

# Chapter 3

(Week 5)

## The Data Link Layer

ANDREW S. TANENBAUM  
COMPUTER NETWORKS  
FOURTH EDITION  
PP. 183-211

# Previous Layer: **The Physical Layer**

- **The physical layer** is concerned with transmitting raw bits over a communication channel.

# Next Layer: **The Data Link Layer**

- The main task of **the data link layer** is to transform a raw transmission facility into a line that appears free of undetected transmission errors to the network layer.

# Data Link Layer (1/5)

- We will study the design principles for layer 2, **the data link layer**.
- This study deals the algorithms for achieving **reliable, efficient communication** between two adjacent machines at the data link layer

## Data Link Layer (2/5)

- Why **Data Link Layer**?
- You might think this problem is so trivial that there is no software to study – machine **A** just puts the bits on the wire, and machine **B** just takes them off.
- Unfortunately, communication circuits make **errors** occasionally.

## Data Link Layer (3/5)

- Machines have only a finite data rate, and there is a nonzero propagation delay between the time a bit is sent and the time it is received.
- These limitations have important implications for the efficiency of the data transfer.

## Data Link Layer (4/5)

- **The protocols** used for communication must take all these factors into consideration.
- **These protocols** are the subject of this chapter.

# Data Link Layer (5/5)

## Objectives:

- 1) An introduction to the key design issues present in the data link layer;
- 2) A study of protocols by looking at the nature of errors, their causes, and how they can be detected and corrected;
- 3) A study of increasingly complex protocols, each one solving more and more of the problems present in this layer;
- 4) An examination of protocol modeling and correctness.

# Data Link Layer Design Issues (1/4)

The data link layer's **specific functions**:

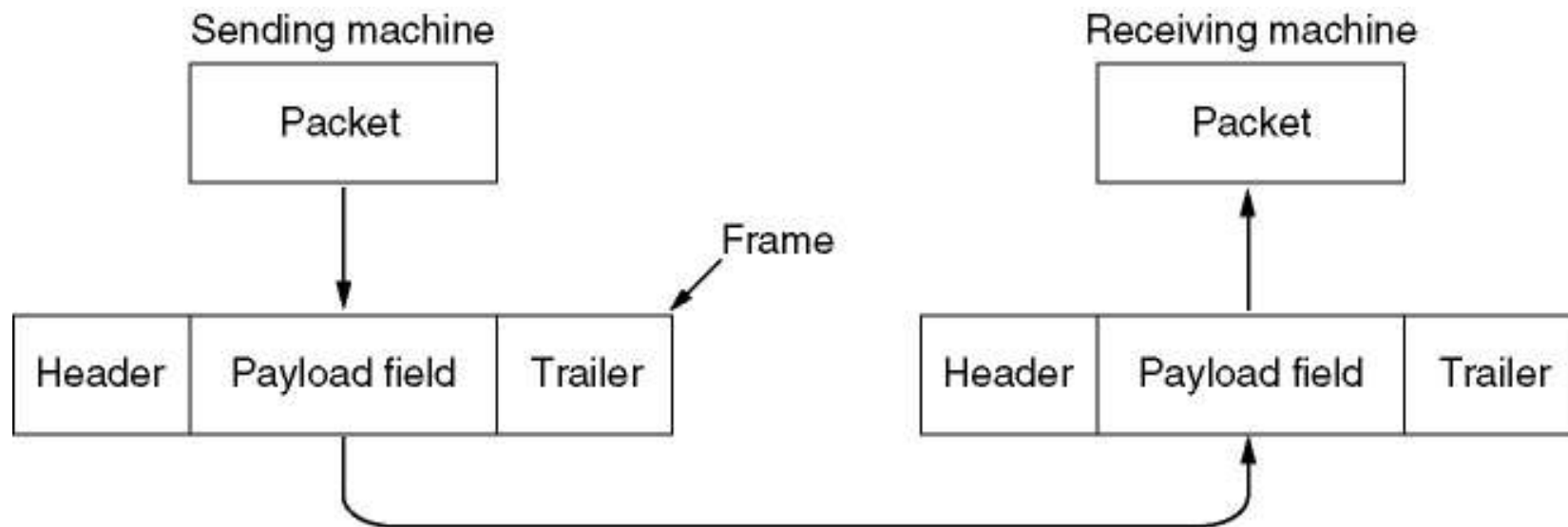
- Providing a well-defined service interface to the network layer
- Dealing with transmission errors
- Regulating the flow of data so that slow receivers are not swamped by fast senders.



# Data Link Layer Design Issues (2/4)

- The data link layer takes **the packets** it gets from the network layer and encapsulates them into **the frames** for transmission.
- **Each frame** consists of **a frame header**, **a payload field** for holding the packet, and **a frame trailer**.
- **Frame management** forms the heart of what the data link layer does.

