Data Communication and Computer Networks

8. Link Layer
PART-A

Dr. Aiman Hanna

Department of Computer Science & Software Engineering
Concordia University, Montreal, Canada

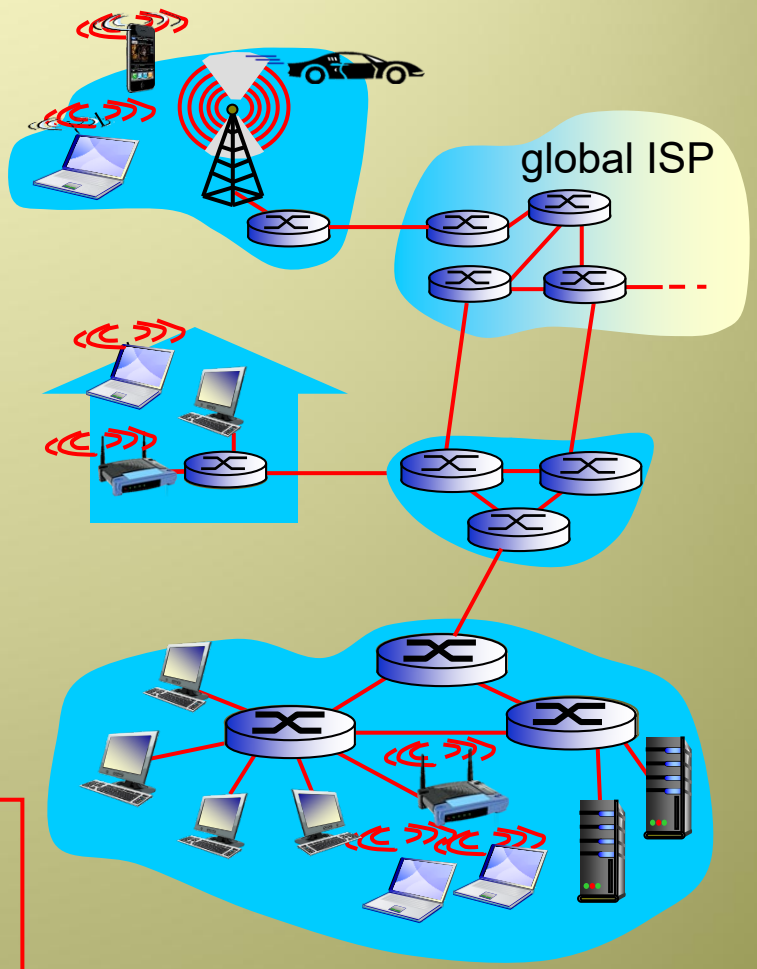
These slides have mainly been extracted, modified and updated from original slides of :
Computer Networking: A Top Down Approach, 6th edition Jim Kurose, Keith Ross
Addison-Wesley, 2013

Additional materials have been extracted, modified and updated from:
Understanding Communications and Networking, 3e by William A. Shay 2005

Link layer: introduction

terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
  - LANs
- layer-2 packet: frame, encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link

Link layer: context

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- each link protocol provides different services
  - e.g., may or may not provide rdt over link

transportation analogy:

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

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!

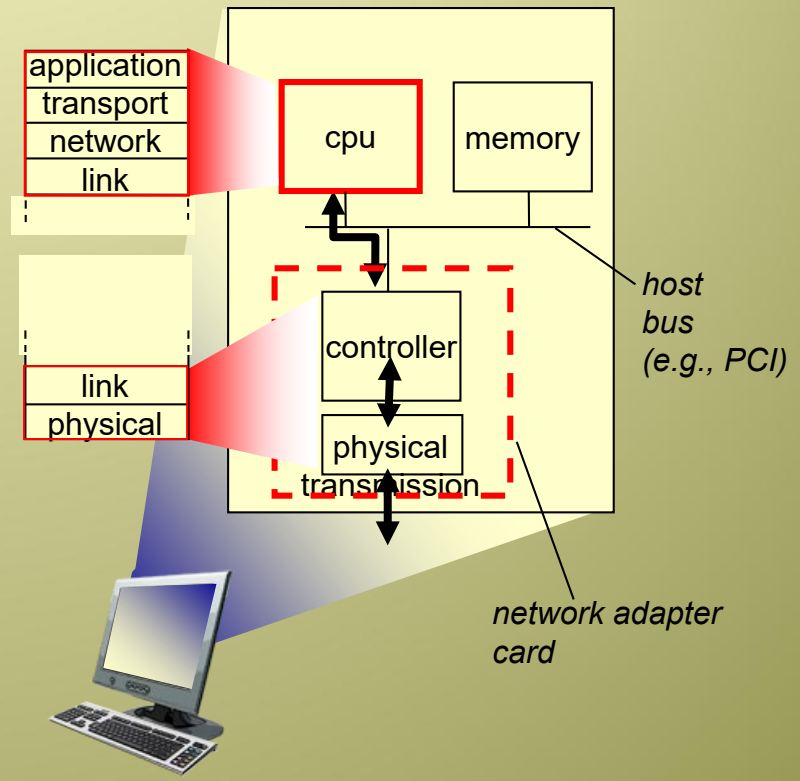
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?

Link layer services (more)

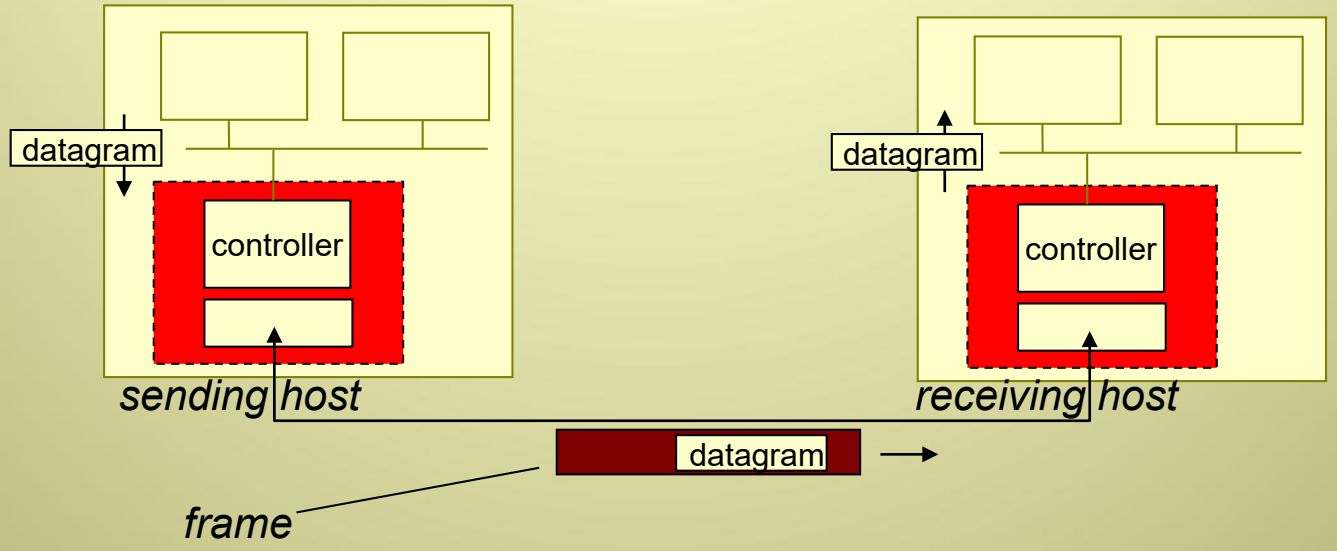
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection:*
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit 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

Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Adaptors communicating



- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

Data Integrity

- *a possible reality:*

  Data transferred → data altered → data received

- *electrical interference, power fluctuation, sunspots, storms, …etc. are all factors*

- ***error detection** is hence needed*

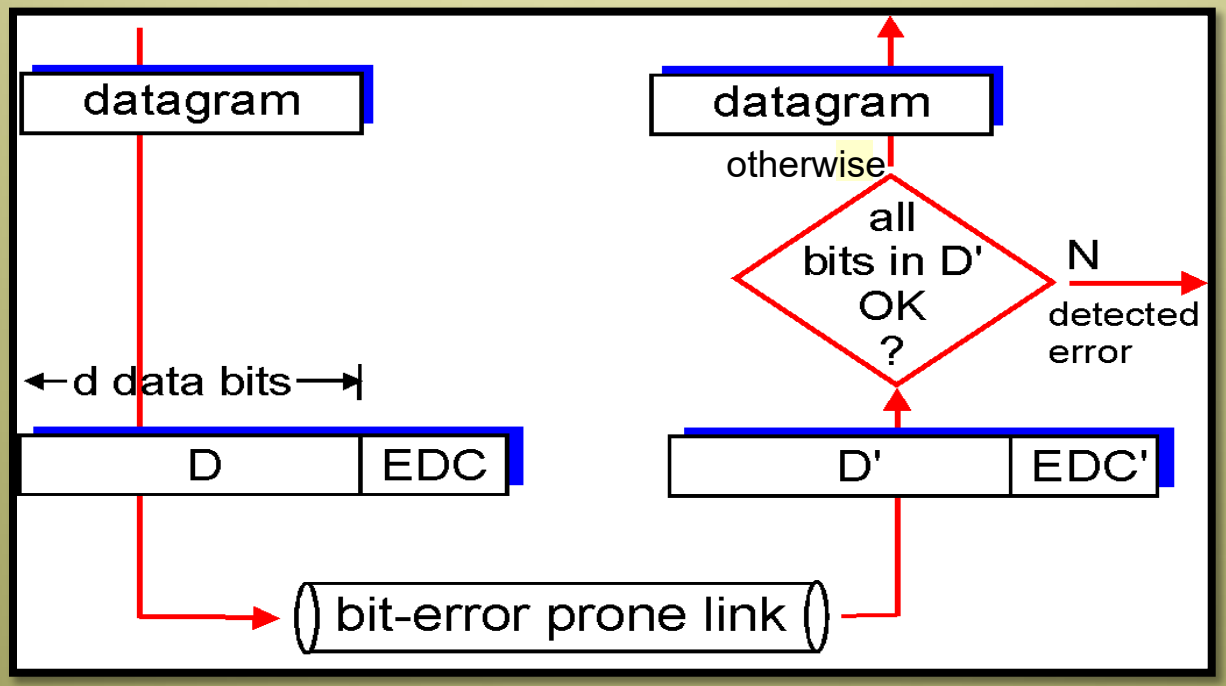
- *what can be done if errors are detected?*

- ***error correction** may sometimes be the most suitable solution*

Error detection

EDC = Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



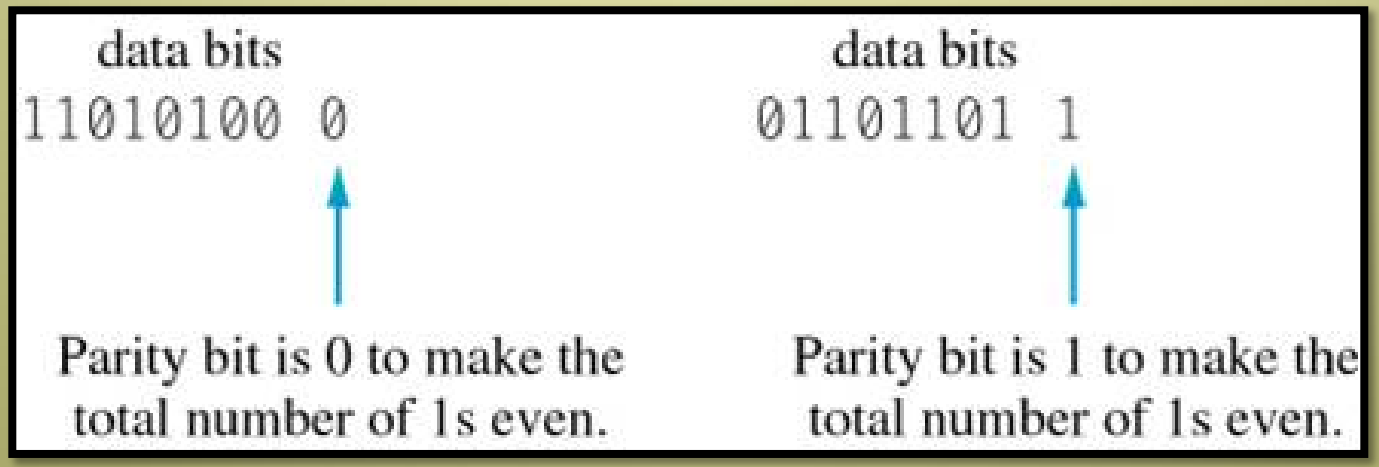
Parity checking

- *simple/naïve*
- *send additional bits along with data*
- *the value of those additional bits depend on the data itself*
- *theoretically, if data is alerted, the value of the additional bits will no longer correspond to the new data*

Parity checking

Single-bit parity:

- detect single bit errors
- *parity checking* – ***even parity*** & ***odd parity***
- *one* ***parity bit*** *is added to achieve that*



Detecting Single-bit Errors Using Even Parity Checking

Parity checking

- how many errors can a parity check detect?

