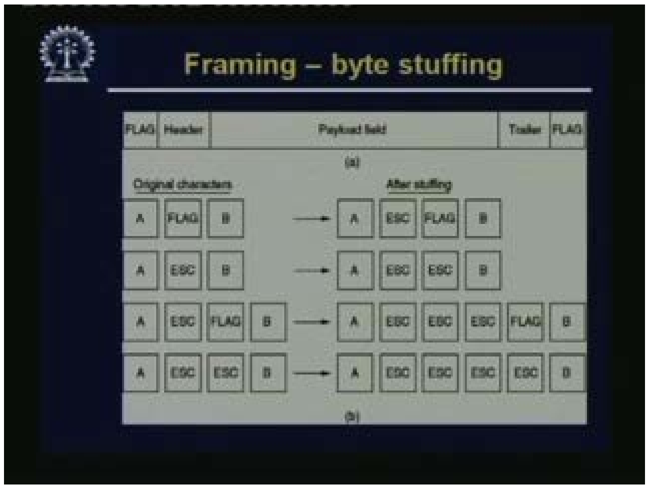


Framing - Bit Stuffing

- Problem: What happens if the reserved delimiter happens to appear in the frame itself? If we don't remove it from the data, the receiver will think that the incoming frame is actually two smaller frames!
- Solution: Use bit stuffing. Within the frame, after every occurrence of two consecutive 1's insert 0. E.g., append a zero bit after each pair of 1's in the data. This prevents 3 consecutive 1's from ever appearing in the frame

The problem with this is as follows: what happens if the reserved delimiter happens to appear in the frame itself? If we do not remove it from the data, the receiver will think that the incoming frame is actually two smaller frames. Suppose we have the 0111 as the delimiter and the delimiter may contain data that came from the user in any bit pattern. You have to allow any bit pattern to the user. It could be that a picture is being sent in several bits and the bit pattern may be arbitrary. In that case, 0111 may appear in the body of the data; this is where the bit stuffing part comes. We introduce a new set of pattern, say, 0111, for each existing pattern. So now, the solution is to use bit stuffing. Within the frame, after every occurrence of two consecutive 1s, insert a 0. For example, append a 0 bit after each pair of 1s in the data. This prevents three consecutive 1s from ever appearing in the frame.

(Refer Slide Time: 41:54)



Framing – byte stuffing

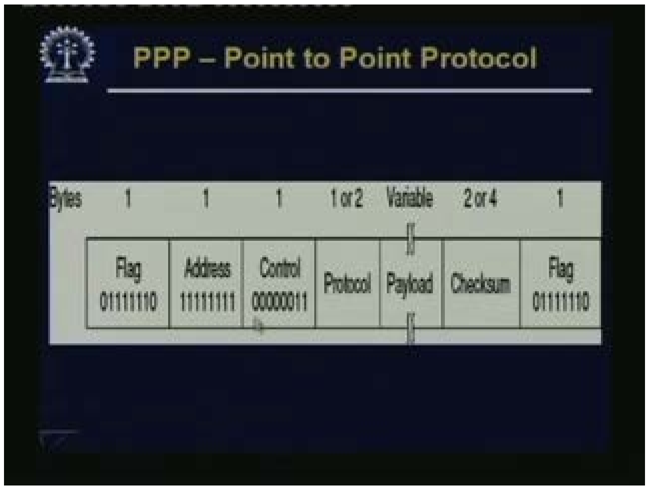
Similar to bit stuffing we may have byte stuffing. For example, let us say a flag say some character is there. So, say the flag, which is also a part of the regular header, just happens to appear in the body of the frame or body of the packet. So what we do is that, we use this other character. These are character introductions, it is called byte stuffing. One byte is one character. So we introduce the character, escape character, just before the flag, and what happens if escape itself appears in the body? Well, we put escape. So if there are two escapes side by side we know that we have to interpret it as only one escape. If there are two escapes in the original data packet, just by chance, then actually this will be center 4 escapes and just one after the other, and at the receiving side, for every 2 escapes, it will reduce it to one escape and know that this is the just a part of the data. Only at the end, we will get flag, etc., bytes. If there is escape flag, we have escape escape escape flag escape escape and so on. So this is known as byte stuffing.

