

# Chapter 5

## Link Layer and LANs

### A note on the use of these ppt slides:

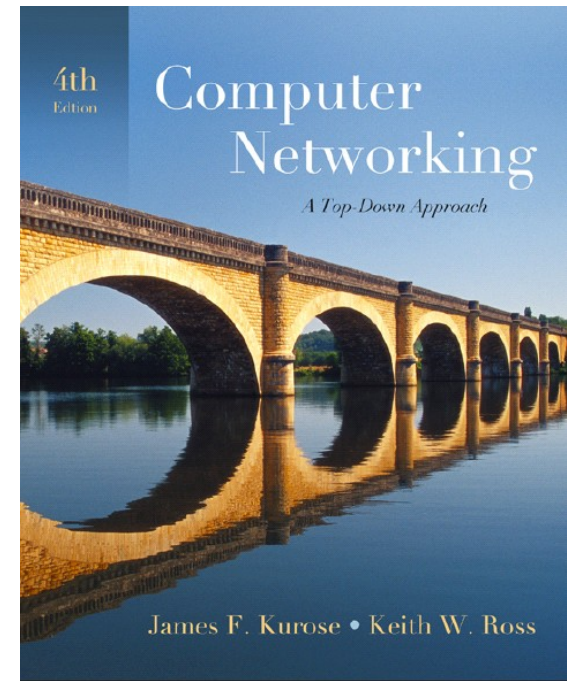
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- ❑ If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- ❑ If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2007

J.F Kurose and K.W. Ross, All Rights Reserved



*Computer  
Networking: A Top  
Down Approach*  
4<sup>th</sup> edition.  
Jim Kurose, Keith  
Ross  
Addison-Wesley, July  
2007.

# Chapter 5: The Data Link Layer

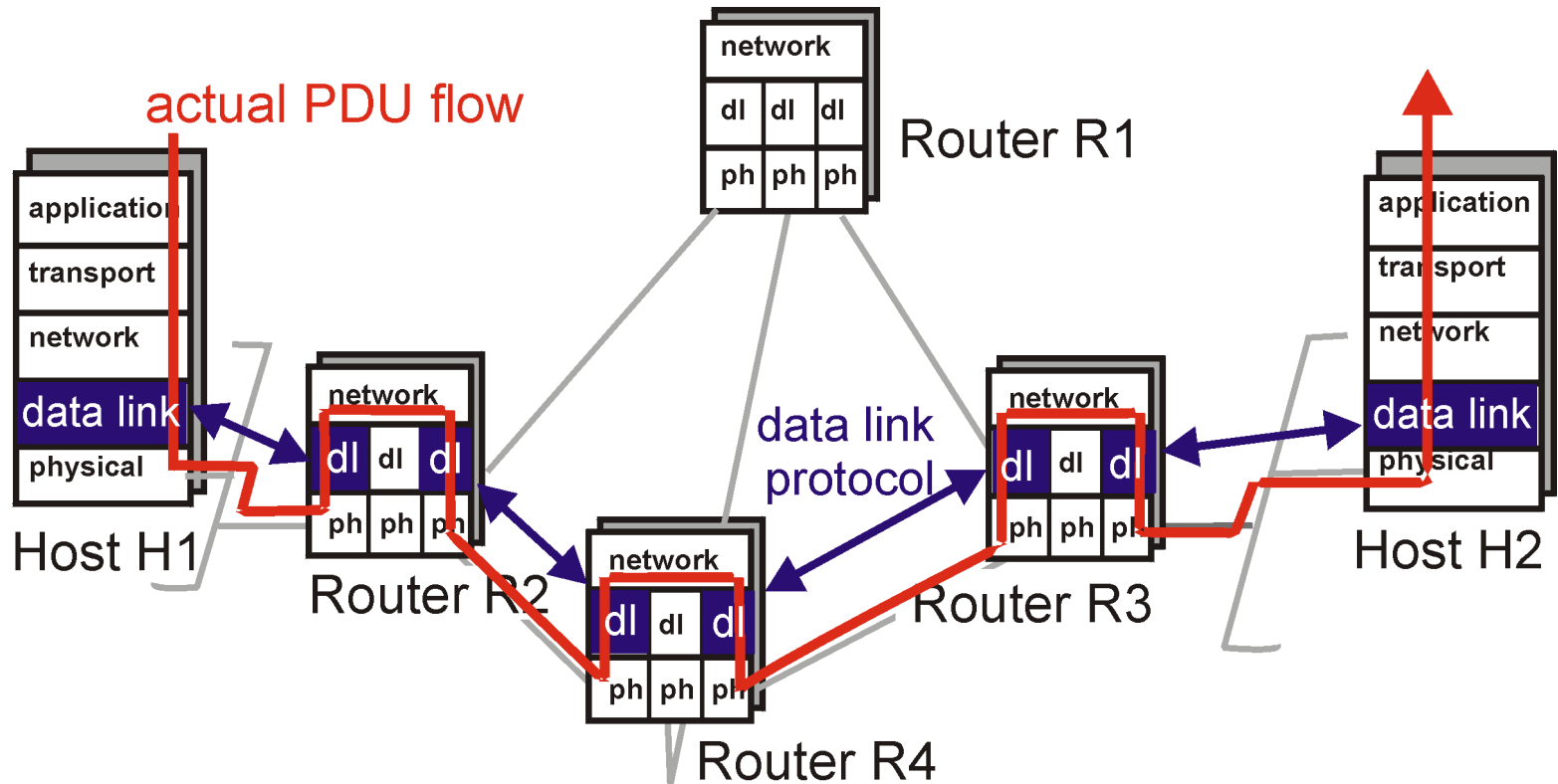
## Our goals:

- understand principles behind data link layer services:
  - error detection, correction
    - sharing a broadcast channel: multiple access
    - link layer addressing
    - reliable data transfer, flow control: *done!*
- instantiation and implementation of various link layer technologies

## Overview:

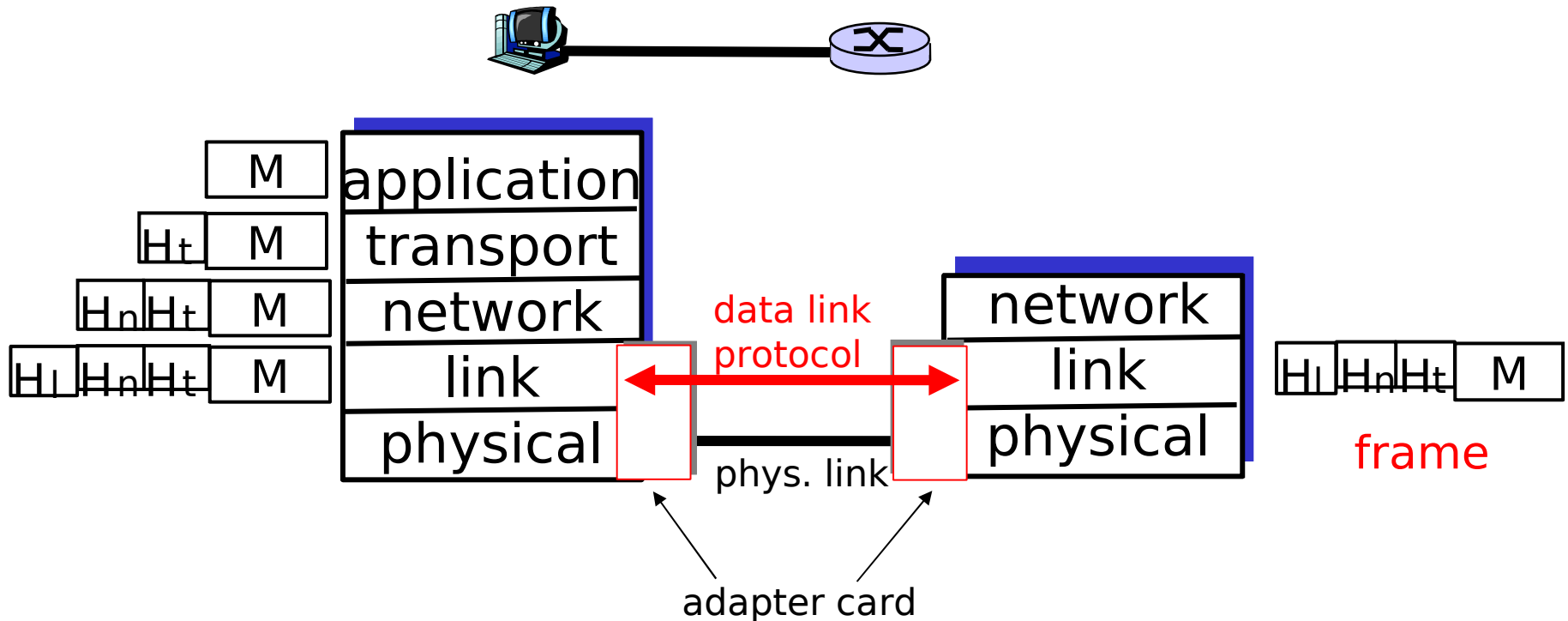
- link layer services
- error detection, correction
- multiple access protocols and LANs
- link layer addressing, ARP
- specific link layer technologies:
  - Ethernet
  - hubs, bridges, switches
  - IEEE 802.11 LANs
  - PPP