- Examples: Assume the use of even parity
  Sent: 10100010 **1** → Received: 10101010 **1** ✋
  Sent: 10011110 **1** → Received: 10101110 **1** ☑
  Sent: 10001110 **0** → Received: 11111110 **0** ✋
  Sent: 10001110 **0** → Received: 11111110 **1** ☑

✋ ≡ **Errors detected**
☑ ≡ **No errors detected**

Notice that all the above examples do have errors!

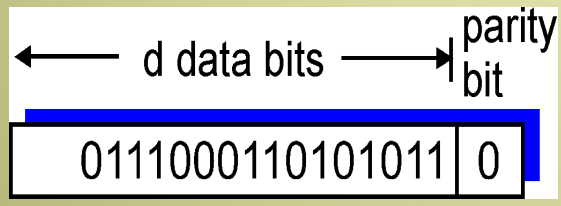
Parity checking

- in reality, single-bit errors are rare
- an error for a duration of 1/100 second over a 10Mbps line may affect 10,000 bits
- when many bits are damaged, we refer to that as ***burst errors***
- in average, parity check would catch about 50% of burst errors
- in other words, it is 50% accurate, which is simply not good for communications networks
- this however does not mean that parity checking is useless. Why?

Parity checking

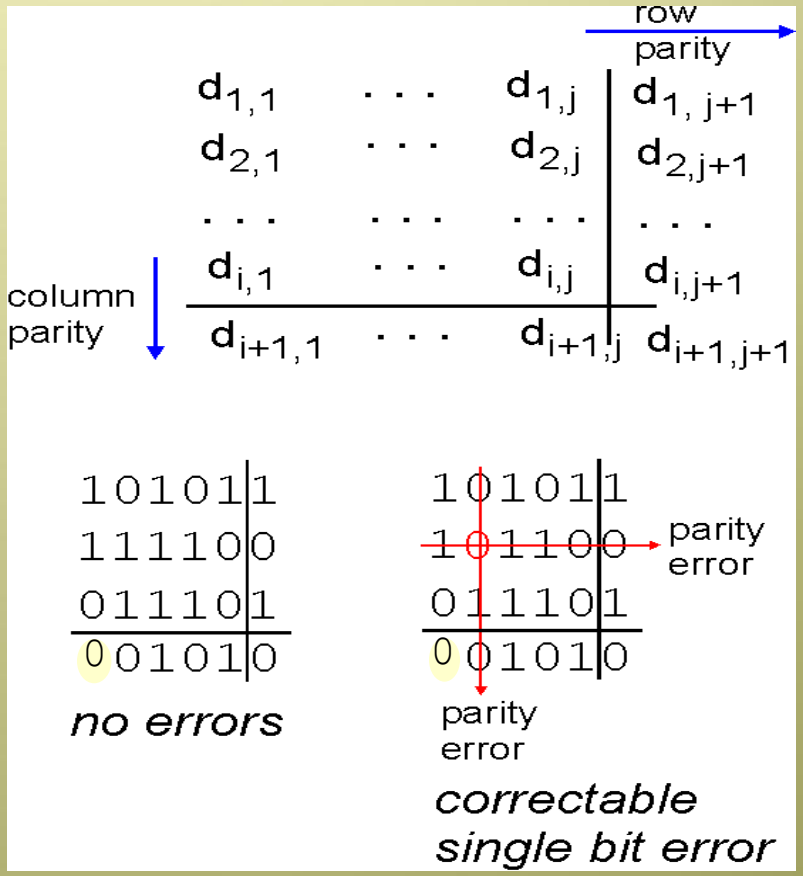
single bit parity:

- *d*etect single bit errors



two-dimensional bit parity:

- detect and correct single bit errors



Checksum

- divide all data bits into groups (i.e. 32-bit groups)
- treat each group as an integer value
- the values are then added together to give a ***checksum***
- the checksum is then appended to the sent data and tested at the receiving end
- more accurate than parity checking, however it may not detect all errors. Why?

Internet checksum (review)

goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

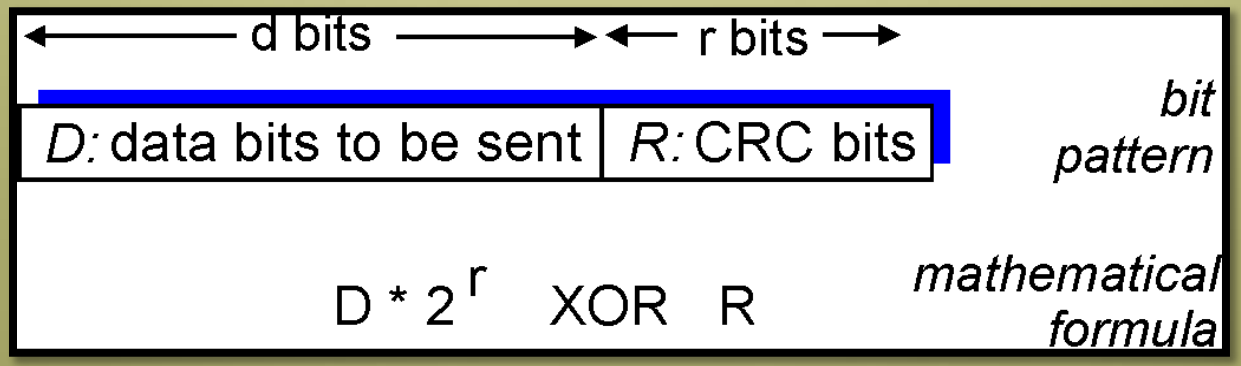
- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

Cyclic Redundancy Check (CRC)

- more powerful error-detection coding
- view data bits, D, as a binary number
- choose bit pattern (generator), G

- goal: choose CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides received <D,R> by G. If non-zero remainder: error detected!

- widely used in practice (Ethernet, 802.11 WiFi, ATM)



Cyclic Redundancy Check (CRC)

Cyclic Redundancy Checks (CRC) - Details

- errors are detected via ***polynomial division***
- in general, a polynomial interpretation is made for any bit string
- for example, the bit string

  $$b_{n-1}b_{n-2}b_{n-3}\ldots. b_2b_1b_0$$

  is interpreted as the polynomial

  $$b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + b_{n-3}x^{n-3} + \ldots.. + b_2x^2 + b_1x + b_0$$

- since each bit here is either 0 or 1, we can consider x^i only when b=1
- for example, the bit string **10010101110**

  is interpreted as

  $$x^{10} + x^7 + x^5 + x^3 + x^2 + x^1$$

Cyclic Redundancy Check (CRC)

Cyclic Redundancy Checks (CRC) - *Polynomial Division*

- Modulo 2 addition and subtraction (Ex-Or arithmetic) are as follows:

  Addition: 0+0=0; 1+0=1; 0+1=1; 1+1=0

  Subtraction: 0-0=0; 1-0=1; 0-1=1; 1-1=0

- Assume *T(x)* and *G(x)* are two polynomials as follows:

  $T(x) = x^{10} + x^9 + x^7 + x^5 + x^4$

  $G(x) = x^4 + x^3 + x^0$

- What is *T(x)/G(x)*?

Cyclic Redundancy Check (CRC)

Cyclic Redundancy Checks (CRC) - *Polynomial Division*

What is *T(x)/G(x)?*

$$
\begin{array}{r}
 & & & & & & & x^6 & & + x^3 & & + x \\
\hline
x^4 + x^3 + 1 \;\big)\; x^{10} + x^9 & & & + x^7 & & + x^5 + x^4 & & & & & & \\
x^{10} + x^9 & & & + x^6 & & & & & & & & \\
\hline
 & & & x^7 + x^6 & + x^5 + x^4 & & & & & & & \\
 & & & x^7 + x^6 & & + x^3 & & & & & & \\
\hline
 & & & & x^5 + x^4 + x^3 & & & & & & & \\
 & & & & x^5 + x^4 & & + x & & & & & \\
\hline
 & & & & & & x^3 & & + x & & \text{remainder} &
\end{array}
$$

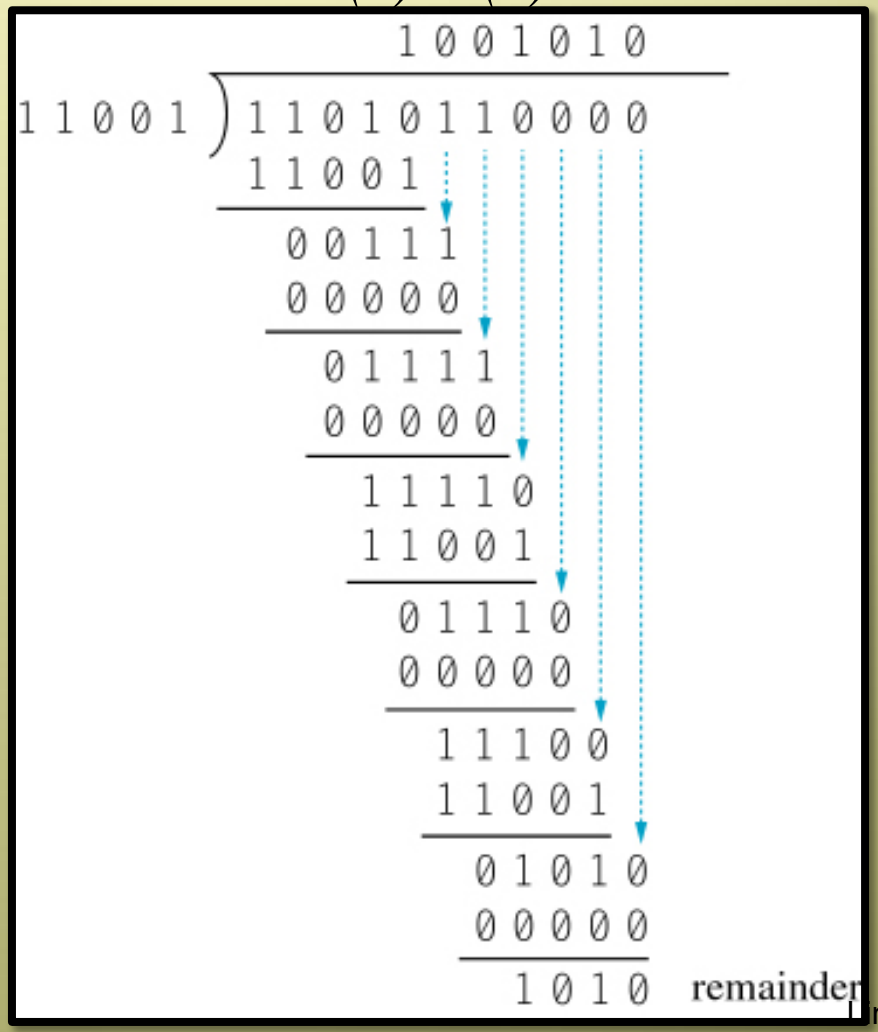
Calculation of $(x^{10} + x^9 + x^7 + x^5 + x^4)/(x^4 + x^3 + 1)$

Cyclic Redundancy Check (CRC)

Cyclic Redundancy Checks (CRC) - *Polynomial Division*

The equivalent synthetic division of *T(x)/G(x) would look as follow*

Synthetic Division of
$(x^{10} + x^9 + x^7 + x^5 + x^4)/(x^4 + x^3 + 1)$



Cyclic Redundancy Check (CRC)

Cyclic Redundancy Checks (CRC)

