

Information Network

Lecture 12: Link Layer

Holger Thies

KYOTO UNIVERSITY

京都
大学



Schedule

- Three more classes
 - Today
 - January 8
 - January 22
- Homework 5 due January 7
- Short Test in class on January 22
- Final report due January 24

Network-layer functions

Recall: two network-layer functions:

- *forwarding*: move packets from router's input to appropriate router output *data plane*
- *routing*: determine route taken by packets from source to destination *control plane*

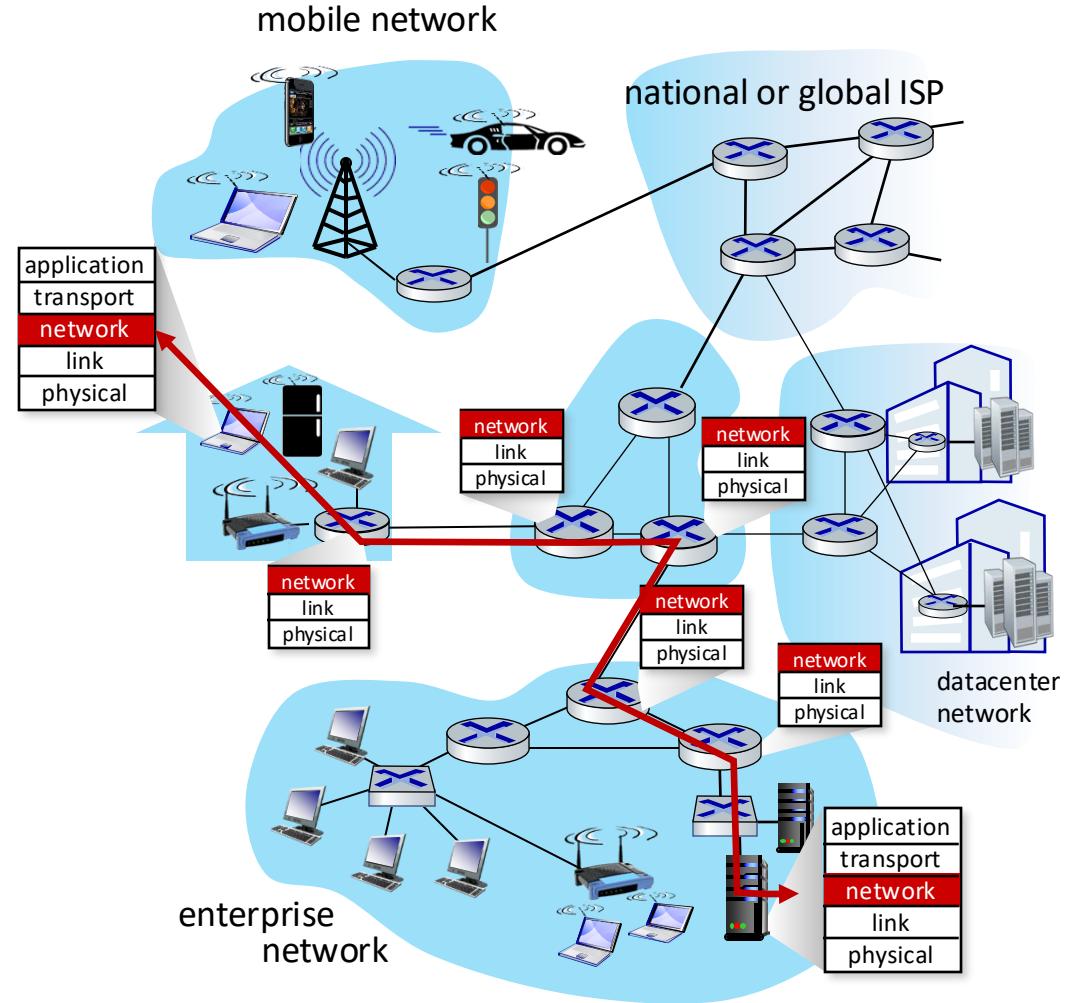
Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

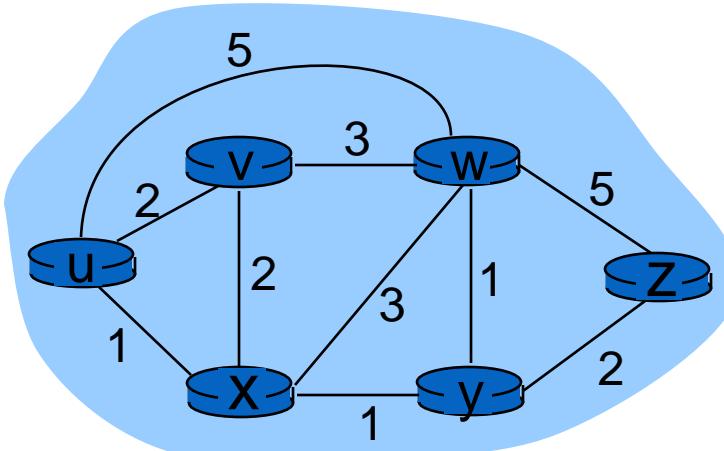
Routing protocols

Routing protocol goal: determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- **path:** sequence of routers packets traverse from given initial source host to final destination host
- **“good”:** least “cost”, “fastest”, “least congested”