# Data Link Layer Design Issues (3/4)



Relationship between packets and frames.

# Data Link Layer Design Issues (4/4)

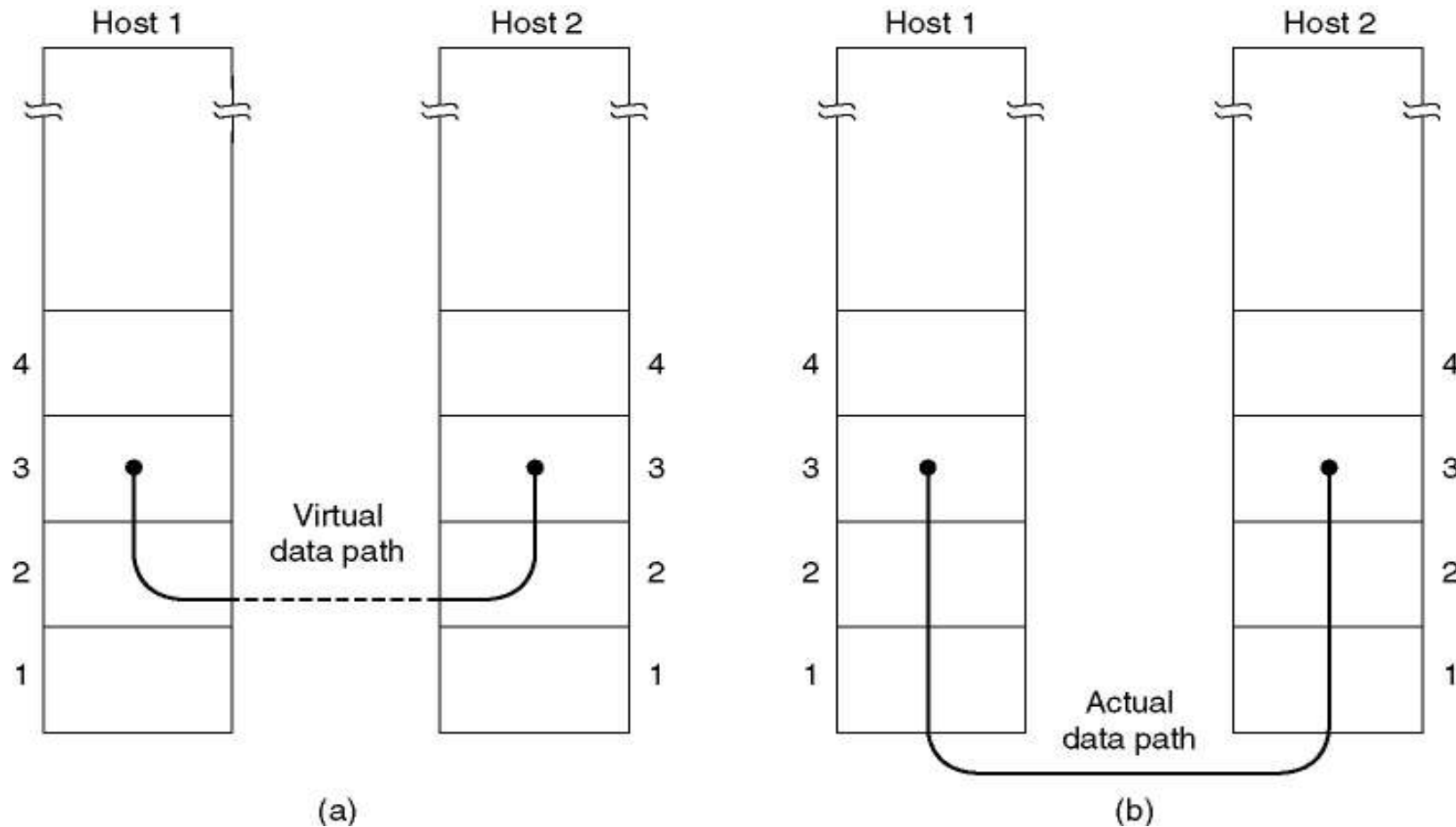
## Subtopics:

- Services Provided to the Network Layer
- Framing
- Error Control
- Flow Control

# Services Provided to Network Layer (1/8)

- The principal service is **transferring data** from the network layer on the source machine to the network layer on the destination machine.
- The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer.

# Services Provided to Network Layer (2/8)



(a) Virtual communication.

(b) Actual communication.

# Services Provided to Network Layer (3/8)

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

- a) Unacknowledged connectionless service.
- b) Acknowledged connectionless service.
- c) Acknowledged connection-oriented service.