- How does CRC work?
  - *Given a bit string, append several 0s to it, and call be **B***
  - *Find the polynomial interpretation of B; that is **B(x)***
  - *Agree on some polynomial value **G(x)** (called **generator polynomial**)*
  - *Divide **B(x)/G(x)** and find the remainder polynomial **R(x)***
  - *Define **T(x) = B(x) – R(x)***
  - *Find the bit string of T(x); call it **T***
  - *Transmit **T***
  - *Let **T'** is the received string*
  - *Divide **T'(x)/G(x)***
  - ➔ *if there is a **zero remainder** then **T = T'**, which means no errors*
  - *Otherwise, errors are detected*

Cyclic Redundancy Check (CRC)

- Example: Suppose the bit string 1101011 is to be sent
  - *Assume* $G(x) = x^4 + x^3 + 1$
  - *Append four 0s at the end of the string* ➔ $B = 11010110000$
  - $B(x) = x^{10} + x^9 + x^7 + x^5 + x^4$
  - $B(x)/G(x)$ ➔ *The remainder* $R(x) = x^3 + x$
  - $T(x) = B(x) - R(x)$; *this can be done by subtracting over the bit strings*

    11010110000 bit string B

    -1010 bit string R

    11010111010 bit string T ***(Notice that T(x)/G(x) has a zero remainder)***

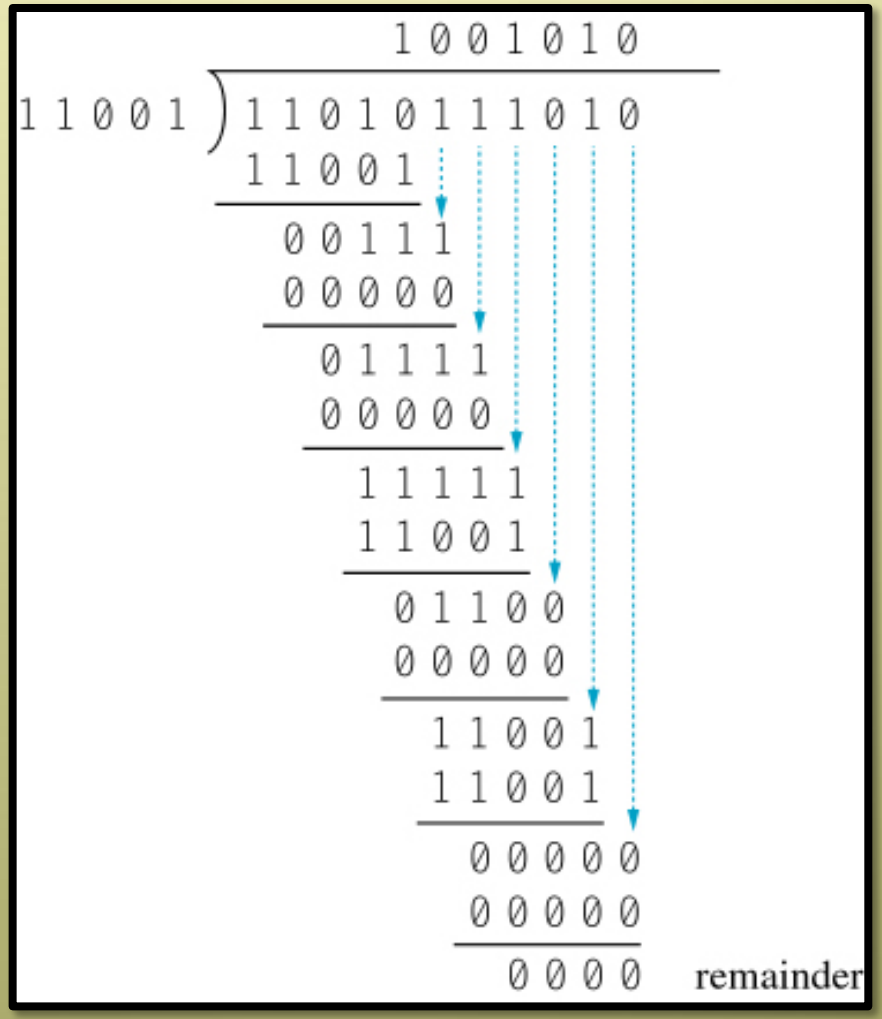
In algebra the following analogy is true: If *p* and *q* are integers and *r* is the remainder of *p/q*, then *p-r* is evenly divisible by *q*. for example, 8/3 generates a remainder of 2. 8-2 is evenly divisible by 3.

Cyclic Redundancy Check (CRC)

Cyclic Redundancy Checks (CRC)

- Example (continues…):

Dividing *T(x)/G(x)*



Cyclic Redundancy Check (CRC)

Cyclic Redundancy Checks (CRC)

- Example (continues…):
  - *Transmit **T***
  - *Let **T'** is the received string*
  - *Divide **T'(x)/G(x)***
  - ➔ *if there is a **zero remainder** then **T = T'**, which means no errors*
  - *Otherwise, errors are detected*

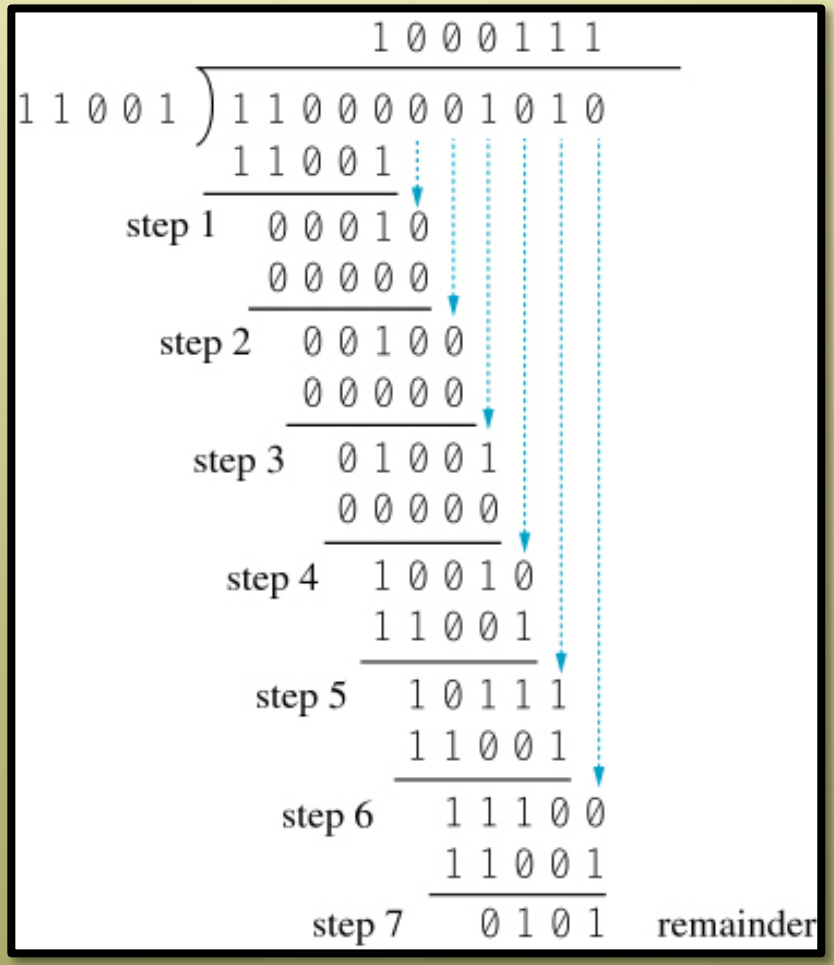
  - *Notes: In practice the following holds true:*
    - *A non-zero remainder concludes that errors have occurred*
    - *A zero remainder however is not that conclusive. Unless G(x) is chosen carefully, it is possible that a damaged T' may still result in a zero remainder when T'(x) is divided by G(x)*

Cyclic Redundancy Check (CRC)

CRC Implementation Using Circular Shifts

- Performing CRC check for each data arrival is a massive overhead
- How fast can the polynomial division be performed? Does it have to be complete?

Dividing *T(x)/G(x)*
When Error Occurs



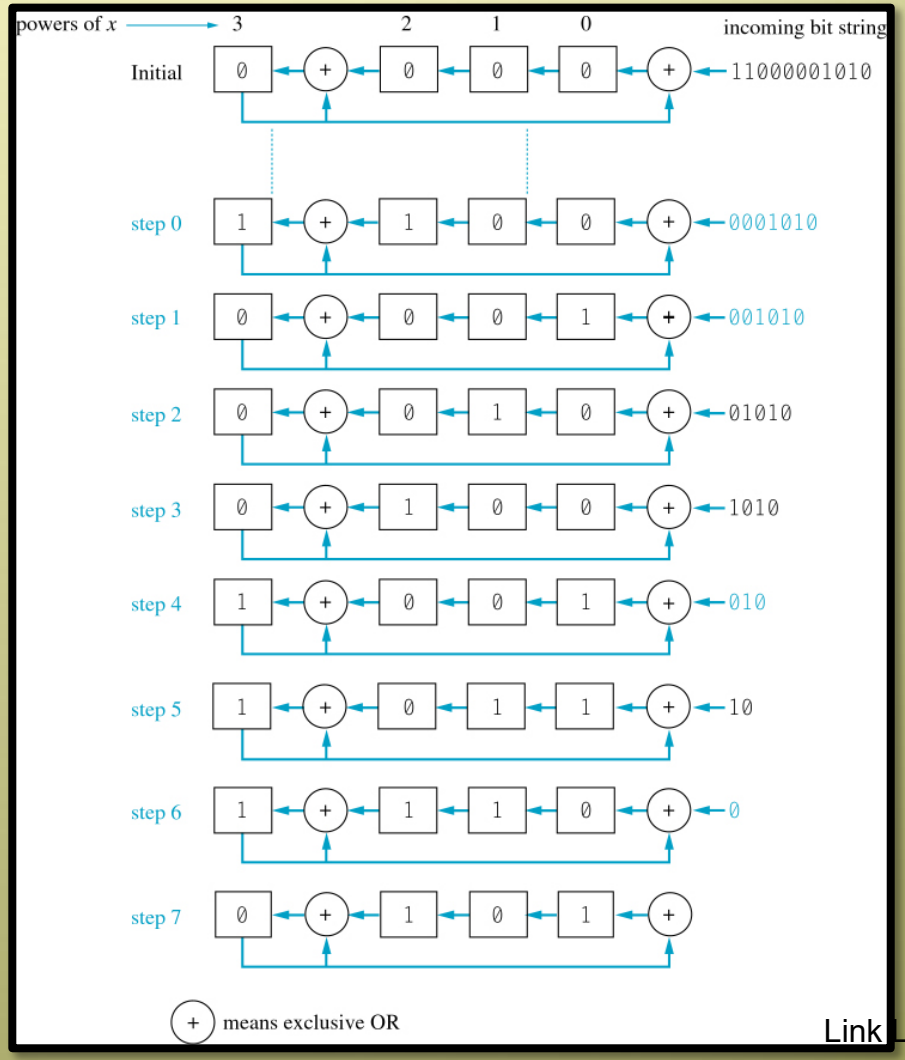
Cyclic Redundancy Check (CRC)

CRC Implementation Using Circular Shifts

- division using circular shifts

Dividing *T(x)/G(x)*
Using Circular Shift

[G(x) = $x^4 + x^3 + 1$
& hardware has 4 registers]



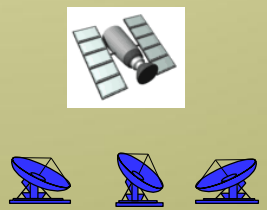
Multiple access links, protocols

two types of "links":

- point-to-point
  - i.e. point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - old-fashioned Ethernet
  - 802.11 wireless LAN

