Business Data Communications and Networking

11th Edition

Jerry Fitzgerald and Alan Dennis

John Wiley & Sons, Inc

Dwayne Whitten, D.B.A
Mays Business School
Texas A&M University

Copyright 2011 John Wiley & Sons, Inc

Chapter 4

Data Link Layer

Copyright 2011 John Wiley & Sons, Inc

Chapter 4 Outline

4.1 - Introduction

4.2 - Media Access Control

- **Contention, Controlled Access, Relative Performance**

4.3 - Error Control

- **Sources of Errors, Error Prevention, Error Detection, Error Correction via Retransmission, Forward Error Correction**

4.4 - Data Link Protocols

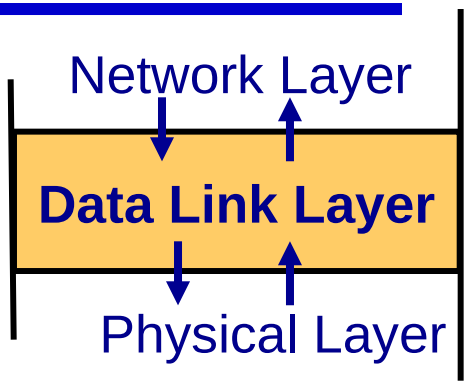
- **Asynchronous Transmission, Synchronous Transmission**

4.5 - Transmission Efficiency

4.6 – Implications for Management

Copyright 2011 John Wiley & Sons, Inc

4.1 Introduction



- **Responsible for moving messages from one device to another**
- **Controls the way messages are on media**
- **Organizes physical layer bit streams into coherent messages for the network layer**
- **Major functions of a data link layer protocol**
  - **Media Access Control**
    - **Controlling when computers transmit**
  - **Error Control**
    - **Detecting and correcting transmission errors**
  - **Message Delineation**
    - **Identifying the beginning and end of a message**

Copyright 2011 John Wiley & Sons, Inc

4.2 Media Access Control (MAC)

- **Controlling when and what computer transmit**
  - **Important when more than one computer wants to send data at the same time over the same, shared circuit**
    - **Point-to-point half duplex links**
      - **computers take turns**
    - **Multipoint configurations**
      - **Ensure that no two computers attempt to transmit data at the same time**
- **Two possible approaches**
  - **Contention based access**
  - **Controlled access**

Copyright 2011 John Wiley & Sons, Inc

Contention

- **Transmit whenever the circuit is free**
- **Collisions**
  - **Occur when more than one computer transmits at the same time**
  - **Need to determine which computer is allowed to transmit first after the collision**
- **Used commonly in Ethernet LANs**
- **Can be problematic in heavy usage networks**

Copyright 2011 John Wiley & Sons, Inc

Controlled Access

- **Controlling access to shared resources**
  - **Acts like a stop light**
- **Commonly used by mainframes (or its front end processor)**
  - **Determines which circuits have access to mainframe at a given time**
- **Also used by some LAN protocols**
  - **Token ring**
- **Major controlled access methods**
  - **Access request and polling**

Copyright 2011 John Wiley & Sons, Inc

Access Request

- **Clients wanting to transmit data first send a request to the device controlling the circuit**
- **The central device will grant permission for one device at a time to transmit**

Copyright 2011 John Wiley & Sons, Inc

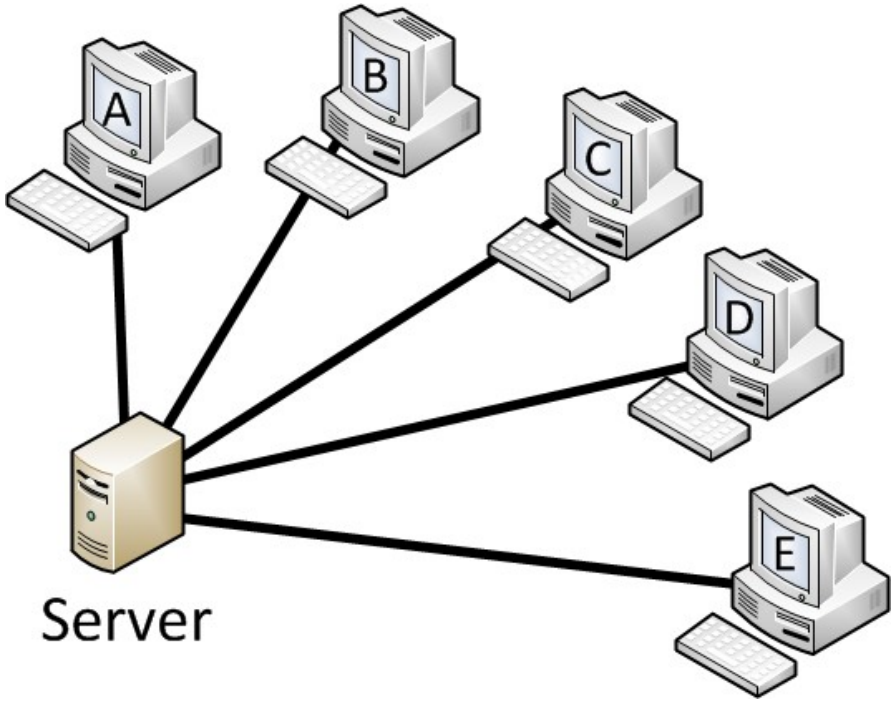
Polling

- **Process of transmitting to a client only if asked and/or permitted**
  - **Client stores the information to be transmitted**
  - **Server (periodically) polls the client if it has data to send**
  - **Client, if it has any, sends the data**
  - **If no data to send, client responds negatively, and server asks the next client**
- **Types of polling**
  - **Roll call polling**
  - **Hub polling (also called token passing)**

Copyright 2011 John Wiley & Sons, Inc

Roll Call Polling

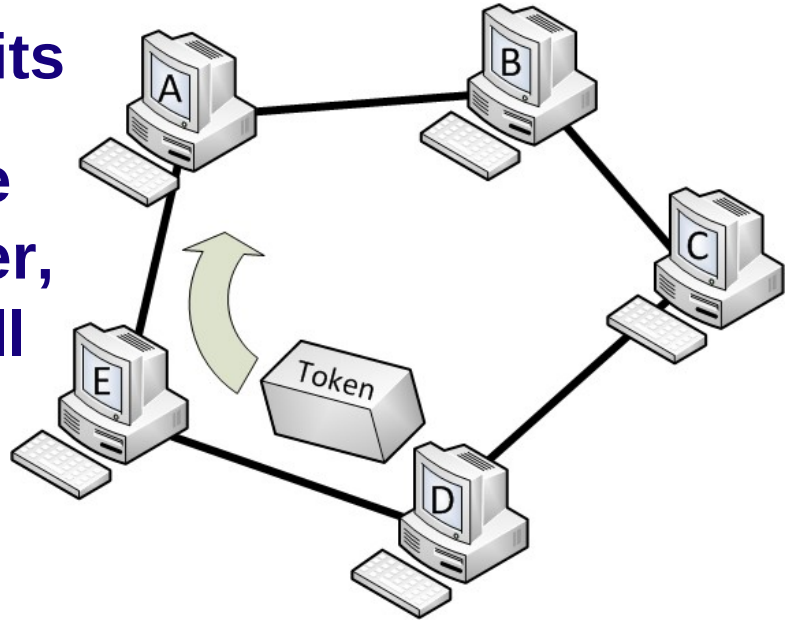
- **Check each client (consecutively and periodically) to see if it wants to transmit : A, B, C, D, E, A, B, …**
- **Clients can also be prioritized so that they are polled more frequently: A, B, A, C, A, D, A, E, A, B, ..**
- **Involves waiting: Poll and wait for a response**
- **Needs a timer to prevent lock-up (by client not answering)**



Copyright 2011 John Wiley & Sons, Inc

Hub Polling (Token Passing)