# Services Provided to Network Layer (4/8)

## a) Unacknowledged connectionless service:

- The source machine sends independent frames to the destination machines without having the destination machine acknowledge them.
- No logical connection is established beforehand or released afterward.
- If a frame is lost due to noise on the line, no attempt is made to detect the loss or recover from it in the data link layer.
- It is used where error rate is very low, for real-time traffic such as voice, most LANs use it.

# Services Provided to Network Layer (5/8)

## b) Acknowledged connectionless service:

- Reliable connectionless service
- When this service is offered, there are still no logical connections used, but each frame sent is individually acknowledged.
- In this way, the sender knows whether a frame has arrived correctly.
- If it has not arrived within a specified time interval, it can be sent again.
- For unreliable channels, such as wireless systems.



# Services Provided to Network Layer (6/8)

## c) Acknowledged connection-oriented service.

- The source and destination machines establish a connection before and data are transferred.
- Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received.
- Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order.

# Services Provided to Network Layer (7/8)

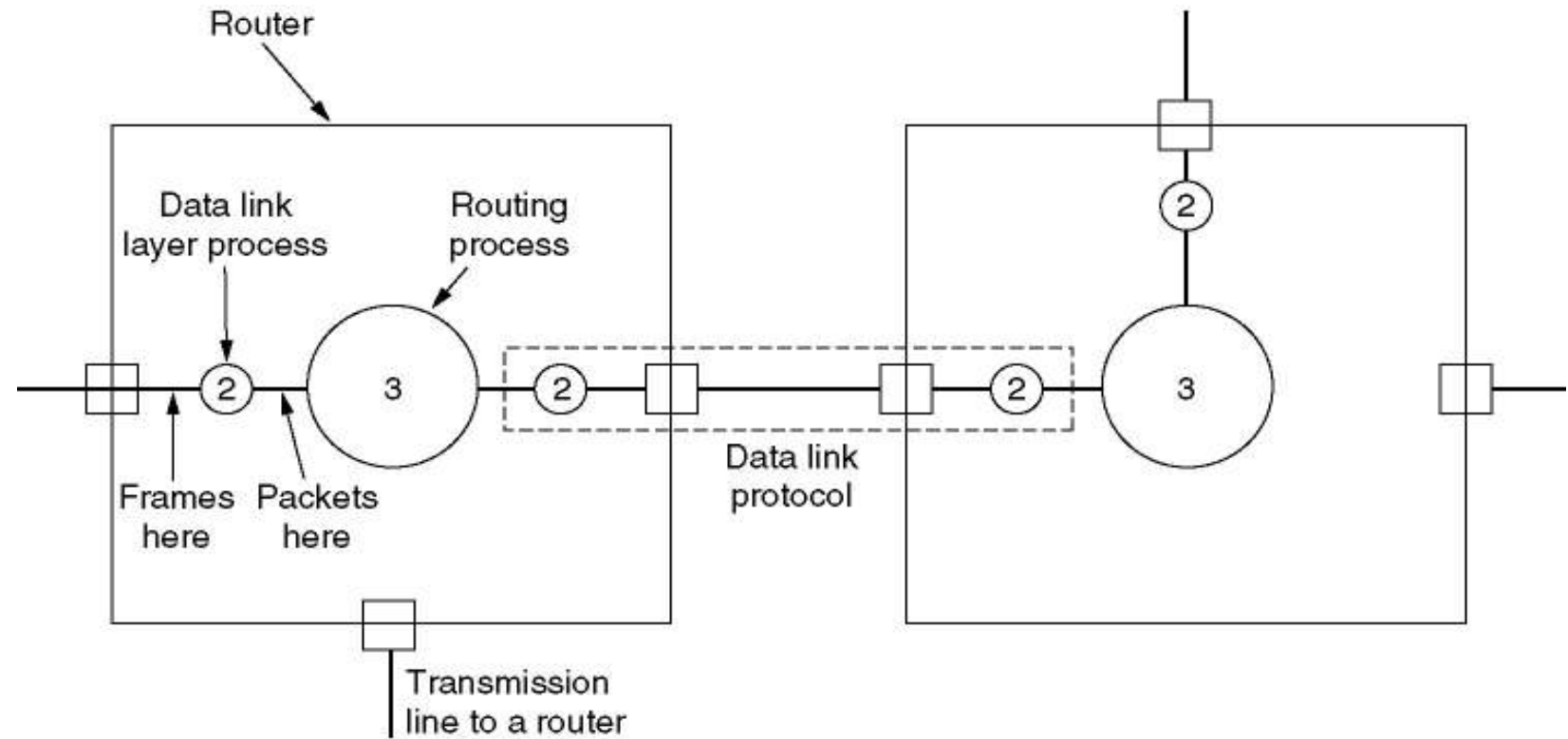
## c) Acknowledged connection-oriented service.

There are three distinct phases in connection-oriented service.

- The connection is established;
- Frames are transmitted by this connection;
- The connection is released.