# Link Layer: setting the context



# Link Layer: setting the context

- two *physically connected* devices:
  - host-router, router-router, host-host
- unit of data: *frame*



# Link Layer Services

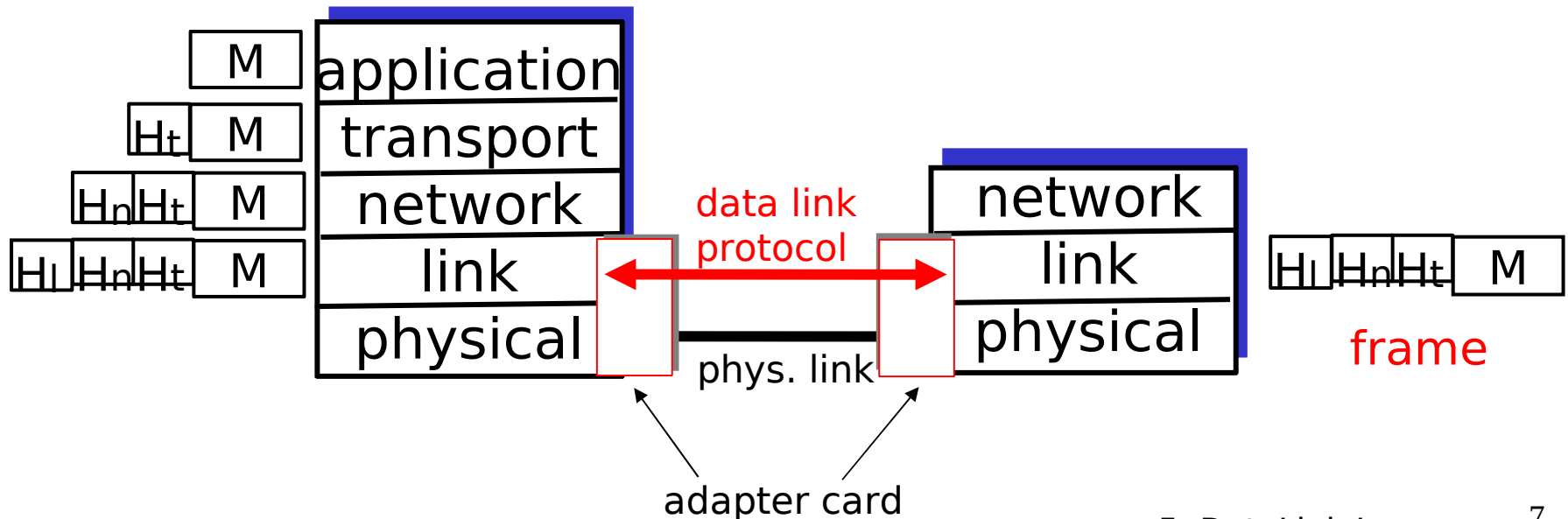
- **Framing, link access:**
  - encapsulate datagram into frame, adding header, trailer
  - implement channel access if shared medium,
  - ‘physical addresses’ used in frame headers to identify **source, destination**
    - different from IP address!
- **Reliable delivery between two physically connected devices:**
  - 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 sender and receivers
- **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

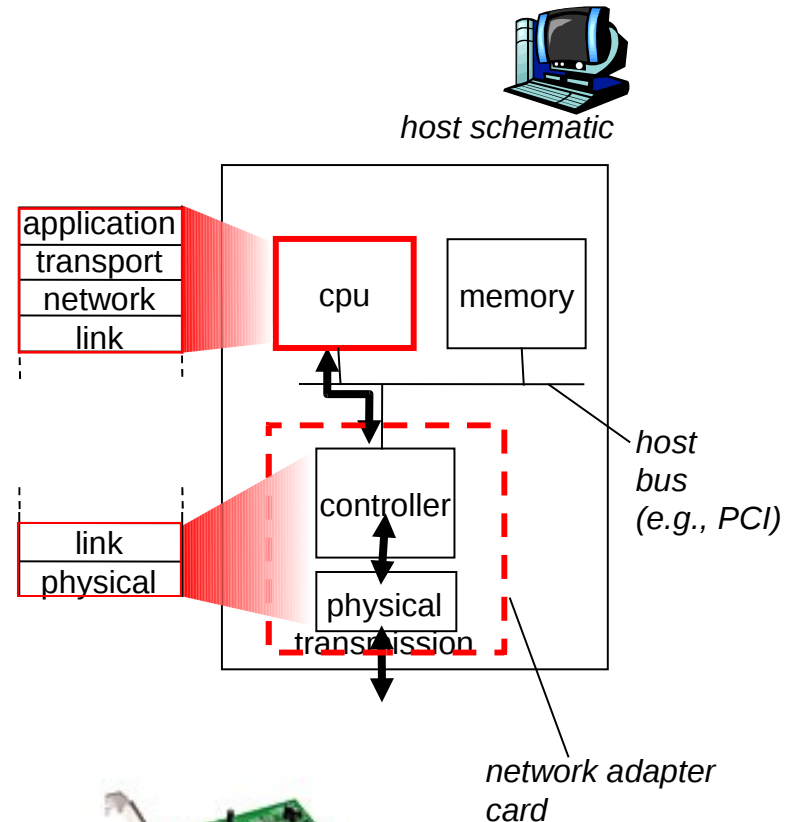
# Link Layer: Implementation

- implemented in adapter
  - e.g., PCMCIA card, Ethernet card
  - typically includes: RAM, DSP chips, host bus interface, and link interface

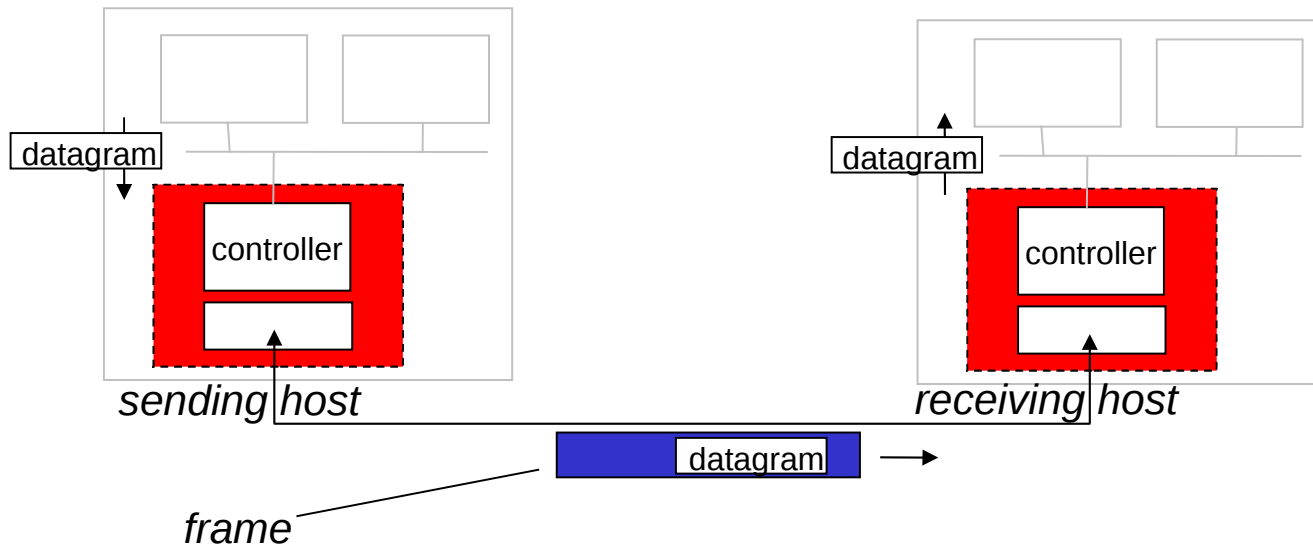


# Where is the link layer implemented?

- in each and every host
- link layer implemented in “adaptor” (aka *network interface card* NIC)
  - Ethernet card, PCMCIA card, 802.11 card
  - 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.