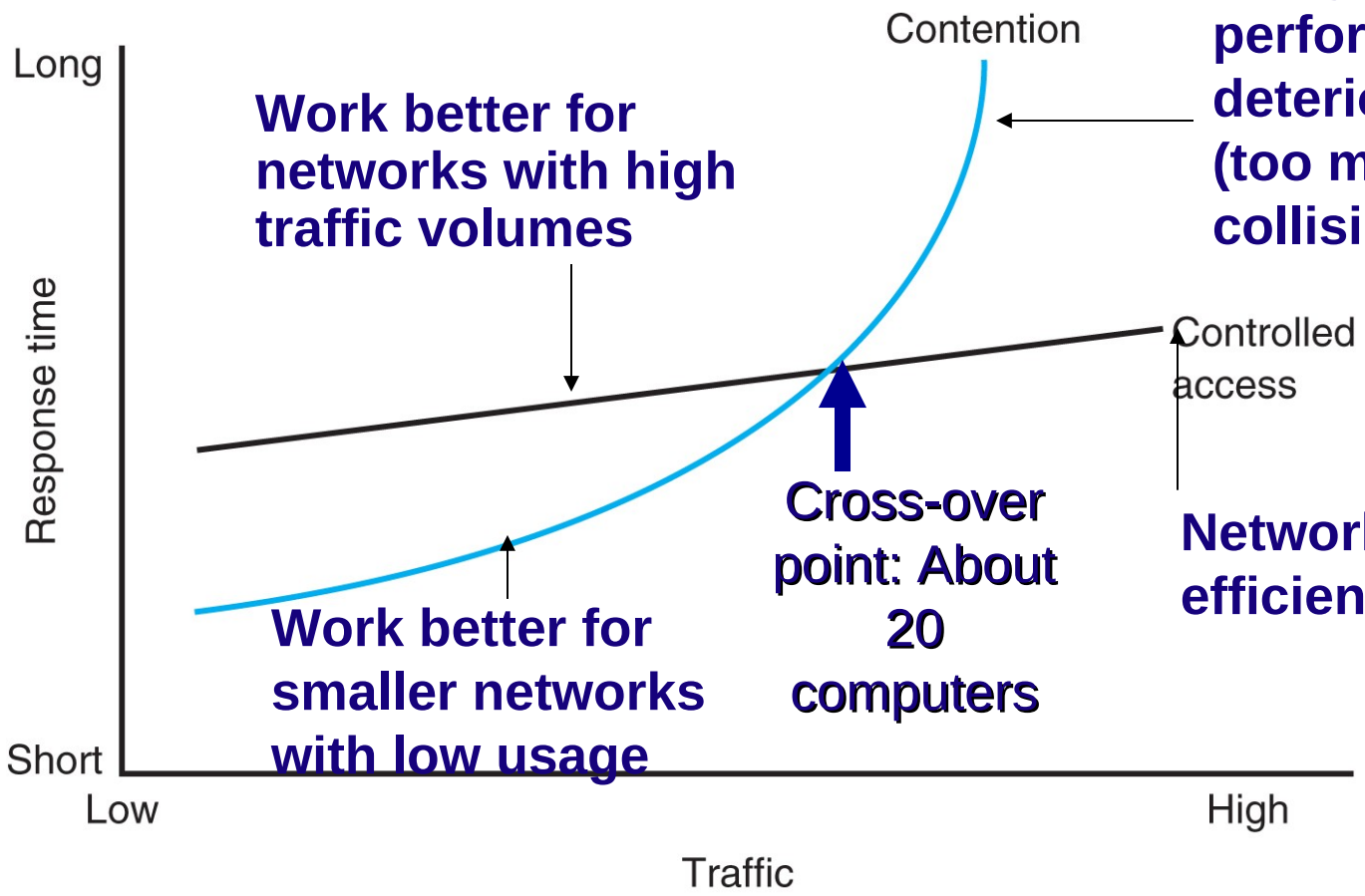
- **One computer starts the poll:**
  - **sends message (if any) then**
  - **passes the token to the next computer**
  - **token is a unique series of bits**
- **Continues in sequence until the token reaches the first computer, which starts the polling cycle all over again**



Copyright 2011 John Wiley & Sons, Inc

Relative Performance

Depends on network conditions



Copyright 2011 John Wiley & Sons, Inc

4.3 – Error Control

- **Handling of network errors caused by problems in transmission**
  - **Network errors**
    - **Can be a bit value change during transmission**
    - **Controlled by network hardware and software**
  - **Human errors:**
    - **Can be a mistake in typing a number**
    - **Controlled by application programs**
- **Categories of Network Errors**
  - **Corrupted (data that has been changed)**
  - **Lost data (cannot find the data at all)**

Copyright 2011 John Wiley & Sons, Inc

Error Control (Cont.)

- **Error Rate**
  - **1 bit error in n bits transmitted, e.g., 1 in 500,000**
- **Burst error** (more common)
  - **Many bits are corrupted at the same time**
  - **Errors not uniformly distributed**
    - **e.g., 100 in 50,000,000 → 1 in 500,000**

Copyright 2011 John Wiley & Sons, Inc

Sources of Errors

- **Line noise and distortion – major cause**
  - **More likely on electrical media**
  - **Undesirable electrical signal**
  - **Introduced by equipment and natural disturbances**
  - **Degrades performance of a circuit**
  - **Manifestation**
    - **Extra bits**
    - **“Flipped” bits**
    - **Missing bits**

Copyright 2011 John Wiley & Sons, Inc

Major Functions of Error Control

- **Error prevention**
- **Error detection**
- **Error correction**

Copyright 2011 John Wiley & Sons, Inc

Sources of Errors and Prevention

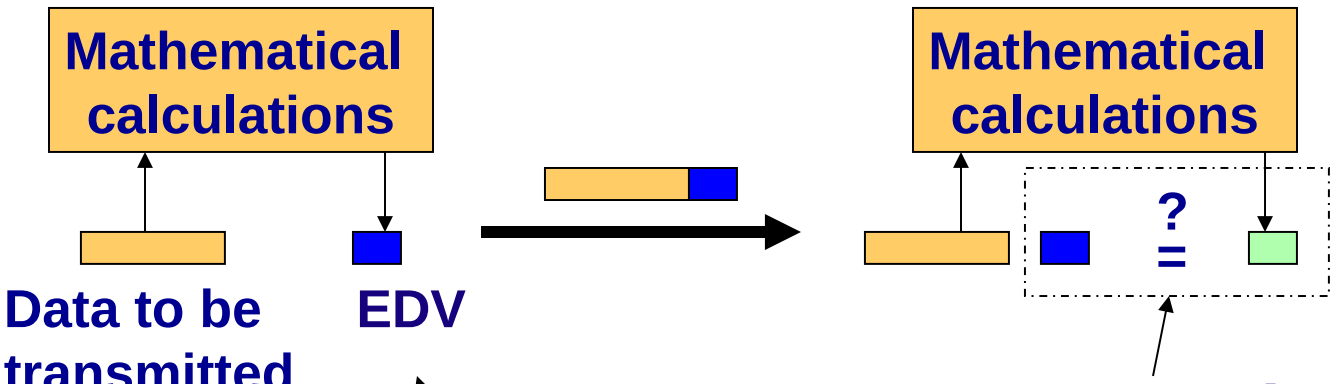
Source of error	Cause	Prevention
White noise	Movement of electrons (thermal energy)	Increase signal strength (increase SNR)
Impulse noise	Sudden increases in electricity (e.g., lightning, power surges)	Shield or move the wires
Cross-talk	Multiplexer guard bands are too small or wires too close together	Increase the guard bands, or move or shield the wires
Echo	Poor connections (causing signal to be reflected back to the source)	Fix the connections, or tune equipment
Attenuation	Gradual decrease in signal over distance (weakening of a signal)	Use repeaters or amplifiers
Intermodulation noise	Signals from several circuits combine	Move or shield the wires

Copyright 2011 John Wiley & Sons, Inc

Error Detection

Sender calculates an Error Detection Value (EDV) and transmits it along with data

Receiver recalculates EDV and checks it against the received EDV



Larger the size, better error detection (but lower efficiency)

- **If the same → No errors in transmission**
- **If different → Error(s) in transmission**