(Refer Slide Time: 43:26)

Point to Point Protocol

Point to point protocol.

(Refer Slide Time: 43:58)

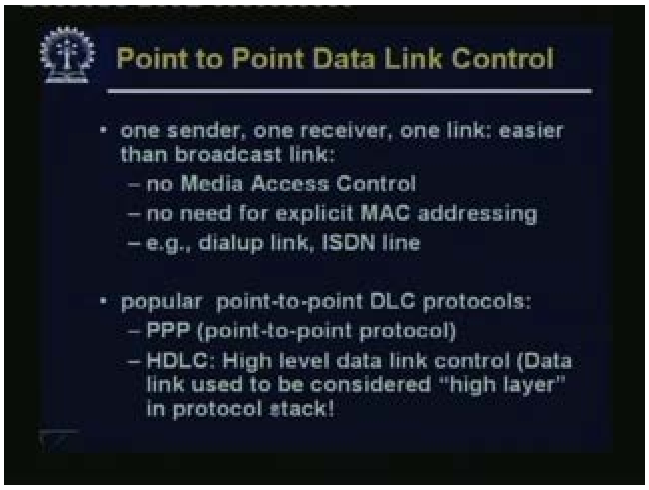


PPP – Point to Point Protocol

Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

We will now discuss point-to-point protocol. There will be a flag field, address field, control field, protocol field, and payload field. This payload is the one which is actually coming from the higher layers. That means from the network layer, some of it will actually come from the user, from the application layer itself. This is the payload so far as the data link layer is concerned; it ends with Checksum and then flag. This is what the general things look like and we will come to discussing how they are used.

(Refer Slide Time: 44:41)

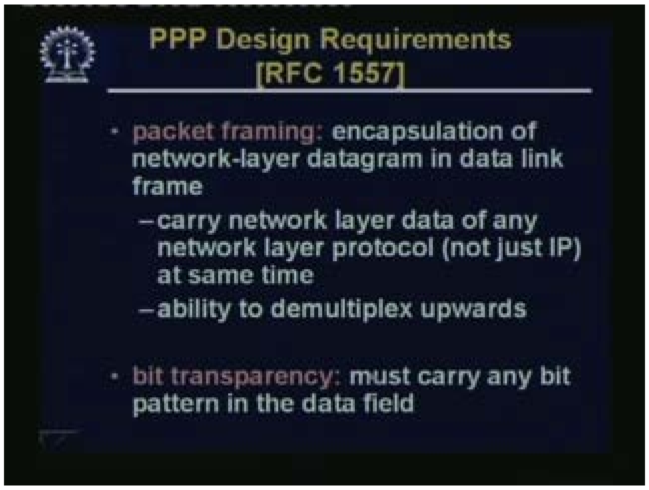


Point to Point Data Link Control

- one sender, one receiver, one link: easier than broadcast link:
  - no Media Access Control
  - no need for explicit MAC addressing
  - e.g., dialup link, ISDN line
- popular point-to-point DLC protocols:
  - PPP (point-to-point protocol)
  - HDLC: High level data link control (Data link used to be considered "high layer" in protocol stack!

In a point to point data link control, there is one sender, one receiver and one link, which is easier than broadcast link. It has no media access control, no need for explicit MAC addressing e.g. dial-up link and ISDN line. The popular point to point DLC protocols are PPP, which is point to point protocol, and HDLC, which is high level data link control.