# Services Provided to Network Layer (8/8)

**Example:** A WAN subnet consisting of routers connected by point-to-point leased telephone lines..



Placement of the data link protocol.

# Framing (1/12)

- To provide service to the network layer, the data link layer must use the service provided to it by the physical layer.
- What the physical layer does is accept a row bit stream and attempt to deliver it to the destination.
- The number of bits received may be less than, equal to, or more than the number of bits transmitted, and they may have different values.
- It is up to the data link layer to detect and, if necessary, correct errors.

# Framing (2/12)

- The usual approach is for the data link layer to break the bit stream up into discrete frames and compute the checksum for each frame.
- When a frame arrives at the destination, the checksum is recomputed.
- If the newly – computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it.

# Framing (3/12)

The methods to break the bit stream up into frames:

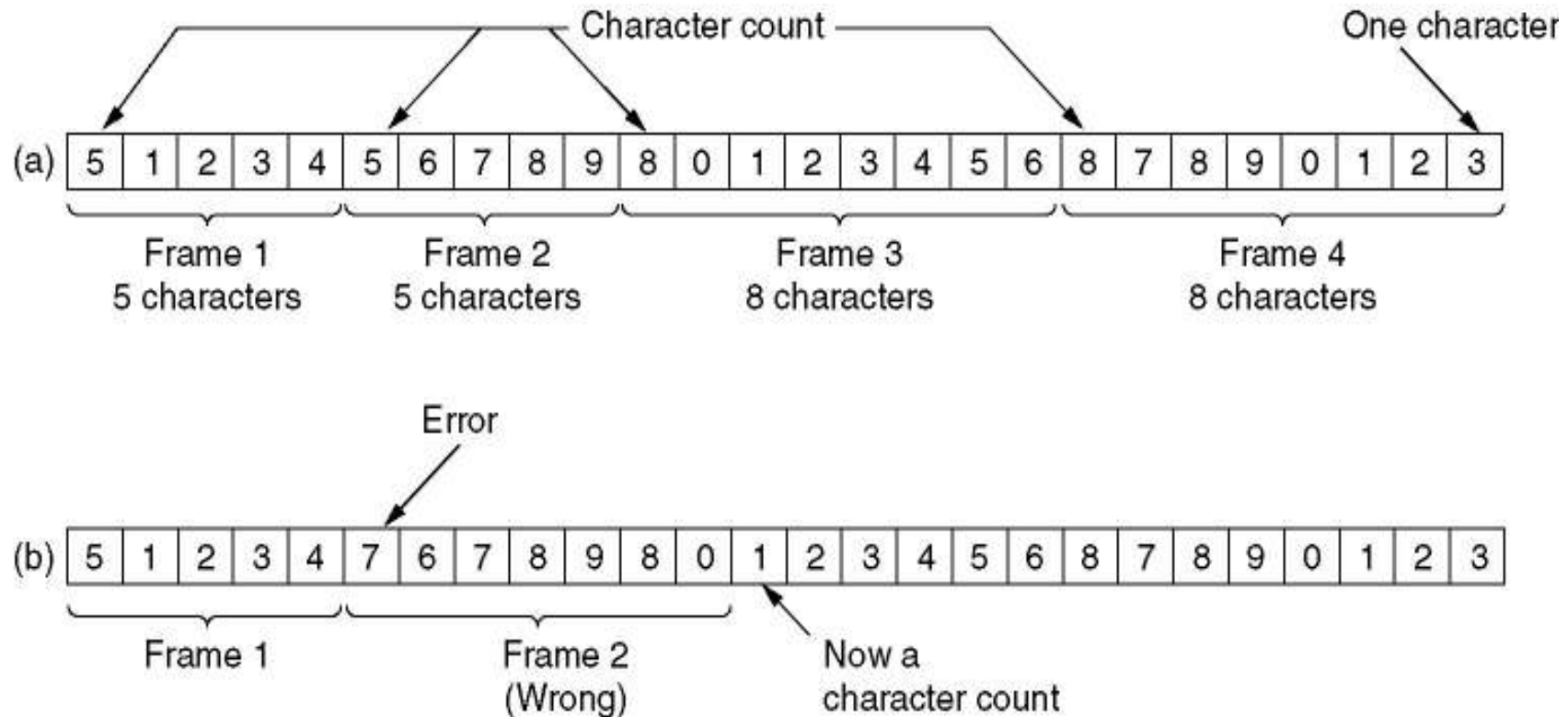
- 1) Character count;
- 2) Flag bytes with byte stuffing;
- 3) Starting and ending flags, with bit stuffing;
- 4) Physical layer coding violations.

# Framing (4/12)

## 1) Character count:

- This framing method uses a field in the header to specify the number of characters in the frame.
- When the data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is.
- The trouble with this algorithm is that the count can be garbled by a transmission error.

