Routing

What is it?

Dijkstra : Guru of Shortest path Routing

Link State Routing

Bellman Ford : Distance Vector Routing

Analysis of LSR and DVR

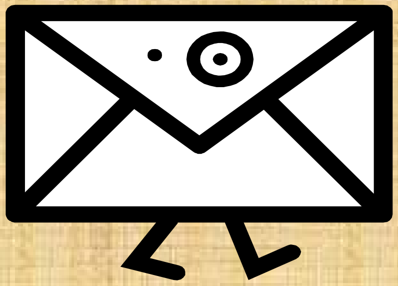
Goals of Today's Lecture

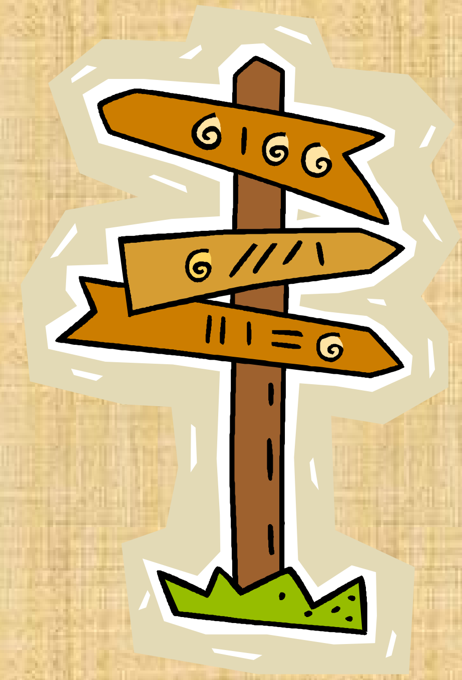
- Inside a router
  - Control plane: routing protocols
  - Data plane: packet forwarding
- Path selection
  - Minimum-hop and shortest-path routing
  - Dijkstra's algorithm
- Topology change
  - Using beacons to detect topology changes
  - Propagating topology information

What is Routing?

- A famous quotation from RFC 791

  "A *name* indicates what we seek.
  An *address* indicates where it is.
  A *route* indicates how we get there."
  -- Jon Postel

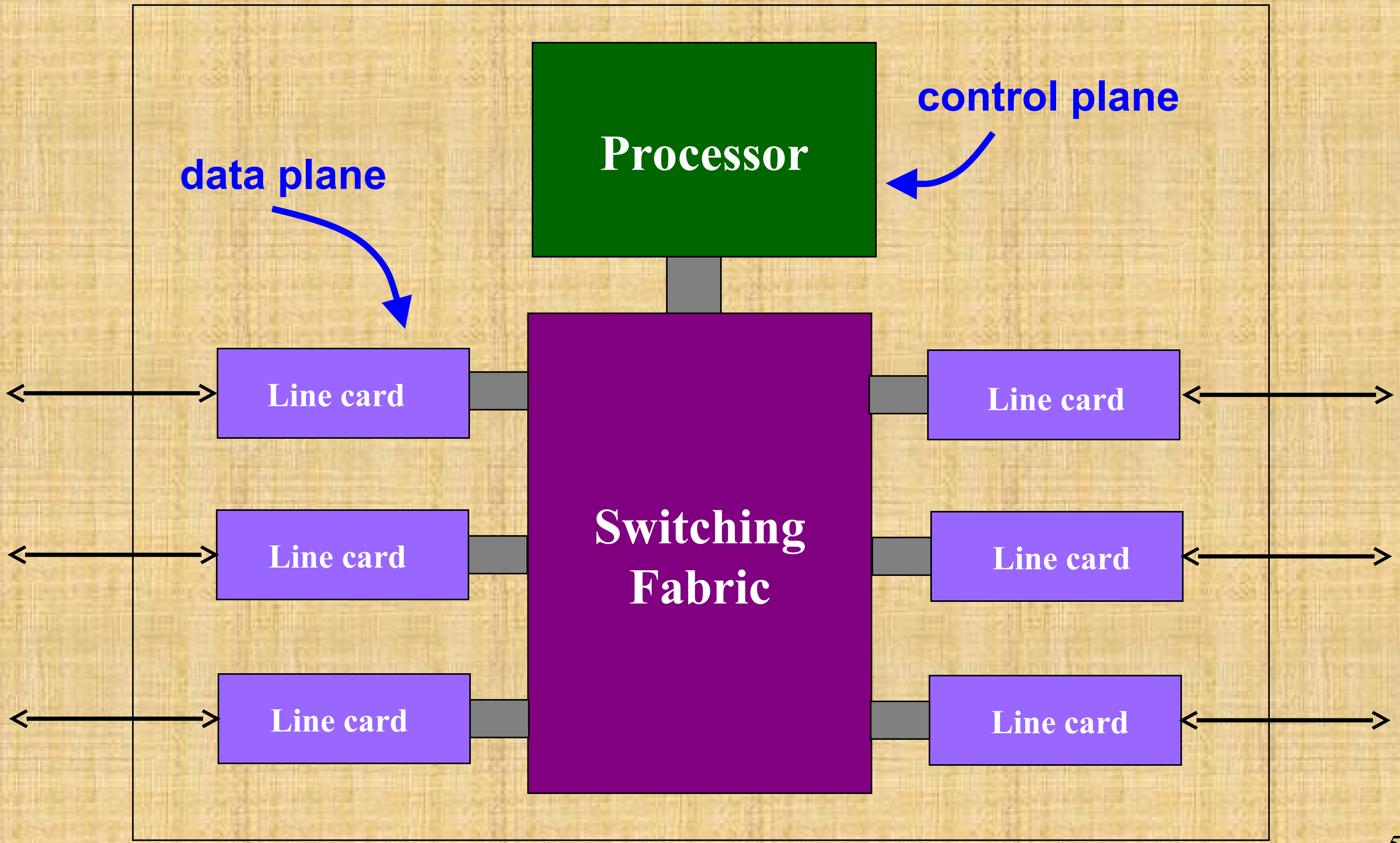




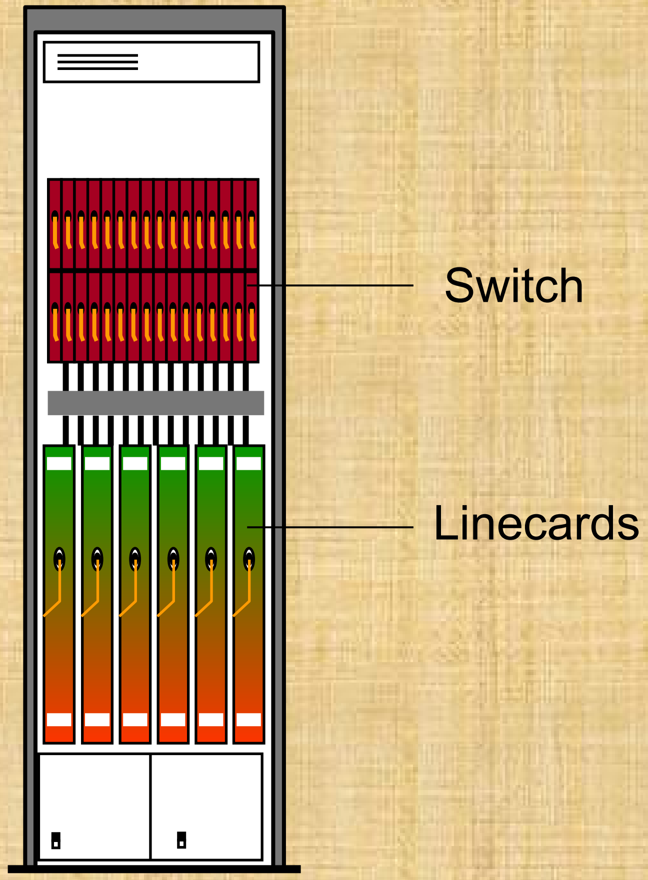
Routing vs. Forwarding

- Routing: control plane
  - –Computing paths the packets will follow
  - –Routers talking amongst themselves
  - –Individual router *creating* a forwarding table
- Forwarding: data plane
  - –Directing a data packet to an outgoing link
  - –Individual router *using* a forwarding table

Data and Control Planes



Router Physical Layout

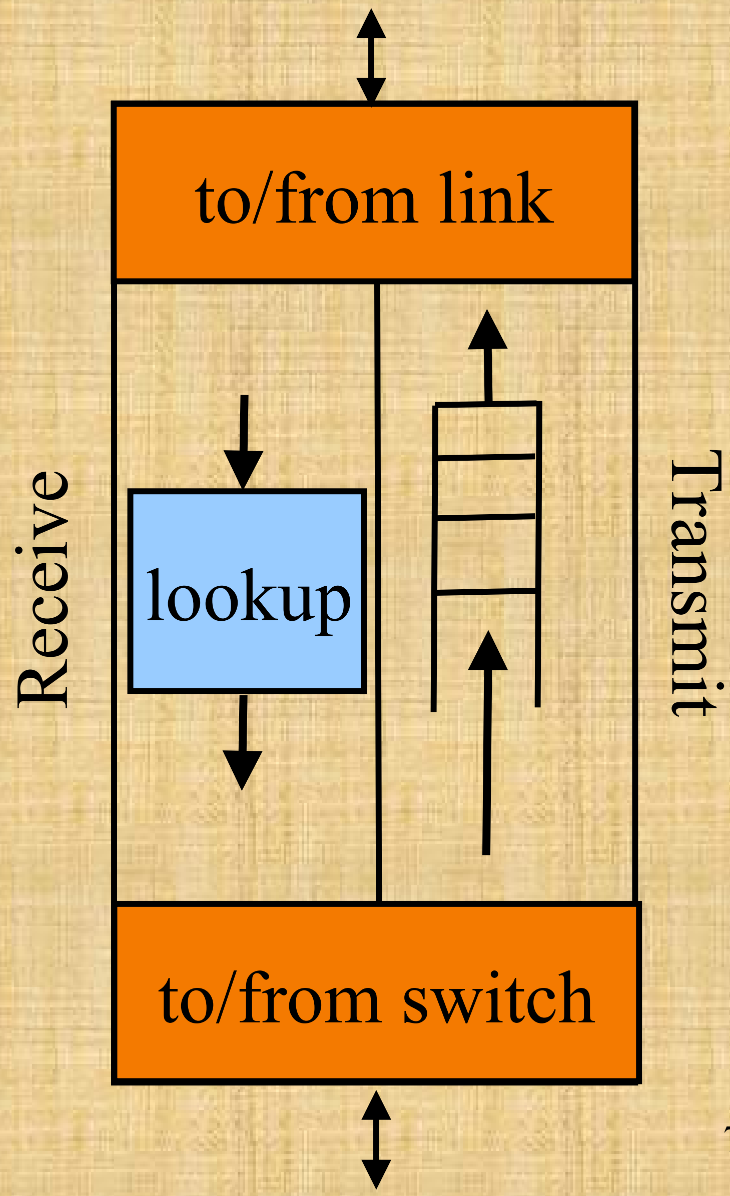


Juniper T series

Cisco 12000

Line Cards (Interface Cards, Adaptors)

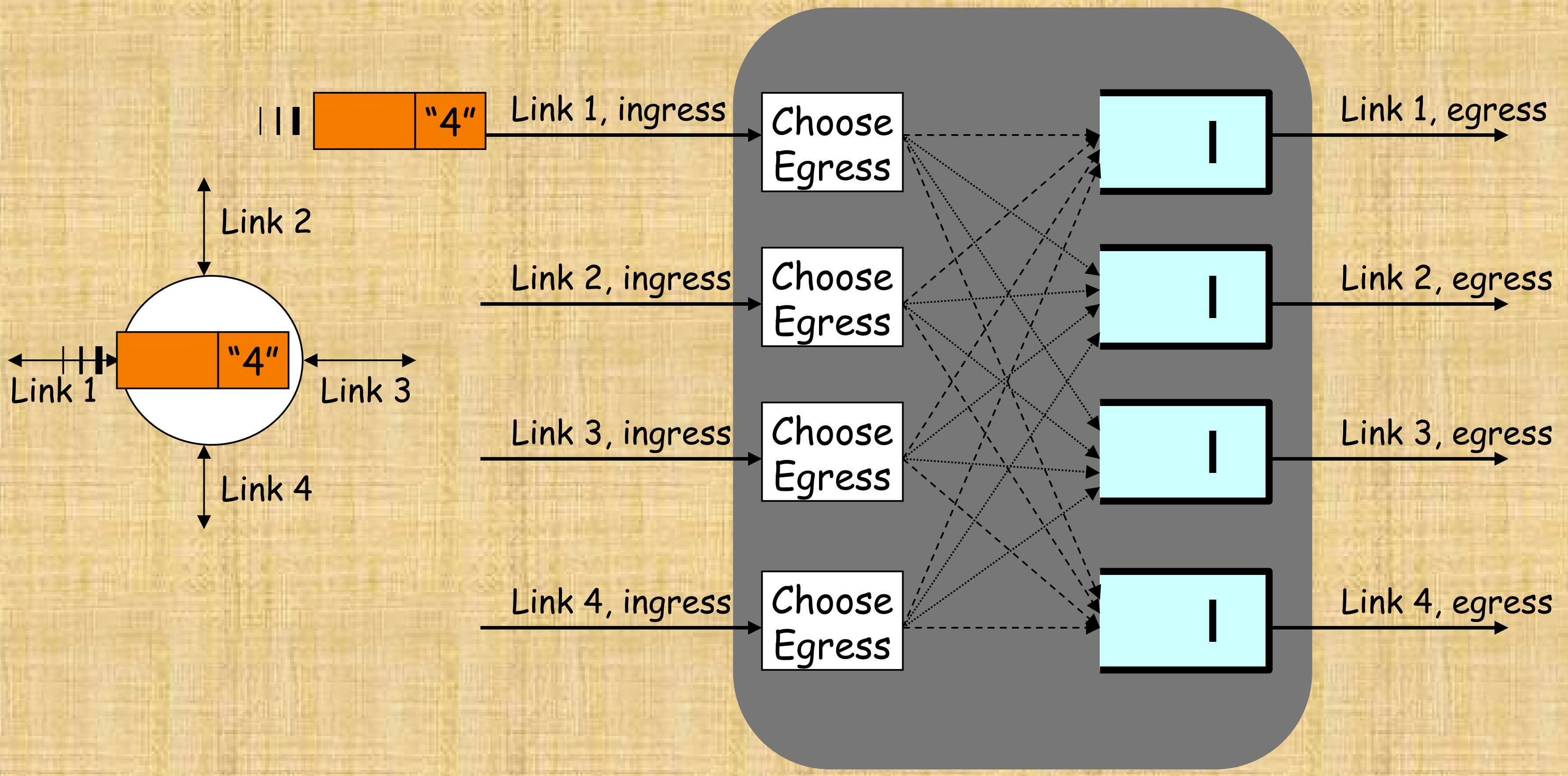
- Interfacing
  - Physical link
  - Switching fabric
- Packet handling
  - Packet forwarding
  - Decrement time-to-live
  - Buffer management
  - Link scheduling
  - Packet filtering
  - Rate limiting
  - Packet marking
  - Measurement