Copyright 2011 John Wiley & Sons, Inc

Error Detection Techniques

- **Parity checks**
- **Checksum**
- **Cyclic Redundancy Check (CRC)**

Copyright 2011 John Wiley & Sons, Inc

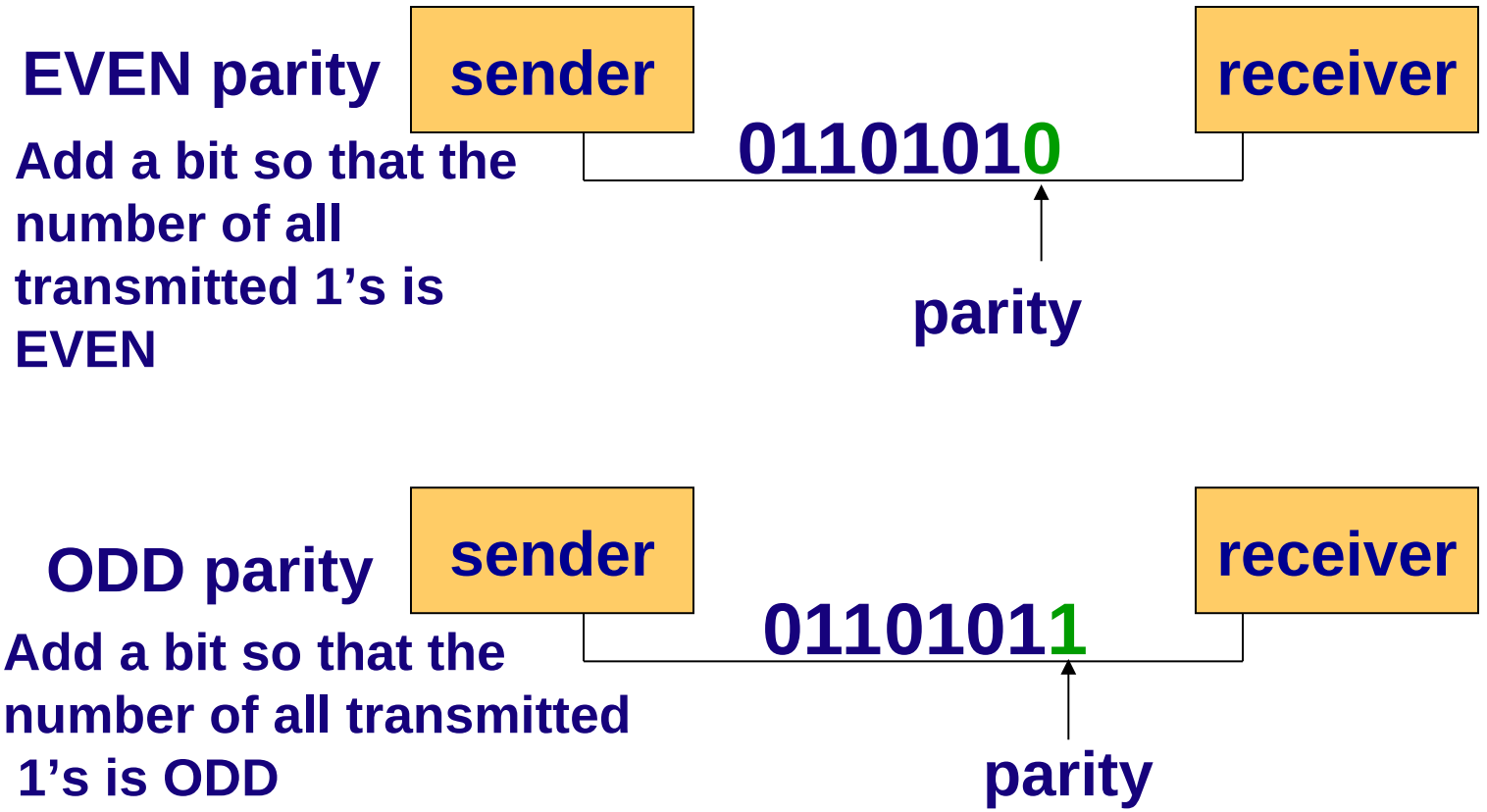
Parity Checking

- **One of the oldest and simplest**
- **A single bit added to each character**
  - **Even parity: number of 1's remains even**
  - **Odd parity: number of 1's remains odd**
- **Receiving end recalculates parity bit**
  - **If one bit has been transmitted in error the received parity bit will differ from the recalculated one**
- **Simple, but doesn't catch all errors**
  - **If two (or an even number of) bits have been transmitted in error at the same time, the parity check appears to be correct**
  - **Detects about 50% of errors**

Copyright 2011 John Wiley & Sons, Inc

Examples of Using Parity

To be sent: Letter V in 7-bit ASCII: 0110101



Copyright 2011 John Wiley & Sons, Inc

Checksum

- **Probably one of the oldest methods of ensuring that data is correct**
- **Checksums also provide a form of authentication because an invalid checksum suggests that the data has been compromised in some fashion.**
- **A checksum (usually 1 byte) is added to the end of the message**
- **It is 95% effective**
- **Method:**
  - **Add decimal values of each character in the message**
  - **Divide the sum by 256 The remainder is the checksum value**

Copyright 2011 John Wiley & Sons, Inc

Checksum

- **A checksum is determined in one of two ways:**
- **Let's say the checksum of a packet is 1 byte long. A byte is made up of 8 bits, and each bit can be in one of two states (0 or 1), leading to a total of 256 (28) possible combinations. Since the first combination equals zero, a byte can have a maximum value of 255.**
- **If the sum of the other bytes in the packet is 255 or less, then the checksum contains that exact value.**
- **If the sum of the other bytes is more than 255, then the checksum is the remainder of the total value after it has been divided by 256.**

For example:

Bytes total 1,151

1,151 / 256 = 4.496 (round to 4)

4 x 256 = 1,024

1,151 - 1,024 = 127 checksum (remainder of total value after divided by 256)

Cyclic Redundancy Check (CRC)

- **CRCs are similar in concept to checksums,**
- **but they use polynomial division to determine the value of the CRC,**
- **which is usually 8, 16, 24 or 32 bits in length.**
- **CRC is that it is very accurate.**
- **If a single bit is incorrect, the CRC value will not match up.**
- **Both checksum and CRC are good for preventing random errors in transmission**

Cyclic Redundancy Check (CRC)

$$P / G = Q + R / G = 7.25$$

Example:
P = 58
G = 8
Q = 7
R = 2

Message (treated as one long binary number)

A fixed number (determines the length of the R)

Quotient* (whole number)

Remainder:

- **added to the message as Error Detection Value (EDV)**
- **could be 8 bits, 16 bits, 24 bits, or 32 bits long**
- **CRC16 has R of 16 bits**

- **Most powerful and most common**
- **Detects 100% of errors (if number of errors <= size of R)**

*: a Quotient is a **number** obtained by dividing one quantity by another. In 45 ÷ 3 = 15, 15 is the **quotient**.

Error Correction

- **Once detected, the error must be corrected**
- **Error correction techniques**
  - **Retransmission (or, backward error correction)**
    - **Simplest, most effective, least expensive, most commonly used**
    - **Corrected by retransmission of the data**
      - **Receiver, when detecting an error, asks the sender to retransmit the message**
    - **Often called Automatic Repeat reQuest (ARQ)**
  - **Forward Error Correction**
    - **Receiving device can correct incoming messages without retransmission**

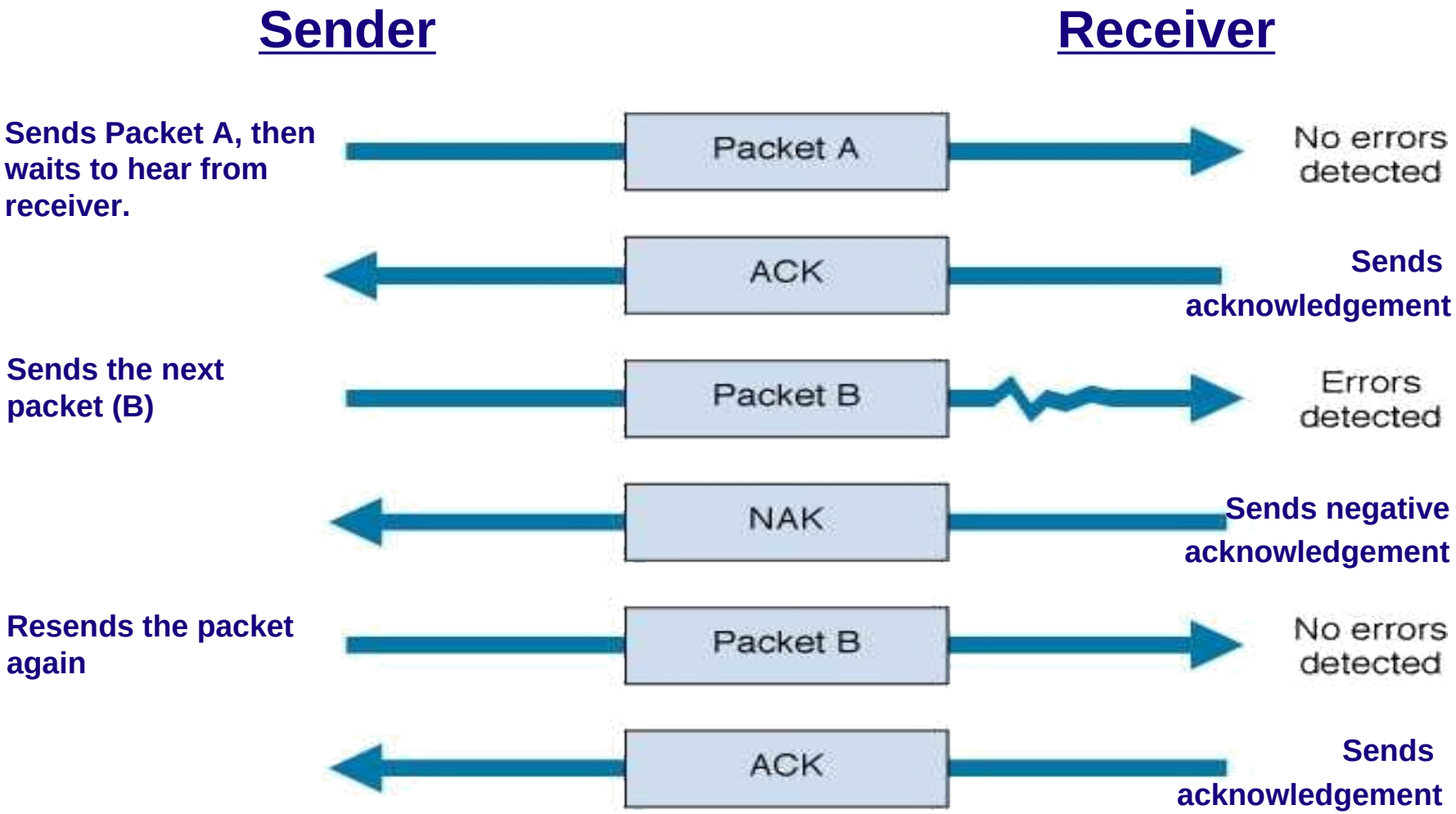
Copyright 2011 John Wiley & Sons, Inc

Automatic Repeat reQuest (ARQ)

- **Process of requesting a data transmission be resent**
- **Main ARQ protocols**
  - **Stop and Wait ARQ (A half duplex technique)**
    - **Sender sends a message and waits for acknowledgment, then sends the next message**
    - **Receiver receives the message and sends an acknowledgement, then waits for the next message**
  - **Continuous ARQ (A full duplex technique)**
    - **Sender continues sending packets without waiting for the receiver to acknowledge**
    - **Receiver continues receiving messages without acknowledging them right away**