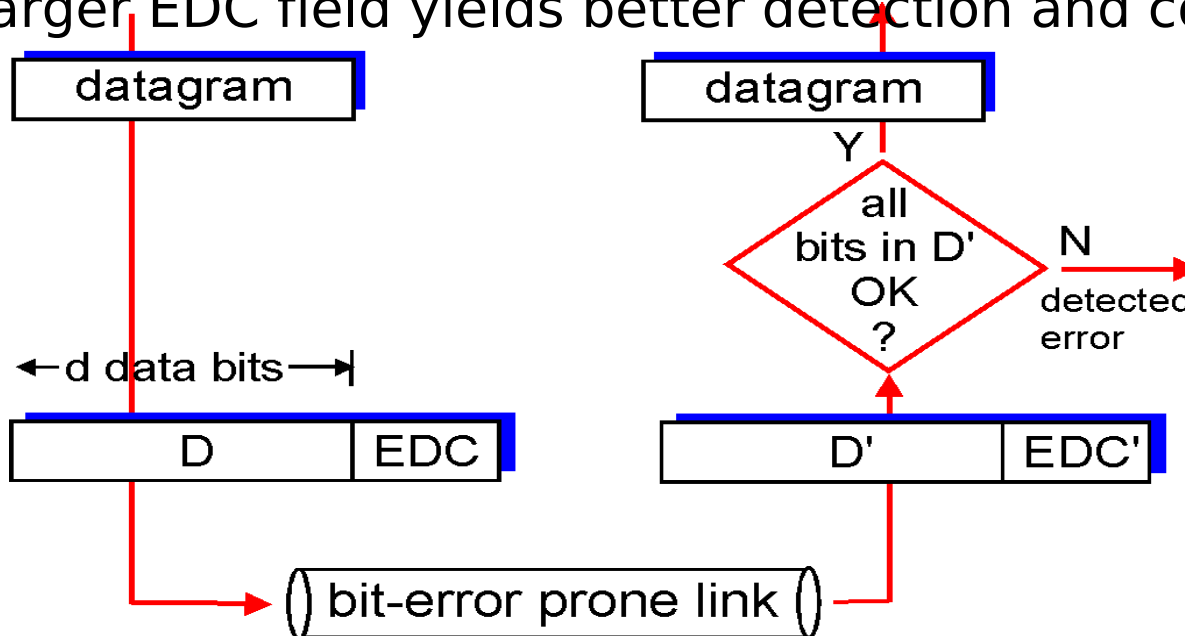
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

# Error Detection

EDC= Error Detection and Correction bits (redundancy)

D = Data protected by error checking, may include header fields

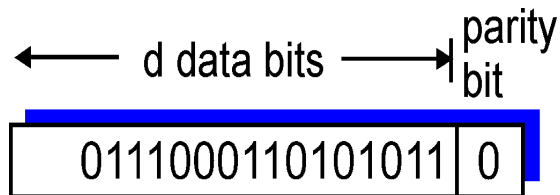
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



# Parity Checking

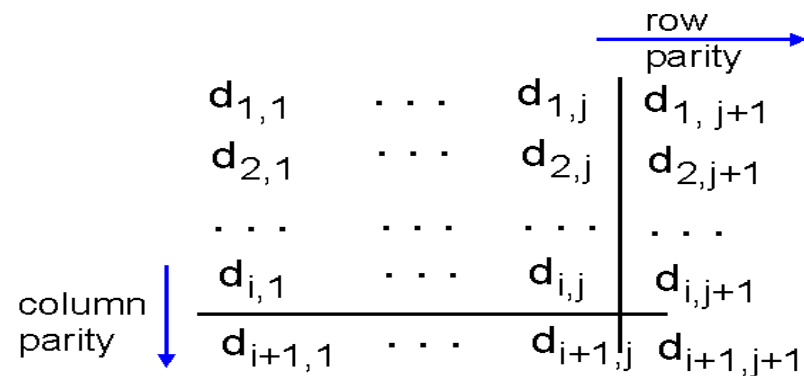
## Single Bit Parity:

**Detect single bit errors**



## Two Dimensional Bit Parity:

**Detect *and correct* single bit errors**



1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

*no errors*

1	0	1	0	1	1
1	⊖	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

↓ parity error  
→ parity error  
*correctable  
single bit error*

# Internet checksum

Goal: detect errors (e.g., flipped bits) in transmitted segment (note: used at transport layer *only*)

## Sender:

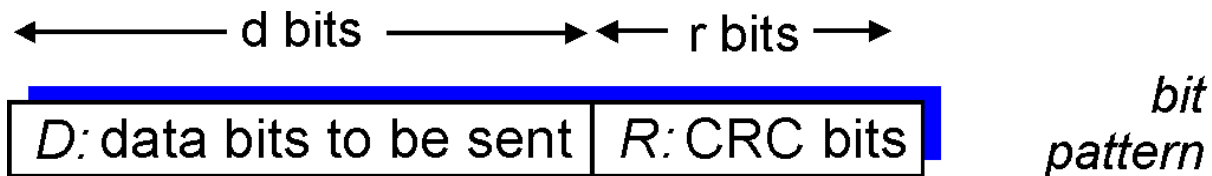
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

## 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? More later*
- ....

# Check summing: Cyclic Redundancy Check

- view data bits, **D**, as a binary number
- choose  $r+1$  bit pattern (generator), **G**
- goal: choose  $r$  CRC bits, **R**, such that
  - $\langle D, R \rangle$  exactly divisible by  $G$  (modulo 2)
  - receiver knows  $G$ , divides  $\langle D, R \rangle$  by  $G$ . If non-zero remainder: error detected!
  - can detect all burst errors less than  $r+1$  bits
- widely used in practice (ATM, HDCL)



$$D * 2^r \text{ XOR } R$$

*mathematical formula*





# Multiple Access protocols

- single shared communication channel
- two or more simultaneous transmissions by nodes: interference
  - only one node can send **successfully** at a time
- *multiple access protocol:*
  - distributed algorithm that determines how stations share channel, i.e., determine when station can transmit
  - communication about channel sharing must use channel itself!
  - what to look for in multiple access protocols:
    - synchronous or asynchronous
    - information needed about other stations
    - robustness (e.g., to channel errors)
    - performance

# MAC Protocols: a taxonomy

Three broad classes:

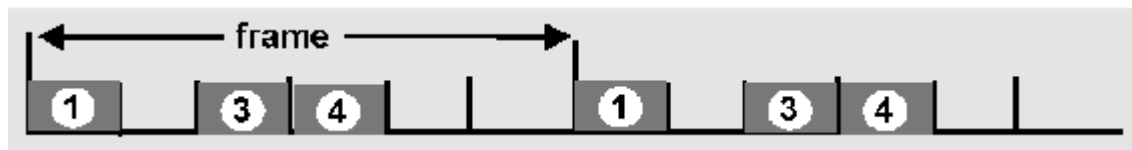
- **Channel Partitioning**
  - divide channel into smaller pieces (time slots, frequency)
  - allocate piece to node for exclusive use
- **Random Access**
  - allow collisions
  - recover from collisions
- **Taking turns**
  - tightly coordinate shared access to avoid collisions

**Goal:** efficient, fair, simple, decentralized

# Channel Partitioning MAC protocols: TDMA

## TDMA: time division multiple access

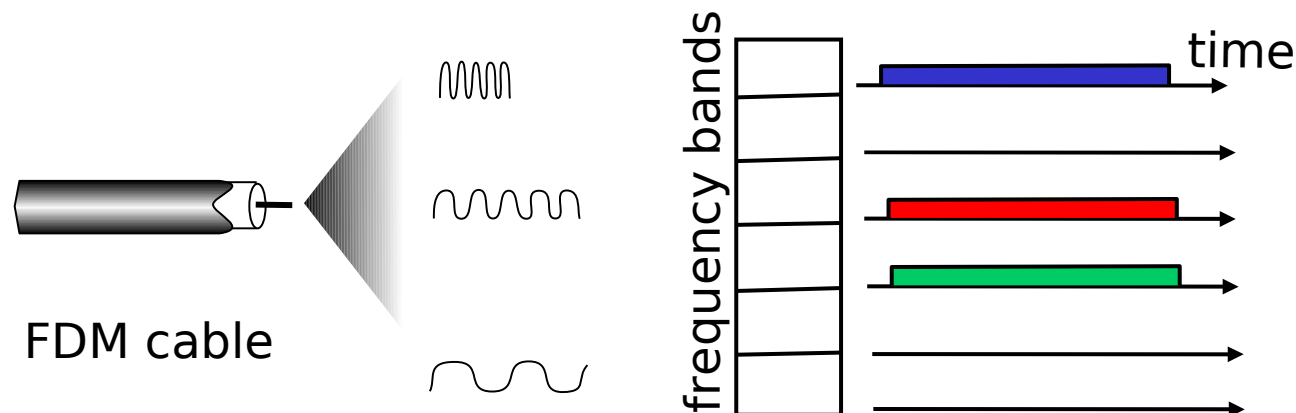
- access to channel in rounds
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