(Refer Slide Time: 46:16)

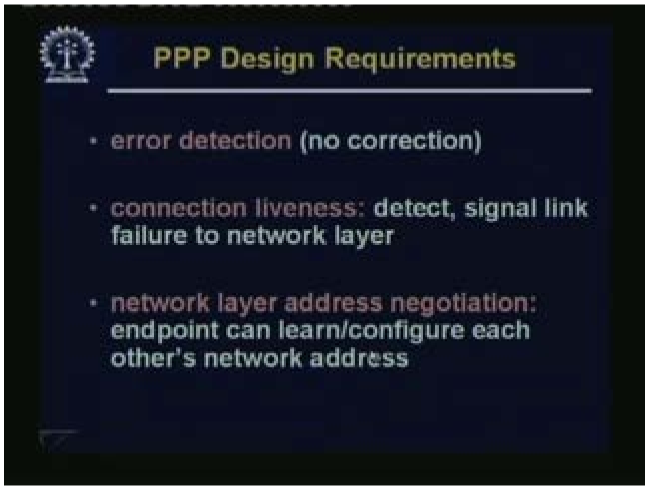


PPP Design Requirements [RFC 1557]

- packet framing: encapsulation of network-layer datagram in data link frame
  - carry network layer data of any network layer protocol (not just IP) at same time
  - ability to demultiplex upwards
- bit transparency: must carry any bit pattern in the data field

In PPP design requirements are given in RFC 1551. RFC stands for Request For Comment and forms a very important part of networking. PPP uses packet framing; its requirements are encapsulation of network layer datagram in data link frame. This carries network layer data of any network layer protocol at the same time. It should have the ability to demultiplex upwards. PPP uses bit transparency also, which means, it must carry any bit pattern in the data field.

(Refer Slide Time: 49:14)

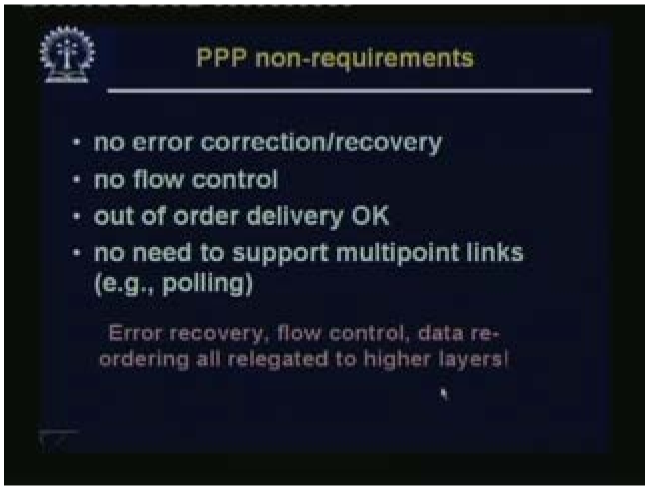


PPP Design Requirements

- error detection (no correction)
- connection liveness: detect, signal link failure to network layer
- network layer address negotiation: endpoint can learn/configure each other's network address

We only require error detection, but no correction at the receiving end. The connection liveness: it should be able to detect, signal link failure to network layer. Network layer address negotiation means endpoint can learn/configure each other's network address.

(Refer Slide Rime: 49:53)



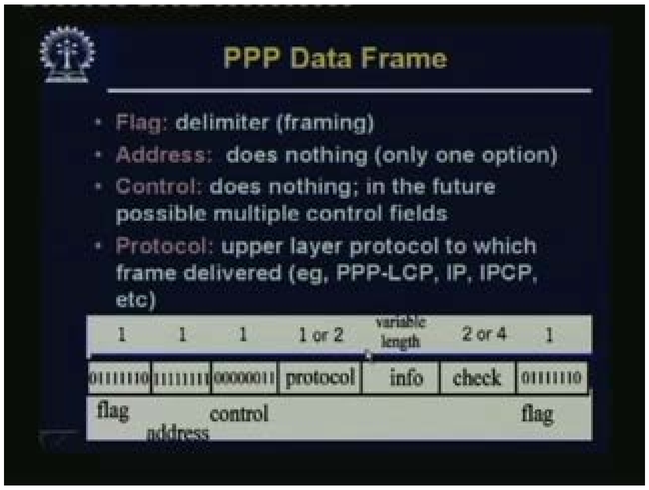
PPP non-requirements

- no error correction/recovery
- no flow control
- out of order delivery OK
- no need to support multipoint links (e.g., polling)

Error recovery, flow control, data re-ordering all relegated to higher layers!