Switching Fabric

- Deliver packet inside the router
  - From incoming interface to outgoing interface
  - A small network in and of itself
- Must operate very quickly
  - Multiple packets going to same outgoing interface
  - Switch scheduling to match inputs to outputs
- Implementation techniques
  - Bus, crossbar, interconnection network, …
  - Running at a faster speed (e.g., 2X) than links
  - Dividing variable-length packets into fixed-size cells

Packet Switching



Router Processor

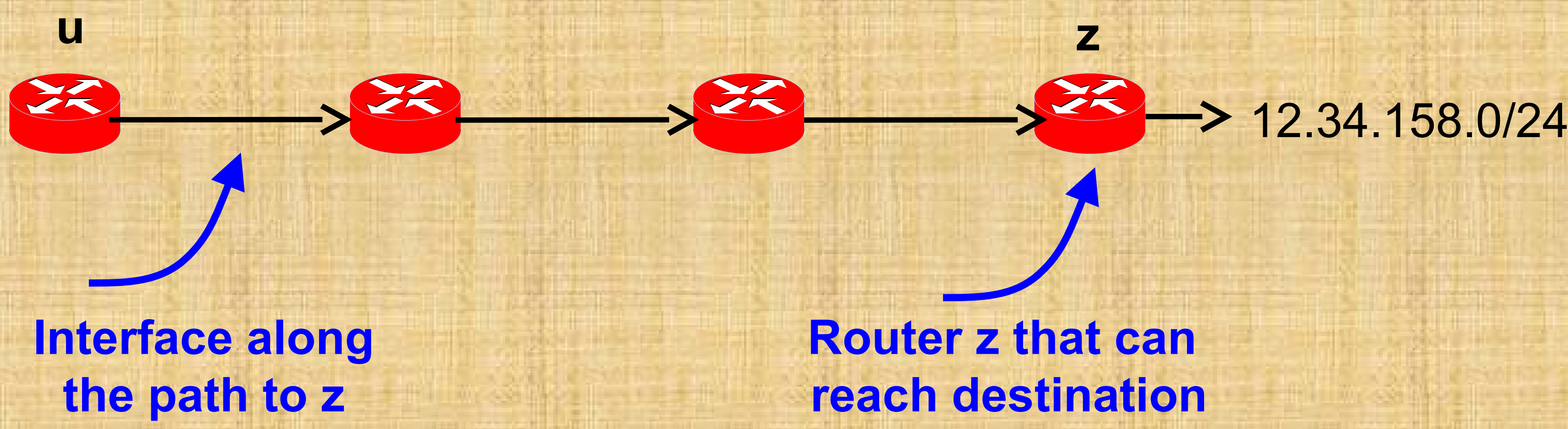
- So-called “Loopback” interface
  - IP address of the CPU on the router
- Interface to network administrators
  - Command-line interface for configuration
  - Transmission of measurement statistics
- Handling of special data packets
  - Packets with IP options enabled
  - Packets with expired Time-To-Live field
- Control-plane software
  - Implementation of the routing protocols
  - Creation of forwarding table for the line cards

Where do Forwarding Tables Come From?

- Routers have forwarding tables
  - Map IP prefix to outgoing link(s)
- Entries can be statically configured
  - E.g., “map 12.34.158.0/24 to Serial0/0.1”
- But, this doesn’t adapt
  - To failures
  - To new equipment
  - To the need to balance load
- That is where routing protocols come in

Computing Paths Between Routers

- Routers need to know two things
  - Which router to use to reach a destination prefix
  - Which outgoing interface to use to reach that router



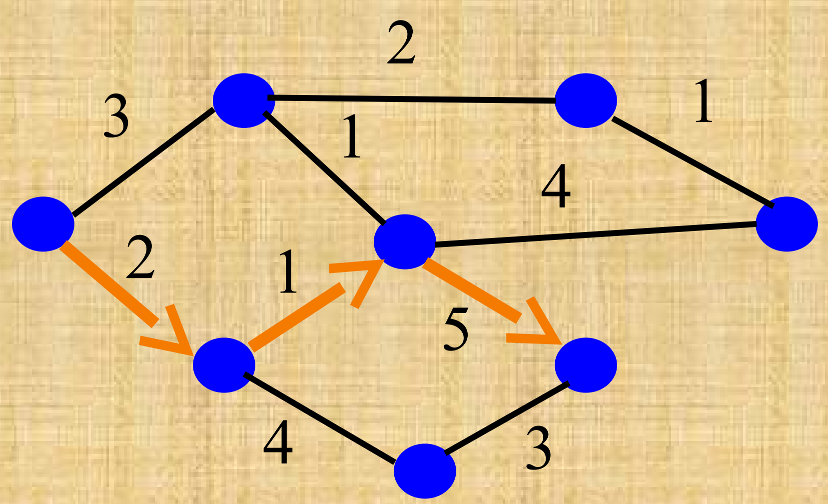
- Today's class: just how routers reach each other
  - How u knows how to forward packets toward z

Computing the Shortest Paths

(assuming you already know the topology)

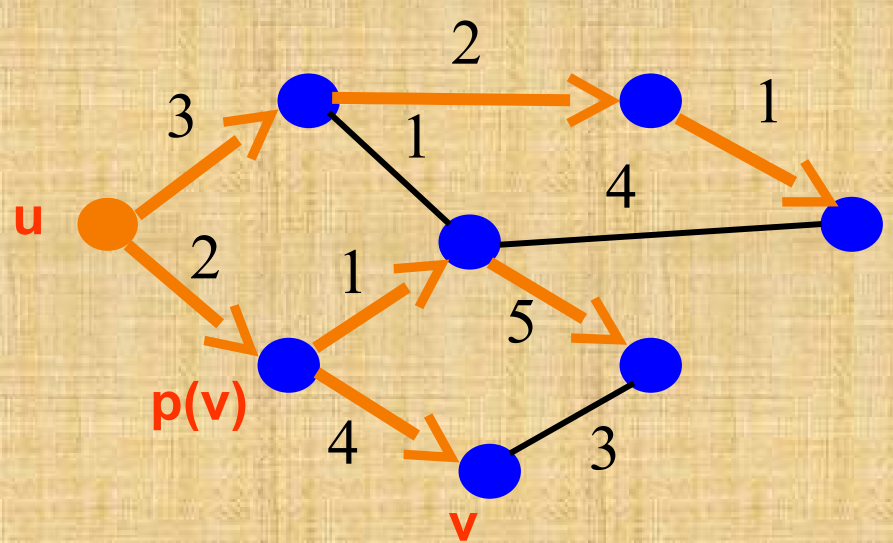
Shortest-Path Routing

- Path-selection model
  - Destination-based
  - Load-insensitive (e.g., static link weights)
  - Minimum hop count or sum of link weights



Shortest-Path Problem

- Given: network topology with link costs
  - **c(x,y)**: link cost from node x to node y
  - Infinity if x and y are not direct neighbors
- Compute: least-cost paths to all nodes
  - From a given source u to all other nodes
  - **p(v)**: predecessor node along path from source to v



Dijkstra's Shortest-Path Algorithm

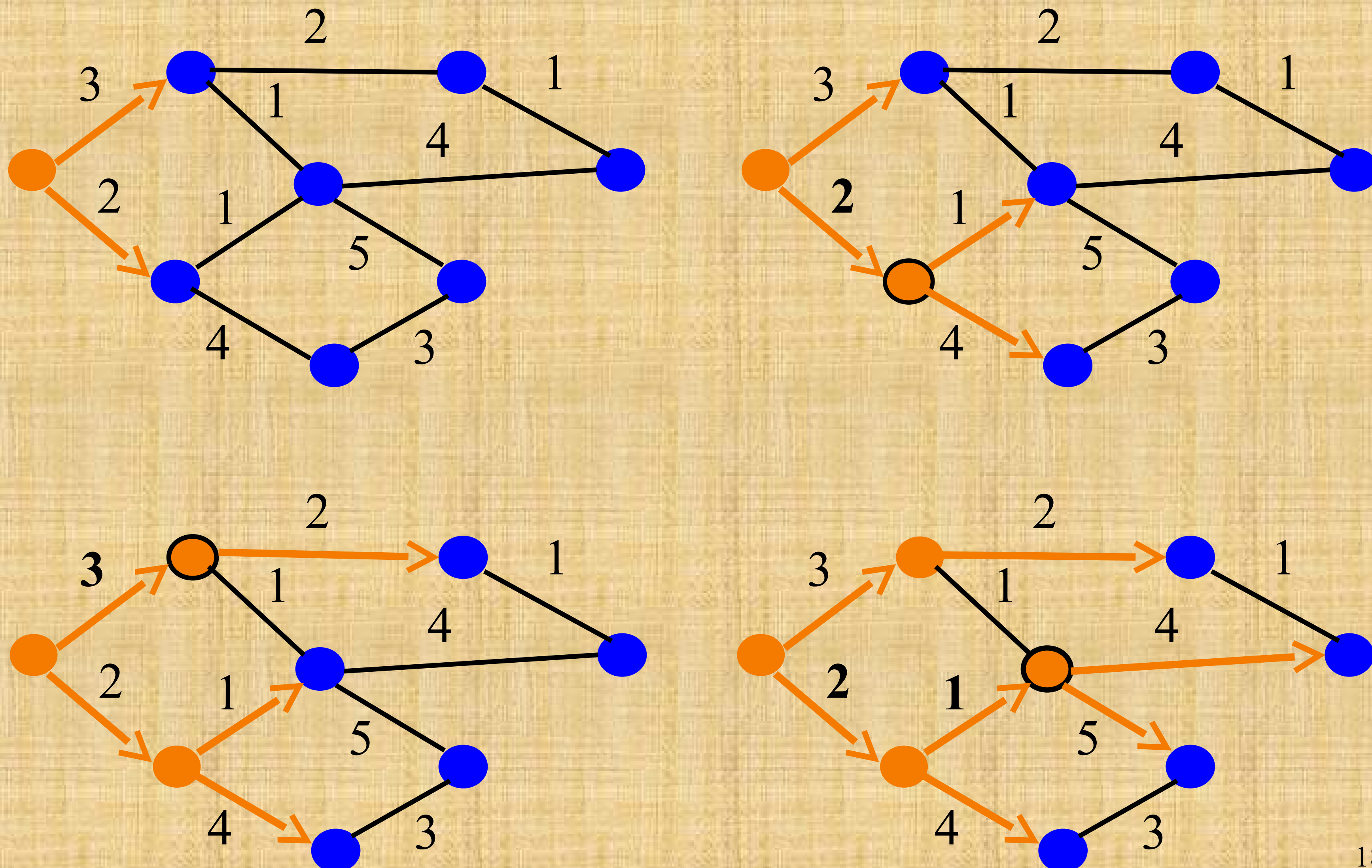
- Iterative algorithm
  - After k iterations, know least-cost path to k nodes
- **S**: nodes whose least-cost path definitively known
  - Initially, **S = {u}** where u is the source node
  - Add one node to S in each iteration
- **D(v)**: current cost of path from source to node v
  - Initially, **D(v) = c(u,v)** for all nodes v adjacent to u
  - … and **D(v) = ∞** for all other nodes v
  - Continually update D(v) as shorter paths are learned

Dijsktra's Algorithm

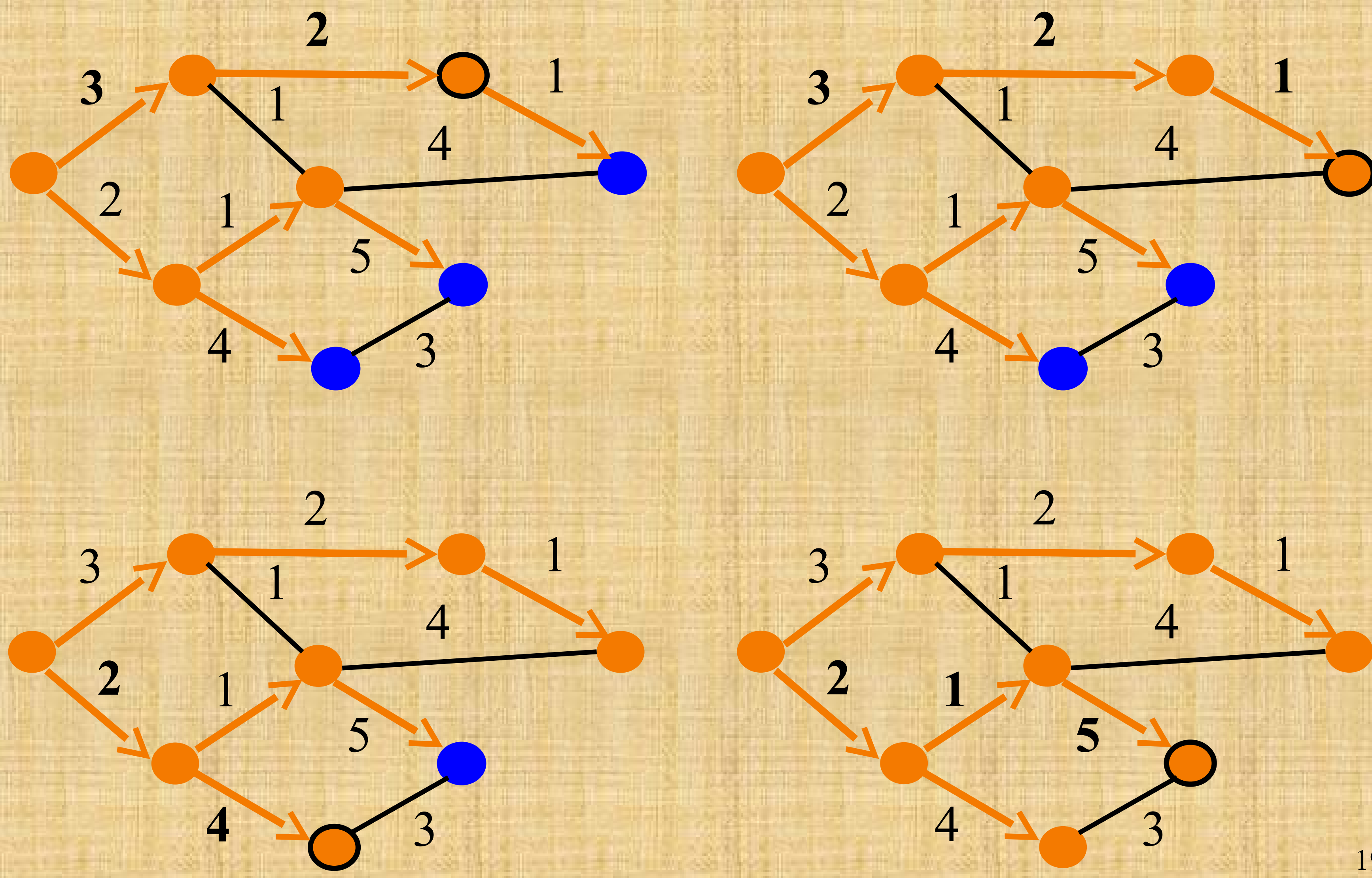
```
Initialization:
  S = {u}
  for all nodes v
    if (v is adjacent to u)
        D(v) = c(u,v)
    else D(v) = ∞

 Loop
   find w not in S with the smallest D(w)
   add w to S
   update D(v) for all v adjacent to w and not in S:
       D(v) = min{D(v), D(w) + c(w,v)}
 until all nodes in S
```