Copyright 2011 John Wiley & Sons, Inc

Stop and Wait ARQ



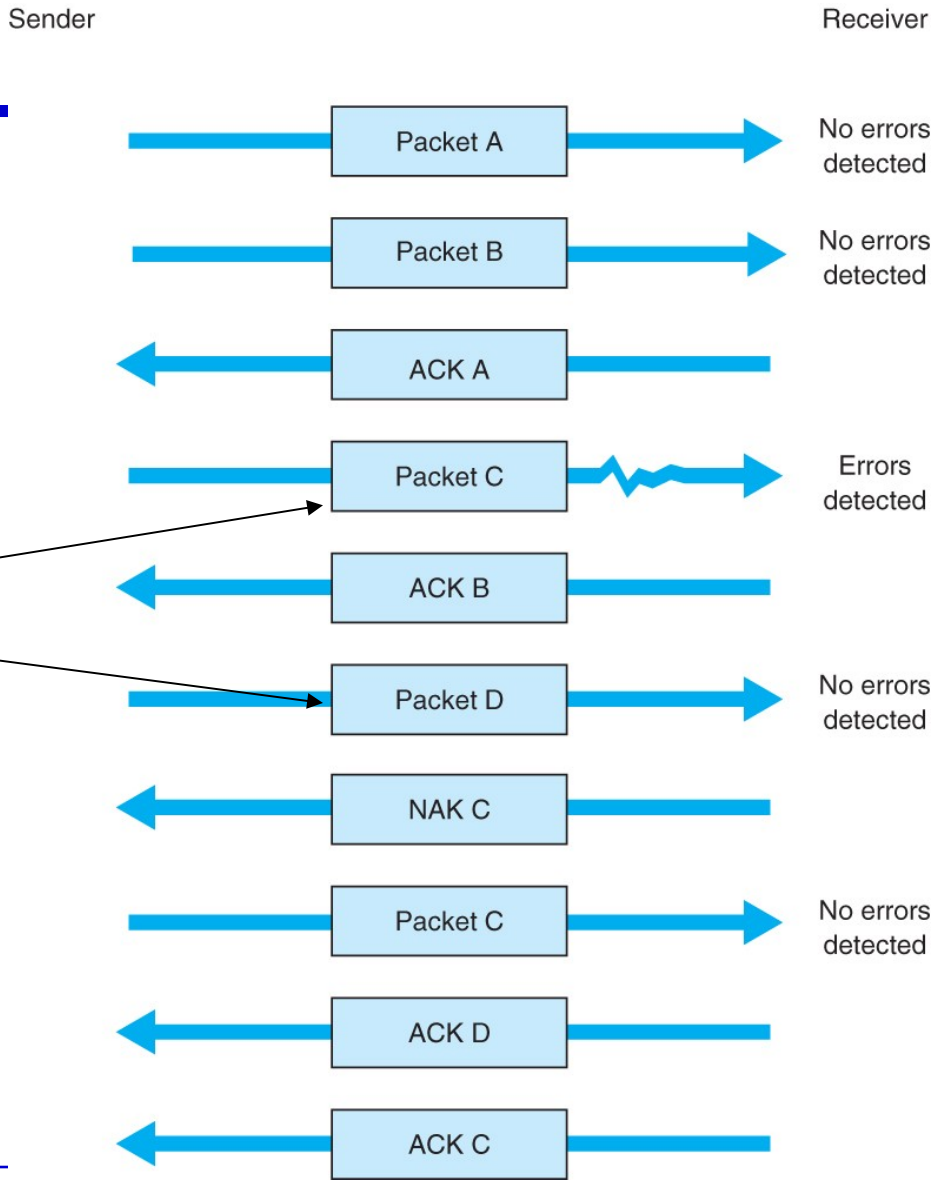
Copyright 2011 John Wiley & Sons, Inc

Continuous ARQ

Sender sends packets continuously without waiting for receiver to acknowledge

Notice that acknowledgments now identify the packet being acknowledged.

Receiver sends back a NAK for a specific packet to be resent.



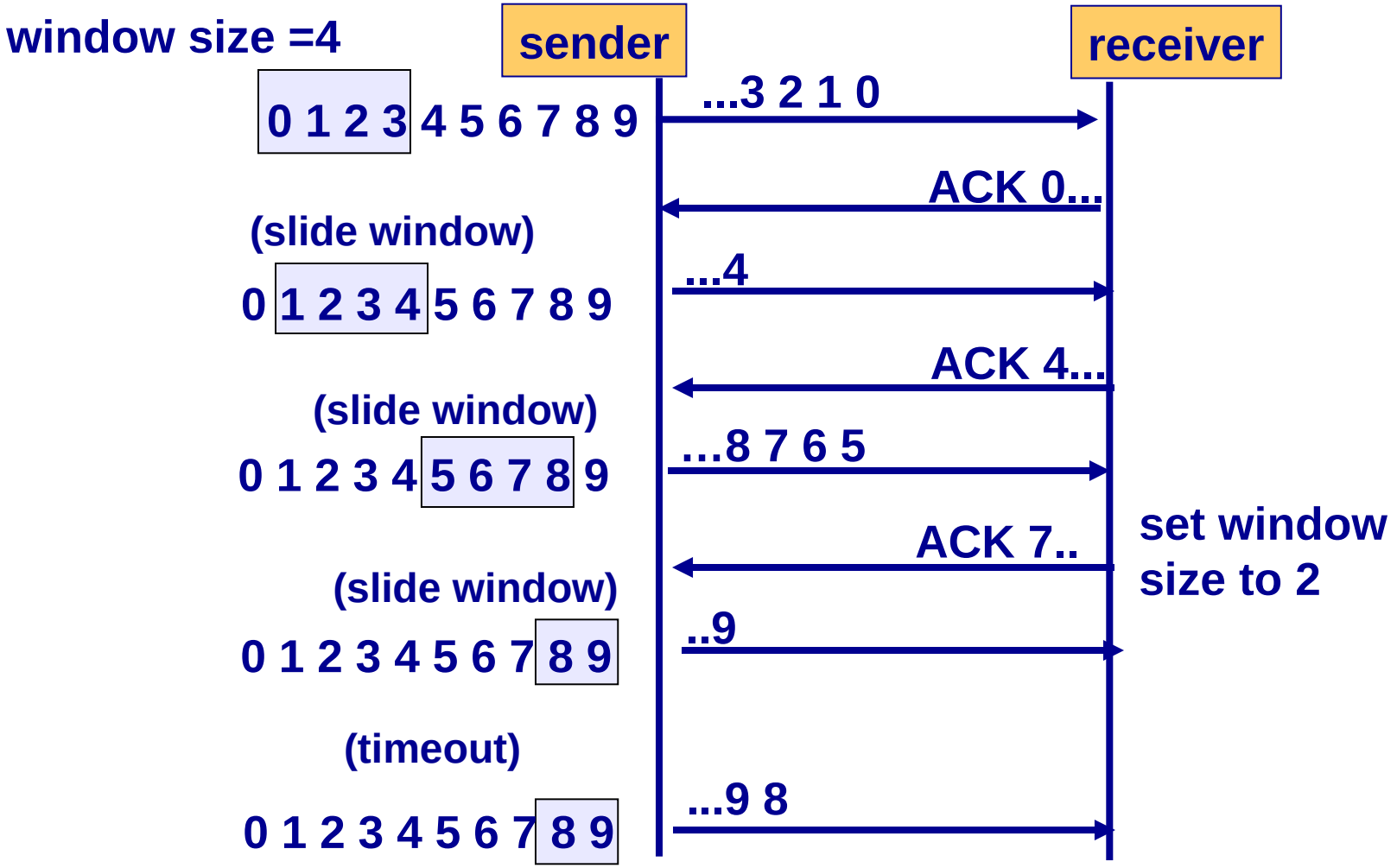
Copyright 2011 John Wiley & Sons, Inc

Flow Control with ARQ

- **Ensuring that sender is not transmitting too quickly for the receiver**
  - **Stop-and-wait ARQ**
    - **Receiver sends an ACK or NAK when it is ready to receive more packets**
  - **Continuous ARQ:**
    - **Both sides agree on the size of the "sliding window"**
      - **Number of messages that can be handled by the receiver without causing significant delays**

Copyright 2011 John Wiley & Sons, Inc

Flow Control Example



Copyright 2011 John Wiley & Sons, Inc

Forward Error Correction (FEC)

- **Receiving device can correct incoming messages itself (without retransmission)**
- **Requires extra corrective information**
  - **Sent along with the data**
  - **Allows data to be checked and corrected by the receiver**
  - **Amount of extra information: usually 50-100% of the data**
- **Used in the following situations:**
  - **One way transmissions (retransmission not possible)**
  - **Transmission times are very long (satellite)**
  - **In this situation, relatively insignificant cost of FEC**

Copyright 2011 John Wiley & Sons, Inc

Hamming Code

- **Before we discuss Hamming Codes it is a good idea to discuss “Parity bit” further**
- A parity bit is an extra bit that is attached to the information that is being sent
- This bit is attached to detect information error during the transmission.
- Added to make number of 1’s in the message or information either even or odd.
- If any change in any bit during transmission then there would be change in number of 1’s hence error would change the number of 1’s from odd to even or even to odd which can be detected.
- When we send information we’ll pass the information firstly through PARITY GENERATOR CIRCUIT and then send.
- At the receiver end information received is passed trough the PARITY CHECK CIRCUIT and if parity matches, then we use the information. If we make number of 1’s even then it is called even parity and if number of 1’s is made odd then it is called odd parity.