The PPP non-requirements are no error correction/recovery, no flow control, out of order delivery is acceptable; and no need to support multipoint links; e.g., polling. Error recovery, flow control and data ordering are all relegated to higher layers.

(Refer Slide Time: 50:49)



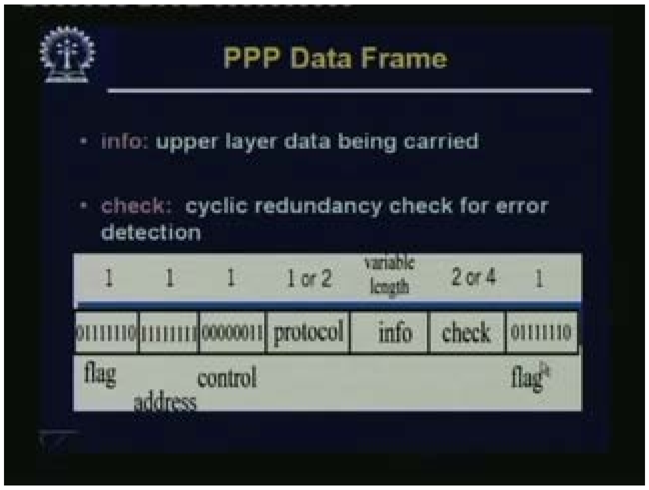
PPP Data Frame

- Flag: delimiter (framing)
- Address: does nothing (only one option)
- Control: does nothing; in the future possible multiple control fields
- Protocol: upper layer protocol to which frame delivered (eg, PPP-LCP, IP, IPCP, etc)

1	1	1	1 or 2	variable length	2 or 4	1
01111110	11111111	00000011	protocol	info	check	01111110
flag	address	control				flag

The PPP data frame has flag, address, and control and protocol bits. The flag is the delimiter. The address does nothing. The control also does nothing; in the future possible multiple control fields. The upper layer protocol is where frame is delivered. The check is for detecting errors.

(Refer Slide Time: 52:37)



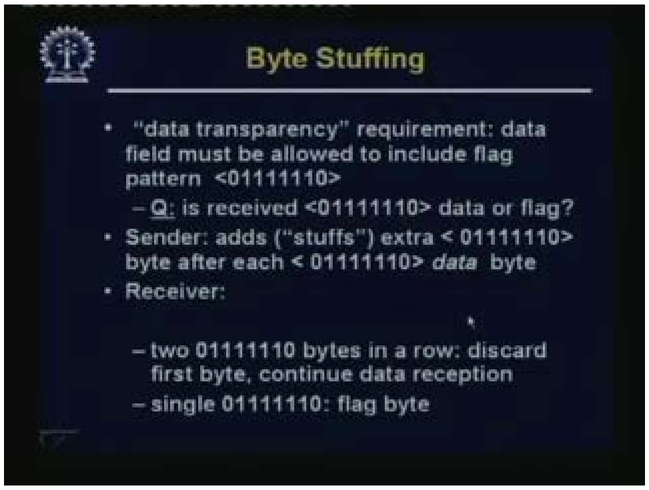
PPP Data Frame

- info: upper layer data being carried
- check: cyclic redundancy check for error detection

1	1	1	1 or 2	variable length	2 or 4	1
01111110	11111111	00000011	protocol	info	check	01111110
flag	address	control				flag

In the data frame some info that is upper layer data being carried is required and the check is the cyclic redundancy check for error. So info is the upper layer data. So this is the main body or the payload which is being carried. The check is for error detection.