Dijkstra’s Algorithm Example

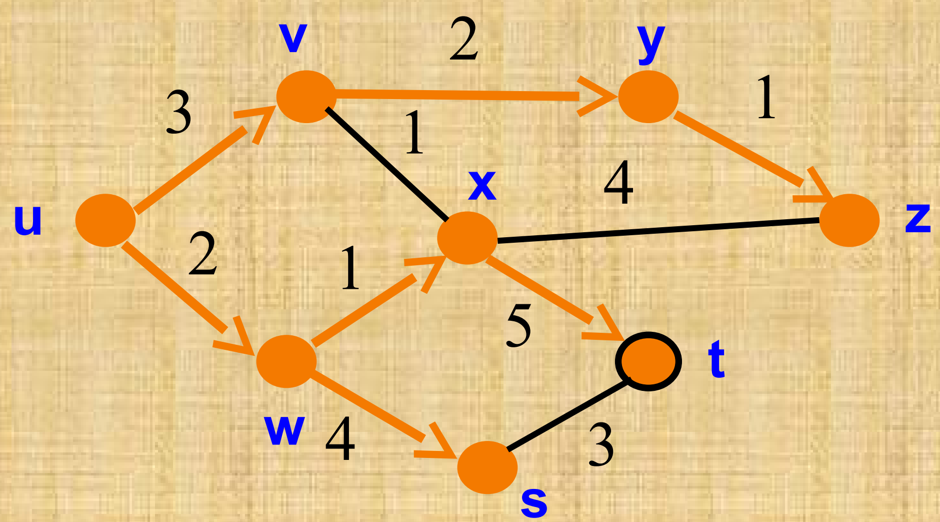


Dijkstra’s Algorithm Example



Shortest-Path Tree

- Shortest-path tree from u



- Forwarding table at u

	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

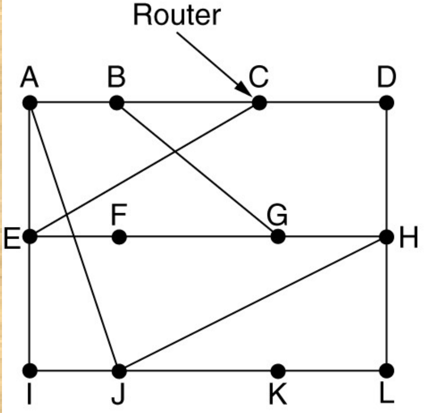
Learning the Topology

(how the routers talk amongst themselves)

1. Distance Vector Routing (DVR)

2. Link State Routing (LSR)

Distance Vector Routing



(a)

Vectors received from J's four neighbors, and the new routing table for J:

To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

(b)

(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

Distance Vector Algorithm

- $c(x,v)$ = cost for direct link from x to v
  - Node x maintains costs of direct links $c(x,v)$
- $D_x(y)$ = estimate of least cost from x to y
  - Node x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$
- Each node v periodically sends D_v to its neighbors
  - And neighbors update their own distance vectors
  - $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$ for each node $y \in N$
- Over time, the distance vector D_x converges

Distance Vector Algorithm

Iterative, asynchronous:

each local iteration caused by:

- Local link cost change
- Distance vector update message from neighbor

Distributed:

- Each node notifies neighbors *only* when its DV changes
- Neighbors then notify their neighbors if necessary

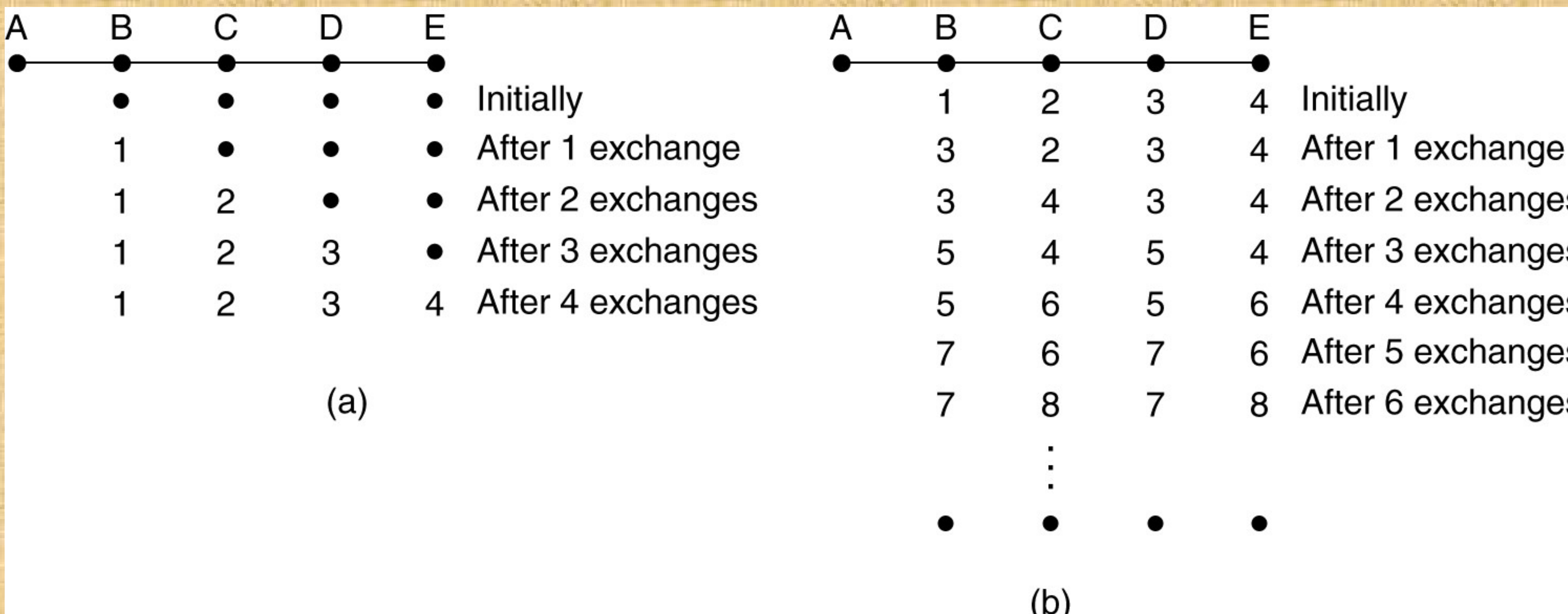
Each node:

wait for (change in local link cost or message from neighbor)

recompute estimates

if distance to any destination has changed, *notify* neighbors

Distance Vector Routing (2)



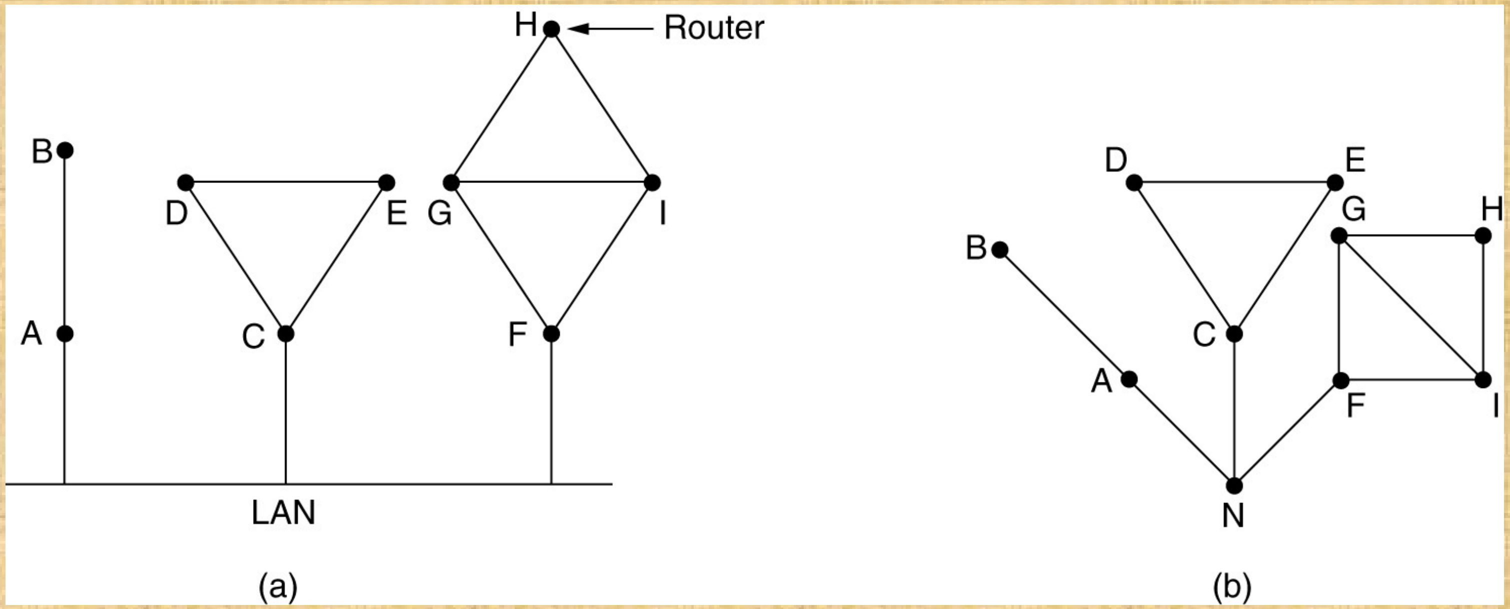
The count-to-infinity problem.

Link State Routing

Each router must do the following:

1. Discover its neighbors, learn their network address.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.

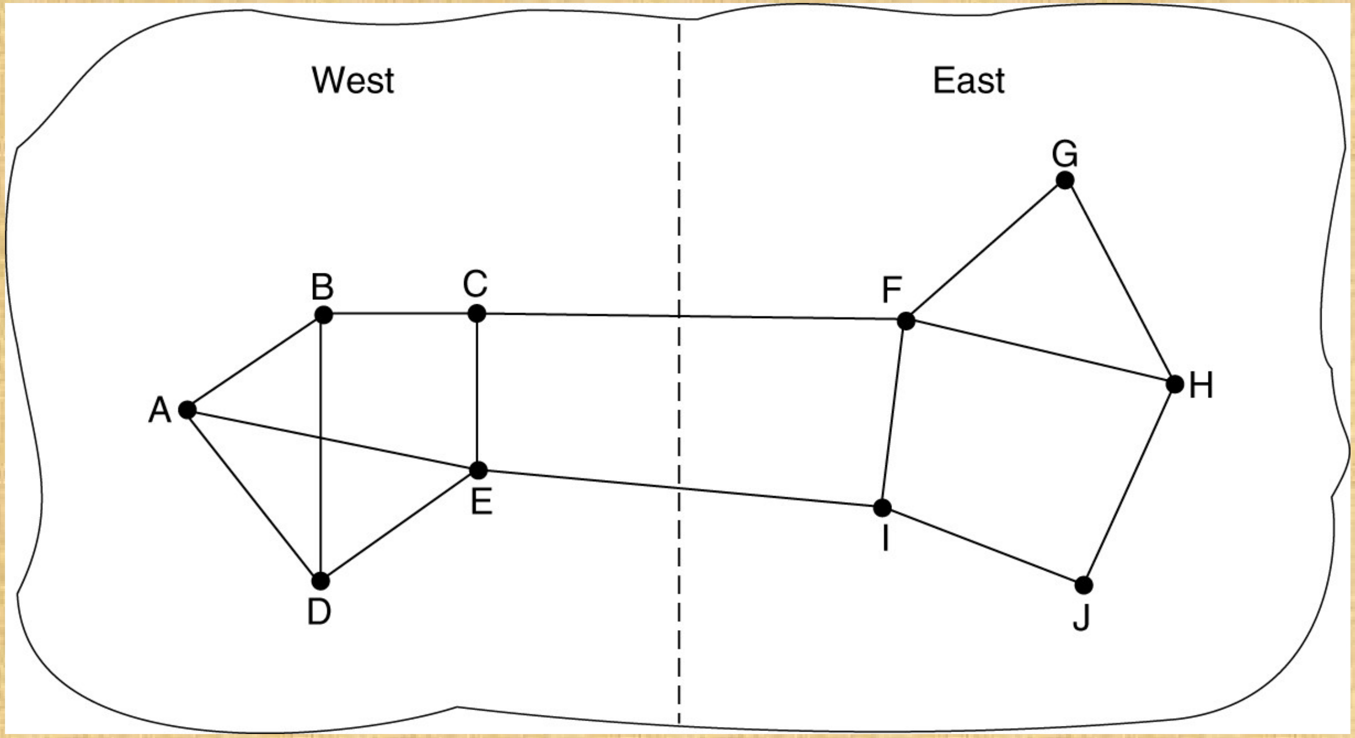
Learning about the Neighbors



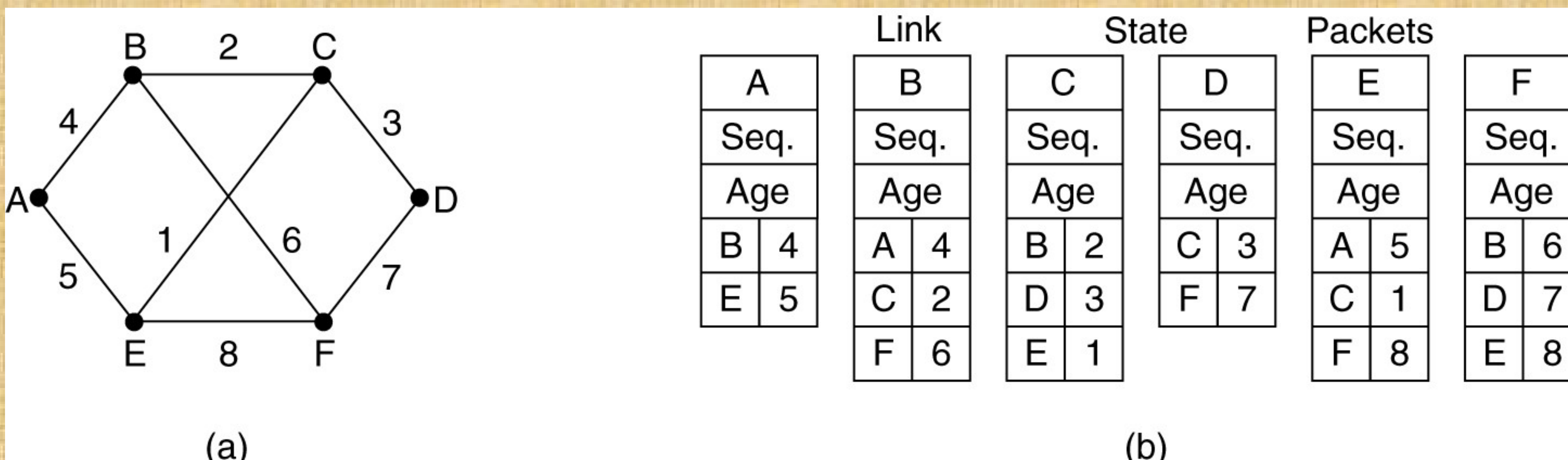
(a) Nine routers and a LAN. (b) A graph model of (a).

Measuring Line Cost

A subnet in which the East and West parts are connected by two lines.



Building Link State Packets



(a) A subnet. (b) The link state packets for this subnet.

Distributing the Link State Packets

The packet buffer for router B in the previous slide (Fig. 5-13).

Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Conclusions

- Routing is a distributed algorithm
  - React to changes in the topology
  - Compute the paths through the network
- Shortest-path link state routing
  - Flood link weights throughout the network
  - Compute shortest paths as a sum of link weights
  - Forward packets on next hop in the shortest path
- Convergence process
  - Changing from one topology to another
  - Transient periods of inconsistency across routers

Comparison of LS and DV Routing

Message complexity

- LS: with n nodes, E links, O(nE) messages sent
- DV: exchange between neighbors only

Speed of Convergence

- LS: relatively fast
- DV: convergence time varies
  - May be routing loops
  - Count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- Node can advertise incorrect *link* cost
- Each node computes only its *own* table

DV:

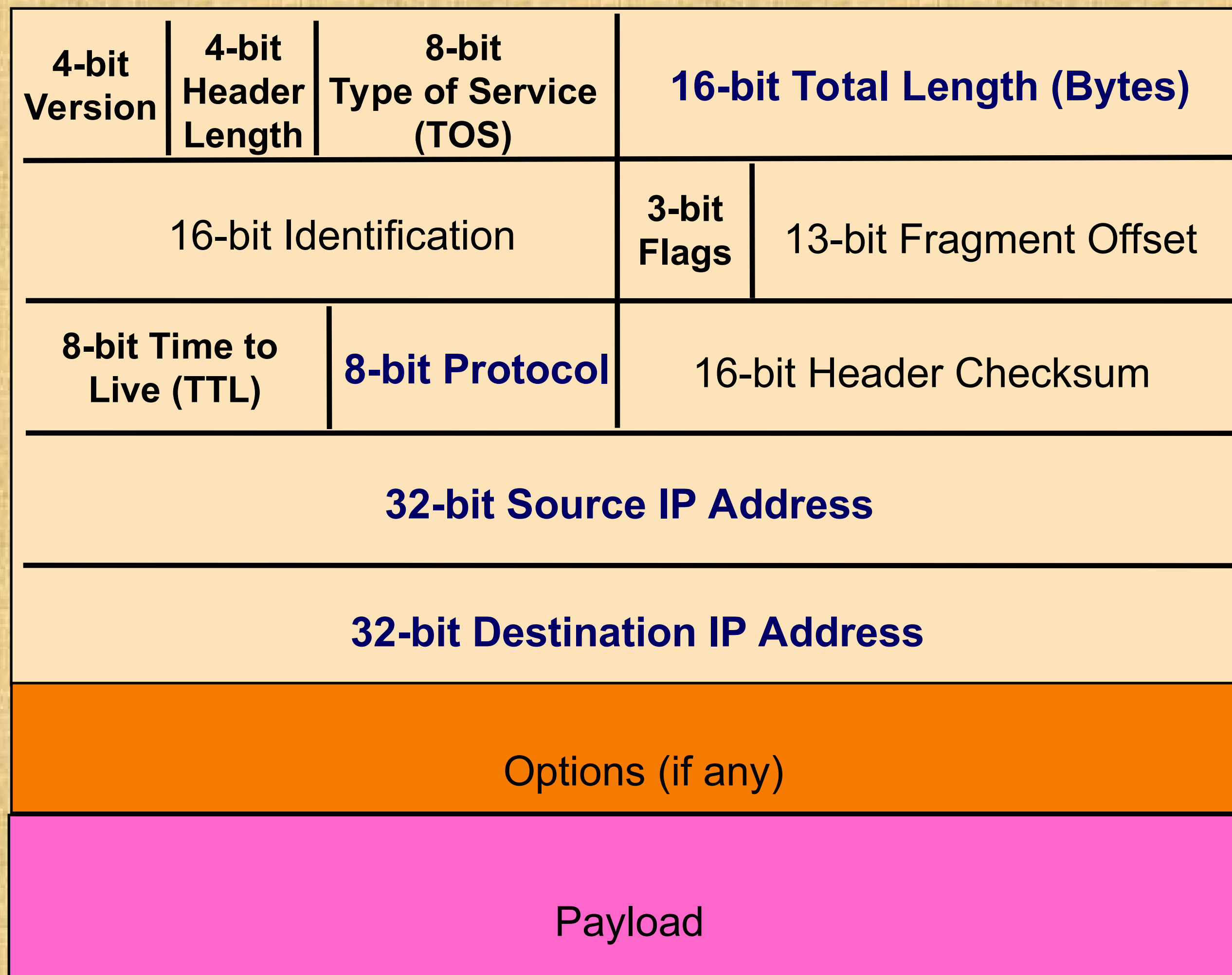
- DV node can advertise incorrect *path* cost
- Each node's table used by others (error propagates)

Similarities of LS and DV Routing

- Shortest-path routing
  - Metric-based, using link weights
  - Routers share a common view of how good a path is
- As such, commonly used *inside* an organization
  - RIP and OSPF are mostly used as *intra*domain protocols
  - E.g., Princeton uses RIP, and AT&T uses OSPF
- But the Internet is a "network of networks"
  - How to stitch the many networks together?
  - When networks may not have common goals
  - … and may not want to share information

IP Packet Structure

IP Packet Structure



IP Header: Version, Length, ToS

- Version number (4 bits)
  - Indicates the version of the IP protocol
  - Necessary to know what other fields to expect
  - Typically "4" (for IPv4), and sometimes "6" (for IPv6)
- Header length (4 bits)
  - Number of 32-bit words in the header
  - Typically "5" (for a 20-byte IPv4 header)
  - Can be more when "IP options" are used
- Type-of-Service (8 bits)
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high bandwidth for bulk transfer

IP Header: Length, Fragments, TTL

- Total length (16 bits)
  - Number of bytes in the packet
  - Maximum size is 63,535 bytes (2^{16} -1)
  - … though underlying links may impose harder limits
- Fragmentation information (32 bits)
  - Packet identifier, flags, and fragment offset
  - Supports dividing a large IP packet into fragments
  - … in case a link cannot handle a large IP packet
- Time-To-Live (8 bits)
  - Used to identify packets stuck in forwarding loops
  - … and eventually discard them from the network

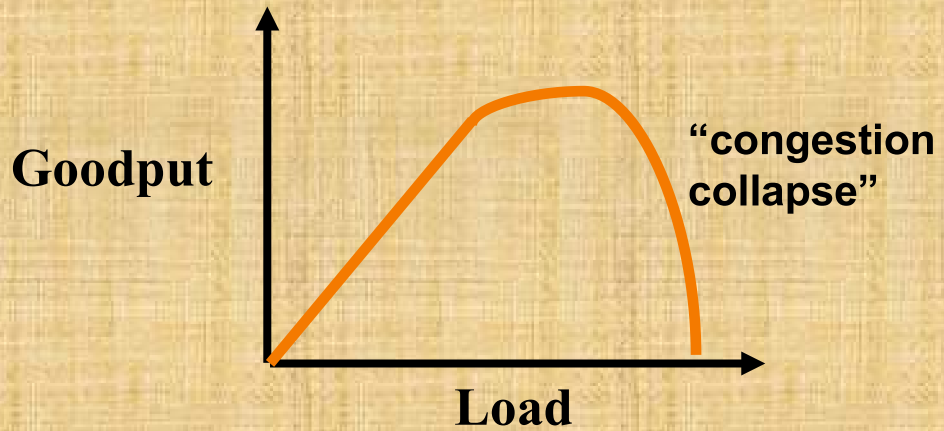
Congestion Control

IP Best-Effort Design Philosophy

- Best-effort delivery
  - Let everybody send
  - Try to deliver what you can
  - … and just drop the rest

The Problem of Congestion

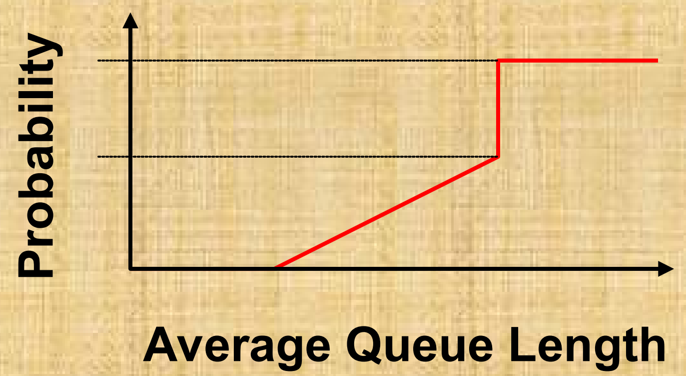
- What is congestion?
  - Load is higher than capacity
- What do IP routers do?
  - Drop the excess packets
- Why is this bad?
  - Wasted bandwidth for retransmissions



Increase in load that results in a *decrease* in useful work done.

Random Early Detection (RED)

- Basic idea of RED
  - Router notices that the queue is getting backlogged
  - … and randomly drops packets to signal congestion
- Packet drop probability
  - Drop probability increases as queue length increases
  - If buffer is below some level, don't drop anything
  - … otherwise, set drop probability as function of queue



Properties of RED

- Drops packets before queue is full
  - In the hope of reducing the rates of some flows
- Drops packet in proportion to each flow's rate
  - High-rate flows have more packets
  - … and, hence, a higher chance of being selected
- Drops are spaced out in time
  - Which should help desynchronize the TCP senders
- Tolerant of burstiness in the traffic
  - By basing the decisions on *average* queue length

Problems With RED

- Hard to get the tunable parameters just right
  - How early to start dropping packets?
  - What slope for the increase in drop probability?
  - What time scale for averaging the queue length?
- Sometimes RED helps but sometimes not
  - If the parameters aren't set right, RED doesn't help
  - And it is hard to know how to set the parameters
- RED is implemented in practice
  - But, often not used due to the challenges of tuning right
- Many variations in the research community
  - With cute names like "Blue" and "FRED"… ☺