# 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

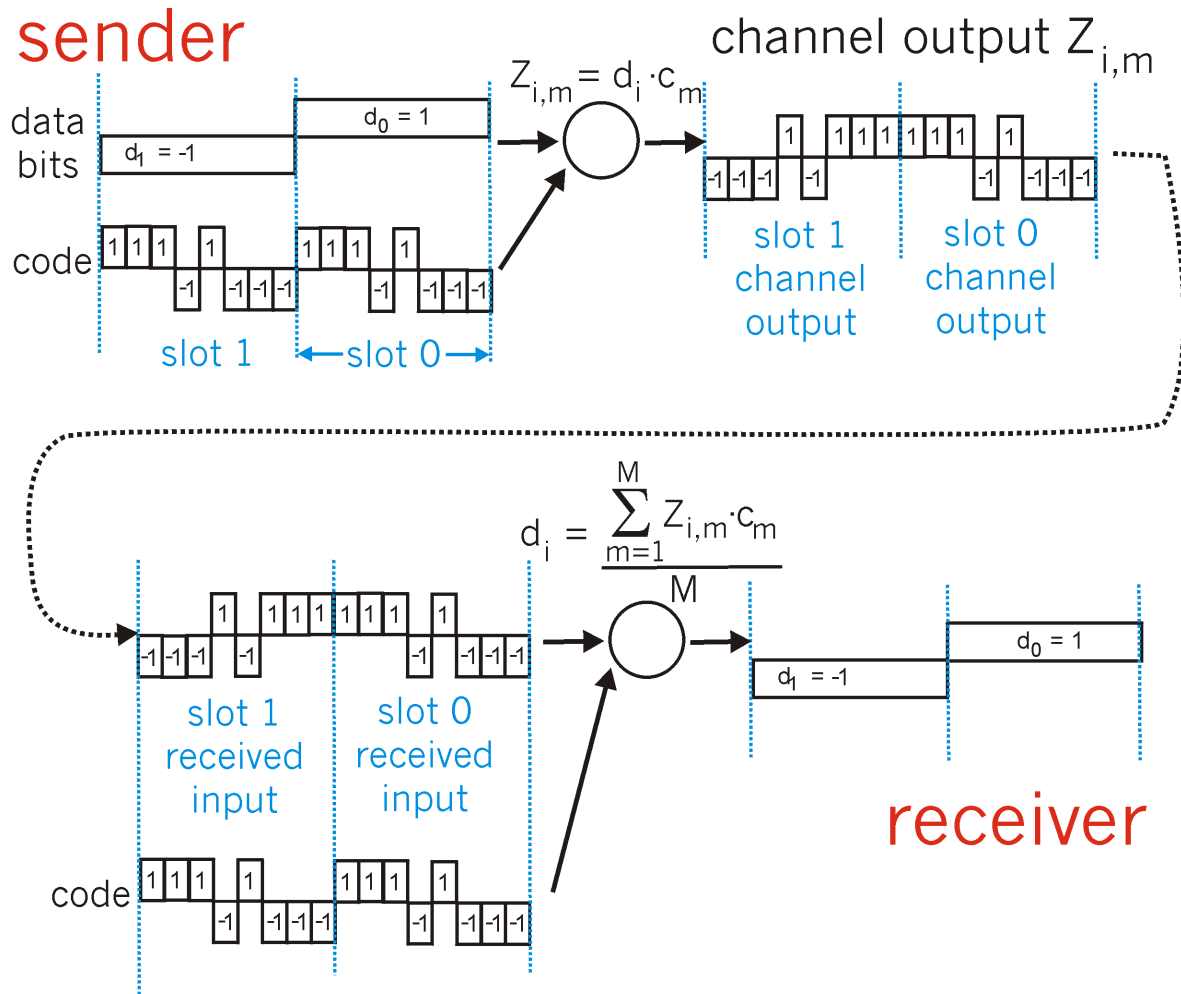


# Channel Partitioning (CDMA)

## CDMA (Code Division Multiple Access)

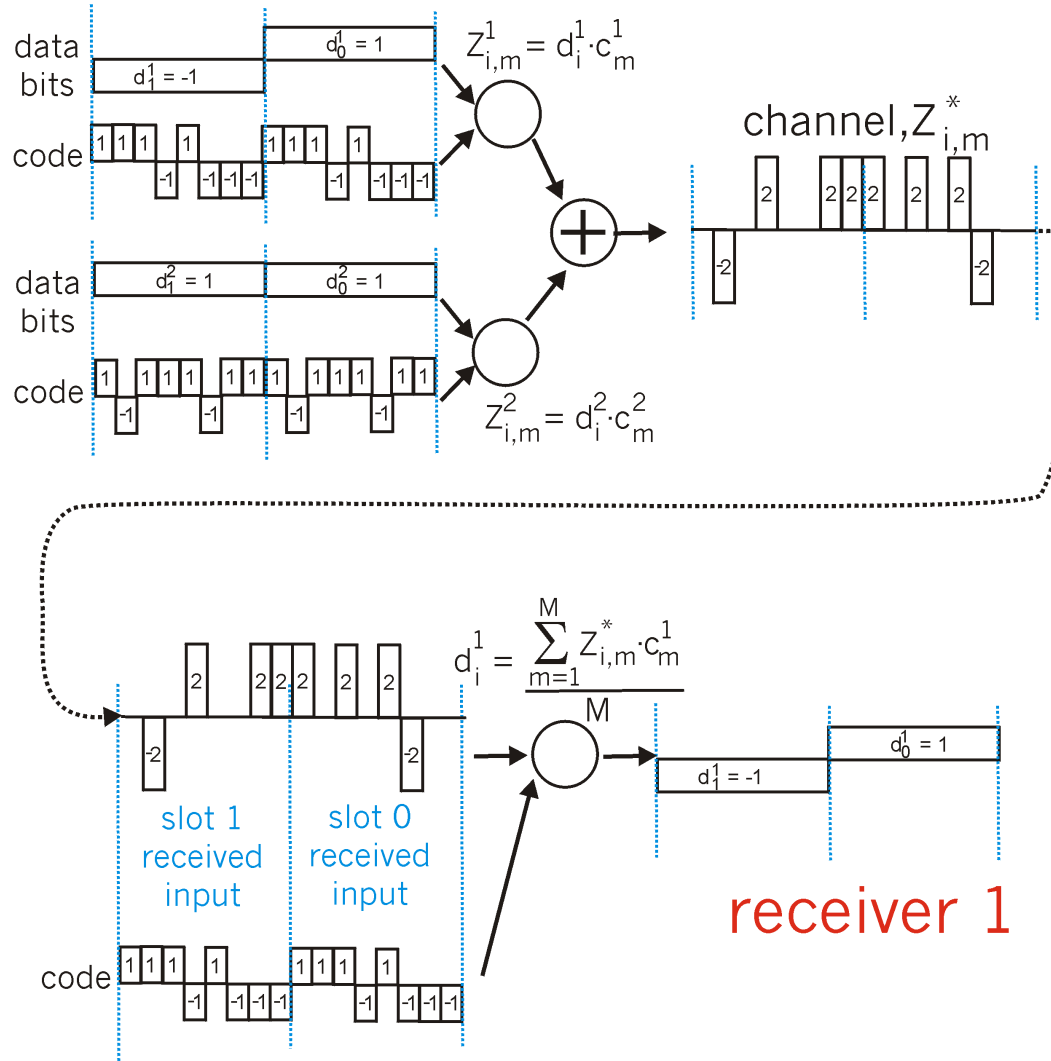
- unique code assigned to each user; i.e., code set partitioning
  - used mostly in wireless broadcast channels (cellular, satellite, etc)
  - all users share same frequency, but each user has own "chipping" sequence (ie, code) to encode data
  - *encoded signal* = (original data) X (chipping sequence)
  - *decoding*: inner-product of encoded signal and chipping sequence
  - allows multiple users to coexist and transmit simultaneously with minimal interference (if codes are orthogonal)