(Refer Slide Time: 52:59)

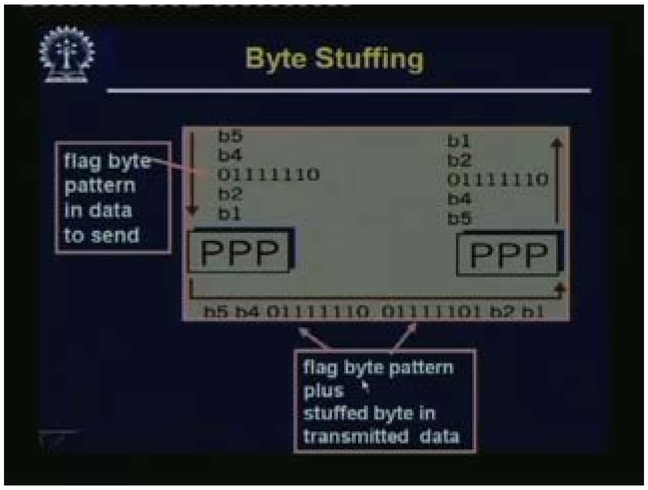


Byte Stuffing

- "data transparency" requirement: data field must be allowed to include flag pattern <01111110>
  - Q: is received <01111110> data or flag?
- Sender: adds ("stuffs") extra < 01111110> byte after each < 01111110> *data* byte
- Receiver:
  - two 01111110 bytes in a row: discard first byte, continue data reception
  - single 01111110: flag byte

It uses byte stuffing. The data transparency requirement data field must be allowed to include flag pattern <01111110>. Now we can have the question, is the received <01111110> data or flag? Sender adds extra byte after each <01111110> data byte. At the receiver side two 01111110 bytes in a row, discard first byte, continue data reception and single 01111110 is flag byte.

(Refer Slide Time: 53:51)

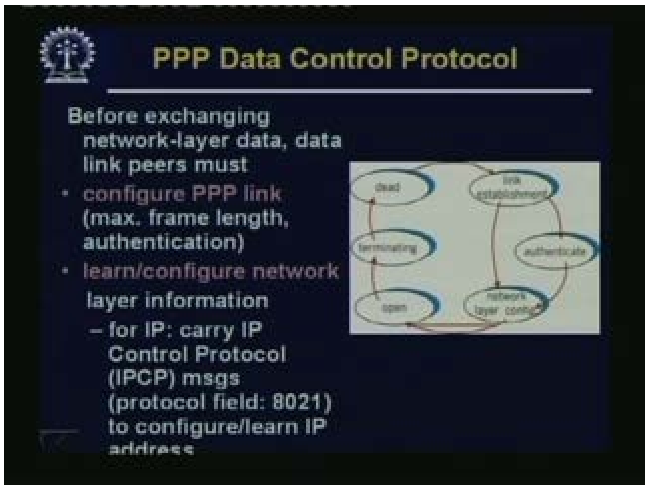


Byte Stuffing

flag byte pattern in data to send
b5
b4
01111110
b2
b1
PPP
b1
b2
01111110
b4
b5
PPP
b5 b4 01111110 01111101 b2 b1
flag byte pattern plus stuffed byte in transmitted data

So after PPP, instead of sending it will send first this B1 then B2 etc., and then instead of sending one of them 0 1110, it sends two of them and then B4 and B5. This is byte stuffing, which is used by PPP.

(Refer Slide Time: 54:19)



PPP Data Control Protocol

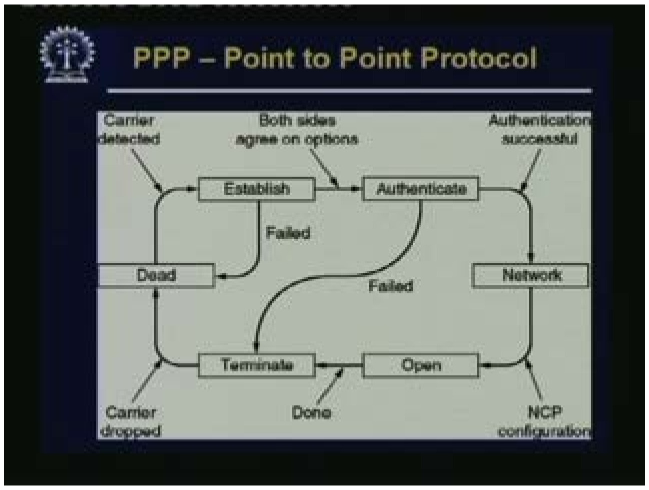
Before exchanging network-layer data, data link peers must

- configure PPP link (max. frame length, authentication)
- learn/configure network layer information
  - for IP: carry IP Control Protocol (IPCP) msgs (protocol field: 8021) to configure/learn IP address

dead
link establishment
terminating
authenticate
open
network layer config

There are a few control issues like before exchanging network layer data: data link peers must continue PPP link and learn/configure network.