Graph abstraction of the network

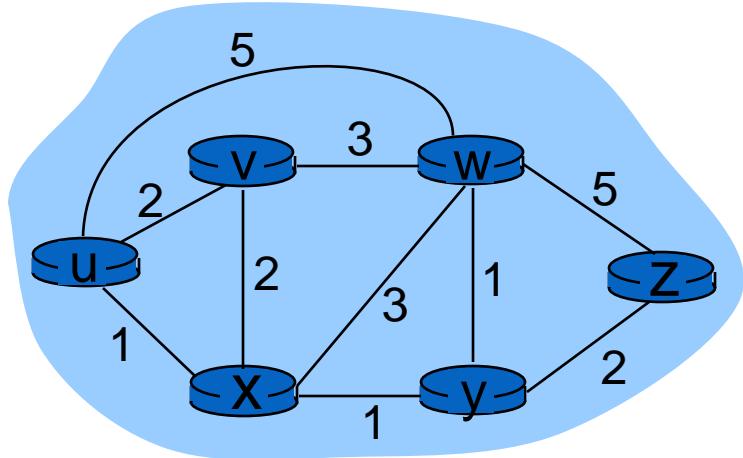


graph: $G = (N, E)$

$N = \text{set of routers} = \{ u, v, w, x, y, z \}$

$E = \text{set of links} = \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Graph abstraction: costs



$c(x,x')$ = cost of link (x,x')
e.g., $c(w,z) = 5$

cost could always be 1, or
inversely related to bandwidth,
or inversely related to
congestion

$$\text{cost of path } (x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$$

key question: what is the least-cost path between u and z ?

routing algorithm: algorithm that finds that least cost path

Sometimes least-cost path is also called **shortest path**.

Classification of routing algorithms

Centralized routing algorithm:

- Computes the least-cost path between source and destination using complete, global knowledge about the network.
- The algorithm knows all the nodes and edges of the graphs, and each of the edge costs.
- Often called link-state algorithm because the algorithm must be aware of the costs of each of the links in the network.
- Example: Dijkstra's Algorithm

Classification of routing algorithms

Decentralized routing algorithm:

- Every router only has information about its neighbors.
- Can exchange information with neighbors.
- Calculation done in an iterative, distributed manner.
- Each router starts with only information about the cost to its direct neighbors.
- By receiving information from its neighbors, the router gradually calculates the least-cost path to other destinations.
- Example: Distance vector algorithm/Bellman-Ford Algorithm

Classification of routing algorithms

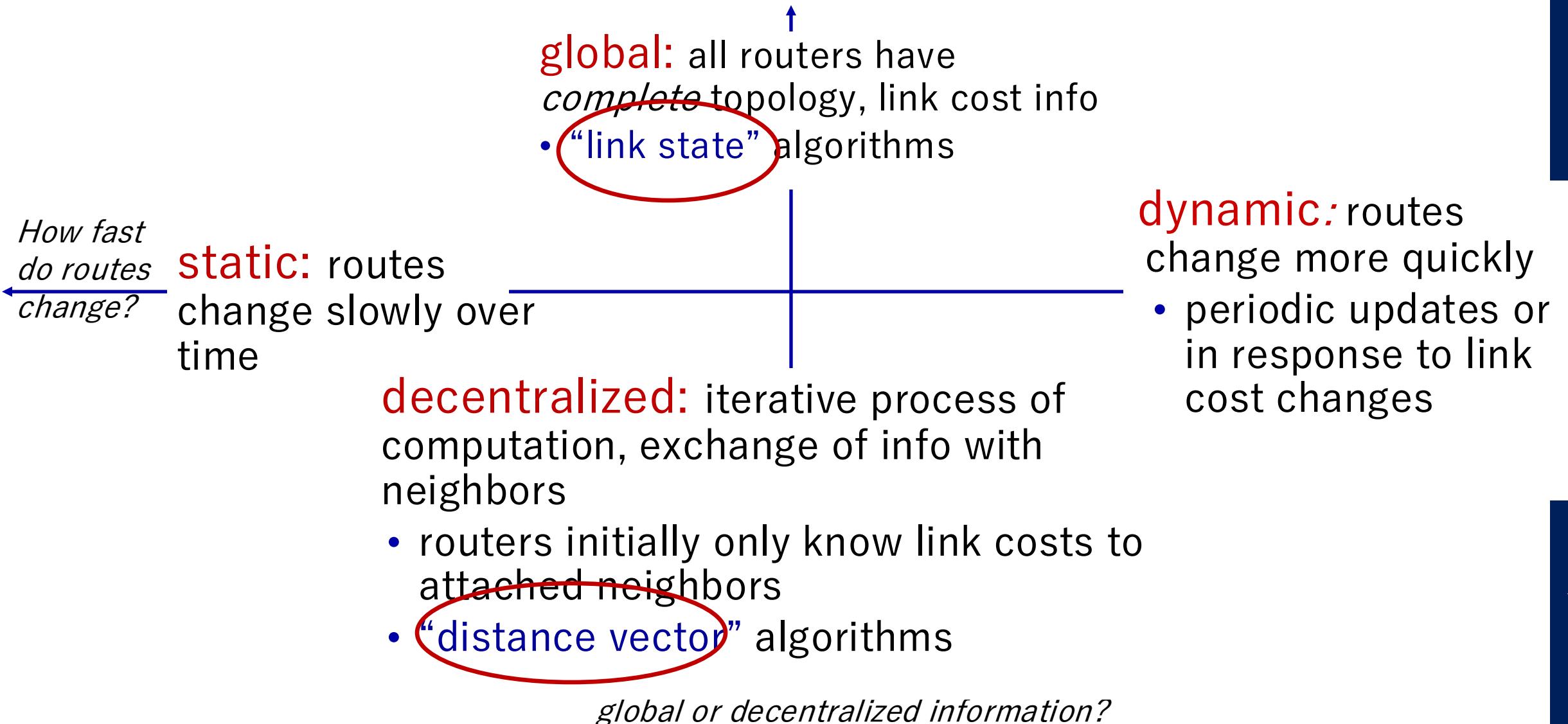
Static routing algorithms:

- Routes usually do not change or only change very slowly over time.
- Routing tables are only updated if requested manually.