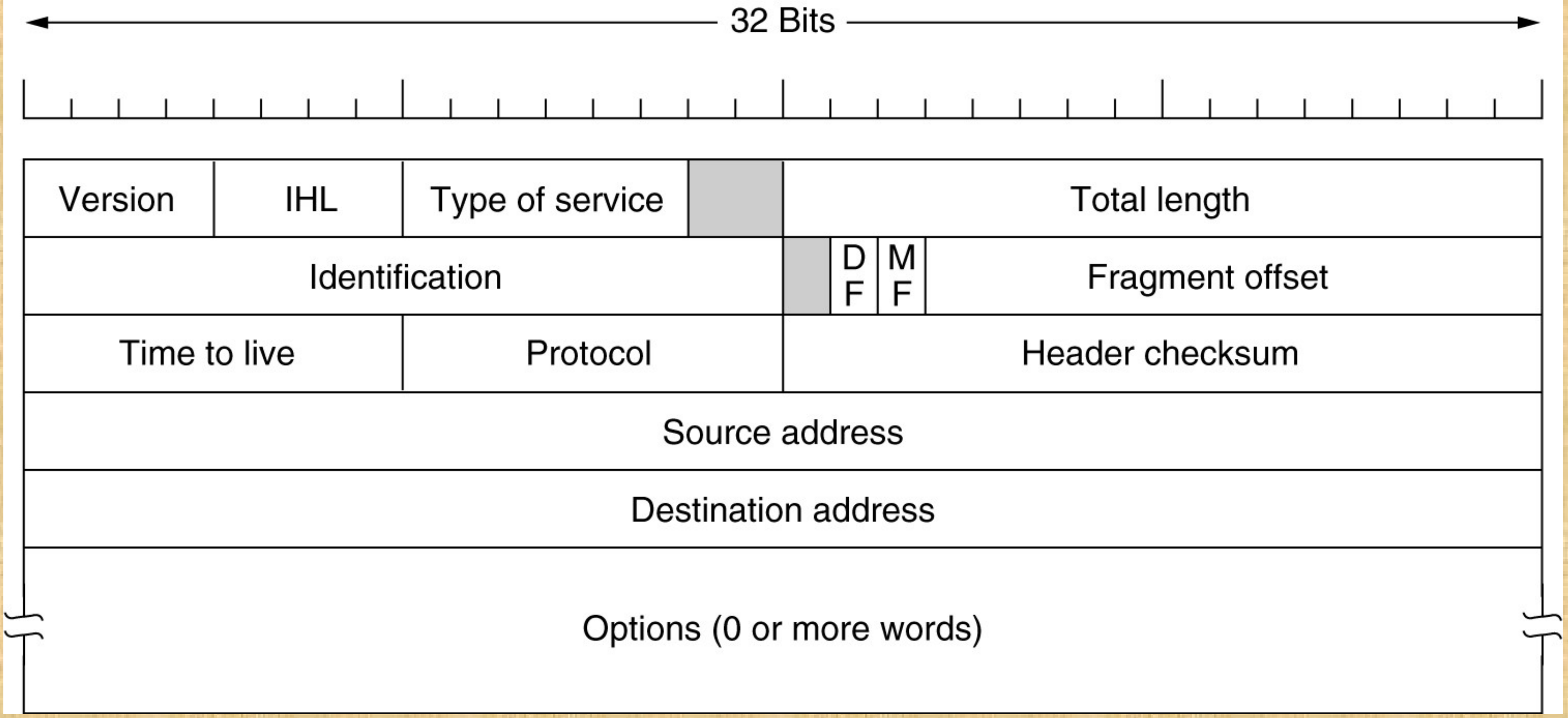
Explicit Congestion Notification

- Early dropping of packets
  - Good: gives early feedback
  - Bad: has to drop the packet to give the feedback
- Explicit Congestion Notification
  - Router marks the packet with an ECN bit
  - … and sending host interprets as a sign of congestion
- Surmounting the challenges
  - Must be supported by the end hosts and the routers
  - Requires two bits in the IP header (one for the ECN mark, and one to indicate the ECN capability)
  - Solution: borrow two of the Type-Of-Service bits in the IPv4 packet header

IP Addresses

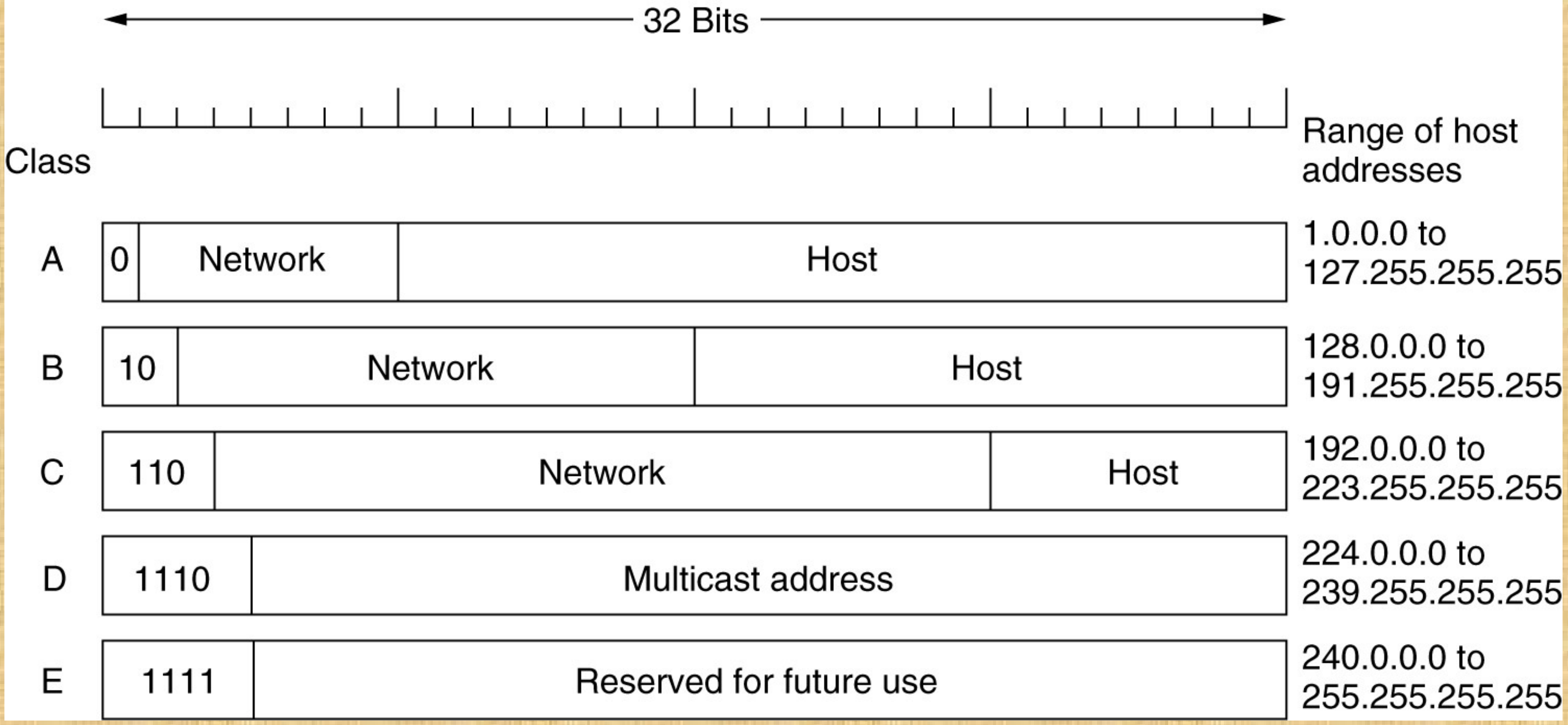
The IP Protocol

The IPv4 (Internet Protocol) header.



IP Addresses

IP address formats.



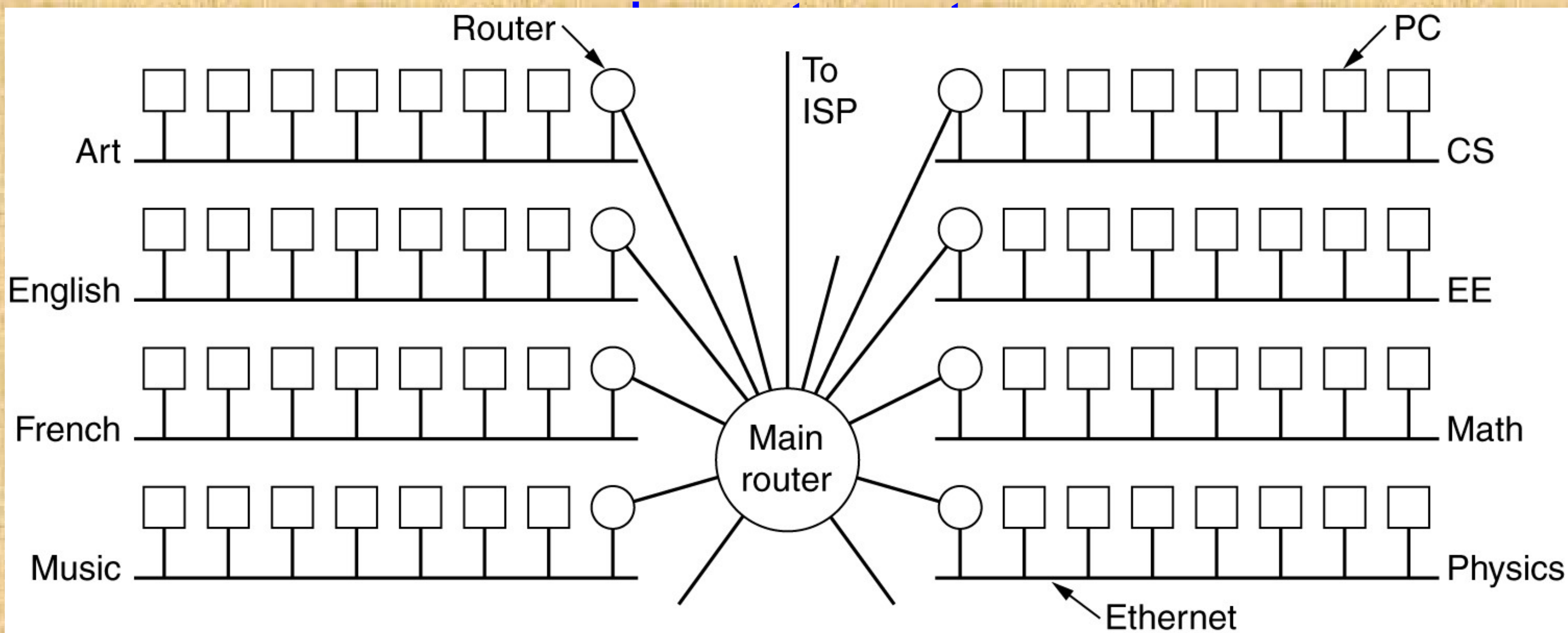
IP Addresses (2)

Special IP addresses.

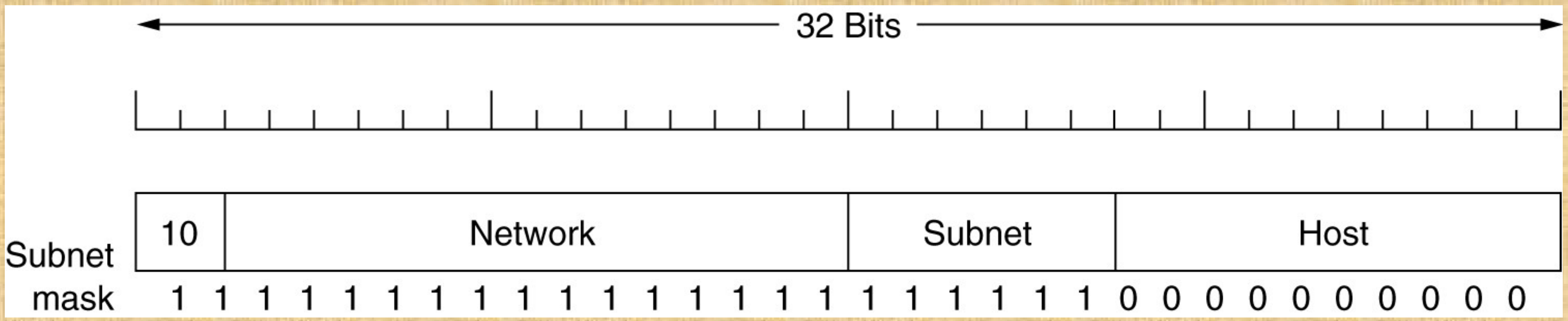
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		This host
0 0 . . . 0 0	Host	A host on this network
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		Broadcast on the local network
Network	1 1 1 1 . . . 1 1 1 1	Broadcast on a distant network
127	(Anything)	Loopback

Subnets

A campus network consisting of LANs for various



Subnets (2)



A class B network subnetted into 64 subnets.

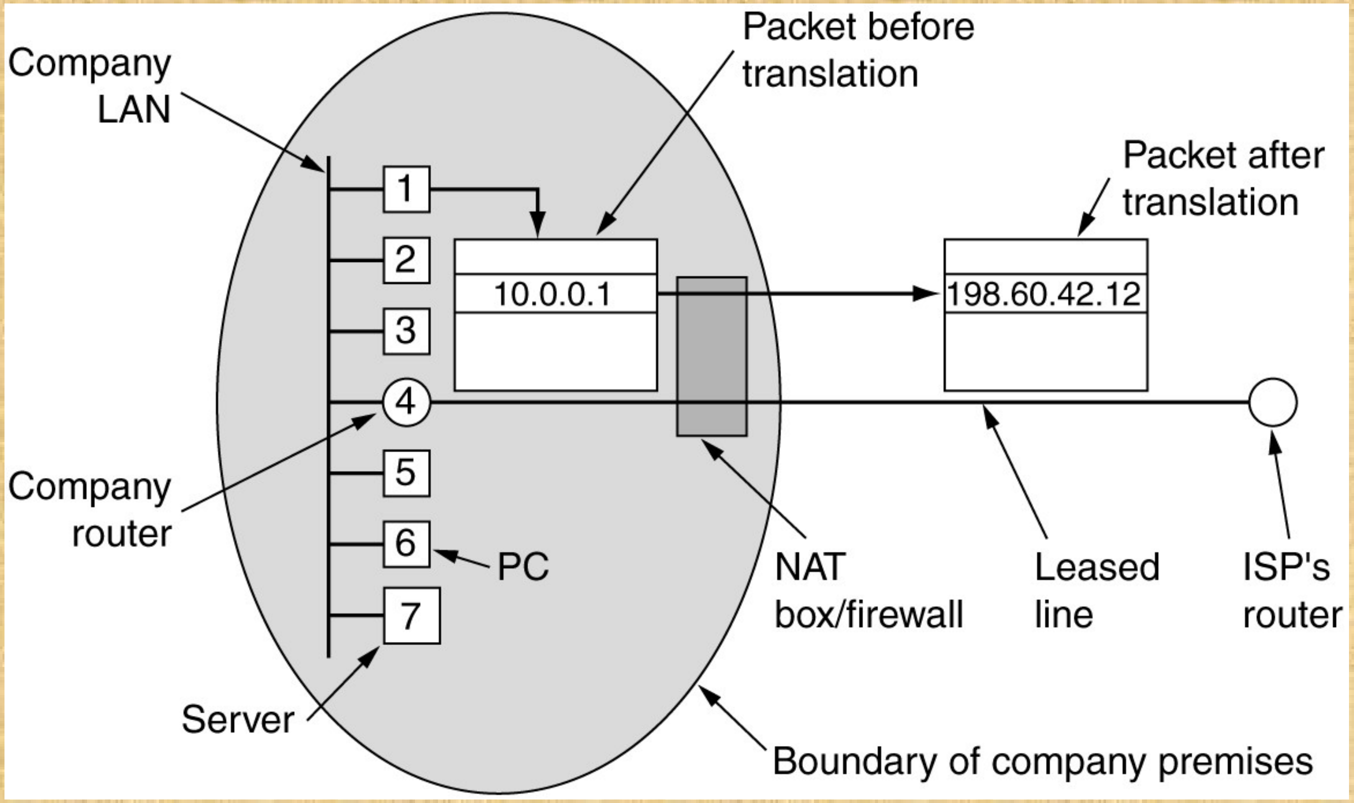
CIDR – Classless InterDomain Routing

A set of IP address assignments.