# Framing (5/12)



A character stream. (a) Without errors for four frames of sizes 5, 5, 8, and 8 characters, respectively. (b) With one error.



# Framing (6/12)

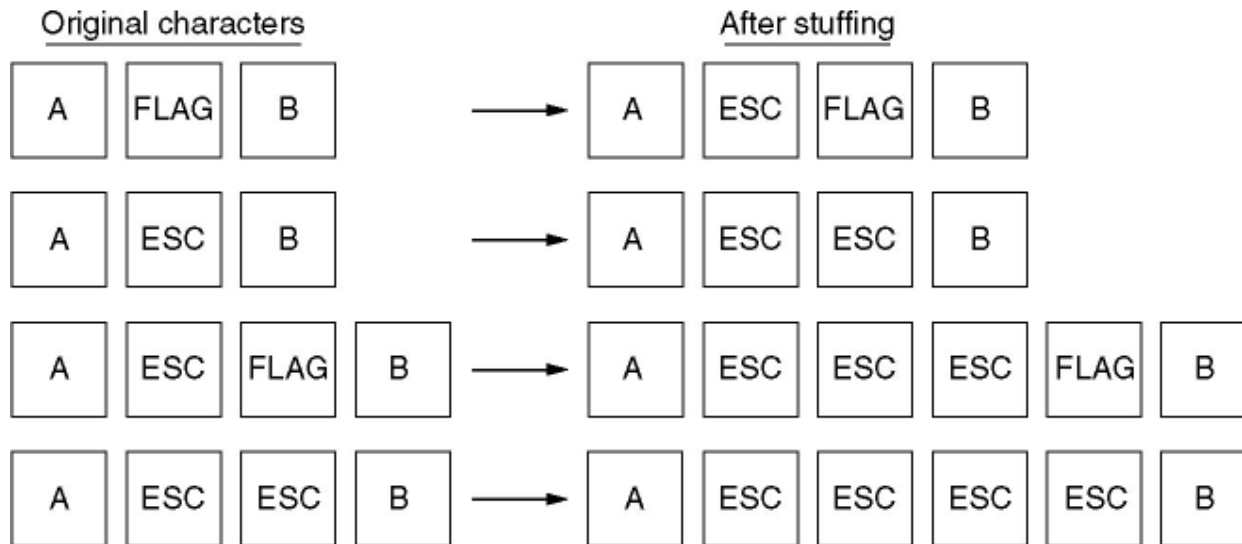
## 2) Flag bytes with byte stuffing.

- This method gets around the problem of **resynchronization** after an error by having each frame start and end with special bytes.
- The starting and ending Flag Bytes are same in recent years.
- If the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame.
- 2 consecutive flag bytes indicate the end of one frame and start of next one.

# Framing (7/12)



(a)



(b)

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

# Framing (8/12)

## 2) Flag bytes with byte stuffing – cont.

- A serious problem occurs with this method when binary data, such as object programs or floating-point numbers, are being transmitted.
- It may easily happen that the flag byte's bit pattern occurs in the data.
- This situation will usually interfere with the framing.
- One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each "accidental" flag byte in the data.

# Framing (9/12)

## 2) Flag bytes with byte stuffing – cont.

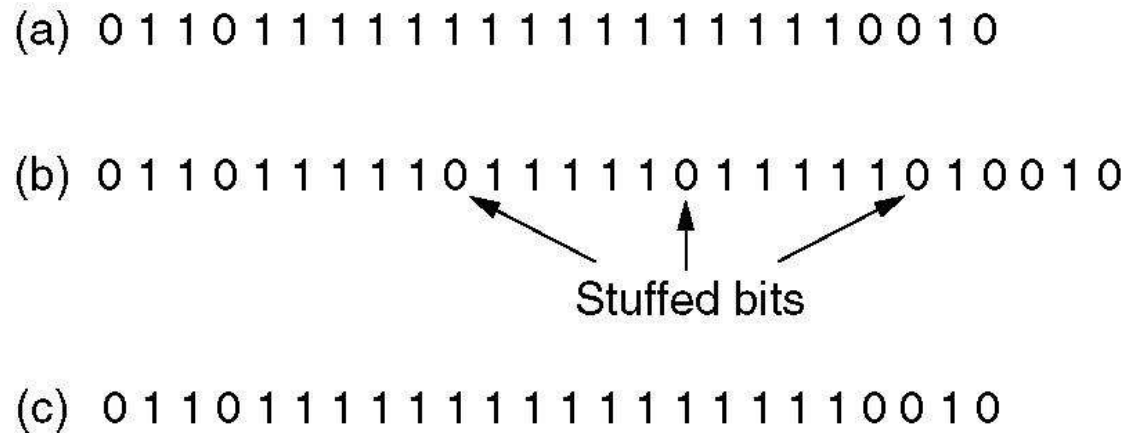
- A major disadvantage of using this framing method is that it is closely tied to the use of 8-bit characters.
- Not all character code use 8-bit characters.
- For example, UNICODE uses 16-bit characters.
- As networks developed, the disadvantages of embedding the character code length in the framing mechanism became more and more obvious, so a new technique had to be developed to allow arbitrary sized characters.