Dynamic routing algorithms:

- Routes can change dynamically, for example to adapt to traffic or changes in the network.
- More responsive to network changes but also more susceptible to problems and needs more complex algorithm.

Routing algorithm classification



Today's lecture

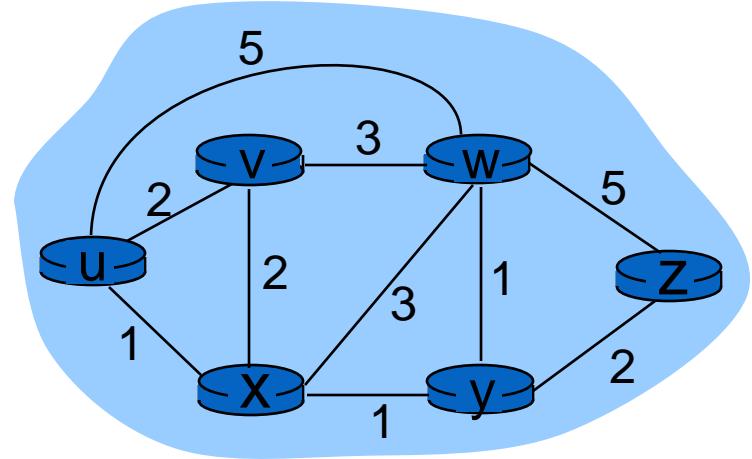
5.1 introduction

5.2 routing protocols

- link state
- distance vector

Link state algorithm

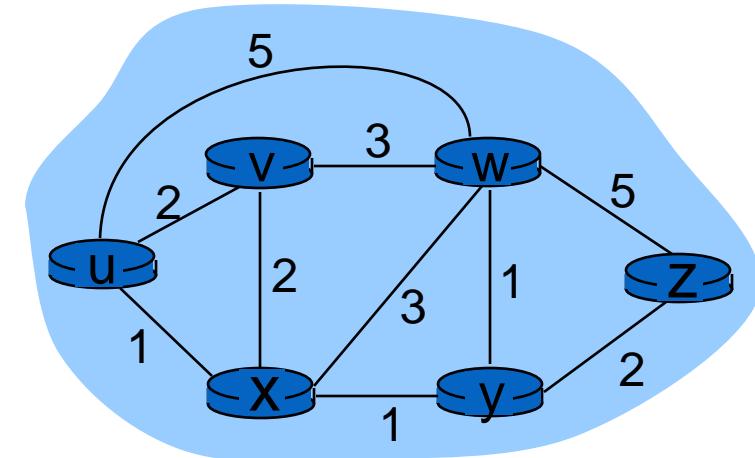
- In a link state algorithm, the algorithm knows the network topology and all link costs.
- In practice, each node has to broadcast link-state packets to all other nodes in the network.
- A link-state packet contains the identities of the node's neighbors and the costs to each neighbor.
- After receiving the packets, all nodes have an identical and complete view of the network.
- Each node can then run the link-state algorithm to find the least-cost path to any other node.



Dijkstra's algorithm

- Dijkstra's algorithm computes the least cost to go from a source node s to each of the other nodes.
- For example, if the source node is u , it computes the table below:

destination	cost	previous node
v	2	u
x	1	u
y	2	x
w	3	y
z	4	y



A link-state routing algorithm

Network topology, link costs known to all nodes

- accomplished via “link state broadcast”
- all nodes have same info
- computes least cost paths from one node (“source”) to all other nodes
 - gives *forwarding table* for that node

Today's lecture

5.1 introduction

5.2 routing protocols

- link state
- distance vector

Distance vector algorithm

- The Link-state algorithm uses global information.
- We will next discuss the distance vector algorithm which is **iterative**, **asynchronous** and **distributed**.
- Distributed: Each node receives information from its neighbors and then distributes it back to its neighbors.
- Iterative: The process continues until no more new information is available.
- Asynchronous: not all nodes need to operate at the same time.

Distance vector algorithm

- Each router has knowledge about the network and shares it through the entire network.
- Each router sends its knowledge to its direct neighbors.
- The router uses the information it gets from its neighbors to update its own routing table.
- The information is shared with the neighbors at regular interval, for example, every 30 seconds.

Distance vector algorithm

iterative, asynchronous: each

local iteration caused by:

- local link cost change
- DV 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 msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