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

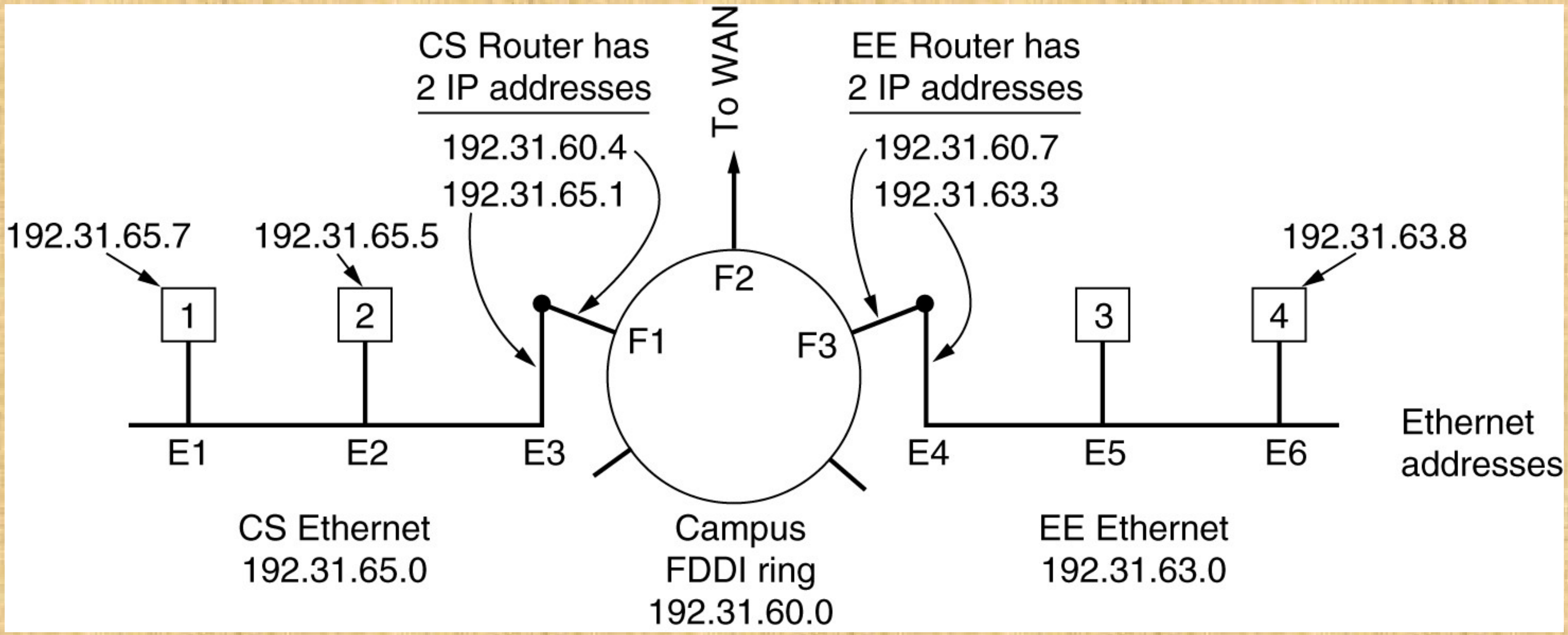
NAT – Network Address Translation

Placement and operation of a NAT box.



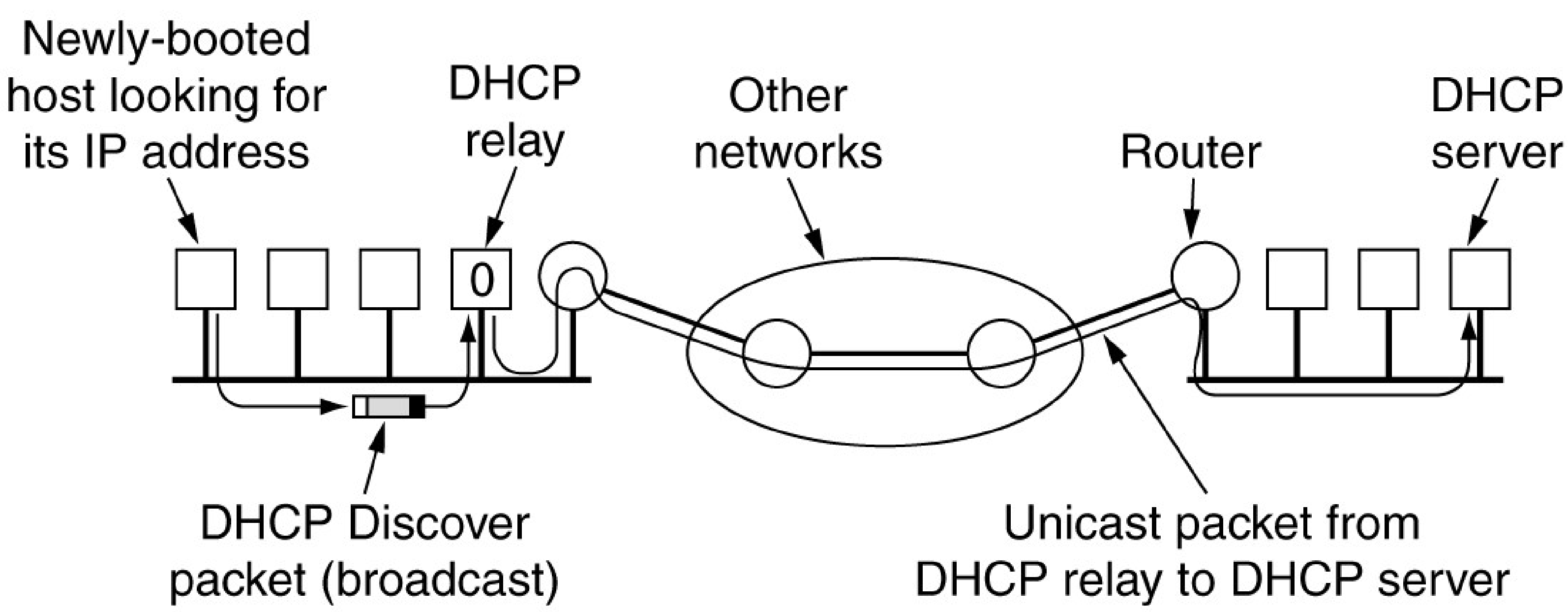
ARP– The Address Resolution Protocol

Three interconnected /24 networks: two Ethernets



Dynamic Host Configuration Protocol

Operation of DHCP.



Overview

- Three different kinds of addresses
  - Host names (e.g., www.cnn.com)
  - IP addresses (e.g., 64.236.16.20)
  - MAC addresses (e.g., 00-15-C5-49-04-A9)
- Protocols for translating between addresses
  - Domain Name System (DNS)
  - Dynamic Host Configuration Protocol (DHCP)
  - Address Resolution Protocol (ARP)
- Two main topics
  - Decentralized management of the name space
  - Boot-strapping an end host that attaches to the ‘net

Separating Names and IP Addresses

- Names are easier (for us!) to remember
  - www.cnn.com vs. 64.236.16.20
- IP addresses can change underneath
  - Move www.cnn.com to 173.15.201.39
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site
- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content
- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu

Separating IP and MAC Addresses

- LANs are designed for arbitrary network protocols
  - Not just for IP (e.g., IPX, Appletalk, X.25, …)
    - Though now IP is the main game in town
  - Different LANs may have different addressing schemes
    - Though now Ethernet address is the main game in town
- A host may move to a new location
  - So, cannot simply assign a static IP address
    - Since IP addresses depend on host's position in topology
  - Instead, must reconfigure the adapter
    - To assign it an IP address based on its current location
- Must identify the adapter during bootstrap process
  - Need to talk to the adapter to assign it an IP address

Three Kinds of Identifiers

- **Host name** (e.g., www.cnn.com)
  - Mnemonic name appreciated *by humans*
  - Provides little (if any) information about location
  - Hierarchical, variable # of alpha-numeric characters
- **IP address** (e.g., 64.236.16.20)
  - Numerical address appreciated *by routers*
  - Related to host’s current location in the topology
  - Hierarchical name space of 32 bits
- **MAC address** (e.g., 00-15-C5-49-04-A9)
  - Numerical address appreciated *within local area network*
  - Unique, hard-coded in the adapter when it is built
  - Flat name space of 48 bits

Three Hierarchical Assignment Processes

- **Host name:** www.cs.princeton.edu
  - Domain: registrar for each top-level domain (e.g., .edu)
  - Host name: local administrator assigns to each host
- **IP addresses:** 128.112.7.156
  - Prefixes: ICANN, regional Internet registries, and ISPs
  - Hosts: static configuration, or dynamic using DHCP
- **MAC addresses:** 00-15-C5-49-04-A9
  - Blocks: assigned to vendors by the IEEE
  - Adapters: assigned by the vendor from its block

Mapping Between Identifiers

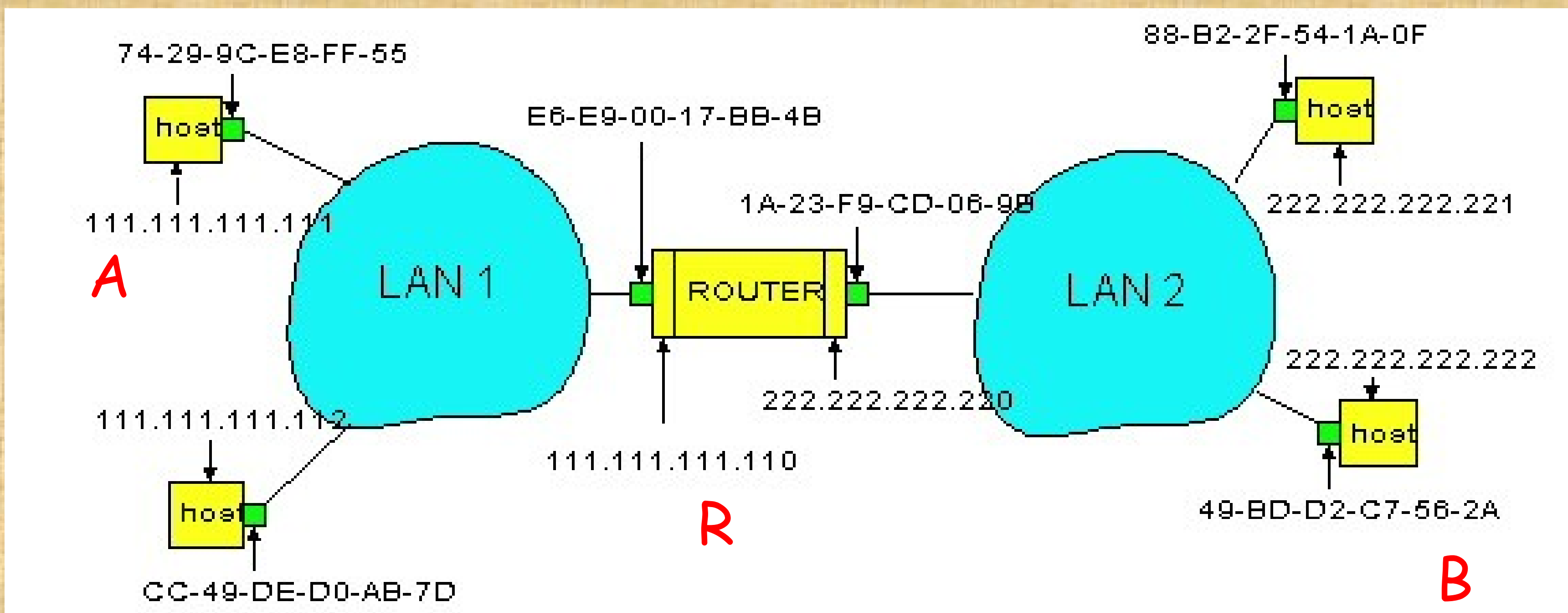
- Domain Name System (DNS)
  - Given a host name, provide the IP address
  - Given an IP address, provide the host name
- Dynamic Host Configuration Protocol (DHCP)
  - Given a MAC address, assign a unique IP address
  - … and tell host other stuff about the Local Area Network
  - To automate the boot-strapping process
- Address Resolution Protocol (ARP)
  - Given an IP address, provide the MAC address
  - To enable communication within the Local Area Network

Address Resolution Protocol Table

- Every node maintains an ARP table
  - (IP address, MAC address) pair
- Consult the table when sending a packet
  - Map destination IP address to destination MAC address
  - Encapsulate and transmit the data packet
- But, what if the IP address is not in the table?
  - Sender broadcasts: "Who has IP address 1.2.3.156?"
  - Receiver responds: "MAC address 58-23-D7-FA-20-B0"
  - Sender caches the result in its ARP table
- No need for network administrator to get involved

Example: A Sending a Packet to B

How does host A send an IP packet to B (www.cnn.com)?