# CDMA Encode/Decode



# CDMA: two-sender interference

senders



# 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:
    - slotted ALOHA
    - ALOHA
    - CSMA, CSMA/CD, CSMA/CA

# Slotted ALOHA

## Assumptions:

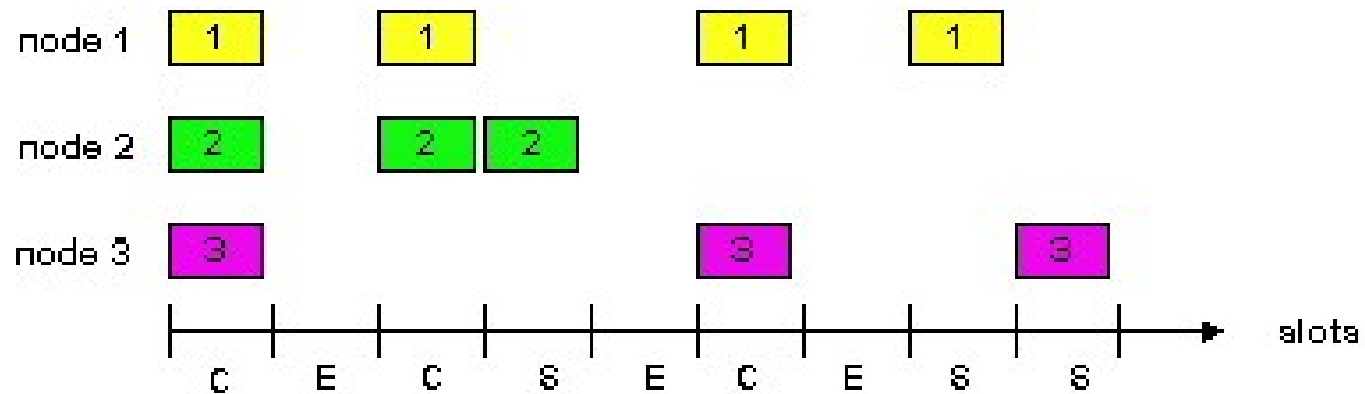
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

## Operation:

- when node obtains fresh frame, transmits in next slot
  - ▣ *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with prob.  $p$  until success

# Slotted Aloha

□ e.g.



Success (S), Collision (C), Empty (E) slots

# Slotted Aloha efficiency

Q: what is max fraction slots successful?

A: Suppose  $N$  stations have packets to send

- each transmits in slot with probability  $p$
- prob. successful transmission  $S$  is:

by single node:  $S = p (1-p)^{(N-1)}$

by any of  $N$  nodes

$S = \text{Prob (only one transmits)}$

$$= N p (1-p)^{(N-1)}$$

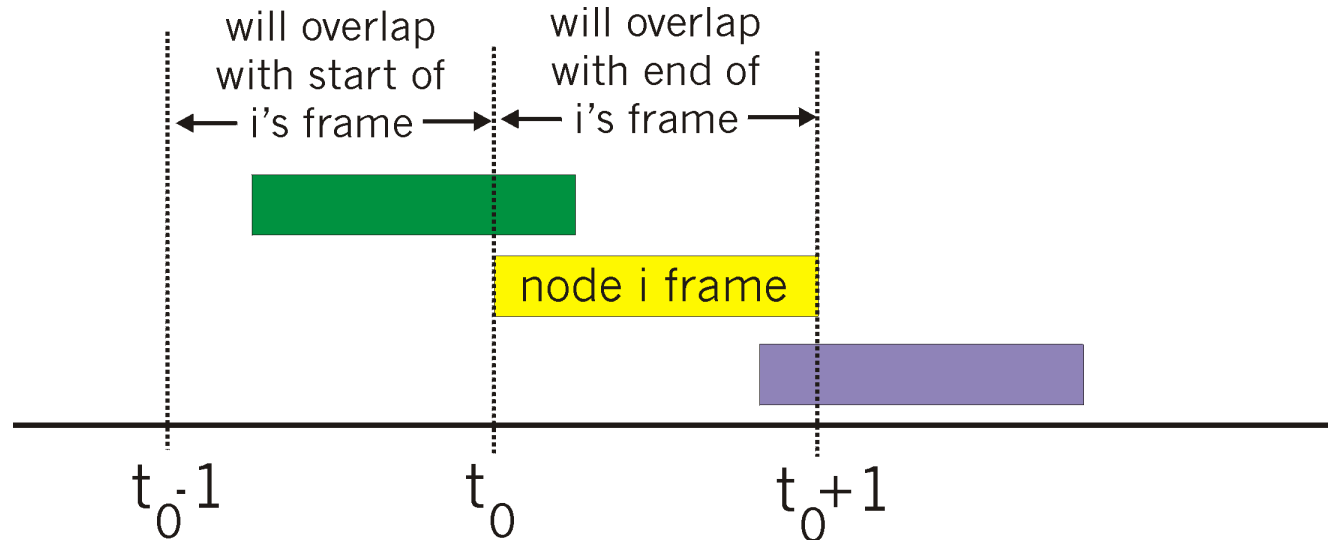
... choosing optimum  $p$  as  $n \rightarrow \infty$  ...

$$= 1/e = .37 \text{ as } N \rightarrow \infty$$

*At best:* channel  
use for useful  
transmissions 37%  
of time!

# Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- pkt needs transmission:
  - send without awaiting for beginning of slot
- collision probability increases:
  - pkt sent at  $t_0$  collide with other pkts sent in  $[t_0-1, t_0+1]$



# Pure Aloha (cont.)

$P(\text{success by given node}) = P(\text{node transmits}) .$

$P(\text{no other node transmits in } [p_0-1, p_0] .$

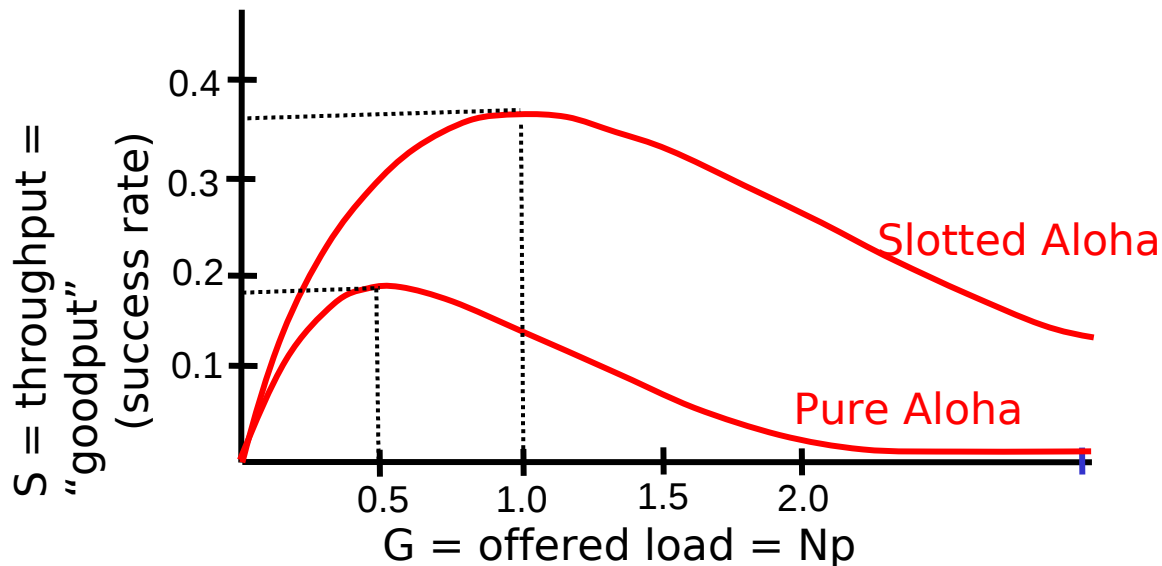
$P(\text{no other node transmits in } [p_0-1, p_0]$

$$= p . (1-p)^{N-1} . (1-p)^{N-1}$$

$P(\text{success by any of } N \text{ nodes}) = p . (1-p)^{2(N-1)}$

... choosing optimum  $p$  as  $n \rightarrow \infty$  ...

$$= 1/(2e) = .18$$



*protocol* constrains effective channel throughput!

# CSMA: Carrier Sense Multiple Access)

**CSMA:** listen before transmit:

- If channel sensed idle: transmit entire pkt
- If channel sensed busy, defer transmission
  - **Persistent CSMA:** retry immediately with probability  $p$  when channel becomes idle (may cause instability)
  - **Non-persistent CSMA:** retry after random interval
- human analogy: don't interrupt others!