- **Example: Find the parity bit (odd) for message 1101 and show us how it helps in detecting errors?**
- As 1101 has 3 odd number of 1's so P=0
- We need to keep the odd number of 1's in the combination of 5 bits (message(4 bits) + parity bit(1 bit))
- Therefore message we send with parity is 11010 (5th bit from left is parity bit)
- Let's now see the effect of errors on this message

Error (1 error):

- Suppose we have error in 3rd bit from left so bit at 3rd position would change from 0 to 1.
- Hence message received would be 11110 instead of 11010
- The receiver checks the parity of message and we see message has even number 1′s in the received signal which should have been odd so we have detected the error
- But we are not sure about the position of the error.
- Now we should make a request for another send of the same message.

Error (2 errors):

- Suppose we have error at positions 1 and 4 from left so we have a bit change from 1 to 0 at 1st and 1 to 0 at 2nd position.
- Therefore messaged received is 01000 instead of 11010
- Now we have odd number of 1's in received message which is also the parity status of sent message
- No detection of errors
- Therefore this method is unable to detect even number of errors

Parity bit drawbacks:

- But there are few drawbacks attached with this method:
  - This system can detect only odd combinations of error, not even combinations. That means if there is 2 or 4 or 6 etc number of then it would go undetected while it can easily detect 1, 3, 5 etc number of errors.
  - We can not check the position of error even if we are able to detect it.

Copyright 2011 John Wiley & Sons, Inc

Hamming Code

- Used for single error correction (only detects 1 bit of error)

If we have num of information bits m and number of parity bits p is determined using following relation:

m = message
p = parity bit

$$2^P >= m + p + 1$$

We have to choose minimum value of p which satisfies this relation.

Hamming Code

- Suppose m=4
- determine number of parity bits using above relation
- 2↑p >= 4 + p + 1
- 2 ↑ p >= 5 + p
- From this we can check for values of p, which one satisfies
- For p=1 2 >= 6 doesn't satisfy
- For p=2 4>= 7 doesn't satisfy
- For p=3 8>=8 satisfies hence we have p=3
- So now we have 4 information bits and 3 parity bits so total of 7 bits. In the parity bit method, we placed the parity bit at rightmost position. But here we don't place the extra bits consecutively but the positions are fixed by following rule:

Copyright 2011 John Wiley & Sons, Inc

Hamming Code

- So now we have 4 information bits and 3 parity bits so total of 7 bits.
- In the parity bit method, we placed the parity bit at rightmost position.
- But here we don’t place the extra bits consecutively but the positions are fixed by following rule:

Parity bits are placed in the positions that are numbered corresponding to power of 2 as (1, 2, 4, 8, 16, 32 and so on)

Hamming Code

As we need only three positions so we have to pick first 3 which are 1, 2, and 4.

So we have the composition of hamming code as follow:

Bit1	bit2	bit3	bit4	bit5	bit6	bit7
Parity	parity		parity			
P_1	P_2	M_1	P_3	M_2	M_3	M_4

- Now we have to decide positions in the hamming code which would be covered by the parity bit
- i.e. the positions considering which value of parity bit would be decided.
- We'll be using following rule for this:

We first write the binary equivalent of position of the parity bit. As positions of parity bit is defined by power 2, so we would have only one 1 in the binary equivalent.

Then we consider only those message bits whose positions have 1 in the same position in it's binary equivalent as the position of 1 in binary equivalent of parity position

Hamming Code

E.g. Consider the parity bit P_1 and we have to find the position of message bits which we'll cover with this parity bit.

Firstly write the binary equivalents of positions of message bit

Bit1	bit2	bit3	bit4	bit5	bit6	bit7
Parity	parity		parity			
P_1	P_2	M_1	P_3	M_2	M_3	M_4
001	010	011	100	101	110	111

Now let's see in the binary equivalent of position of parity bit P_1 that at which position we have 1and we see 1 is at LSB so we select the message bits which have positions with 1 at LSB which are M_1, M_2 and M_4 So P_1 bit would check the parity for M_1, M_2 and M_4

Note: (LSB)less significant bit

Hammin Code

Furthermore

E.g. Consider the parity bit P_2 and we have to find the position of message bits which we'll cover with this parity bit

We have 1 at second position from left so we choose message bits which have 1 at 2nd position n their position's binary equivalent. Hence we get message bits M_1 $M_3$and M_4. So **P_2** checks parity for message bits of M_1 M_3 and M_4

Similarly we have 1 at 3rd position of P_3 message bits with 1 at 3rd position are $M_2M_3M_4$

So we now have

P_1 Checks bit number 1,3,5,7

P_2 Checks bit number 2,3,6,7

P_3 Checks bit number 4,5,6,7

These parity bits can be either even or odd parity bits but all parity bits must be same i.e. all odd or all even

Copyright 2011 John Wiley & Sons, Inc

Hamming code

So form hamming code using 4-bit message bits 1101 with parity bits as even parity bit check how we are able to detect and correct error.

As we have already decided parity bit positions and their corresponding message bits for a 4-bit message

1 2 3 4 5 6 7

For the moment we have hamming code as P_1 P_2 1 P_3 1 0 1

As we have already seen **P_1** Checks bit number 1,3,5,7

So **P_1 = 1 to make number of 1's to 4 i.e. even in positions 1,3,5,7**

P_2 Checks bit number 2,3,6,7

So P_2 = 0 to make number of 1's to 2 i.e. even in positions 2,3,6,7

P_3 Checks bit number 4,5,6,7

So P_3 = 0 to make number of 1's to 2 i.e. even in positions 4,5,6,7

Hence we have the hamming code as 1010101

Hamming Code

- Since hamming code only detects and corrects single error.
- Assume we have error at 5th position which means bit changes from 1 to 0 so we have data changed from 1010101 to 1010001

P1 P2 1 P3 1 0 1

- Now let's start checking all 3 parity bits starting from $\mathbf{P_1}$
- $\mathbf{P_1}$ **Checks** bit number 1,3,5,7 and we see we have number 1's in these bits is 3 i.e. odd which is wrong as it should have been even so put a **1**
- $\mathbf{P_2}$ **Checks** bit number 2,3,6,7 and we see we have number 1's in these bits is 2 i.e. even which is right so put a **0**
- $\mathbf{P_3}$ **Checks** bit number 4,5,6,7 and we see we have number 1's in these bits is 1 i.e. odd which is wrong as it should have been even so put a **1**
- Now we collect all the bits recorded with bit from $\mathbf{P_1}$ as LSB
- So we get **101** hence we get bit 5 with

Hamming Code

- **So as P1 and P3 give wrong results so now we need to find which code bit position is common but not found in code bit position of parity bit P2**
- **We see that 5 and 7 are common for P1 and P3 but 7 is also present in P2 so we are left with code position 5.**

- **Hence we have the bit change in position 5 so re-change it to get the correct value. Hence we change 5th bit of received message which is 1010001, we get 1010101 which is correct one.**
- **So we see hamming code is able to detect and correct single error.**

Copyright 2011 John Wiley & Sons, Inc

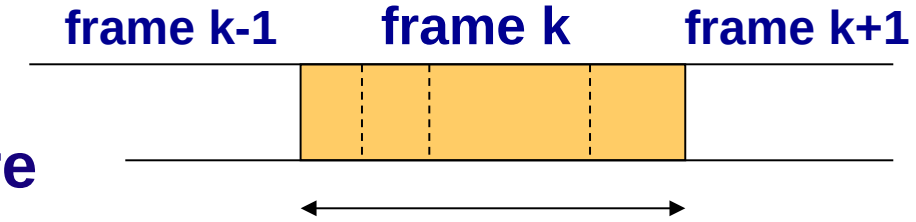
Asynchronous File Transfer

- **Used on**
  - **Point-to-point asynchronous circuits**
  - **Typically over phone lines via modem**
  - **Computer to computer for transfer of data files**
- **Sometimes called Start/Stop Transmission**
- **Characteristics of file transfer protocols**
  - **Designed to transmit error-free data**
  - **Group data into blocks to be transmitted (rather sending character by character)**
- **Popular File transfer Protocols**
  - **Xmodem, Zmodem, and Kermit**

Copyright 2011 John Wiley & Sons, Inc

4.4 Data Link Protocols

- **Classification**
  - **Asynchronous transmission**
  - **Synchronous transmission**
- **Differ by**
  - **Message delineation**
  - **Frame length**
  - **Frame field structure**