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

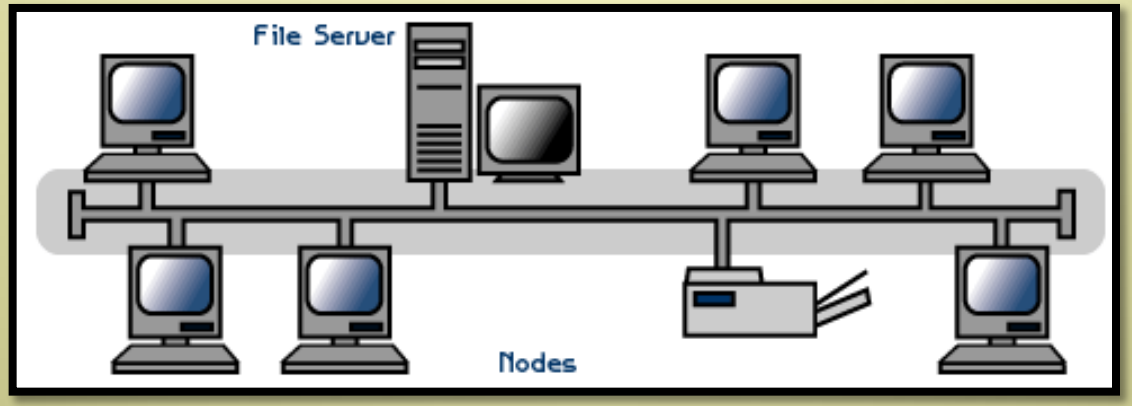
Network topologies

- a network is a system connecting different devices
- the connection strategy to connect these devices is referred to as **Network Topology**

- the best topology depends on the network needs

- possible network topologies are:
  - Common Bus Topology
  - Star Topology
  - Ring Topology
  - Fully Connected Topology
  - Tree Topology
  - Combined Topology

Common bus topology



- simply known as ***Bus Topology*** or ***Linear Bus Topology***
- ***terminators*** are placed at both ends

Common bus topology

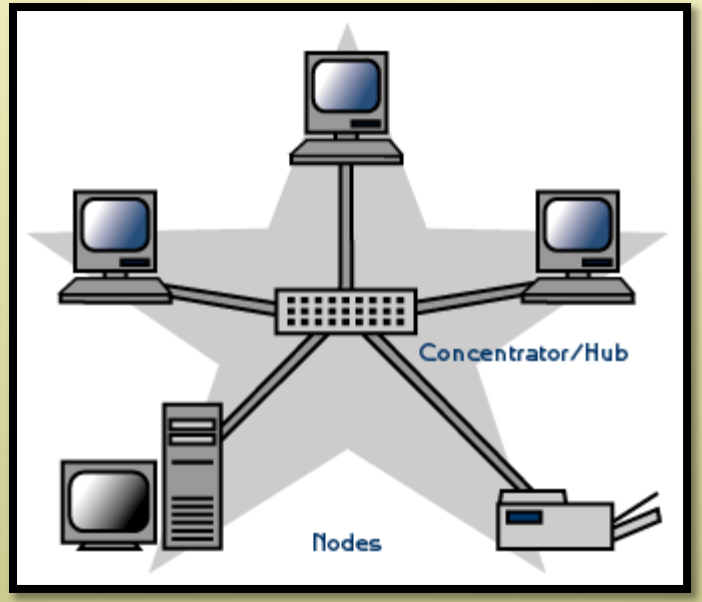
- collision is possible
- a collision results on *noise*
- the interfaces detect the noise on the bus, stop transmission, wait for a random amount of time then attempt transmission again

- this process is called **Carrier Sense Multiple Access with Collision Detection** (**CSMA/CD**)

- the **Ethernet** (original configuration) is an example of a common bus network

- Advantages?
- Disadvantages?

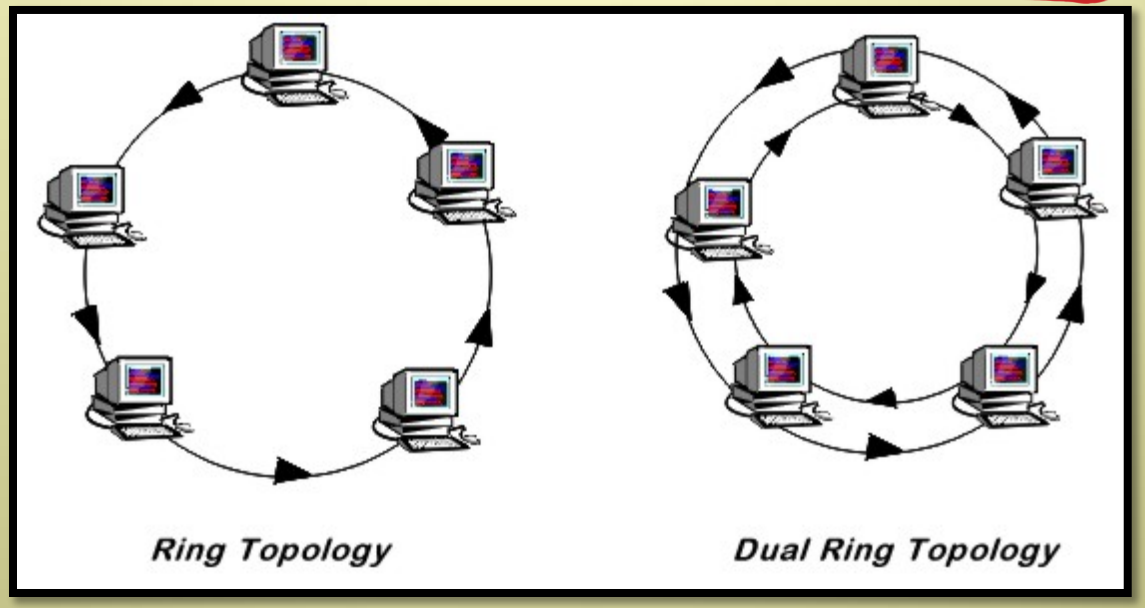
Star Topology



- the central component is commonly called **hubs** or **switches**

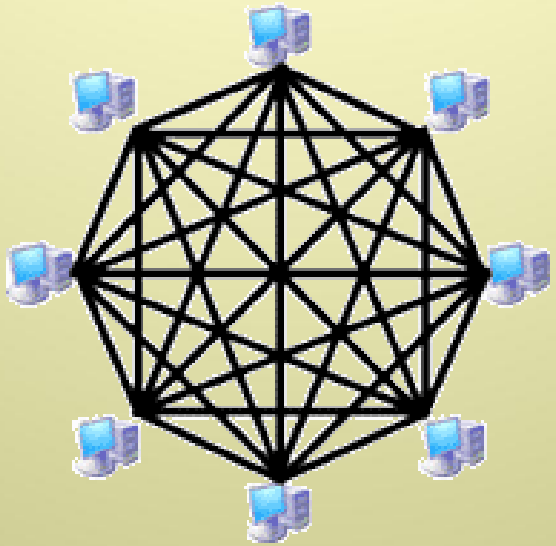
- Advantages?
- Disadvantages?

Ring Topology



- the ring can either be **unidirectional** or **bidirectional (dual)**
- IBM's **Token Ring** network is an example of a ring topology, where a token (sequence of bits) is passed between the devices.
- Advantages?
- Disadvantages?

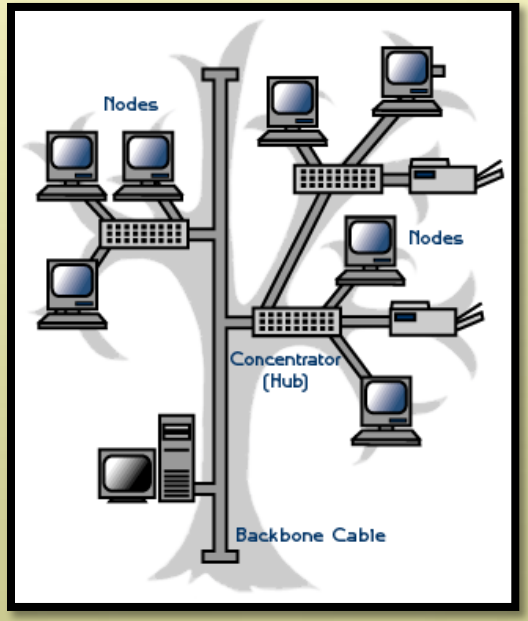
Fully-Connected Topology



- a direct link is there between each pair of devices in the network; this design is extreme

- Advantages?
- Disadvantages?

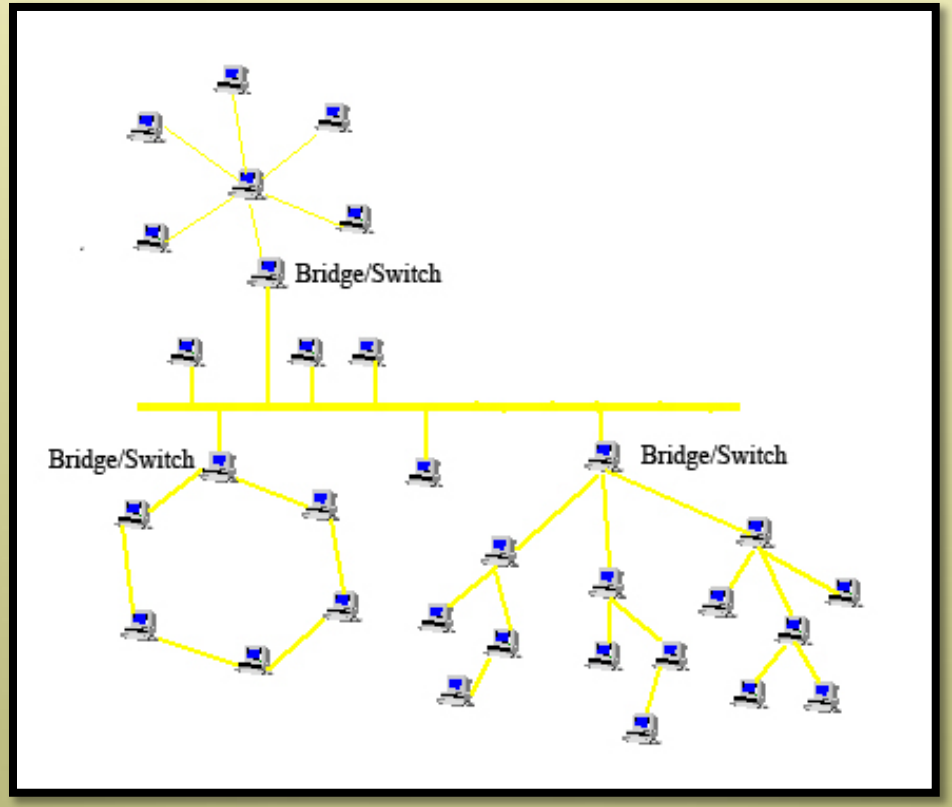
Tree Topology



- combines characteristics of linear bus and star topologies
- allows for the expansion of an existing network

- Advantages?
- Disadvantages?

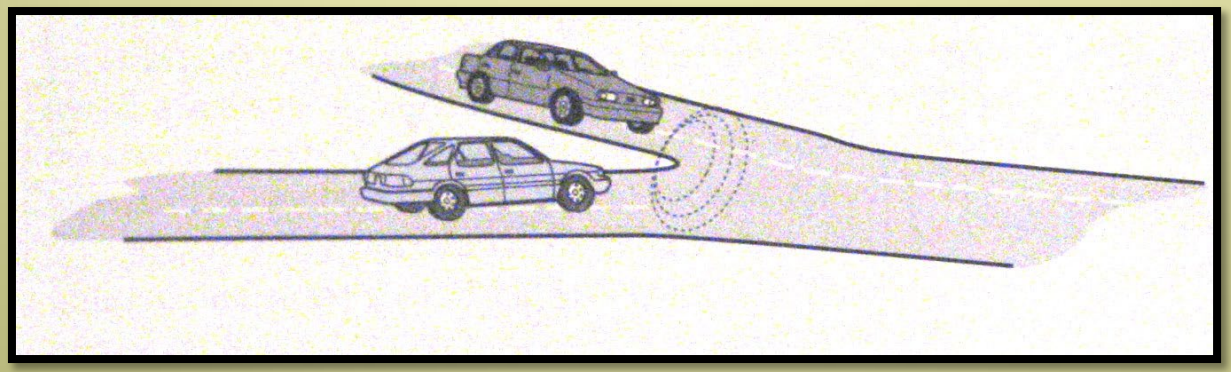
Combined topologies



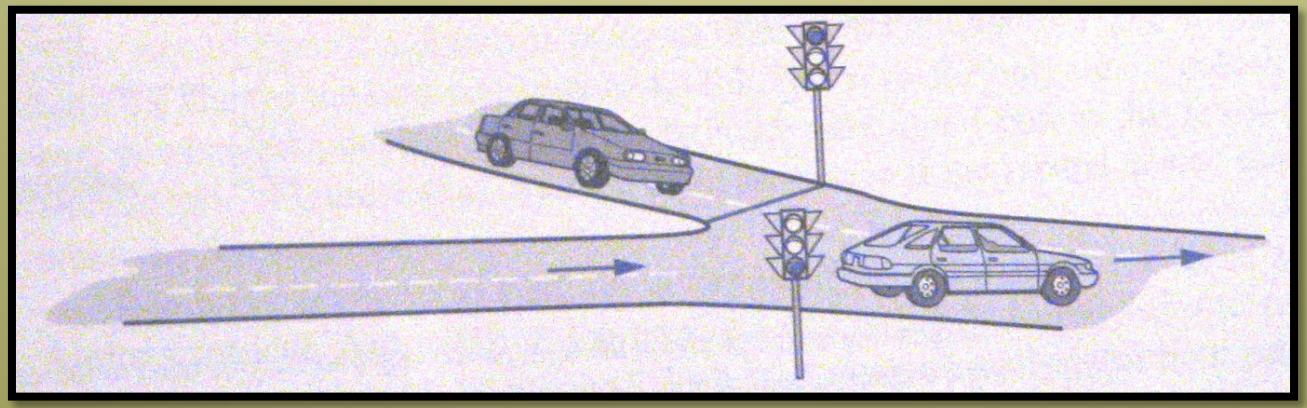
- used by many computer networks
- groups of specialized users may need to have a separate LAN
- different LANs are connected through bridges/switches