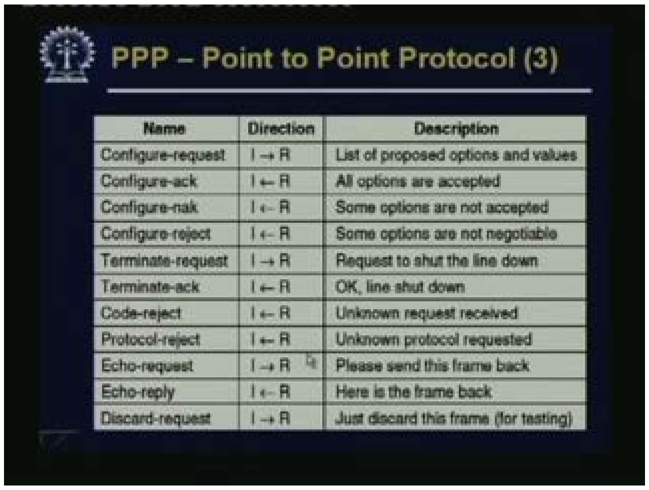
(Refer Slide Time: 54:39)



PPP – Point to Point Protocol

The first thing is the link may be dead. Then the carrier is detected. So it will try to establish the link. For that, they will require some authentication, which means the two sides, configurations, etc. must agree if it fails to establish, then it goes back to dead. If it gets successful authentication, then the network is open and then there is some transmission of data and then finally it will terminate. So for all these, there is a data exchange.

(Refer Slide Time: 55:13)



PPP – Point to Point Protocol (3)

Name	Direction	Description
Configure-request	I → R	List of proposed options and values
Configure-ack	I ← R	All options are accepted
Configure-nak	I ← R	Some options are not accepted
Configure-reject	I ← R	Some options are not negotiable
Terminate-request	I → R	Request to shut the line down
Terminate-ack	I ← R	OK, line shut down
Code-reject	I ← R	Unknown request received
Protocol-reject	I ← R	Unknown protocol requested
Echo-request	I → R	Please send this frame back
Echo-reply	I ← R	Here is the frame back
Discard-request	I → R	Just discard this frame (for testing)

Configure request, configure acknowledgement and configure not acknowledgement -that means your configured thing is not acknowledged. Some of the options are not accepted and some of the options are not negotiable. So this way, the two sides communicate and establish the link. We need not go into the details. This is not really necessary. This is a very simple protocol. Just use byte stuffing and use some data, some error detection and the framing. This is a very simple, but very widely used protocol. In the next lecture, we will see how the error control and error detection can be done by the data link layer. Thank you.

(Refer Slide Time: 56:02)

Preview of the next lecture

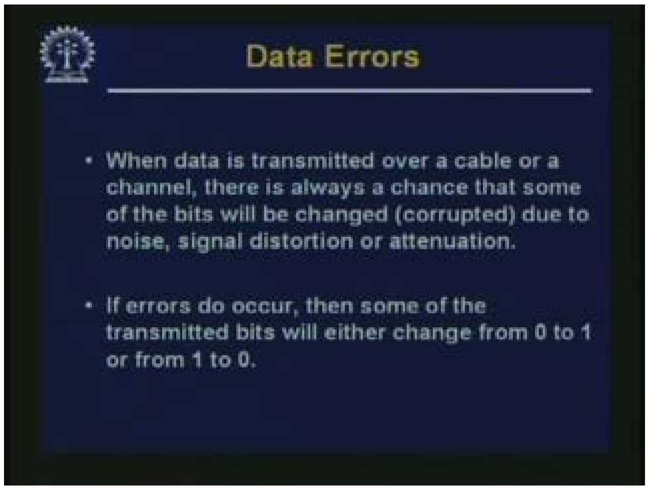
(Refer Slide Time: 56:08)