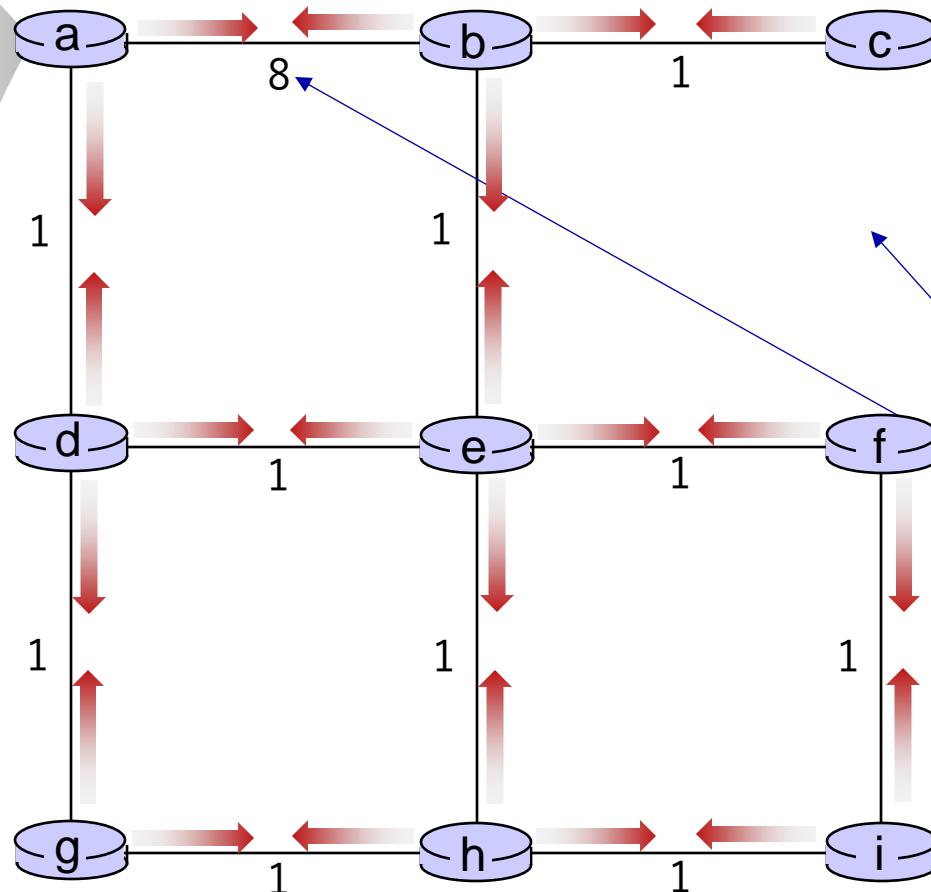
Distance vector: example



$t=0$

- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

DV in a:
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



- A few asymmetries:
- missing link
 - larger cost

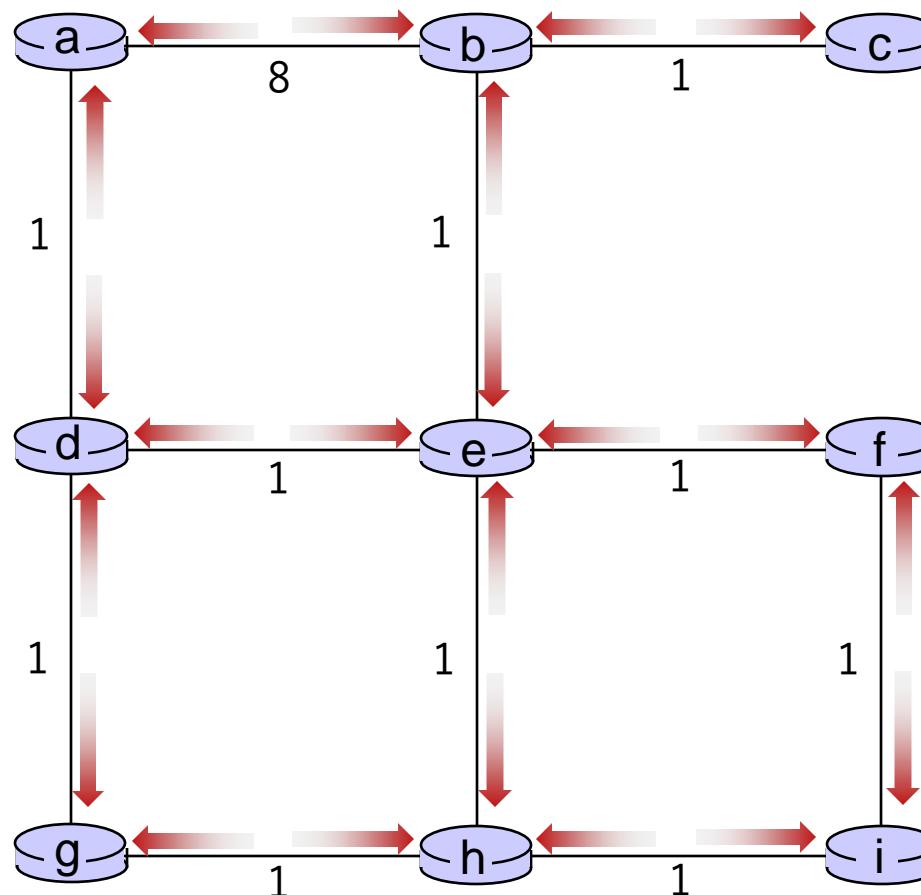
Distance vector example: iteration



$t=1$