Contention protocols

- access to the medium from many entry points is called contention
- unless controlled, contention may lead to fatal problems
- contention protocols are used to avoid such problems



No Contention Protocol



Stop-and-Go Access Protocol

Multiple access protocols

- single shared broadcast channel

- two or more simultaneous transmissions by nodes: interference
  - *collision* if node receives two or more signals at the same time

multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit

- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

An ideal multiple access protocol

given: broadcast channel of rate R bps

desiderata:

1. when one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. simple

MAC protocols: taxonomy

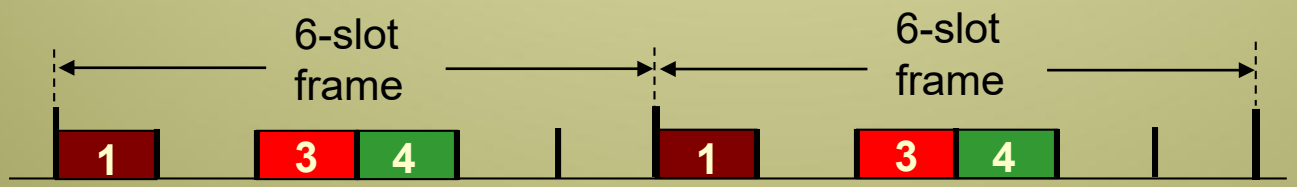
three broad classes:

- *channel partitioning*
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- *random access*
  - channel not divided, allow collisions
  - "recover" from collisions
- *"taking turns"*
  - nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

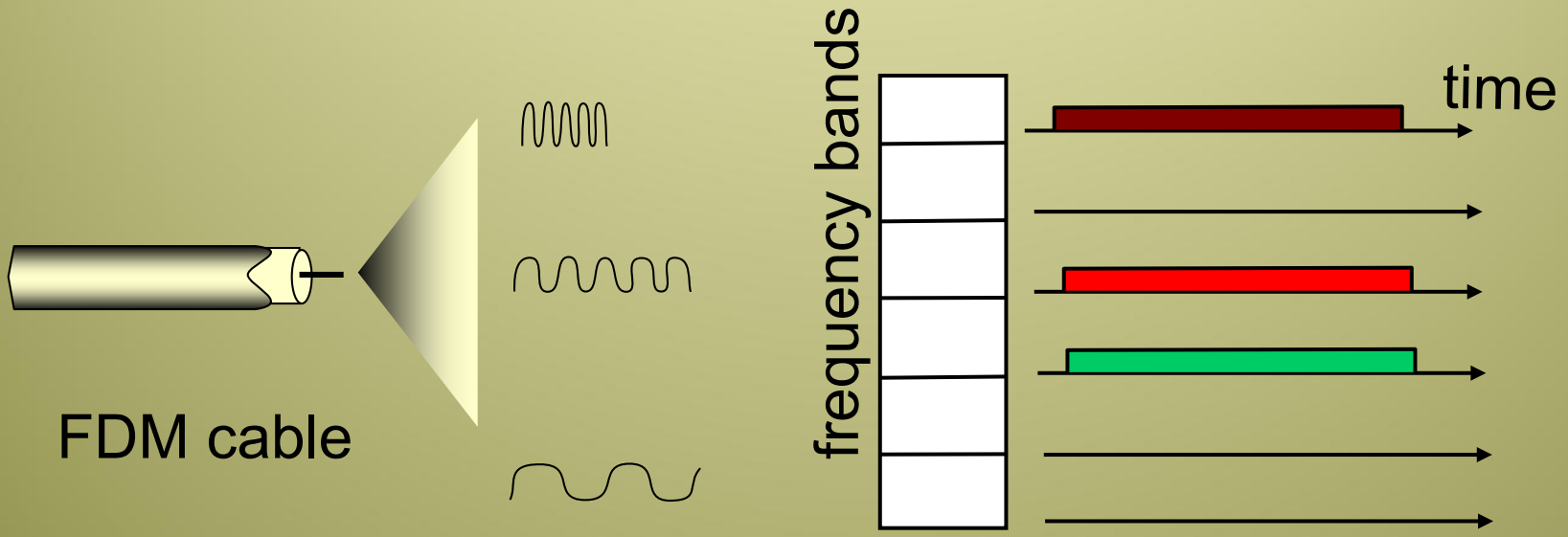
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



Random access protocols

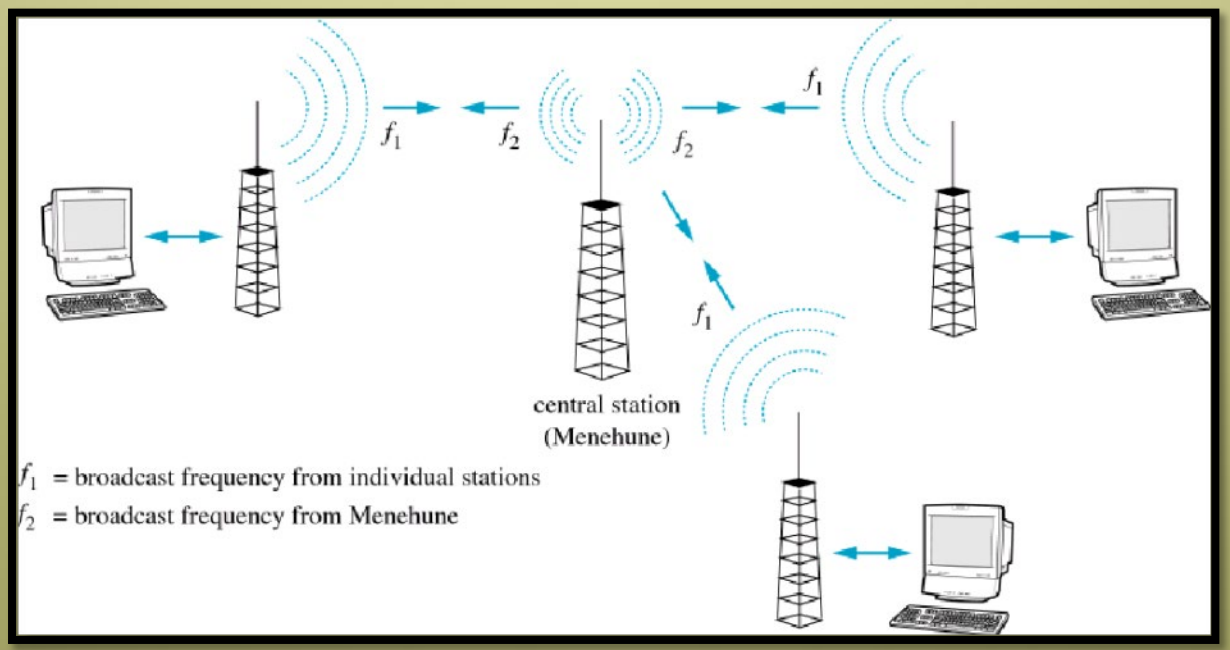
- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "collision"

- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - ALOHA
  - slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA

ALOHA protocols

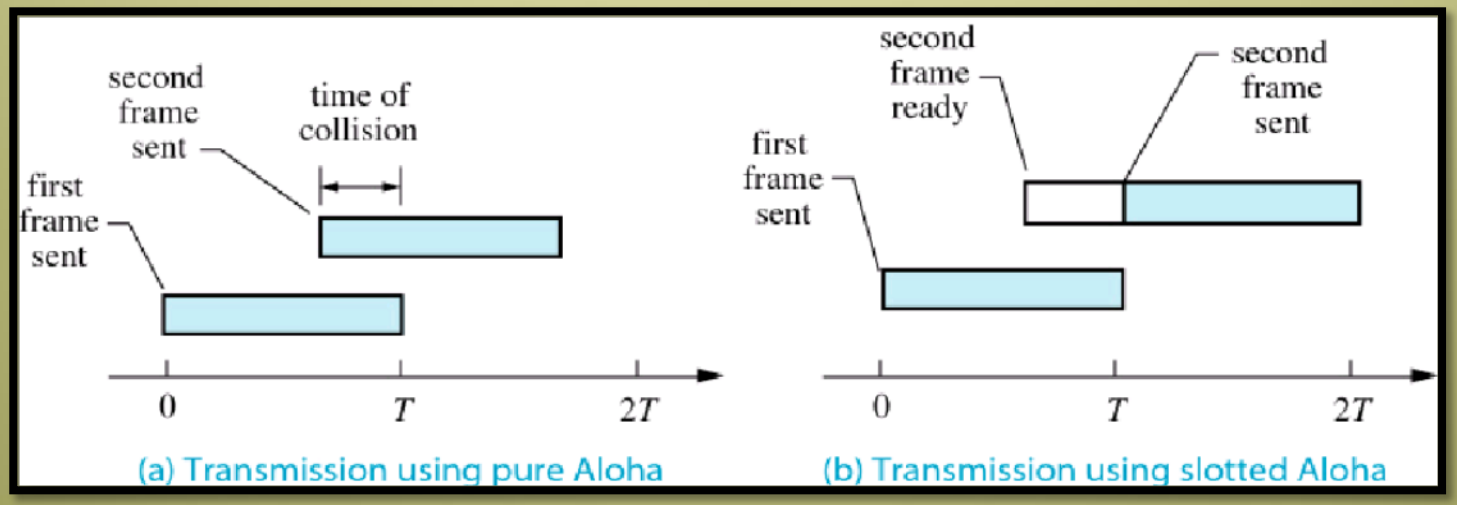
- earliest contention protocol in 1970s by Univ. of Hawaii, called **Pure ALOHA**
- several stations to central station (Menehune) by radio communication
- f_1 for broadcast, f_2 (different frequency than f_1) for ACK
- any station can transmit; if collision then wait random time