# Framing (10/12)

## 3. Starting and ending flags, with bit stuffing:

- The new technique allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character.
- Each frame begins and ends with a special bit pattern, 0111110 (in fact, a flag byte).
- Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream.

# Framing (11/12)



## Bit stuffing

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

# Framing (12/12)

## 4) Physical layer coding violations:

- This method is only applicable to networks in which the encoding on the physical medium contains some redundancy.
- For example, some LANs encode 1 bit of data by using 2 physical bits.
- Normally, a 1 bit is a high-low pair and a 0 bit is a low-high pair.
- 11 or 00 are not used for data but are used for delimiting frames in some protocols.

# Error Control (1/1)

- The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line.
- Typically, the protocol calls for the receiver to send back special control frames.
- If the sender receives a positive acknowledgement about a frame, it knows the frame has arrived safely.
- On the other hand, a negative acknowledgement means that something has gone wrong, and the frame must be transmitted again.



# Flow Control (1/2)

- Another important design issue that occurs in the data link layer (and higher layers as well) is what to do with a sender that systematically wants to transmit frames faster than the receiver can accept them.
- To solve this problem **feedback-based flow control** and **rate-based flow control** are used.

## Flow Control (2/2)

- **Feedback-based flow control:** the receiver sends back information to the sender giving it permission to send more data or at least telling the sender how the receiver is doing.
- **Rate-based flow control:** the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.
- Rate-based flow control is never used in the data link layer.

# Error Detection and Correction (1/5)

- The telephone system has three parts:
  - 1) the switches – almost entirely digital
  - 2) the interoffice trunks - almost entirely digital
  - 3) the local loops – still analog (twisted copper pairs)

# Error Detection and Correction (2/5)

- While errors are rare on the digital part, they are still common on the local loops.
- Furthermore, wireless communication is becoming more common, and the error rates here are orders of magnitude worse than on the interoffice fiber trunks.
- The conclusion is: transmission errors are going to be with us for many years to come.
- **We have to learn how to deal with them.**

# Error Detection and Correction (3/5)

- As a result of the physical processes that generate them, errors on some media (e.g., radio) tend to come in **bursts** rather than **singly**.
- **The advantage of bursts** errors is that the computer data are always sent in blocks of bits.
- **The disadvantage** is that they are much harder to correct than are isolated errors.

# Error Detection and Correction (4/5)

- **Error-Correcting Codes**

To include enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been.

- **Error-Detecting Codes**

To include only enough redundancy to allow the receiver to deduce that an error occurred, but not which error, and have it request a retransmission.

# Error Detection and Correction (5/5)

- On channels that are highly reliable, **such as fiber**, it is cheaper to use an error detecting code and just retransmit the occasional block found to be faulty.
- On channels such as **wireless links** that make many errors, it is better to add enough redundancy to each block for the receiver to be able to figure out what the original block was, rather than relying on a retransmission, which itself may be in error.

# Error-Correcting Codes (1/11)

- What is an error?
- Normally, a frame consists of  $m$  data bits and  $r$  redundant, or check, bits.
- The total length is  $n$  ( $n=m+r$ )
- An  $n$ -bit unit containing data and check bits is often referred to as an  *$n$ -bit codeword*.



# Error-Correcting Codes (2/11)

- Given any two codeword (say, 10001001 and 10110001), it is possible to determine how many corresponding bits differ. As we see in this case 3 bits differ.
- To determine how many bits differ, just exclusive OR the two codewords and count the number of 1 bits in the result, for example:
  - 10001001
  - 10110001
  - -----
  - 00111000

# Error-Correcting Codes (3/11)

- The number of bit positions in which two codewords differ is called **the Hamming distance**.
- Its significance is that if two codewords are a Hamming distance  $d$  apart, it will require  $d$  single-bit errors to convert one into the other.

# Error-Correcting Codes (4/11)

- $2^m$  possible data message are legal;
- But due to the way the check bits are computed, not all of the  $2^n$  possible codewords are used.
- It is possible to construct a complete list of legal codewords by given algorithm for computing the check bits, and from this find the two codewords whose Hamming distance is minimum.
- This distance is the Hamming distance of the complete code.

# Error-Correcting Codes (5/11)

- The error-detecting and error –correcting properties of a code depend on its Hamming distance.
- To detect  $d$  errors, you need a distance  $d+1$  code because with such a code there is no way that  $d$  single-bit errors can change a valid codeword.
- When the receiver sees an invalid codeword, it can tell that a transmission error has occurred.