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



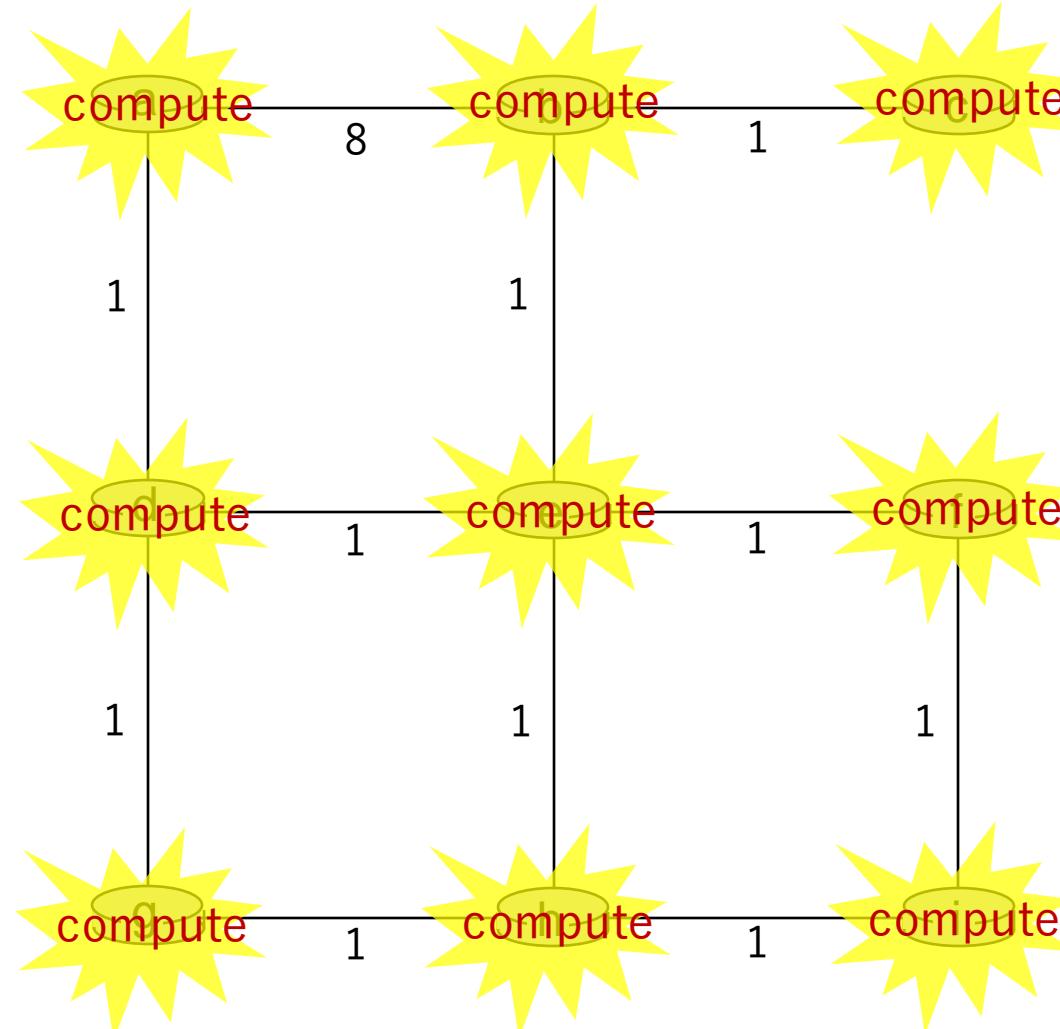
Distance vector example: iteration



$t=1$

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



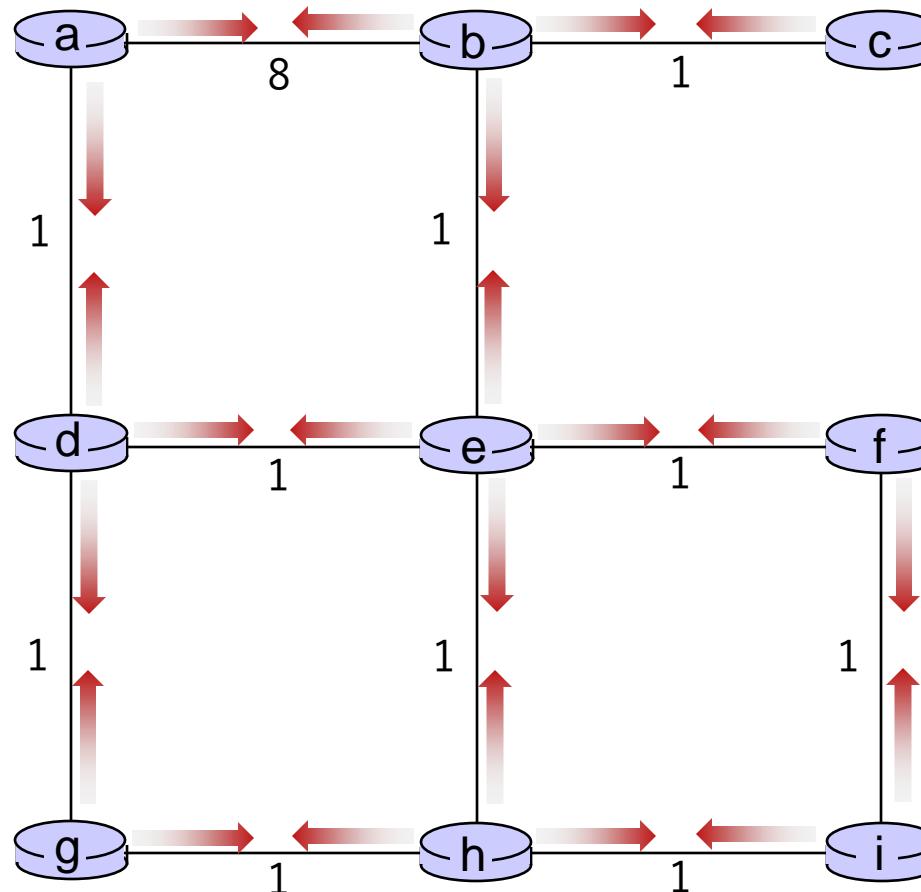
Distance vector example: iteration



$t=1$

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