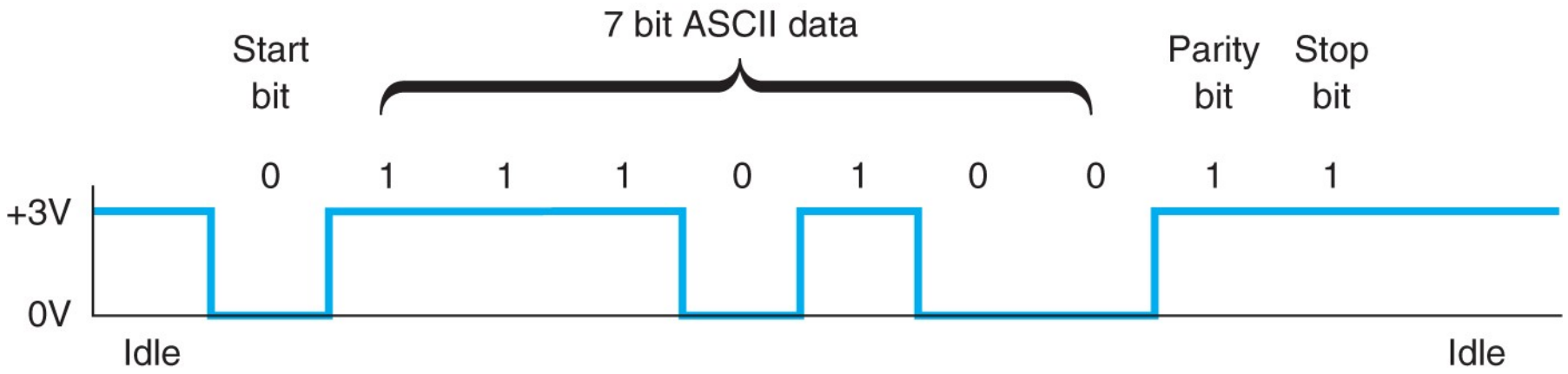
Copyright 2011 John Wiley & Sons, Inc

Asynchronous Transmission

Start bit used by the receiver for separating characters and for synch.

Each character is sent independently

Stop bits sent between transmissions (a series of stop bits)



Copyright 2011 John Wiley & Sons, Inc

File Transfer Protocols

Xmodem:

- One of the oldest async file transfer protocol
- Uses stop-and-wait ARQ (Repeat ReQuest).

Xmodem-CRC:

- uses 1 byte CRC (instead of checksum)

Xmodem-1K:

- Xmodem-CRC + 1024 byte long message field

Zmodem:

- Uses CRC-32 with continuous ARQ
- Dynamic adjustment of packet size

Copyright 2011 John Wiley & Sons, Inc

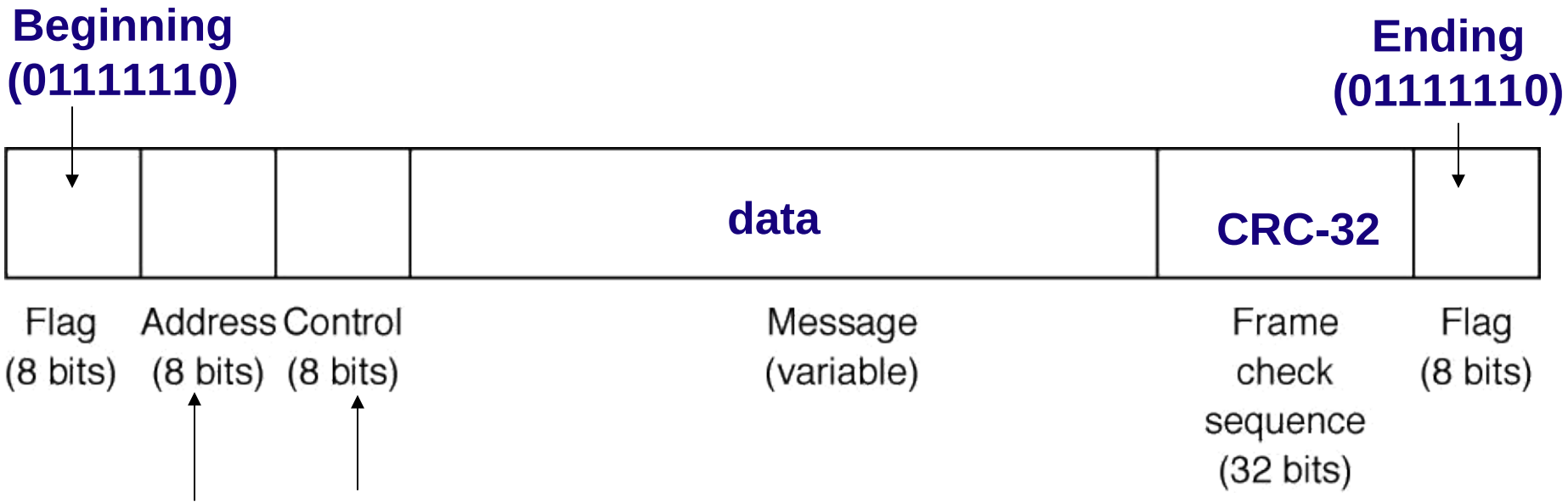
Synchronous Transmission

- **Data sent in a large block**
  - **Called a frame or packet**
  - **Typically about a thousand characters (bytes) long**
- **Includes addressing information**
  - **Especially useful in multipoint circuits**
- **Includes a series of synchronization (SYN) characters**
  - **Used to help the receiver recognize incoming data**
- **Synchronous transmission protocols categories**
  - **Bit-oriented protocols: SDLC, HDLC**
  - **Byte-count protocols: Ethernet**
  - **Byte-oriented protocols: PPP**

Copyright 2011 John Wiley & Sons, Inc

SDLC – Synchronous Data Link Control

- **Bit-oriented protocol developed by IBM**
- **Uses a controlled media access protocol**



Copyright 2011 John Wiley & Sons, Inc

Transparency Problem of SDLC

- **Problem: Transparency**
  - **User data may contain the same bit pattern as the flags (01111110)**
  - **Receiver may interpret it as the end of the frame and ignores the rest**
- **Solution: Bit stuffing (aka zero insertion)**
  - **Sender inserts 0 anytime it detects 11111 (five 1's)**
  - **If receiver sees five 1's, checks next bit(s)**
    - **if 10, end of frame marker (01111110)**
    - **if 11, error (7 1's cannot be in data)**
  - **Works but increases complexity**

Copyright 2011 John Wiley & Sons, Inc

HDLC – High-Level Data Link Control

- **Formal standard developed by ISO**
- **Same as SDLC, except**
  - **Longer address and longer control fields**
  - **Larger sliding window size**
  - **And more**
- **Basis for many other Data Link Layer protocols**
  - **LAP-B (Link Access Protocol – Balanced)**
    - **Used by X.25 technology**
  - **LAP-D (Link Access Protocol – Balanced)**
    - **Used by ISDN technology**
  - **LAP- F (Used by Frame Relay technology)**

Copyright 2011 John Wiley & Sons, Inc

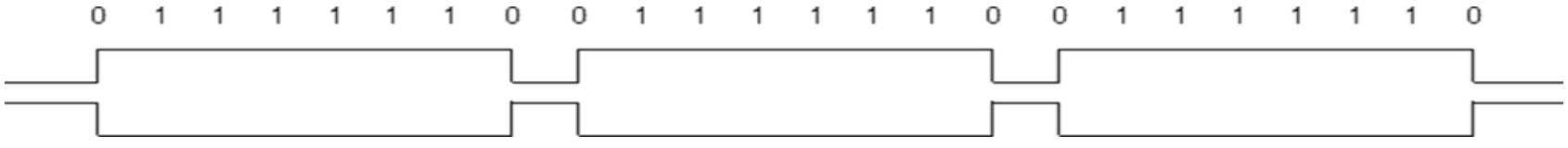
HDLC – High-Level Data Link Control

- **HDLC is a bit-oriented code-transparent synchronous data link layer protocol developed by the International Organization for Standardization (ISO).**
- **HDLC provides both connection-oriented and connectionless service.**
- **HDLC can be used for point to multipoint connections, but is now used almost exclusively to connect one device to another, using what is known as Asynchronous Balanced Mode (ABM).**
  - **synchronous Balanced Mode (ABM) is a communication mode of HDLC and derivative protocols, supporting peer-oriented point-to-point communications between two nodes, where either node can initiate transmission. There is no master/slave relationship. Each station may initialize, supervise, recover from errors, and send frames at any time. The DTE (Data Terminal Equipment) and DCE (Data Circuit Terminating Equipment) are treated as equals.**

HDLC – High-Level Data Link Control

Note: Connection-oriented communication is a network communication mode in telecommunications and computer networking, where a communication session or a semi-permanent connection is established before any useful data can be transferred, and where a stream of data is delivered in the same order as it was sent. Connection-oriented communication may be a circuit switched connection, or a packet-mode virtual circuit connection

On the contrary in connectionless communication, such as in the datagram mode communication used by the IP and UDP protocols, data may be delivered out of order, since different packets are routed independently, and may be delivered over different paths.