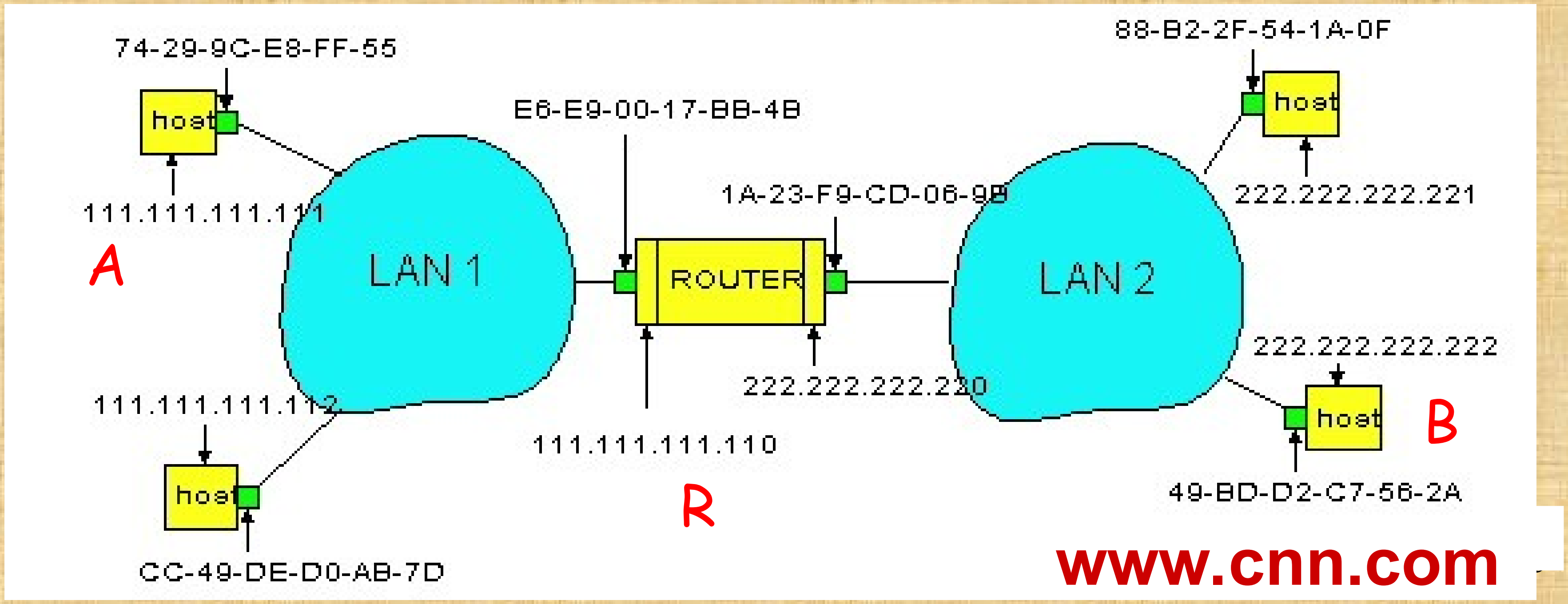


www.cnn.com

A sends packet to R, and R sends packet to B.

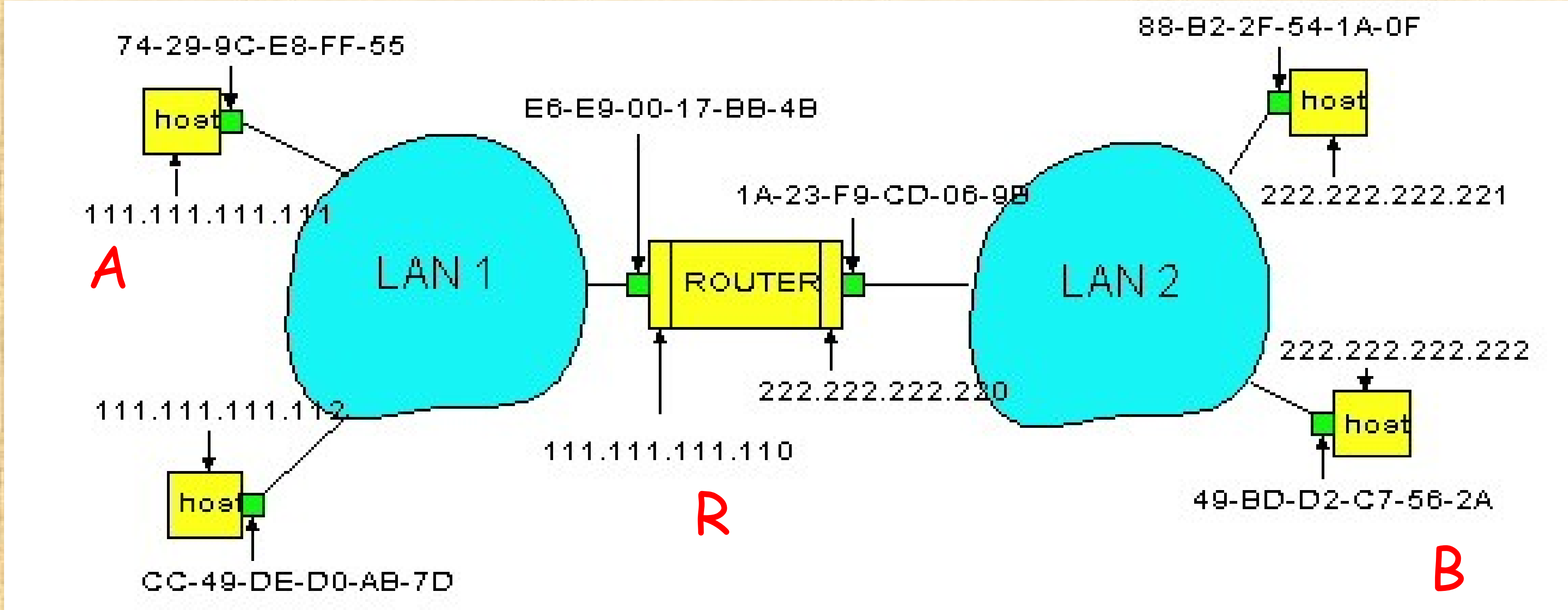
Basic Steps

- Host A must learn the IP address of B via DNS
- Host A uses gateway R to reach external hosts
- Host A sends the frame to R's MAC address
- Router R forwards IP packet to outgoing interface
- Router R learns B's MAC address and forwards frame



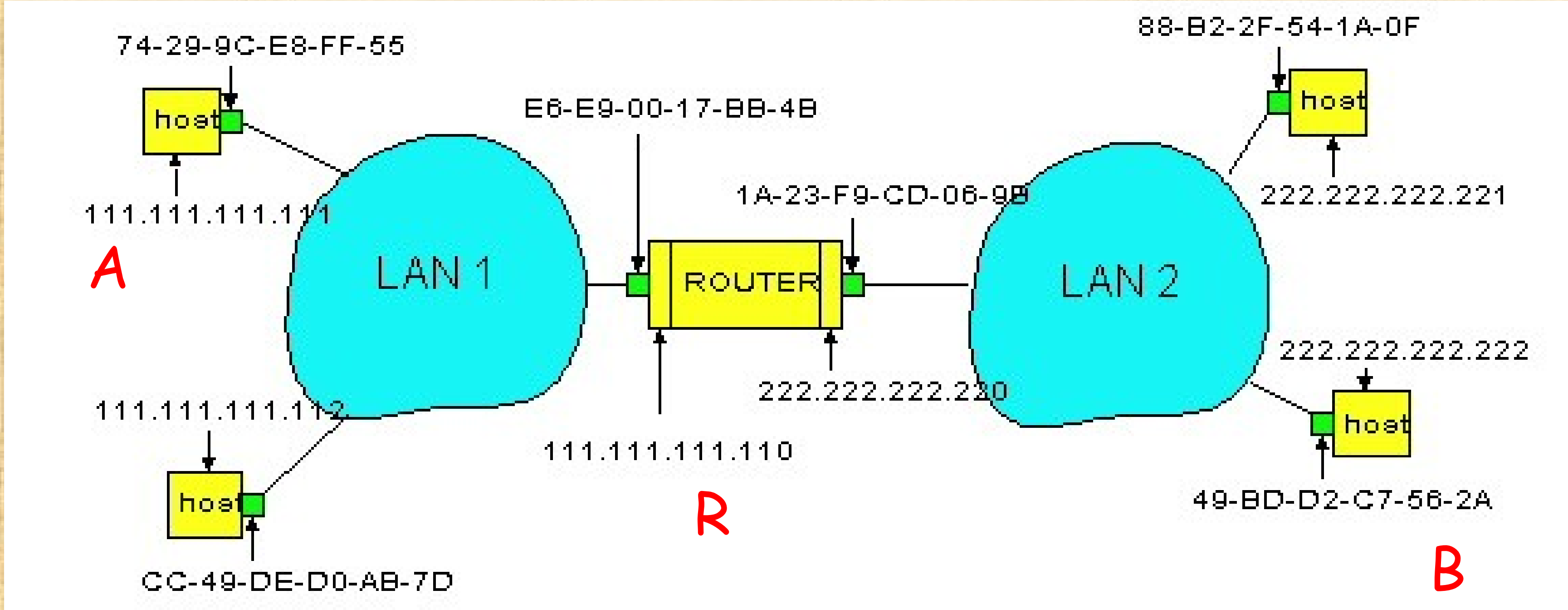
Host A Learns the IP Address of B

- Host A does a DNS query to learn B's address
  - Suppose gethostbyname() returns 222.222.222.222
- Host A constructs an IP packet to send to B
  - Source 111.111.111.111, destination 222.222.222.222



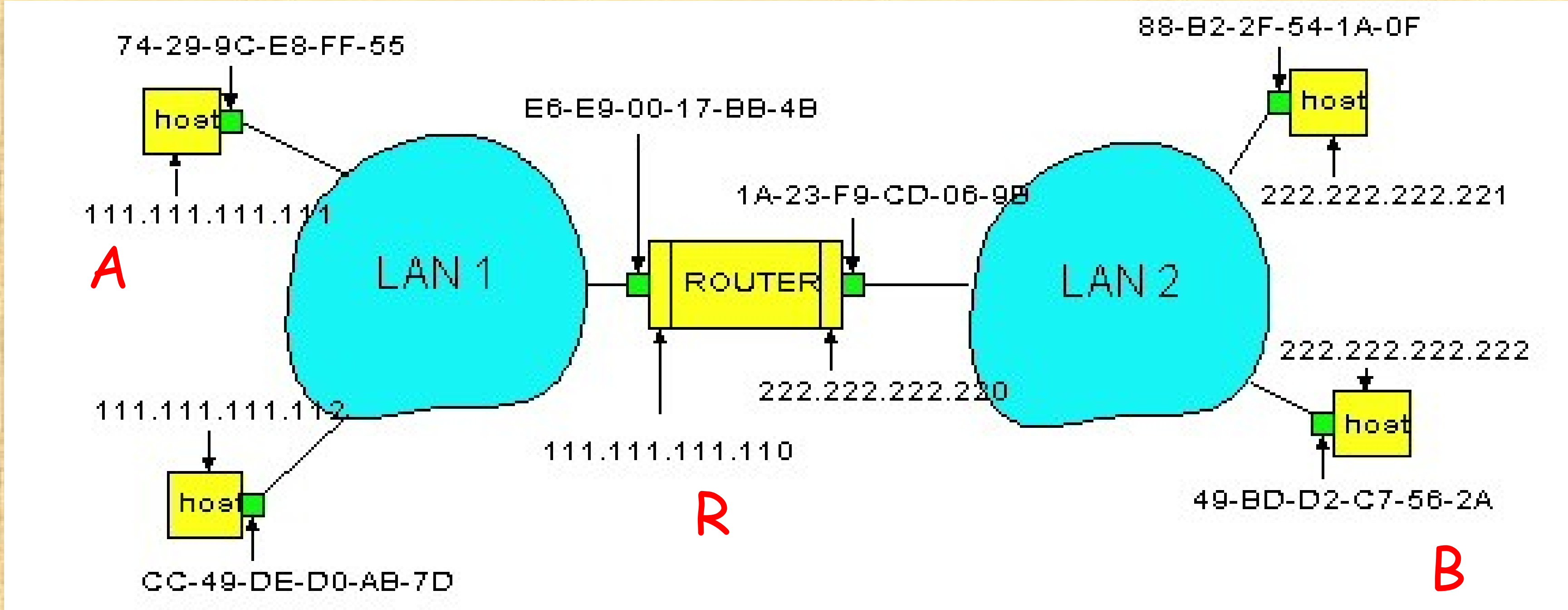
Host A Learns the IP Address of B

- IP header
  - From A: 111.111.111.111
  - To B: 222.222.222.222
- Ethernet frame
  - From A: 74-29-9C-E8-FF-55
  - To gateway: ????



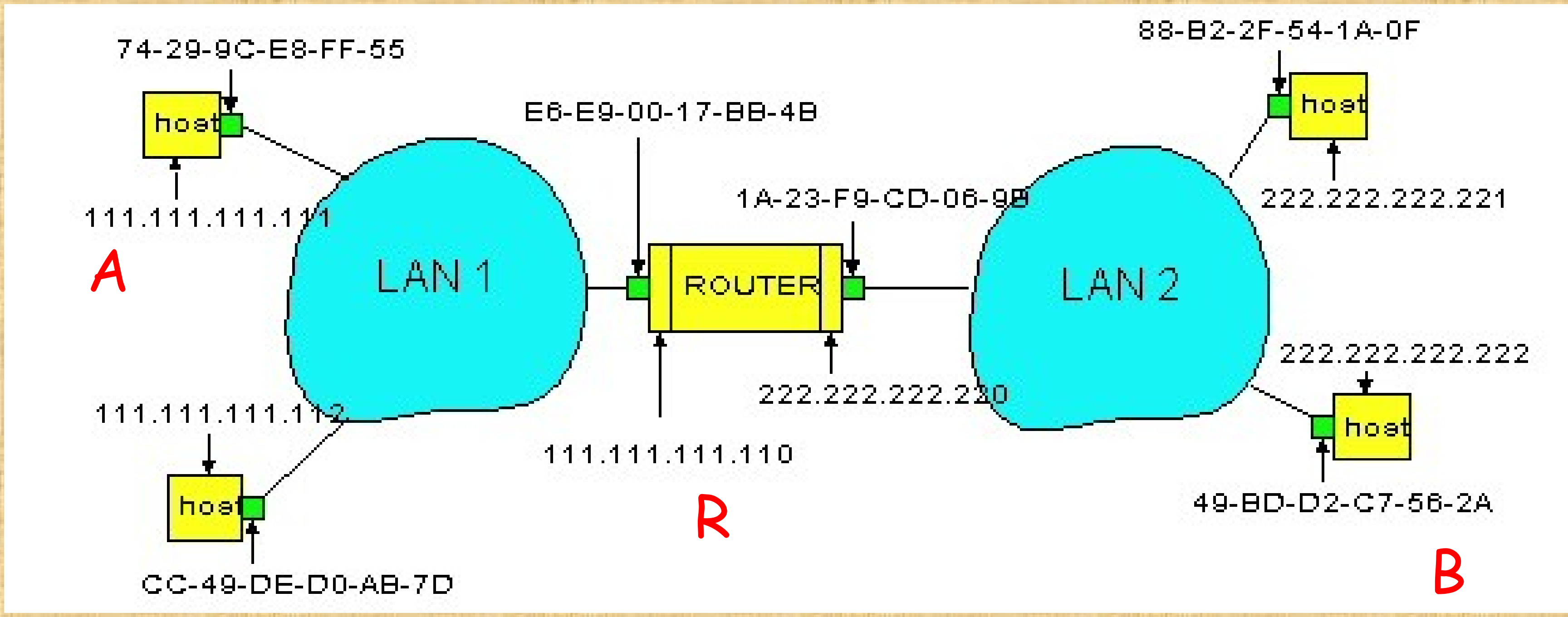
Host A Decides to Send Through R

- Host A has a gateway router R
  - Used to reach destinations outside of 111.111.111.0/24
  - Address 111.111.111.110 for R learned via DHCP
- But, what is the MAC address of the gateway?