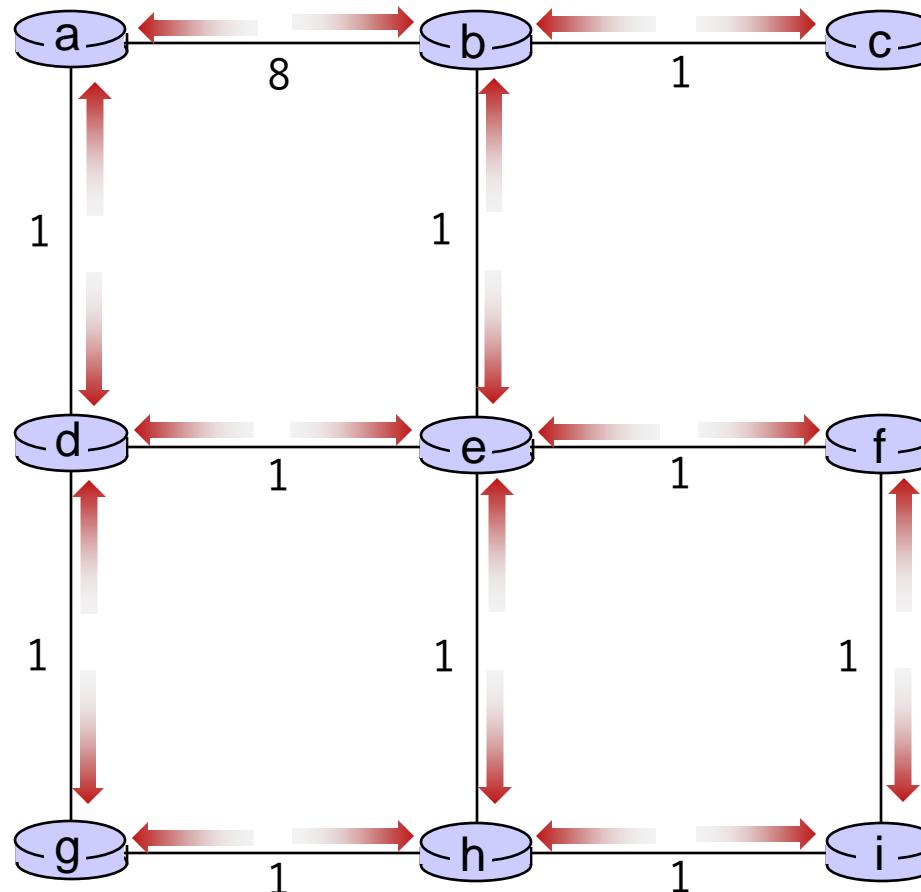
Distance vector example: iteration



$t=2$

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



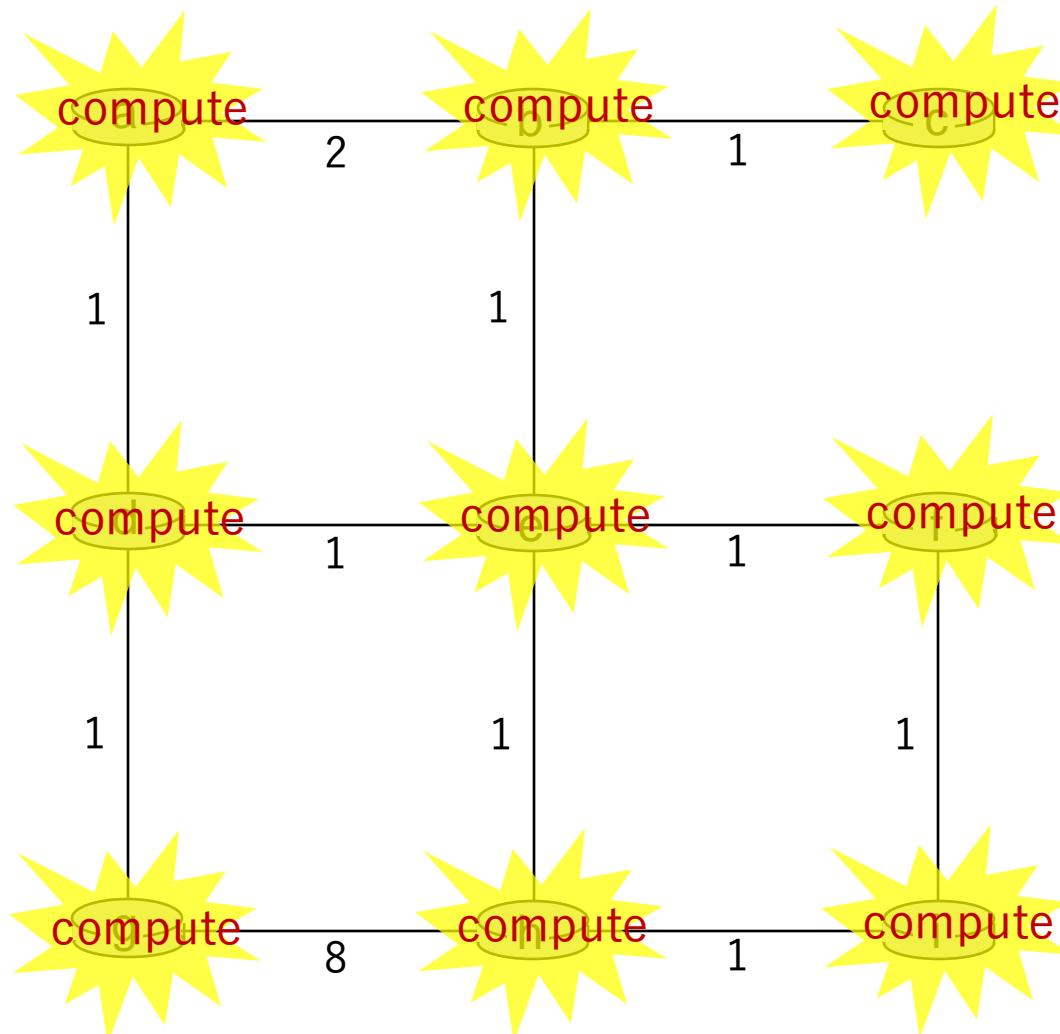
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
 - compute their new local distance vector
 - send their new local distance vector to neighbors