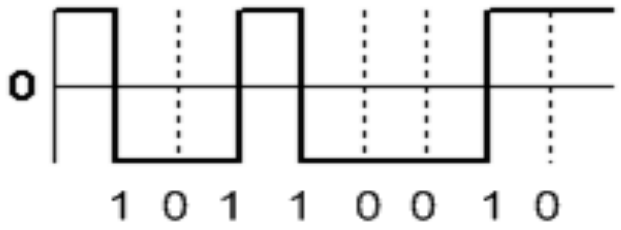
HDLC – High-Level Data Link Control

- **HDLC frames can be transmitted over synchronous or asynchronous links.**
- **Since those links have no mechanism to mark the beginning or end of a frame, the beginning and end of each frame has to be identified.**
- **This is done by using a frame delimiter, or flag, which is a unique sequence of bits that is guaranteed not to be seen inside a frame. This sequence is '01111110'. Each frame begins and ends with a frame delimiter. A frame delimiter at the end of a frame may also mark the start of the next frame. A sequence of 7 or more consecutive 1-bits within a frame will cause the frame to be aborted.**
- **When no frames are being transmitted on a simplex or full-duplex synchronous link, a frame delimiter is continuously transmitted on the link. Using the standard NRZI encoding from bits to line levels (0 bit = transition, 1 bit = no transition), this generates one of two continuous waveforms, depending on the initial state (0 or 1).**

HDLC – High-Level Data Link Control

Note:

- **Non return to zero, inverted (NRZI) is a method of mapping a binary signal to a physical signal for transmission over some transmission media. The two level NRZI signal has a transition at a clock boundary if the bit being transmitted is a logical 1, and does not have a transition if the bit being transmitted is a logical 0.**
- **"One" is represented by a transition of the physical level.**
- **"Zero" has no transition.**



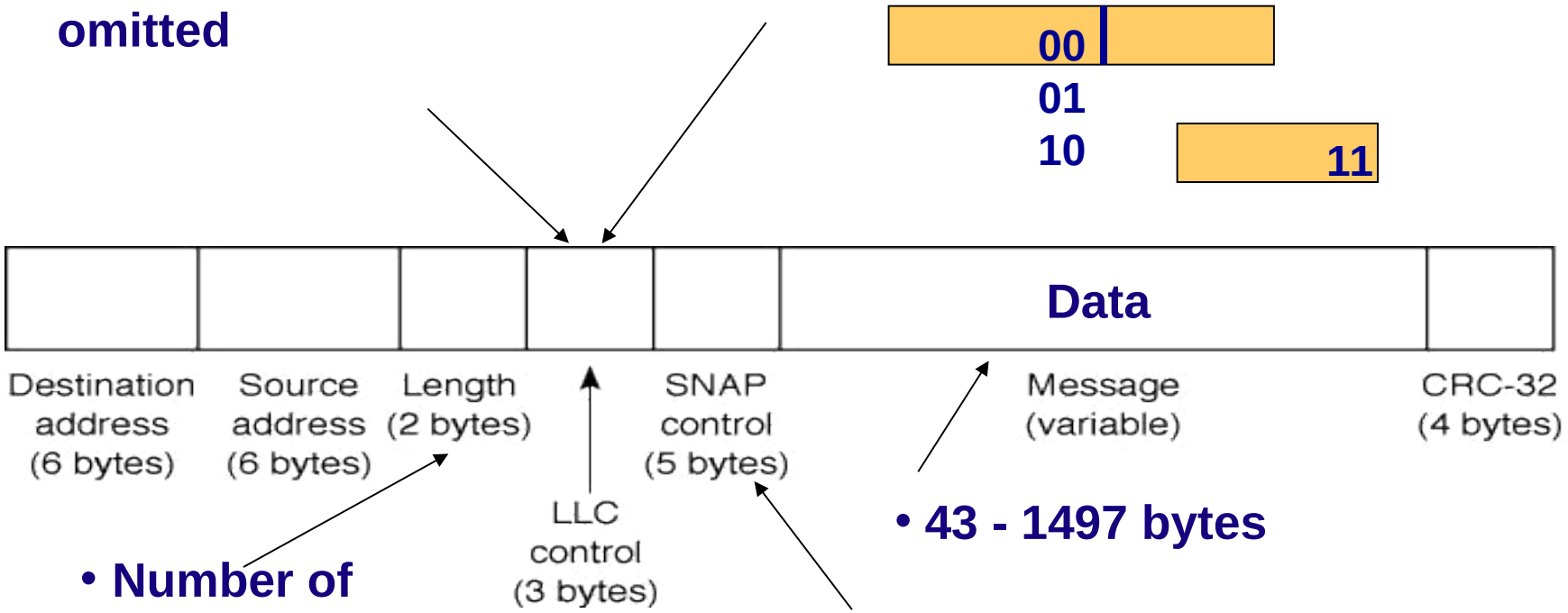
Ethernet (IEEE 802.3)

- **Most widely used LAN protocol, developed jointly by Digital, Intel, and Xerox, now an IEEE standard**
- **Uses contention based media access control**
- **Byte-count data link layer protocol**
- **No transparency problem**
  - **uses a field containing the number of bytes (not flags) to delineate frames**
- **Error correction: optional**

Copyright 2011 John Wiley & Sons, Inc

Ethernet (IEEE 802.3) Frame

- **Used by Virtual LANs; if no vLAN, the field is omitted**
- **Used to hold sequence number, ACK/NAK, (1 or 2 bytes)**



- **43 - 1497 bytes**
- **Number of bytes in the message field**
- **Used to exchange control info (e.g., type of network layer protocol used)**

Note: LLC (Logical Link Control
& SNAP: SubNetwork Access Protocol
is an a standard for the transmission of IP datagrams over IEEE 802 networks. In other words, IP datagrams could be sent on IEEE 802 networks encapsulated within the 802.2 LLC and SNAP data link layers and the 802.3, 802.4 or 802.5 physical network layers.

Point-to-Point Protocol (PPP)

- **Byte-oriented protocol developed in early 90s**
- **Commonly used on dial-up lines from home PCs**
- **Designed mainly for point-to-point phone line (can be used for multipoint lines as well)**



(up to 1500 bytes)

Specifies the network layer protocol used (e.g, IP, IPX)

Copyright 2011 John Wiley & Sons, Inc

Data Link Protocol Summary

Protocol	Size	Error Detection	Retransmission	Media Access
Asynchronous Xmission	1	Parity	Continuous ARQ	Full Duplex
File Transfer Protocols				
XMODEM	132	8-bit Checksum	Stop-and-wait ARQ	Controlled Access
XMODEM-CRC	132	8-bit CRC	Stop-and-wait ARQ	Controlled Access
XMODEM-1K	1028	8-bit CRC	Stop-and-wait ARQ	Controlled Access
ZMODEM	*	32-bit CRC	Continuous ARQ	Controlled Access
KERMIT	*	24-bit CRC	Continuous ARQ	Controlled Access
Synchronous Protocols				
SDLC	*	16-bit CRC	Continuous ARQ	Controlled Access
HDLC	*	16-bit CRC	Continuous ARQ	Controlled Access
Token Ring	*	32-bit CRC	Stop-and wait ARQ	Controlled Access
Ethernet	*	32-bit CRC	Stop-and wait ARQ	Contention
SLIP	*	None	None	Full Duplex
PPP	*	16-bit CRC	Continuous ARQ	Full Duplex

*** Varies depending on message length.**

Copyright 2011 John Wiley & Sons, Inc

4.5 Transmission Efficiency

- **One objective of a data communication network** is to move the highest possible volume of accurate **in**formation through the network. The higher the volume, the greater the result**in**g network's **efficiency** and the lower the cost.
- Network **efficiency** is affected by characteristics of the circuit such as: error rate, maximum **transmission** speed, speed of transmitt**in**g and receiv**in**g equipment, the error-detection and control methodology, and the protocol used by the data l**in**k layer.

Copyright 2011 John Wiley & Sons, Inc

4.5 Transmission Efficiency

- **Each protocol we discussed uses some bits** (or bytes) to del**in**eate the start and end of each message and to control error.
- These bits and bytes are necessary for the **transmission** to occur,
- But they are not part of the message.
- They add no value to the user,
- But they count aga**in**st the total number of bits that can be transmitted.

Copyright 2011 John Wiley & Sons, Inc

4.5 Transmission Efficiency

- **Each communication protocol** has both **in**formation bits and overhead bits.
- **In**formation bits are those used to convey the user's mean**in**g.
- Overhead bits are used for purposes such as error check**in**g and mark**in**g the start and end of characters and packets.
- A parity bit used for error check**in**g is an overhead bit because it is not used to send the user's data;
- If you did not care about errors, the overhead error check**in**g bit could be omitted and the users could still understand the message.

4.5 Transmission Efficiency

- **Transmission efficiency is defined** as the total number of **in**formation bits (i.e., bits **in** the message sent by the user) divided by the total bits **in transmission** (i.e., **in**formation bits plus overhead bits).
- For example, let's calculate the **transmission efficiency** of asynchronous **transmission**.
- Assume we are us**in**g 7-bit ASCII. We have 1 bit for parity, plus 1 start bit and 1 stop bit. Therefore, there are 7 bits of **in**formation **in** each letter, but the total bits per letter is 10 (7 + 3).
- The **efficiency** of the asynchronous **transmission** system is 7 bits of **in**formation divided by 10 total bits, or 70 percent.
- **In other words,** with asynchronous **transmission**, only 70 percent of the data rate is available for the user;
- 30 percent is used by the **transmission** protocol.
- If we have a communication circuit us**in**g a dial-up **modem** receiv**in**g **56** Kbps, the user sees an effective data rate (or throughput) of 39.2 Kbps. This is very **in**efficient.