# CSMA collisions

collisions *can*  
occur:

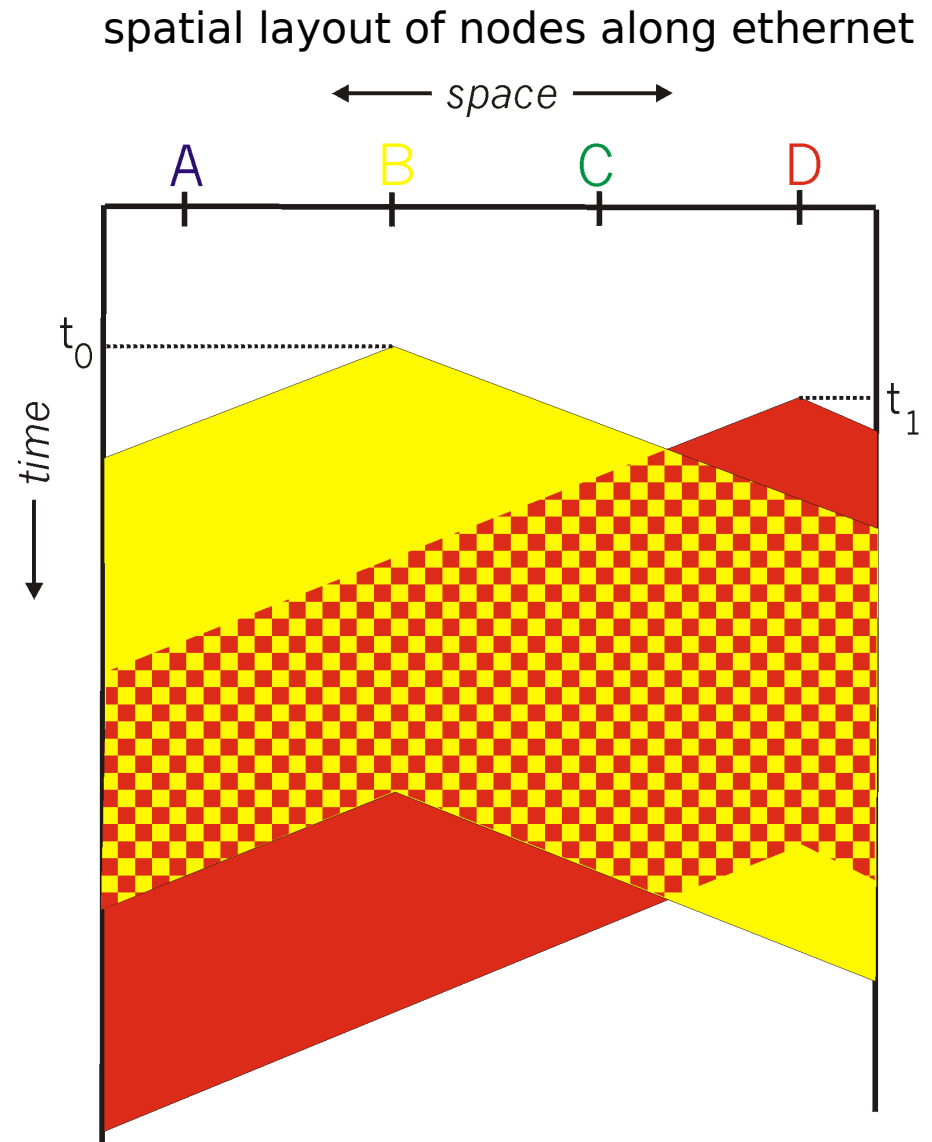
propagation delay  
means  
two nodes may not  
year  
hear each other's  
transmission

**collision:**

entire packet  
transmission  
time wasted

**note:**

role of distance and  
propagation delay in  
determining collision  
prob.

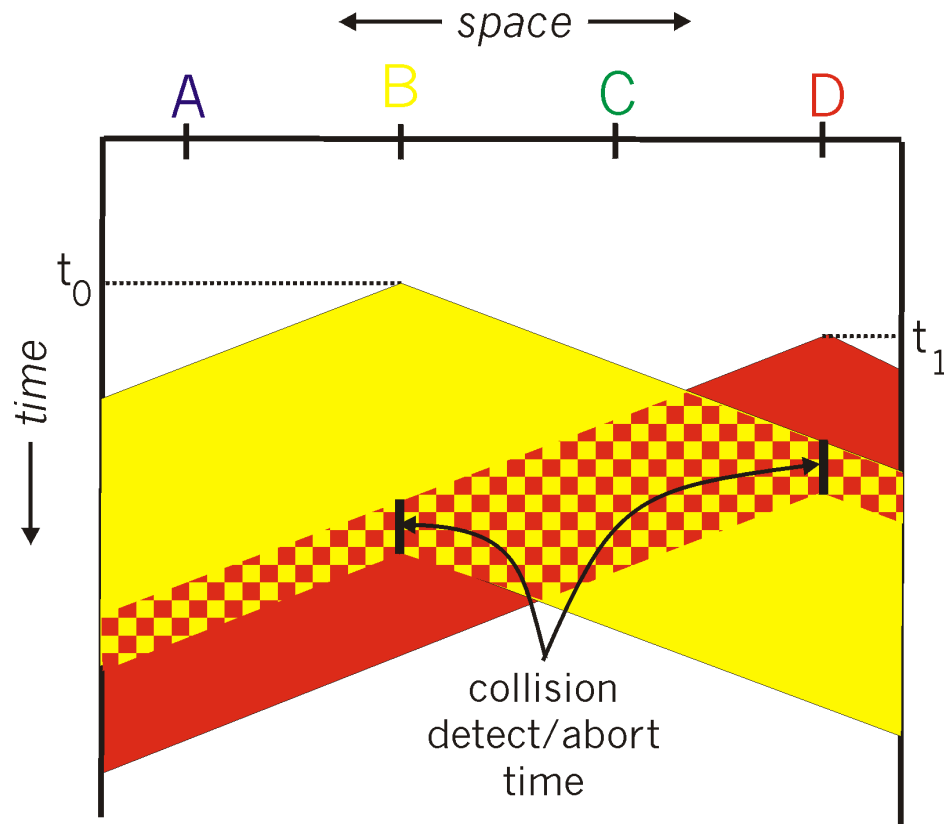


# CSMA/CD (Collision Detection)

**CSMA/CD:** carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- persistent or non-persistent retransmission
- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: receiver shut off while transmitting
- human analogy: the polite conversation

# CSMA/CD collision detection



# Taking Turns MAC protocols

## channel partitioning MAC protocols:

- share channel efficiently at high load
- inefficient at low load: delay in channel access,  $1/N$  bandwidth allocated even if only 1 active node!

## Random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

## taking turns protocols

look for best of both worlds!

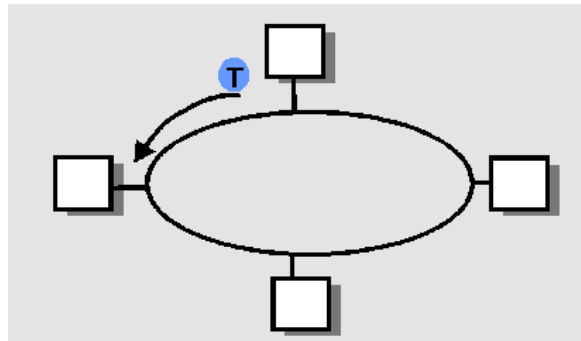
# Taking Turns MAC protocols

## Polling:

- master node invites slave nodes to transmit in turn
- Request to Send, Clear to Send msgs
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)

## Token passing:

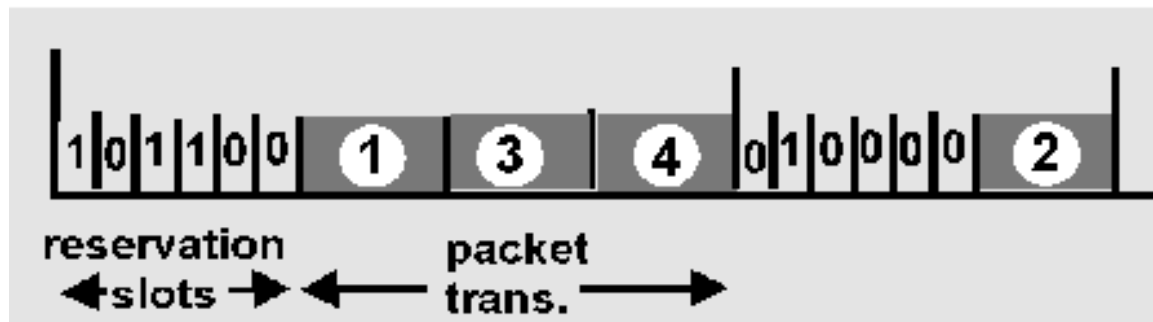
- control **token** passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure



# Reservation-based protocols

## Distributed Polling:

- time divided into slots
- begins with N short **reservation slots**
  - reservation slot time equal to channel end-end propagation delay
  - station with message to send posts reservation
  - reservation seen by all stations
- after reservation slots, message transmissions ordered by known priority