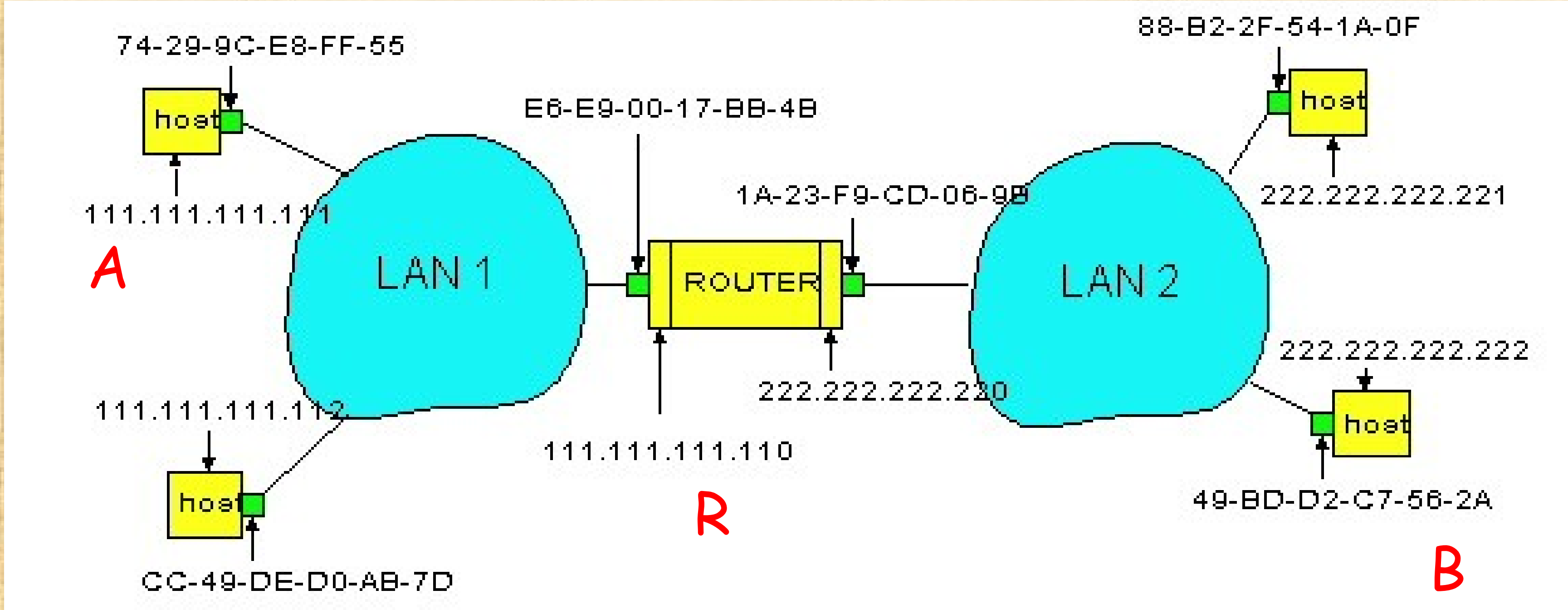
Host A Sends Packet Through R

- Host A learns the MAC address of R's interface
  - ARP request: broadcast request for 111.111.111.110
  - ARP response: R responds with E6-E9-00-17-BB-4B
- Host A encapsulates the packet and sends to R



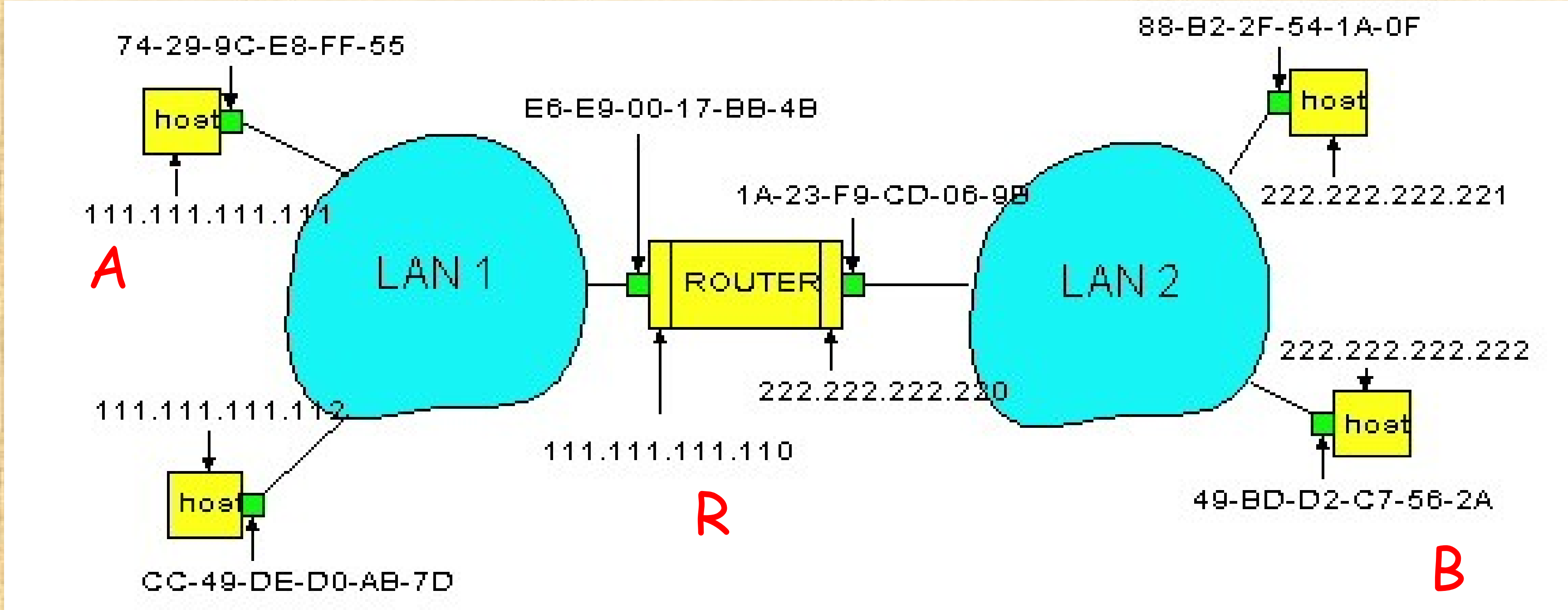
Host A Sends Packet Through R

- IP header
  - From A: 111.111.111.111
  - To B: 222.222.222.222
- Ethernet frame
  - From A: 74-29-9C-E8-FF-55
  - To R: E6-E9-00-17-BB-4B



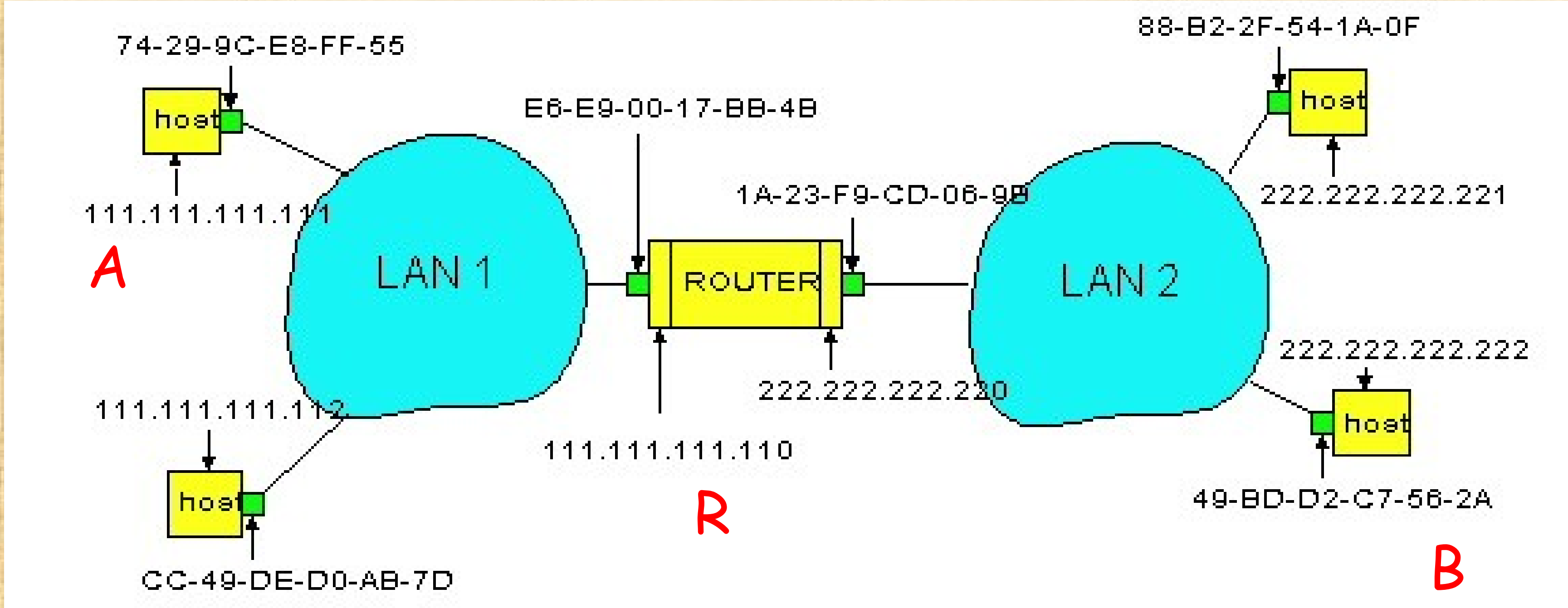
R Decides how to Forward Packet

- Router R's adapter receives the packet
  - R extracts the IP packet from the Ethernet frame
  - R sees the IP packet is destined to 222.222.222.222
- Router R consults its forwarding table
  - Packet matches 222.222.222.0/24 via other adapter



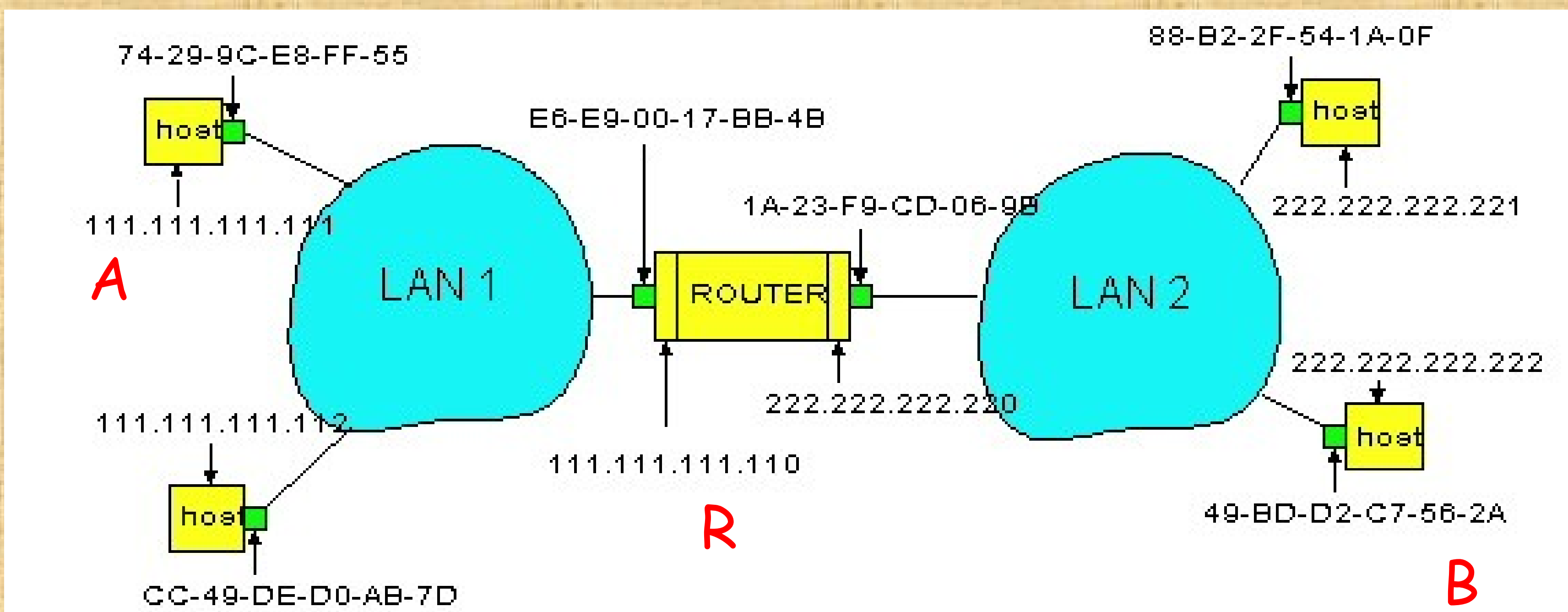
Router R Wants to Forward Packet

- IP header
  - From A: 111.111.111.111
  - To B: 222.222.222.222
- Ethernet frame
  - From R: 1A-23-F9-CD-06-9B
  - To B: ???



R Sends Packet to B

- Router R's learns the MAC address of host B
  - ARP request: broadcast request for 222.222.222.222
  - ARP response: B responds with 49-BD-D2-C7-56-2A
- Router R encapsulates the packet and sends to B



Router R Wants to Forward Packet

- IP header
  - From A: 111.111.111.111
  - To B: 222.222.222.222
- Ethernet frame
  - From R: 1A-23-F9-CD-06-9B
  - To B: 49-BD-D2-C7-56-2A