Aloha System



Slotted ALOHA

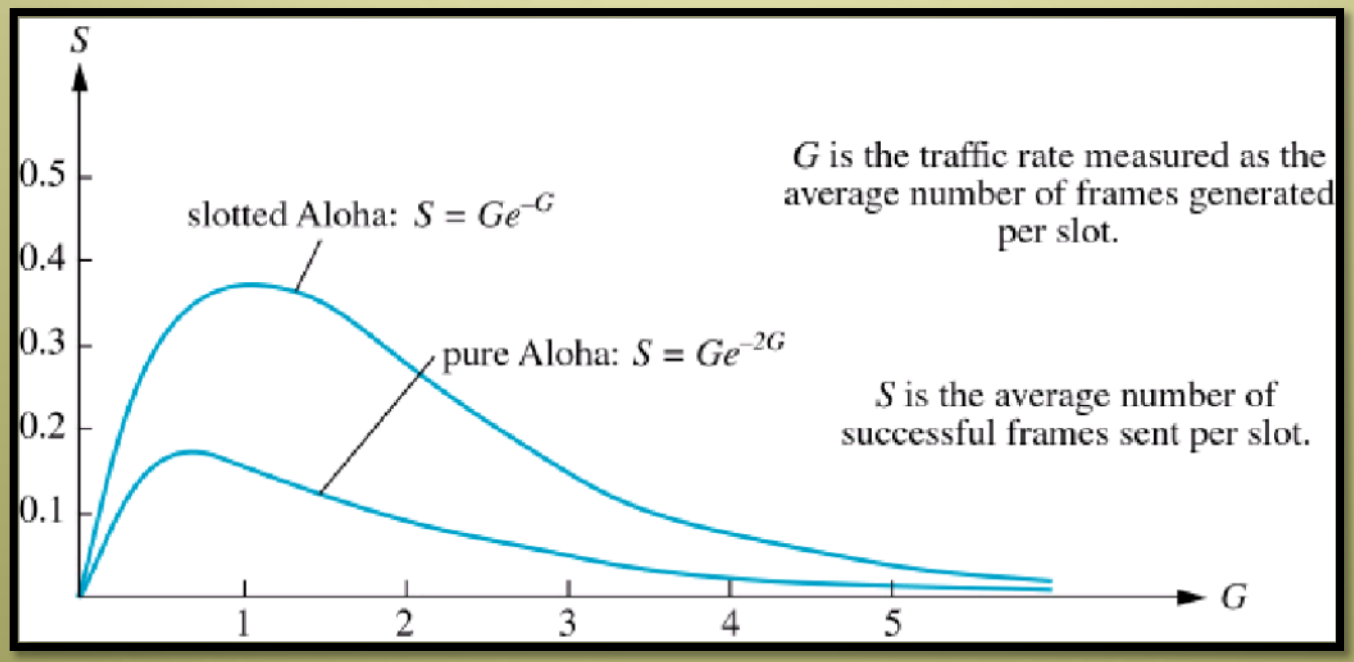
- any overlap in signals, even a small one, would force retransmission
- hence, a minimal safe period to transmit two signals is 2T (T is time period)
- so, to allow a device to transmit, you should reserve 2T for that
- not to waste such time, Slotted Aloha is used
- devices can only send at the beginning of each slot



Transmission Using Pure Aloha & Slotted Aloha

Slotted ALOHA: efficiency

- Slotted Aloha has a higher success rate than Pure Aloha
- however, with increased traffic, the different may not be that significant



Success Rate for Pure Aloha & Slotted Aloha

Carrier Sense Multiple Access (CSMA)

CSMA: listen before transmit:

if channel sensed idle: transmit entire frame

- if channel sensed busy, defer transmission

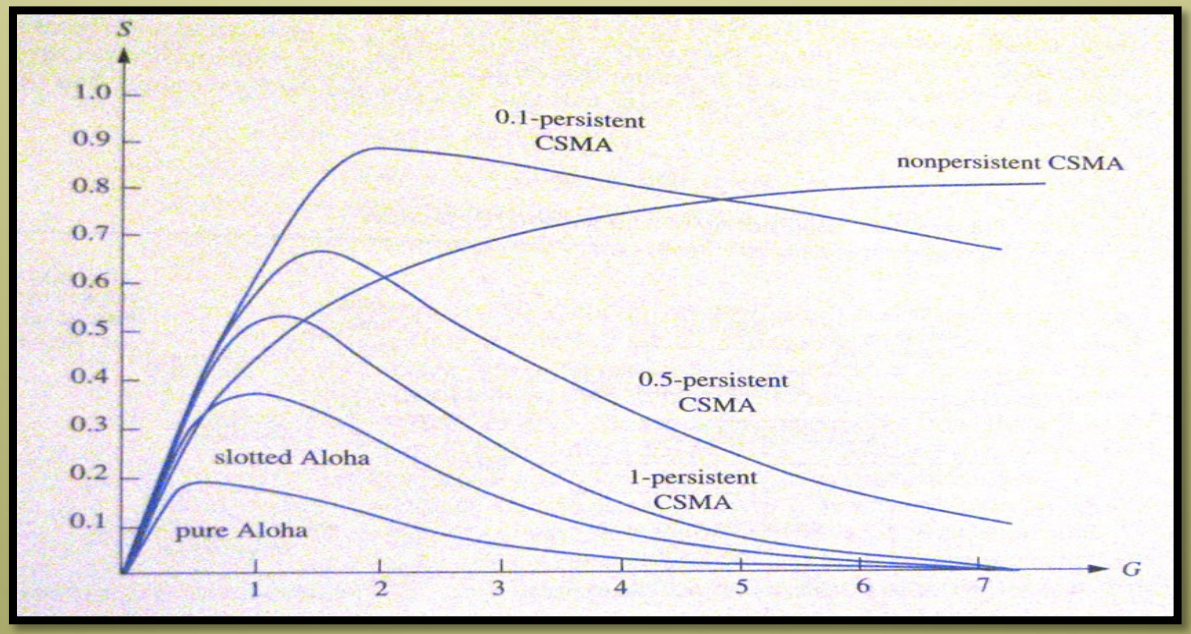
- human analogy: don't interrupt others!

Carrier Sense Multiple Access (CSMA)

Carrier Sense Multiple Access Protocols (CSMA)

- sense the medium at the beginning of a slot, send if the medium is free, else wait for next slot
- ***p-persistent CSMA:***
  - *Continue to sense the active medium*
  - If free, send with a probability p ($0 < p \leq 1$)
    - p=0 never transmits (wait again) ;
    - p=1 always transmits (collision chances are higher)

- ***Nonpersistent CSMA***: check periodically, if free send else wait for one time slot and check again

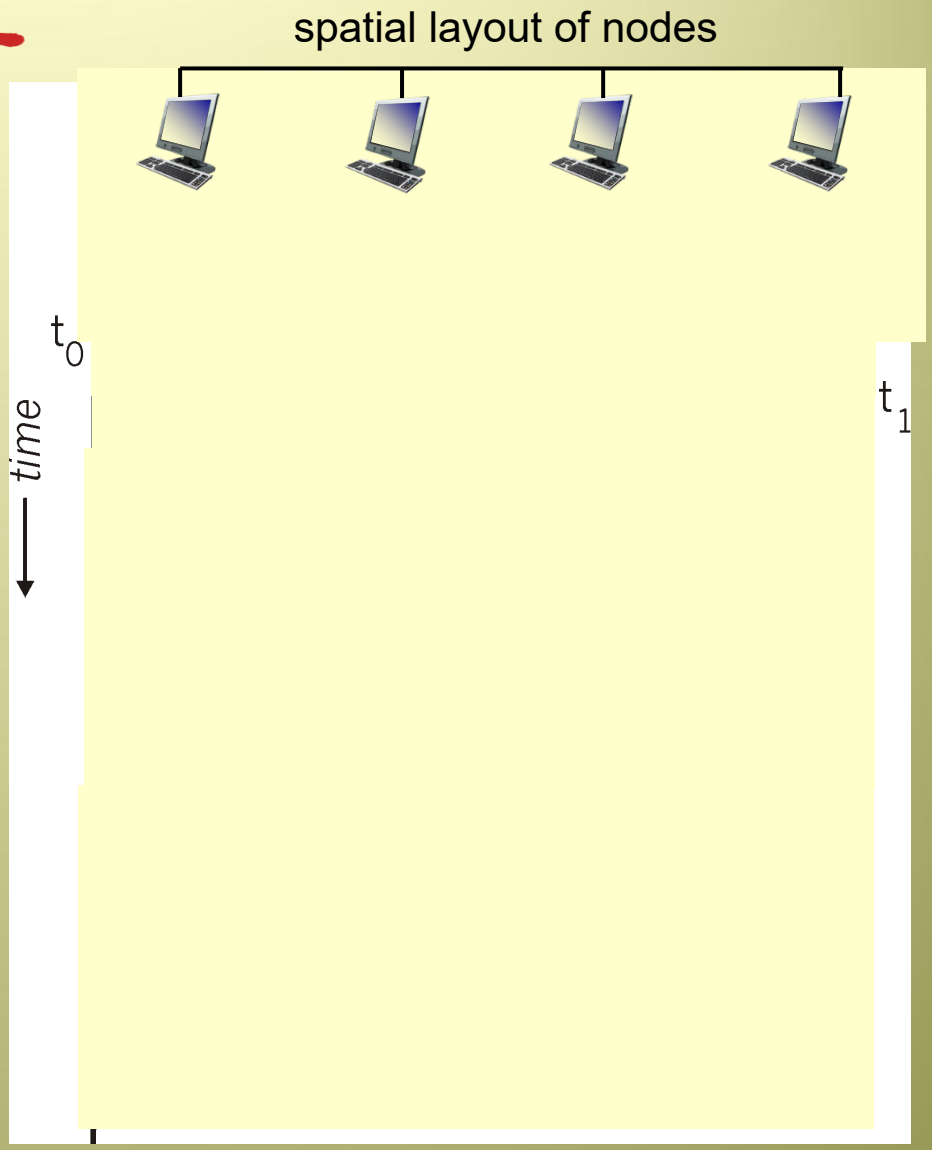


Success Rate for CSMA & Aloha Protocols

CSMA collisions

- collisions *can* still occur: propagation delay means two nodes may not hear each other's transmission

- collision: entire packet transmission time wasted
  - distance & propagation delay play role in in determining collision probability



CSMA/**CD** (**C**ollision **D**etection)

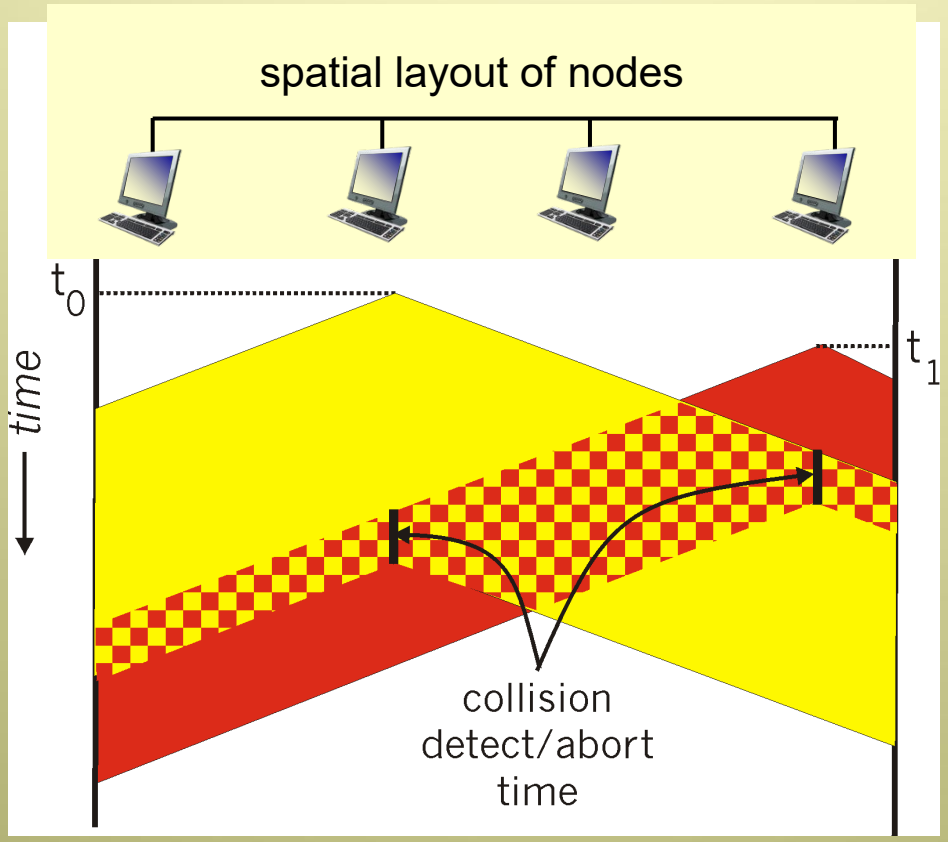
CSMA/CD: carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