# Error-Correcting Codes (6/11)

- Similarly, to correct  $d$  errors, you need a distance  $2d+1$  code because that way the legal codewords are so far apart that even with  $d$  changes, the original codeword is still closer than any other codeword, so it can be uniquely determined.
- As a simple example of an error-detecting code, consider a code in which a single parity bit is appended to the data.
- The parity bit is chosen so that the number of  $1$  bits in the codeword is even (or odd).

# Error-Correcting Codes (7/11)

- For example, when 1011010 is sent in **even parity**, a bit is added to the end to make it 10110100.
- With **odd parity** 1011010 becomes 10110101.
- A code with a single parity bit has a distance 2, since any single-bit error produces a codeword with the wrong parity.
- It can be used to detect single errors.

# Error-Correcting Codes (8/11)

- **Example:**
- Consider a code with only four valid codewords:
- 0000000000, 0000011111, 1111100000, 1111111111
- This code has a distance 5, which means that it can correct double errors, because of  $5=2d+1$ .
- If codeword 0000000111 arrives, the receiver knows that the original must have been 0000011111.
- If however a triple error changes 0000000000 into 0000000111, the error will not be corrected properly.

# Error-Correcting Codes (9/11)

- Imagining that we want to design a code with  $m$  message bits and  $r$  check bits that will allow all single errors to be corrected.
- Each of the  $2^m$  legal message has  $n$  illegal codewords at a distance  $1$  from it.
- These are formed by systematically inverting each of the  $n$  bits in the  $n$ -bit *codeword* formed from it.
- Thus each of  $2^m$  legal messages requires  $n+1$  bit patterns dedicated to it.



# Error-Correcting Codes (10/11)

- Since the total number of bit patterns is  $2^n$ , we must have  $(n+1)2^m \leq 2^n$ .
- Using  $n=m+r$ , this requirement becomes  $(m+r+1) \leq 2^r$ .
- Given  $m$ , this puts a lower limit on the number of check bits needed to correct single errors.

# Error-Correcting Codes (11/11)

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

Use of a Hamming code to correct burst errors.

# Error-Detecting Codes (1/10)

- Error correcting codes are widely used on wireless links, which are noisy and error prone when compared to copper wire or optical fibers.
- Without error correcting codes, it would be hard to get anything through.
- However over copper wire or fiber, the error rate is much lower, so error detection and retransmission is usually more efficient there for dealing with the occasional error.

# Error-Detecting Codes (2/10)

- **Example:**
- Consider a channel on which errors are isolated and the error rate is  $10^{-6}$  per bit.
- Let the block size be 1000 bits.
- To provide error correction for 1000-bit block, 10 check bits are needed: a megabit of data would require 10,000 check bits.
- To merely detect a block with a single 1-bit error, one parity bit per block will suffice.

# Error-Detecting Codes (3/10)

- **Example (cont.):**
- Once every 1000 blocks, an extra block (1001 bits) will have to be transmitted.
- The total overhead for the error detection + retransmission method is only 2001 bits per megabit of data, versus 10,000 bits for a Hamming code.

# Error-Detecting Codes (4/10)

- The polynomial code or *CRC (Cyclic Redundancy Check)* is in widespread use;
- Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of *0* and *1* only.
- A *k-bit* frame is regarded as the coefficient list for a polynomial with *k terms*, ranging from  $x^{k-1}$  to  $x^0$ .
- Such a polynomial is said to be of degree *k-1*.
- For example: *110001* is  $x^5 + x^4 + x^0$

# Error-Detecting Codes (5/10)

- Polynomial arithmetic is done *modulo 2*, according to the rules of algebraic field theory.
- There are no carries for addition or borrows for subtraction.
- Both addition and subtraction are identical to exclusive OR. For example:

- $10011011 \quad 00110011 \quad 11110000 \quad 01010101$
- $+11001010 \quad +11001101 \quad -101001101 \quad -10101111$
- $\text{-----} \quad \text{-----} \quad \text{-----} \quad \text{-----}$
- $01010001 \quad 11111110 \quad 01010110 \quad 11111010$

# Error-Detecting Codes (6/10)

- When the polynomial code method is employed, the sender and receiver must agree upon a generator polynomial,  $G(x)$ , in advance.
- Both high- and low- order bits of the generator must be  $1$ .
- To compute *the checksum* for some frame with  $m$  bits, corresponding to the polynomial  $M(x)$ , the frame must be longer than the generator polynomial.
- The idea is to append a checksum to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisible by  $G(x)$ .
- When the receiver gets the checksummed frame, it tries dividing it by  $G(x)$ .
- If there is a remainder, there has been a transmission error.