7-bit ASCII (info bits), 1 parity bit, 1 stop bit, 1 start bit

Transmission Efficiency = 7 / 10 → 70%

4.5 Transmission Efficiency

- **We can improve efficiency by reducing the number of overhead bits in each message**
- **or by increasing the number of information bits.**
- **For example, if we remove the stop bits from asynchronous transmission, efficiency increases to 7/9, or 77.8 percent.**
- **The throughput of a dial-up modem at 56 Kbps would increase 43.6 Kbps, which is not great but is at least a little better.**

Copyright 2011 John Wiley & Sons, Inc

4.5 Transmission Efficiency

- **The same basic formula can be used to** calculate the **efficiency** of synchronous **transmission**.
- For example, suppose we are us**in**g SDLC. The number of **in**formation bits is calculated by determ**inin**g how many **in**formation characters are **in** the message.
- If the message portion of the frame conta**in**s 100 **in**formation characters and we are us**in**g an 8-bit code, then there are 100 x 8 = 800 bits of **in**formation.
- The total number of bits is the 800 **in**formation bits plus the overhead bits that are **in**serted for del**in**eation and error control.
- SDLC has a beg**in**n**in**g flag (8 bits), an address (8 bits), a control field (8 bits), a frame check sequence (assume we use a CRC-32 with 32 bits), and an end**in**g flag (8 bits). This is a total of 64 overhead bits; thus, **efficiency** is 800/(800 + 64) = 92.6 percent. If the circuit provides a data rate of **56** Kbps, then the effective data rate available to the user is about 51.9 Kbps**.**

4.5 Transmission Efficiency

- **These examples show that synchronous networks are more efficient than asynchronous networks**
- **Also some protocols are more efficient than others.**
- **The longer the message (1,000 characters as opposed to 100), the more efficient the protocol.**
- **For example, suppose the message in the SDLC example were 1,000 bytes. The efficiency here would be 99.2 percent, or 8,000/(8000 + 64), giving an effective data rate of about 55.6 Kbps (almost 56K, equal to modem capacity).**

4.5 Transmission Efficiency

- **An objective of the network:**
  - **Move as many bits as possible with minimum errors**

    → higher efficiency and lower cost
- **Factors affecting network efficiency:**
  - **Characteristics of circuit (error rate, speed)**
  - **Speed of equipment, Error control techniques**
  - **Protocol used**
    - **Information bits (carrying user information)**
    - **Overhead bits (used for error checking, frame delimiting, etc.)**

$$\text{Transmission efficiency} = \frac{\text{Total number of info bits to be transmitted}}{\text{Total number of bits transmitted}}$$

Copyright 2011 John Wiley & Sons, Inc

Transmission Efficiency of Protocols

Async Transmission:

7-bit ASCII (info bits), 1 parity bit, 1 stop bit, 1 start bit

Transmission Efficiency = 7 / 10 → 70%

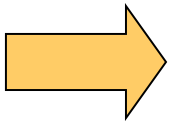
e.g., V.92 modem with 56 Kbps → 39.2 Kbps effective rate

SDLC Transmission

Assume 100 info characters (800 bits), 2 flags (16 bits)
Address (8 bits), Control (8 bits), CRC (32 bits)

Transmission Efficiency = 800 / 64 → 92.6%

e.g., V.92 modem with 56 Kbps → 51.9 Kbps effective rate



Bigger the message length, better the efficiency

However, large packets likely to have more errors and are more likely to require retransmission → wasted capacity

Copyright 2011 John Wiley & Sons, Inc

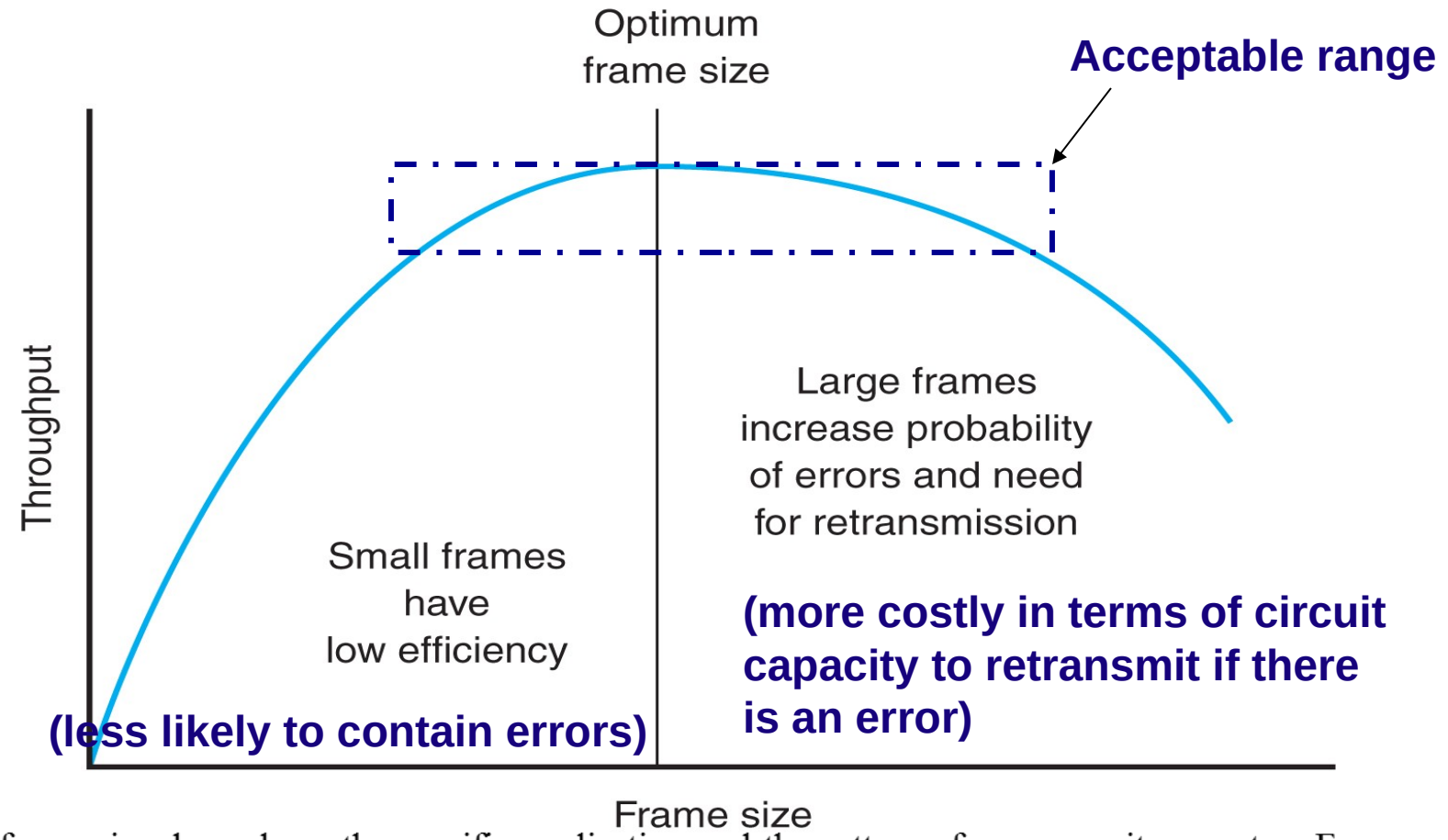
Throughput

- **A more accurate definition of efficiency**
- **Total number of information bits received per second; takes into account:**
  - **Overhead bits (as in transmission efficiency)**
  - **Need to retransmit packets containing errors**
- **Complex to calculate; depends on:**
  - **Transmission efficency**
  - **Error rate**
  - **Number of retransmission**
- **Transmission Rate of Information Bits (TRIB)**
  - **Used as a measurement of throughput**

Copyright 2011 John Wiley & Sons, Inc

Optimum Packet Size

Trade-off between packet size and throughput



The ideal frame size depends on the specific application and the pattern of messages it generates. For Standard Commercial, the ideal frame size appeared to be between 4,000 and 8,000. Unfortunately, not all network software packages enable network managers to fine-tune frame sizes in this way.

Transmission Rate of Information Bits (TRIB)

FORMULA FOR CALCULATING TRIB

$$\text{TRIB} = \frac{\text{Number of information bits accepted}}{\text{Total time required to get the bits accepted}}$$

$$\text{TRIB} = \frac{K(M - C)(1 - P)}{(M/R) + T}$$

where K = information bits per character

M = frame length in characters

R = data transmission rate in characters per second

C = average number of noninformation characters per frame (control characters)

P = probability that a frame will require retransmission because of error

T = time between frames in seconds, such as modem delay/turnaround time on half duplex and propogation delay on satellite transmission. This is the time required to reverse the direction of transmission from send to receive or receive to send on a half-duplex circuit. It can be obtained from the modem specification book and may be referred to as *reclocking time*.

Copyright 2011 John Wiley & Sons, Inc

4.6 Implications for Management

- **Provide a few, widely used data link layer protocols for all networks**
  - **Minimize costly customization**
  - **Minimize costly translation among many protocols**
  - **Less training, simpler network management**
  - **Bigger pool of available experts**
  - **Less expensive, off-the-shelf equipment**

Copyright 2011 John Wiley & Sons, Inc

Copyright 2011 John Wiley & Sons, Inc.

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information herein.