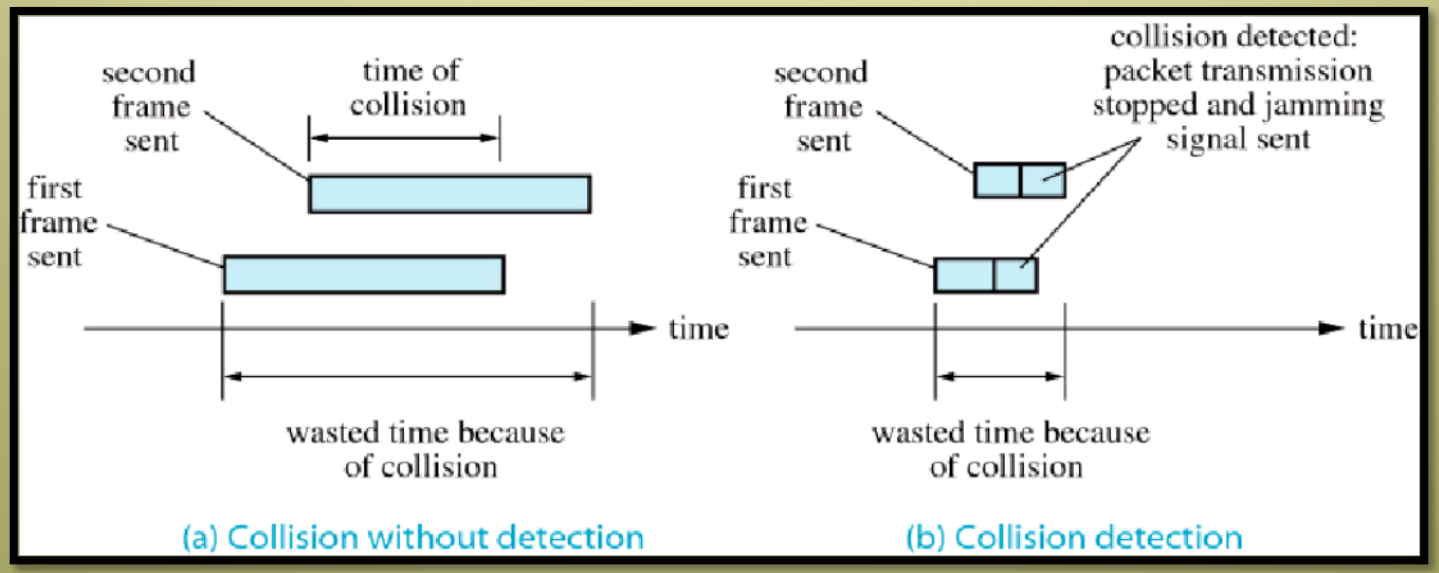
- human analogy: the polite conversationalist

CSMA/CD (collision detection)



Collision detection (CD)

- instead of sending entire frame then discover that collision has occurred when no ack is received, sense the medium for collision and stop transmitting if occurs
- this will avoid the medium from being unusable during collision
- Commonly used with CSMA – ***called CSMA/CD***



Collision with and without Detection

Collision detection (CD)

- two issues worth considering:
  - frame size
  - distance
- frame size:
  - the frame has to be of a minimum size so the device can detect collision before it finishes
  - if too large, a device can monopolize the medium
  - so, how small should a frame be?
    - depends on the maximum time it takes to detect collision

Collision detection (CD)

Example: ***Assume:***

- ***10 Mbps bit rate,***
- ***Largest distance between two devices is 2 KM***
- ***Signal propagate at a rate of 200 meter/µsec***

➔To propagate 2 KM it takes 10 *µsec*

➔To propagate 4 KM (worst case, go & come back), we need 20 *µsec*

➔Rate of 10 Mbps is the same as 10 bits each *µsec*

➔In 20 *µsec* we have 200 bits or 200/8 = 25 bytes

- **This is the minimum size a frame can be so CD can be made**

Collision detection (CD)

- the other issue with CD is *distance*
- for example *CD* does not work well with satellite since the time needed to travel back and forth between ground and satellite is too big due to the large distance

Binary exponential back-off algorithm

- varies the waiting time before sending again if collision occurred
- if first collision then wait 0, or 1 slots
- second collision then wait randomly for 0, 1, 2, or 3 slots
- ……
- ……..
- if *n* successive collisions then wait for random # of slots between 0 and 2^n-1, when n > 16 give-up and signal to higher layer!

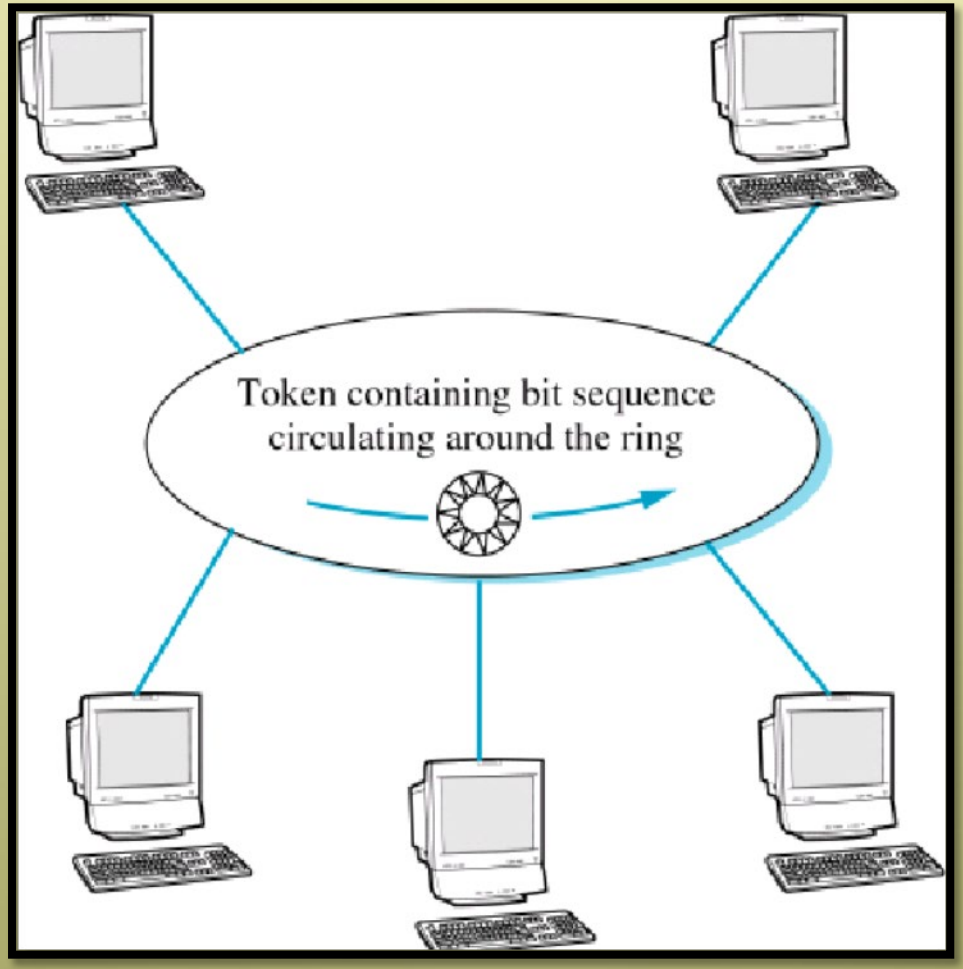
Token passing

- instead of sending whenever it wishes, a device will take turns in sending with the other ones
- capture token to send data frame
- if data then remove token and transmit data frame; else pass token to neighbor
- only sender can put the token back on ring after receiving it back
- one frame per token

- advantage: contention is much controlled than the previous protocols

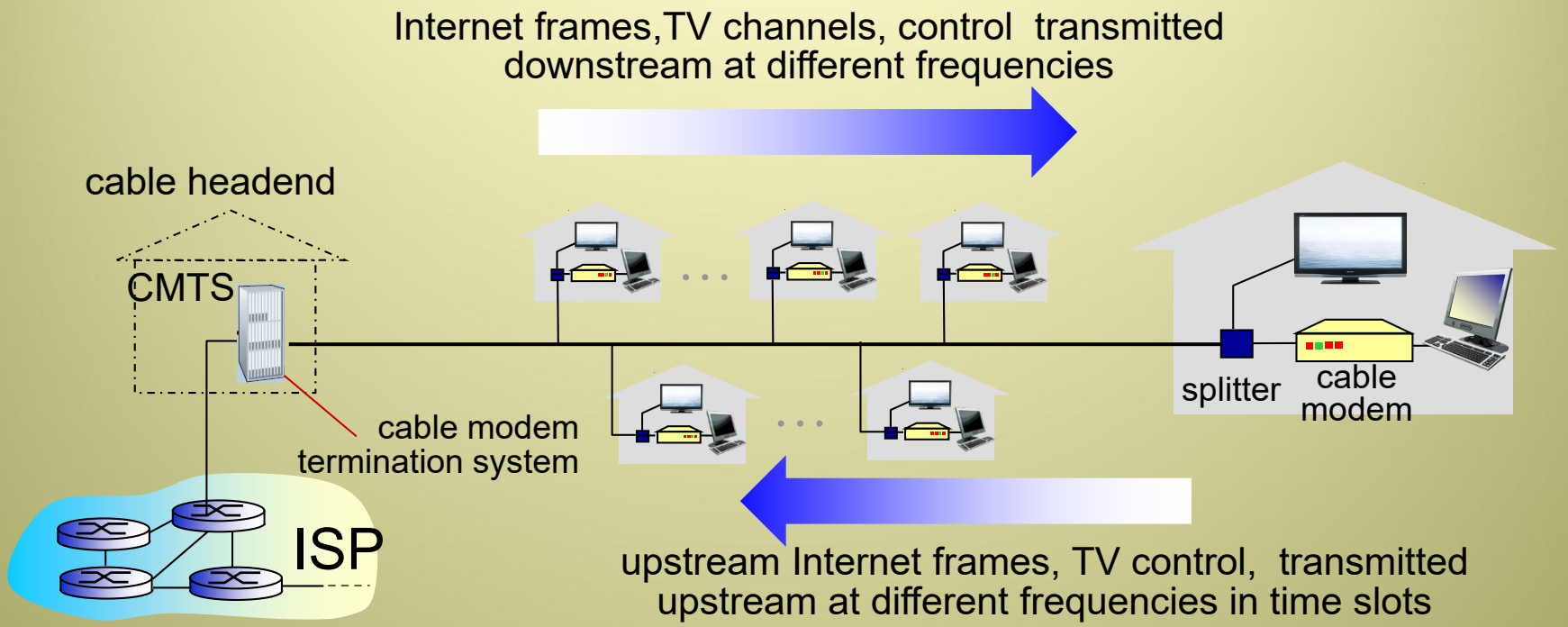
- Disadvantages:
  - All devices must be known
  - Complexity (what happen if the token is lost or if the device that has control over it fails)

Token passing



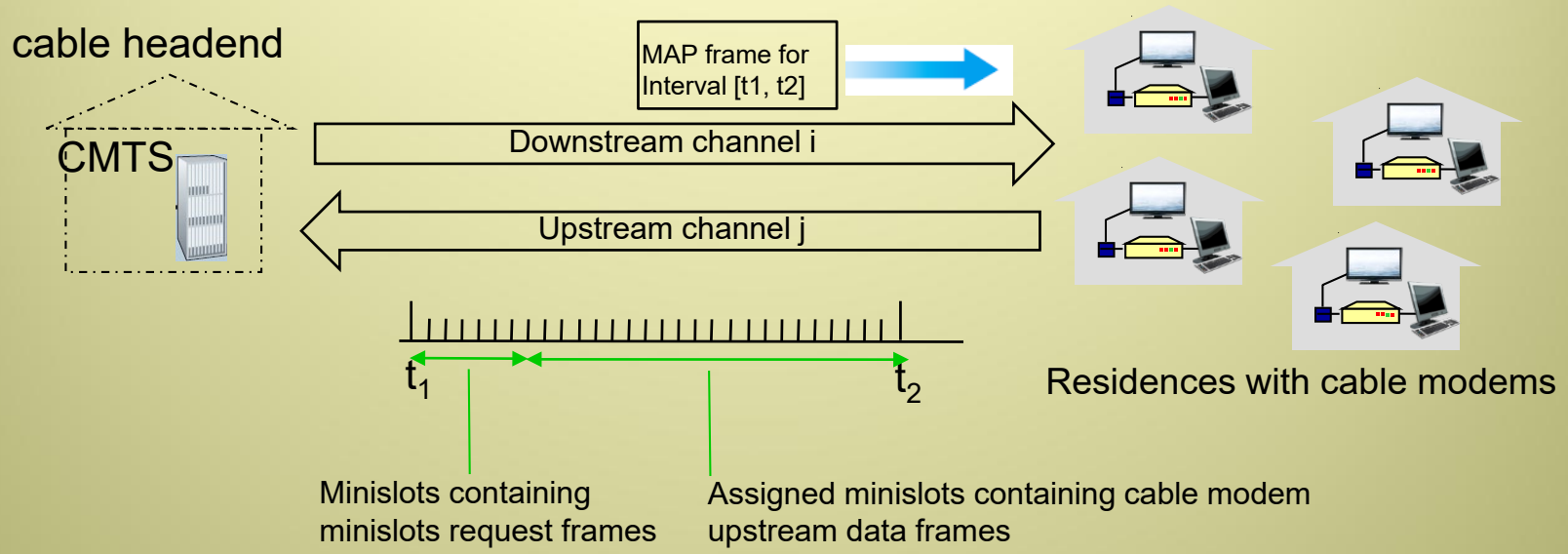
Token Ring Network

Cable access network



- multiple 40Mbps downstream (broadcast) channels
  - single CMTS transmits into channels
- multiple 30 Mbps upstream channels
  - multiple access: all users contend for certain upstream channel time slots (others assigned)