Lecture Name # 16

Error control

Error Control

(Refer Slide Time: 56:47)

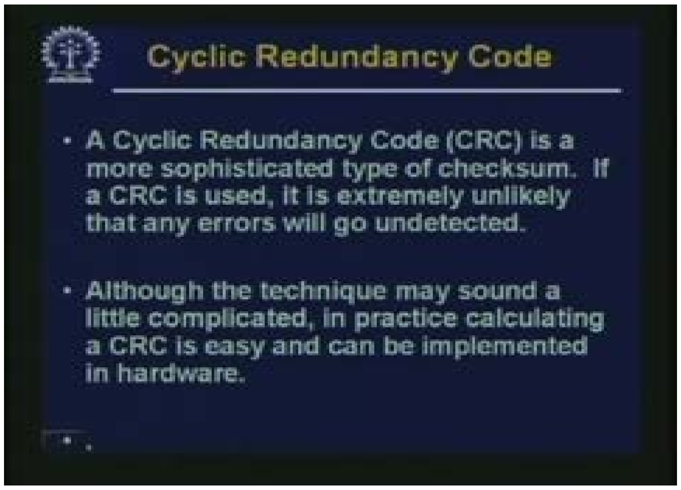


Data Errors

- When data is transmitted over a cable or a channel, there is always a chance that some of the bits will be changed (corrupted) due to noise, signal distortion or attenuation.
- If errors do occur, then some of the transmitted bits will either change from 0 to 1 or from 1 to 0.

When data is transmitted over a cable or a channel, there is always a chance that some of the bits will be changed or corrupted due to noise signal distortion or attenuation. for example, suppose you have a wireless channel and suddenly there is a burst of noise. What will happen is that, some of the data will get garbled. Similarly the data may become very attenuated. It may be due to some loose contact somewhere or something. The one that was sent was not received that way or may be it was received as a one zero or something. So whenever you are sending some data or something, there is some communication going on some transmission over some transmission line. You always have to assume that, a data may not reach the other side in a perfect condition. So that is why CRC is preferred in many data link protocols.

(Refer Slide Time: 57:47)

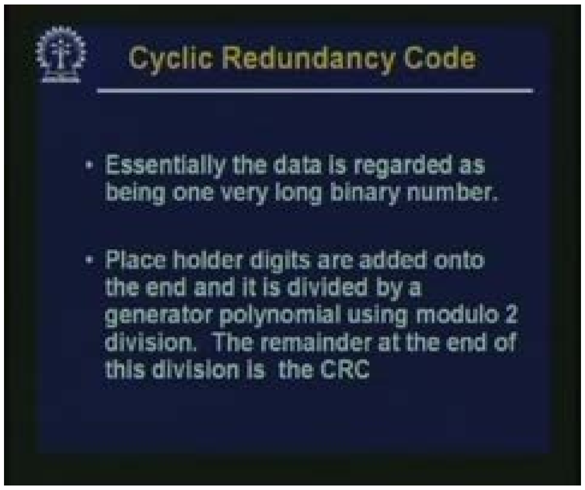


Cyclic Redundancy Code

- A Cyclic Redundancy Code (CRC) is a more sophisticated type of checksum. If a CRC is used, it is extremely unlikely that any errors will go undetected.
- Although the technique may sound a little complicated, in practice calculating a CRC is easy and can be implemented in hardware.

CRC is Cyclic Redundancy Code.

(Refer Slide Time: 57:58)



Cyclic Redundancy Code

- Essentially the data is regarded as being one very long binary number.
- Place holder digits are added onto the end and it is divided by a generator polynomial using modulo 2 division. The remainder at the end of this division is the CRC

In Cyclic Redundancy Code, essentially the data is regarded as being one very long binary number. After all, what you are sending is a string of ones and zeros so you can take a few of them and just look at it as a binary number although the original intention of the user. Place holder digits are added onto the end and it is divided by a generator polynomial using modulo 2 divisions. The remainder at the end of this division is the CRC.