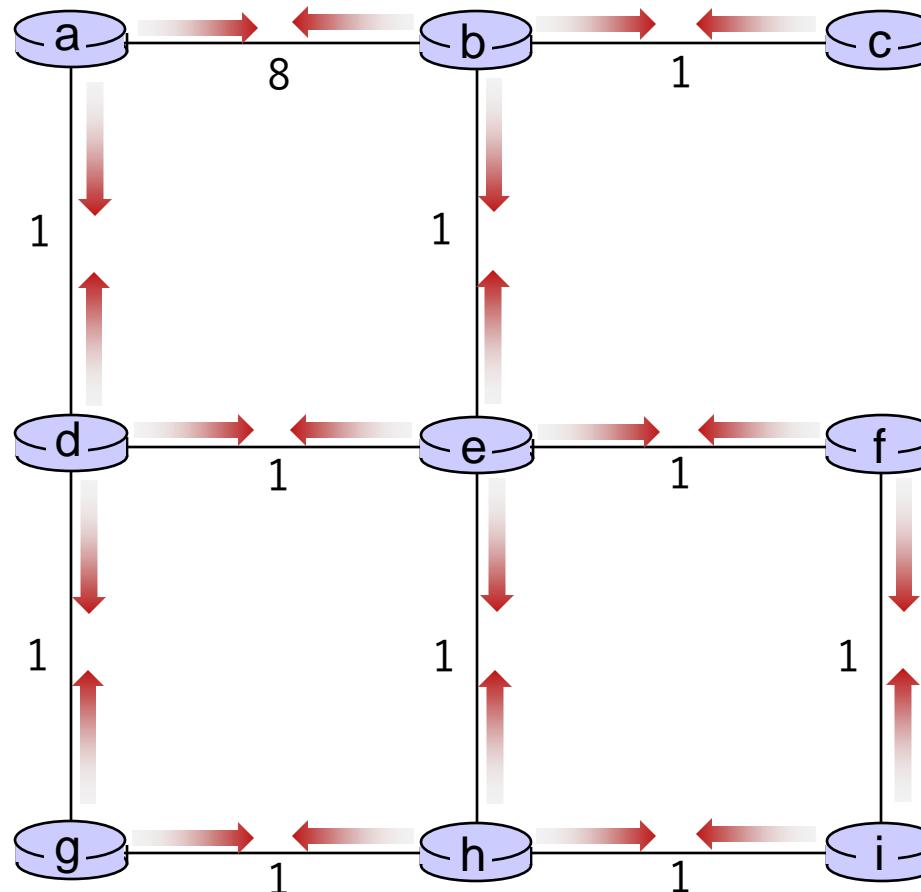
Distance vector example: iteration



$t=2$

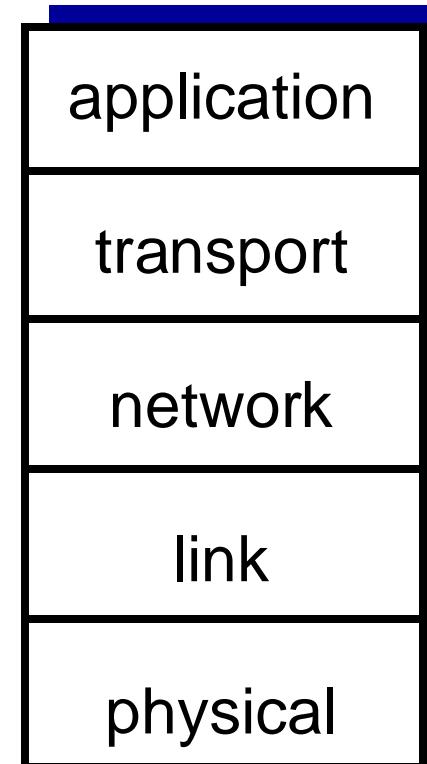
All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