Cable access network



DOCSIS: data over cable service interface spec

- FDM over upstream, downstream frequency channels
- TDM upstream: some slots assigned, some have contention
  - downstream MAP frame: assigns upstream slots
  - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

Error correction

- when errors are detected, either resend the message or correct it
- in many cases correction will be a much better choice
- 1 parity bit check → 2 possibilities
  whether or not an error occurred; but not where
- 2 parity bit checks → 4 possibilities of failures and successes
- 3 parity bit checks → 8 possibilities of failures and successes
- ***n*** parity bit checks → 2^n possibilities of failures and successes

Error correction

- how many *single-error* possibilities are there in the a 3-bit string, for example m_1 m_2 m_3?
- 1 parity check → whether or not an error occurred, but not where – Not enough to correct
- 2 parity checks → 4 possibilities
  - Example: Assume **1 0 1** is sent and there are P1 & P2 parity checks
  - P1 is even parity for m_1 & m_2 ; P2 is even parity for m_1 & m_3
  - P1 is hence set to 1, P2 is set to 0. Assume also that parities are delivered successfully!
  - Possible deliveries with a maximum of a single-bit error are:
    - 0 0 1 → P1 fails, P2 fails
    - 1 1 1 → P1 fails, P2 succeed
    - 1 0 0 → P1 succeed, P2 fails
    - 1 0 1 → P1 succeed, P2 succeed ➔ No error in transmission
    - *As the receiver, can you correct the error?*

Error correction – Hamming Codes

Hamming Codes

- Hamming code requires the insertion of few parity bits in the bit stream before sending it
- the parity bits check the parity in strategic locations
- if bits are altered, the receiver will examine the combinations of failures to discover where the errors are

Error correction – Hamming Codes

Hamming Codes – *Single-bit Error Correction*

- how many *single-error* possibilities are there in an 8-bit string, for example $m_1\ m_2\ m_3\ m_4\ m_5\ m_6\ m_7\ m_8$?
- 3 parity checks *would* be enough to discover a single error in one of 8 positions, however
- errors may as also occur in the parity bits
- 4 parity checks are needed → $2^4 > 12$

Error correction – Hamming Codes

Hamming Codes – *Single-bit Error Correction*

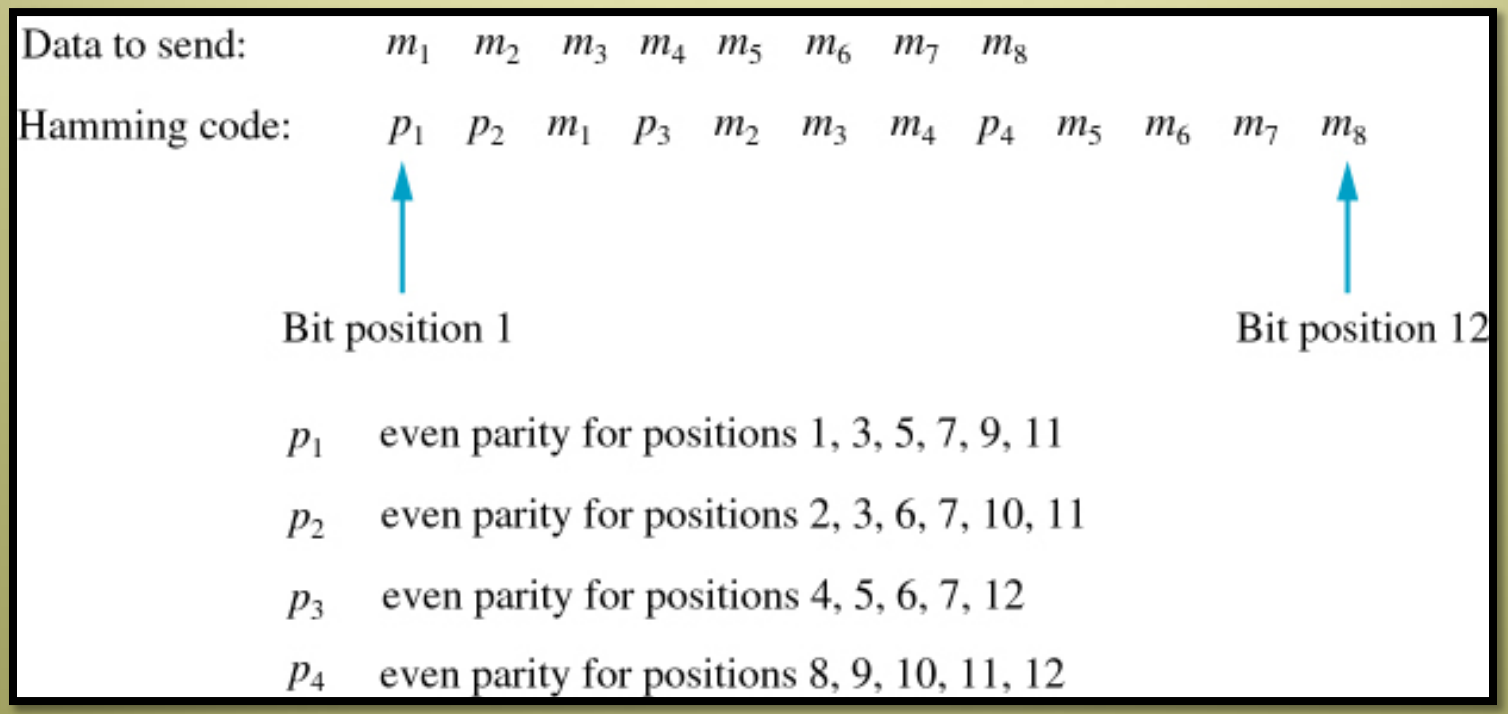
- in general, must have *n* parity checks, where 2^n is larger than the bits sent

n (parity checks)	Bits sent	Success/fail possibilities
1	9	2
2	10	4
3	11	8
4	12	16 (enough to cover the 13 possible events – no error or single-bit error in one of 12 positions)

Error correction – Hamming Codes

Hamming Codes – *Single-bit Error Correction*

- Which positions will parity checks cover? Where are they inserted?



Hamming Code for Single-bit *Errors*

Error correction – Hamming Codes

Hamming Codes – *Single-bit Error Correction*

- assume 4-bit binary number, $b_1\ b_2\ b_3\ b_4$, where $b_i = 0$ if the parity p_i succeed, and 1 otherwise

Error Position	Binary b_4, b_3, b_2 & b_1	Parity P_1 / Position	Parity P_2 / Position	Parity P_3 / Position	Parity P_4 / Position
0 (No Error)	0000				
1	0001	1			
2	0010		2		
3	0011	3	3		
4	0100			4	
5	0101	5		5	
6	0110		6	6	
7	0111	7	7	7	
8	1000				8
9	1001	9			9
10	1010		10		10
11	1011	11	11		11
12	1100			12	12

Error correction – Hamming Codes

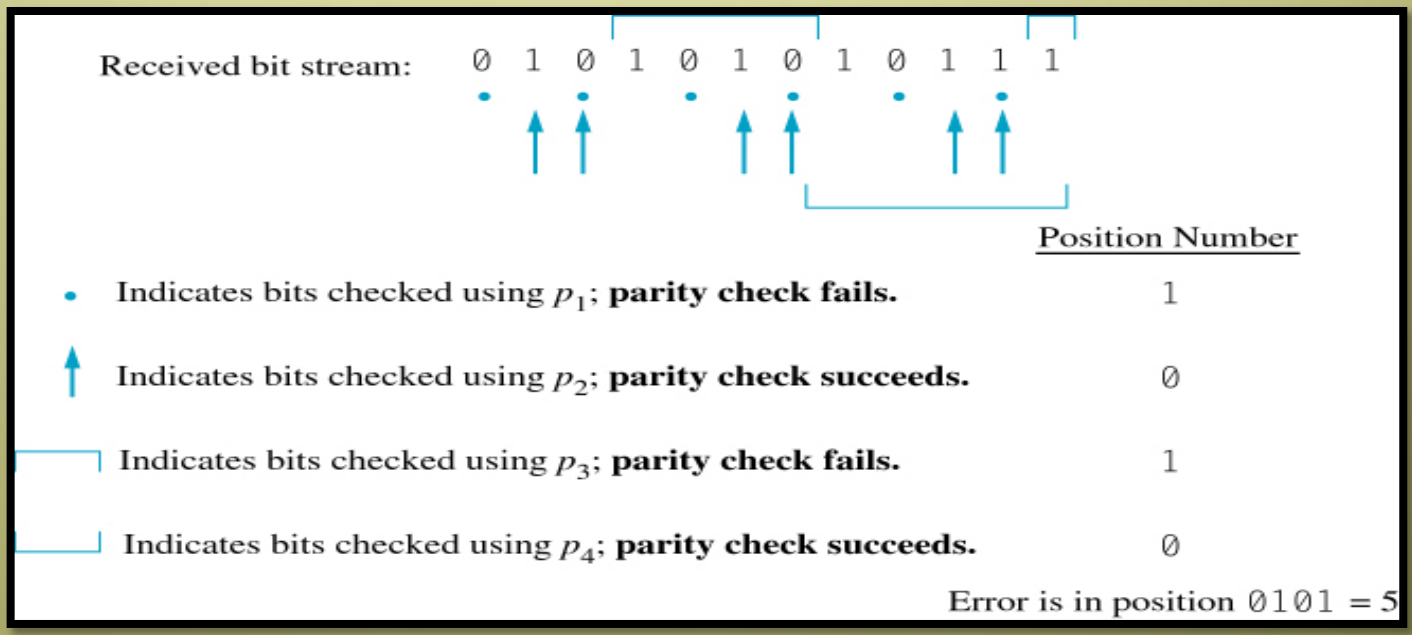
Hamming Codes – *Single-bit Error Correction*

Example:

Bit Stream Before Transmission

Data:	0	1	1	0	0	1	1	1				
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8				
Hamming code:	0	1	0	1	1	1	0	1	0	1	1	1
	p_1	p_2	m_1	p_3	m_2	m_3	m_4	p_4	m_5	m_6	m_7	m_8

Parity Checks of Frame After Transmission



Error correction – Hamming Codes

Hamming Codes – *Multiple-bit Error Correction*

- a more general Hamming code for multiple-bit error do exist
- the number of extra parity bits becomes much larger in this case
- when there are many possible codes, a reduction is needed in order to analyze them quickly
- code Correlation can be used to achieve such reduction
- when you have two, or more, Hamming codes, a Hamming distance is defined as the minimum number of bit changes that can lead one code to be detected as another
- a Radius is defined as Hamming Distance / 2

Error correction – Hamming Codes

Hamming Codes – *Multiple-bit Error Correction*

- Example:
  Note: This topic is a major subject of higher-level/graduate courses. It is **not** a part of this course and you will **not** be examined on it. This example is just to give an extra knowledge on the subject.

Code 1	Code 2	Code 3
1	0	1
0	1	1
0	1	0
1	0	0

- the codes have a minimum Hamming Distance of 2, or a Radius of 1
- a code such as 0100 (which is not among the exact valid codes) can be coded as either code 3 with a missing first bit, or as code 2 with a missing third bit
- a change of 2 bits may lead to two codes to be identical; or in other words decoded incorrectly (for example if code 3 changed to 0110, it will be decoded incorrectly as code 2)