# Summary of MAC protocols

- What do you do with a shared media?
  - Channel Partitioning, by time, frequency or code
    - Time Division, Code Division, Frequency Division
  - Random partitioning (dynamic),
    - ALOHA, S-ALOHA, CSMA, CSMA/CD
    - carrier sensing: easy in some technologies (wire), hard in others (wireless)
    - CSMA/CD used in Ethernet
  - Taking Turns
    - polling from a central site, token passing
    - e.g., Bluetooth, FDDI, IBM Token Ring

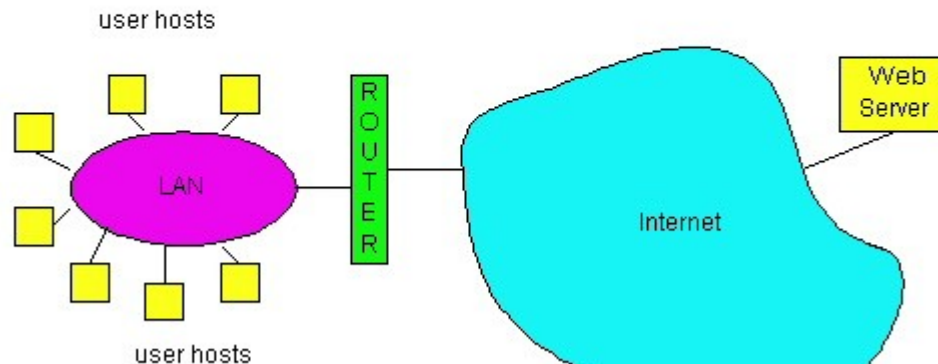
# LAN technologies

Data link layer so far:

- services, error detection/correction, multiple access

Next: LAN technologies

- addressing
- Ethernet
- hubs, bridges, switches
- 802.11
- PPP



# LAN Addresses and ARP

## 32-bit IP address:

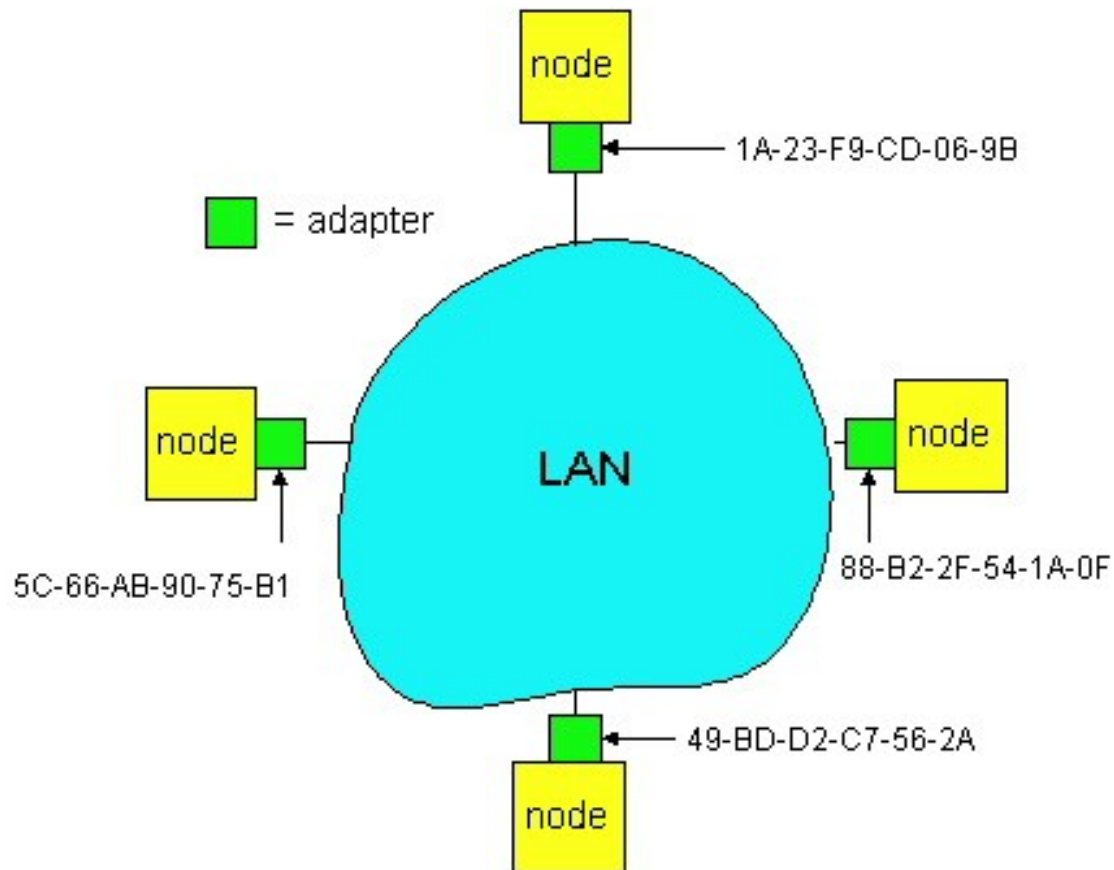
- *network-layer* address
- used to get datagram to destination network (recall IP network definition)

## LAN (or MAC or physical) address:

- used to get datagram from one interface to another physically-connected interface (same network)
- 48 bit MAC address (for most LANs) burned in the adapter ROM

# LAN Addresses and ARP

Each adapter on LAN has unique LAN address



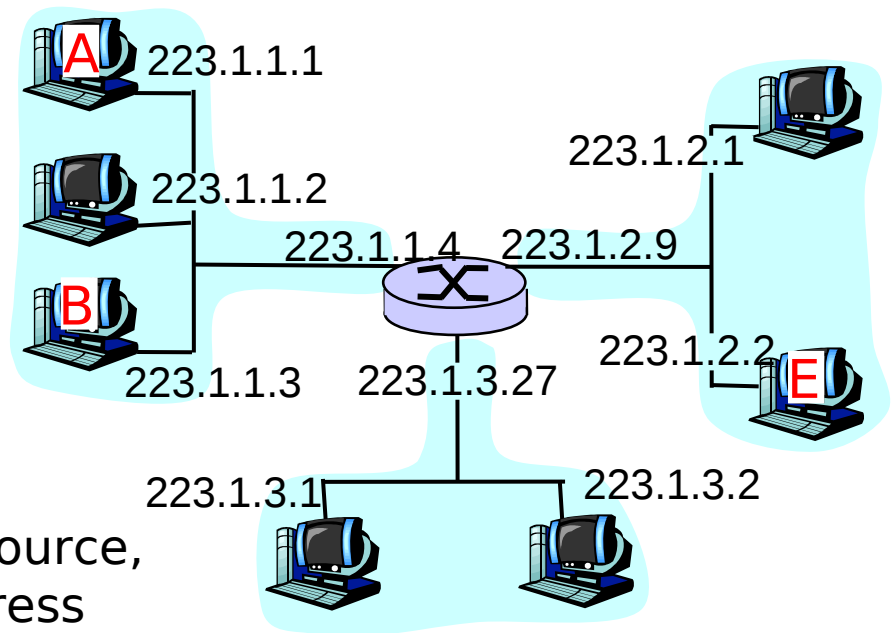
# LAN Address (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- Analogy:
  - (a) MAC address: like Social Security Number
  - (b) IP address: like postal address
- MAC flat address => portability
  - can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - depends on network to which one attach to

# Recall earlier routing discussion

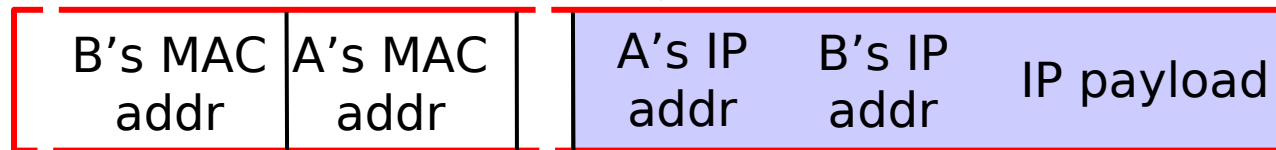
Starting at A, given IP datagram addressed to B:

- look up net. address of B, find B on same net. as A
- **link layer send datagram to B inside link-layer frame**



frame source,  
dest address

datagram source,  
dest address

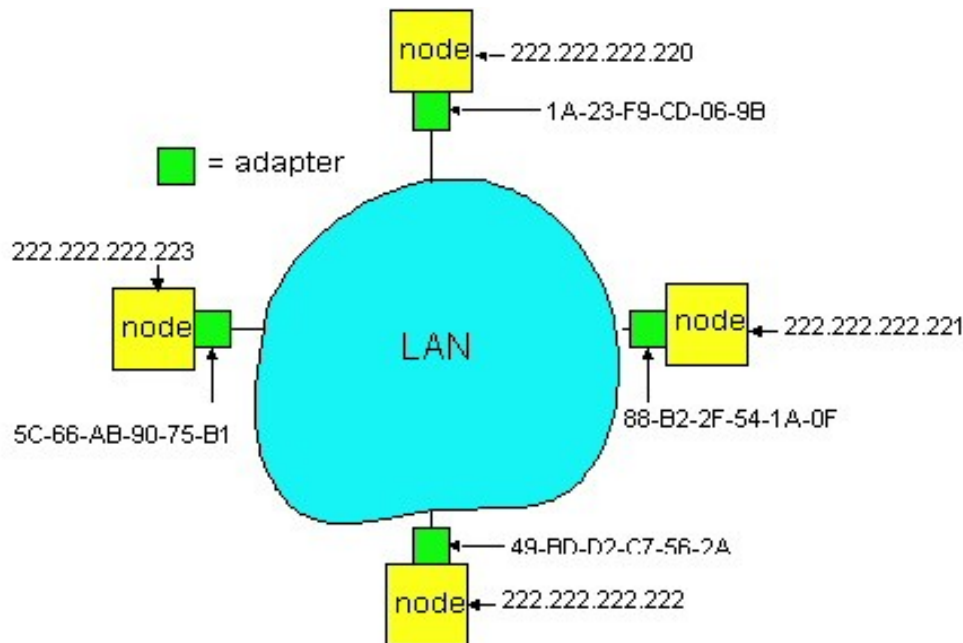


← datagram →

← frame →

# ARP: Address Resolution Protocol

Question: how to determine MAC address of B given B's IP address?



- Each IP node (Host, Router) on LAN has **ARP** module, table
- ARP Table: IP/MAC address mappings for some LAN nodes

< IP address; MAC address; TTL >

< ..... >

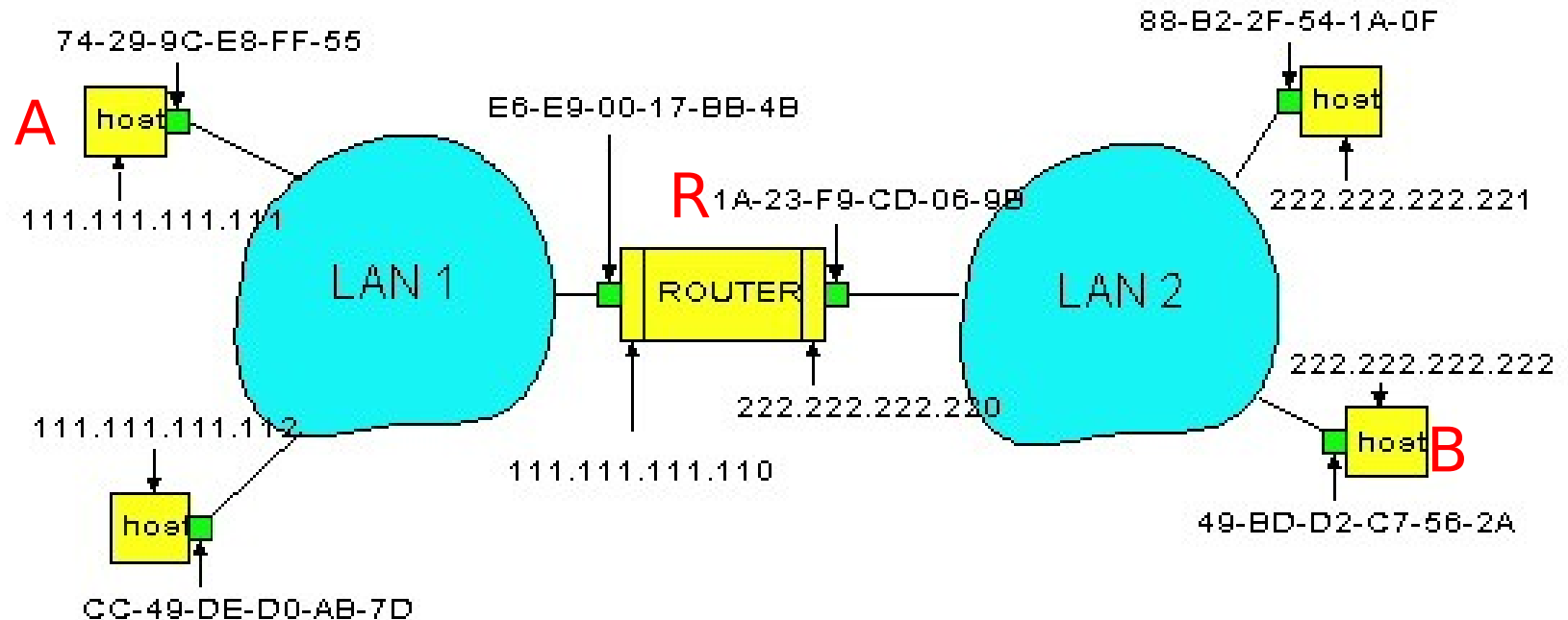
- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

# ARP protocol

- A knows B's IP address, wants to learn physical address of B
- A **broadcasts** ARP query pkt, containing B's IP address
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) physical layer address
- A caches (saves) IP-to-physical address pairs until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is “plug&play”

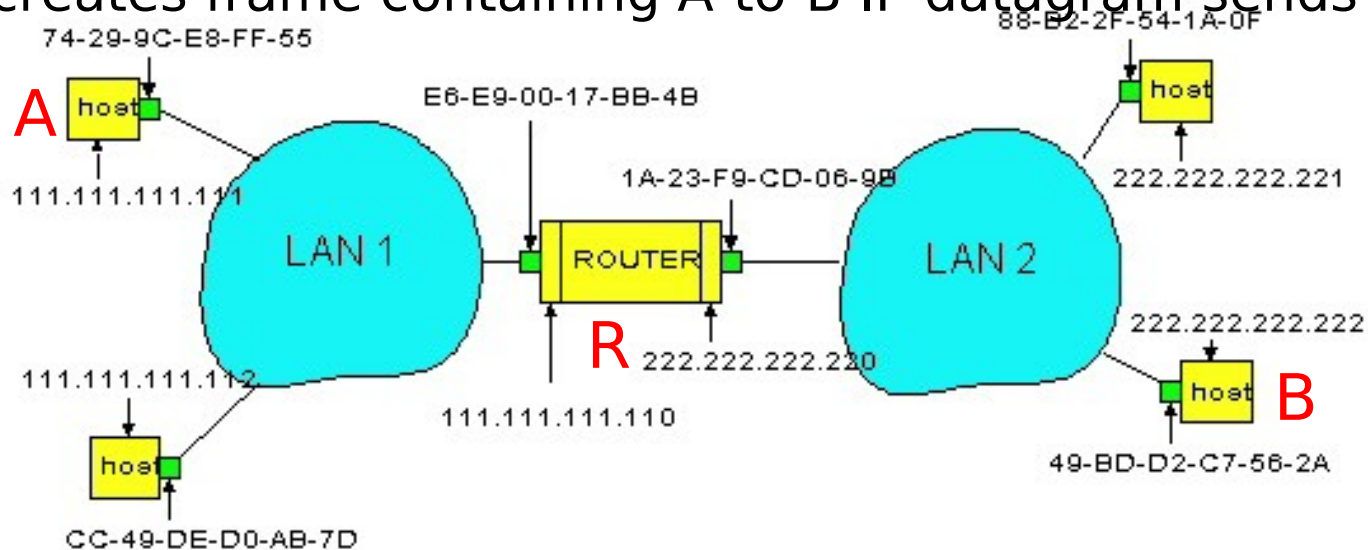
# Routing to another LAN

routing from A to B via R



- In routing table at source Host, find router 111.111.111.110
- In ARP table at source, find MAC address E6-E9-00-17-BB-4B, etc

- A creates IP packet with source A, destination B
- A uses ARP to get R's physical layer address for 111.111.111.110
- A creates Ethernet frame with R's physical address as dest, Ethernet frame contains A-to-B IP datagram
- A's data link layer sends Ethernet frame
- R's data link layer receives Ethernet frame
- R removes IP datagram from Ethernet frame, sees its destined to B
- R uses ARP to get B's physical layer address
- R creates frame containing A-to-B IP datagram sends to B



# See ARP in action:

- Try that in the labs:
  - arp -a
    - This line prints the arp table
    - Your own physical address should appear
  - Ping <neighbour IP>
  - arp -a
    - A new line should appear if the ping was successful
    - The physical address of your neighbour is now known
    - Wait 60+ seconds
  - arp -a
    - The neighbour's line should have disappeared