# Error-Detecting Codes (7/10)

- The algorithm for computing **the checksum** is as follows:
  - 1) Let  $r$  be the degree of  $G(x)$ . Append  $r$  zero bits to the low-order end of the frame so it now contains  $m+r$  bits and corresponds to the polynomial  $x^rM(x)$ .
  - 2) Divide the bit string corresponding to  $G(x)$  into the bit string corresponding to  $x^rM(x)$ , using modulo 2 division.
  - 3) Subtract the remainder (which is always  $r$  or fewer bits) from the bit string corresponding to  $x^rM(x)$  using modulo 2 subtraction. The result is the checksummed frame to be transmitted. Call its polynomial  $T(x)$ .

# Error-Detecting Codes (8/10)

Frame : 1 1 0 1 0 1 1 0 1 1  
 Generator: 1 0 0 1 1  
 Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

This example illustrates the calculation for a frame

$M(t) = 1101011011$  using

the generator

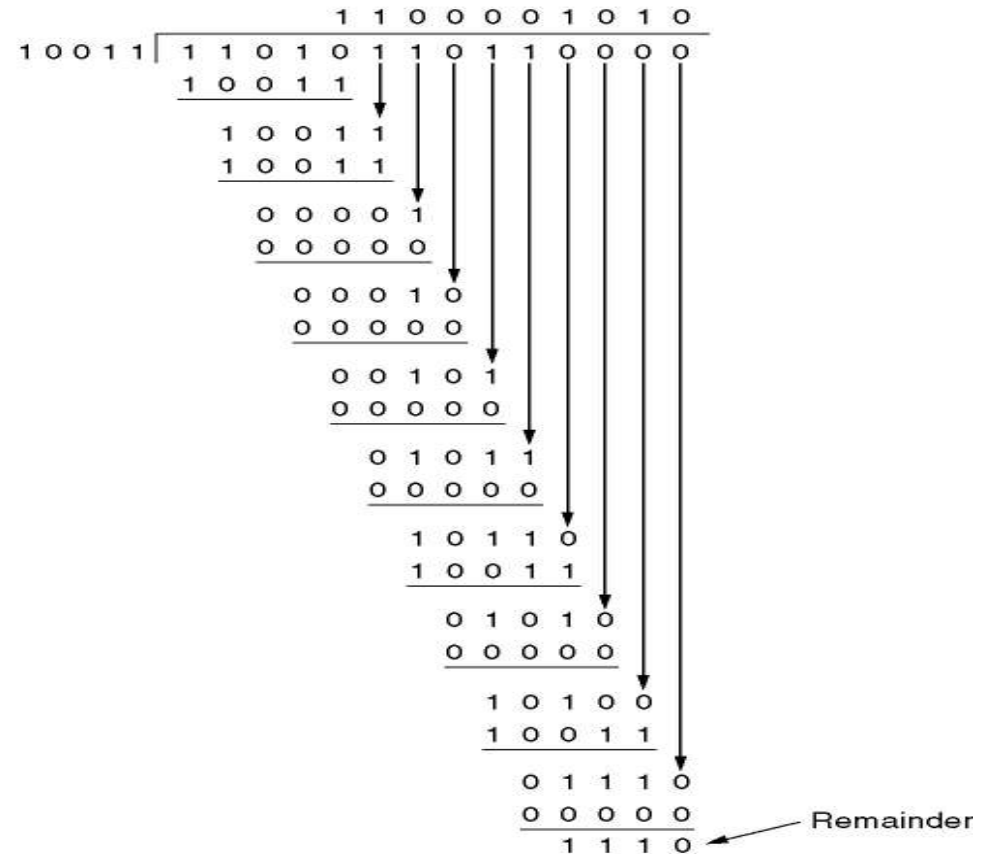
$G(x) = x^4 + x + 1$ .  $r = 4$

$x^r M(x) = 11010110110000$

$T(x) = 11010110111110$

$E(x)$  is error polynomial

Calculation of the polynomial code checksum.



Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

# Error-Detecting Codes (9/10)

- What kinds of errors will be detected?
- Imagining that a transmission error occurs, so that instead of the bit string for  $T(x)$  arriving,  $T(x)+E(x)$  arrives.
- Each *1 bit* in  $E(x)$  corresponds to a bit that has been inverted.
- If there are *k 1 bits* in  $E(x)$ , *k single-bit* errors have occurred.
- A single burst error is characterized by an initial *1*, a mixture of *0s* and *1s*, and a final *1*, with all other bits being *0*.

# Error-Detecting Codes (10/10)

- Upon receiving the checksummed frame, the receiver divides it by  $G(x)$ ; that is, it computes  $[T(x)+E(x)]/G(x)$ .
- $T(x)/G(x)$  is 0, so the result of the computation is simply  $E(x)/G(x)$ .
- Those errors that happen to correspond to polynomials containing  $G(x)$  as a factor will slip by; all other errors will be caught.