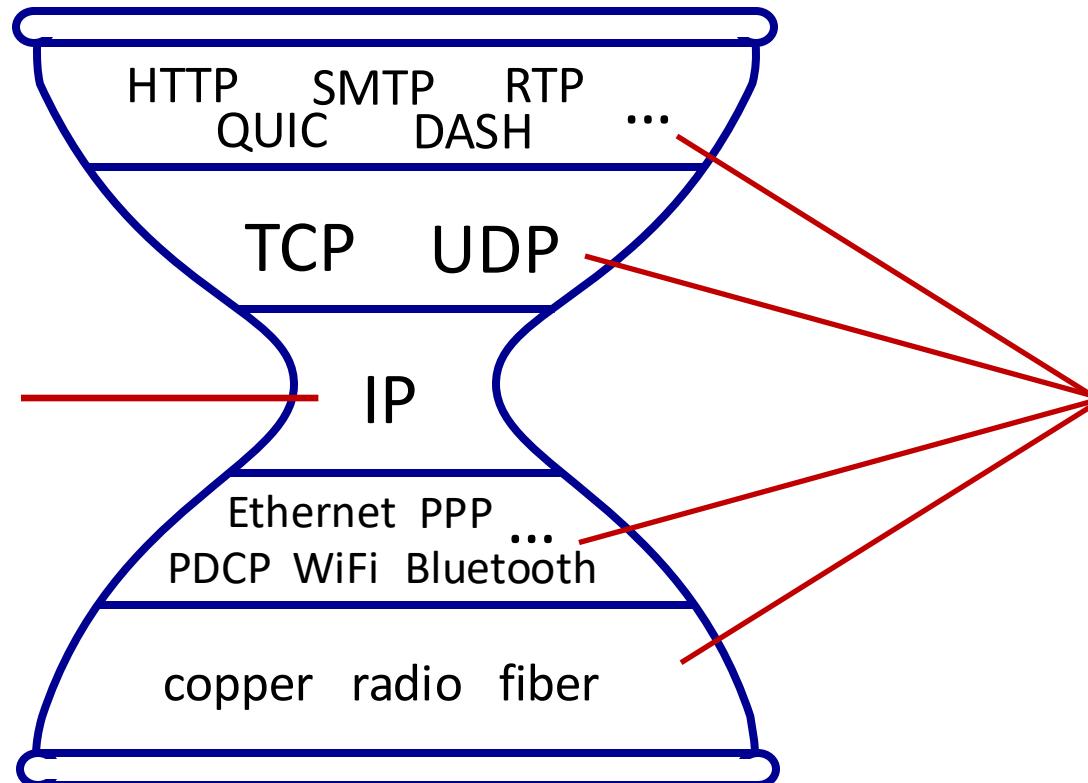
Internet protocol stack

- *application*: supporting network applications
 - FTP, SMTP, HTTP
- *transport*: process-process data transfer
 - TCP, UDP
- *network*: routing of datagrams from source to destination
 - IP, routing protocols
- *link*: data transfer between neighboring network elements
 - Ethernet, 802.111 (WiFi)
- *physical*: bits “on the wire”



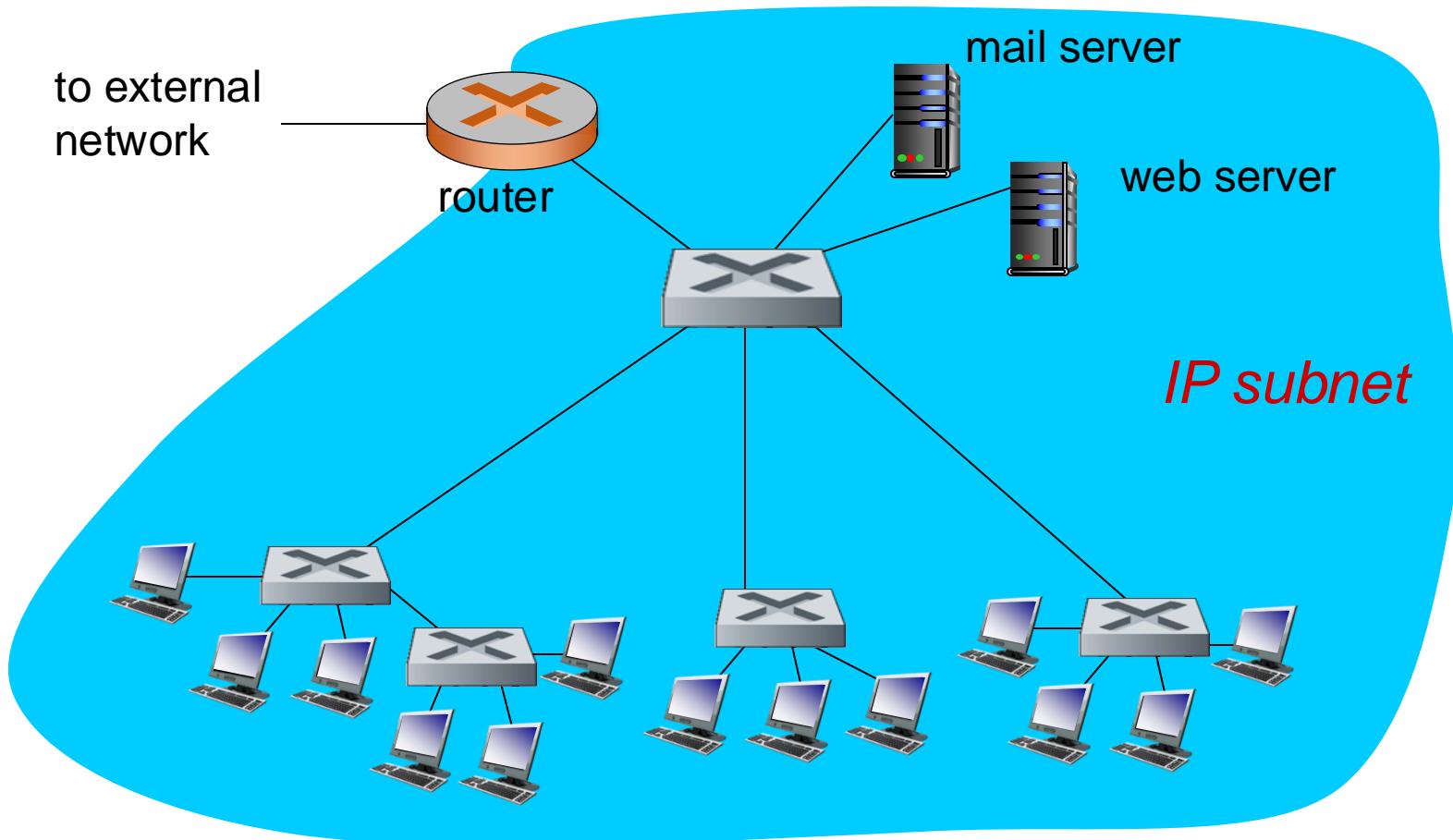
The IP hourglass

- Internet's "thin waist":
- *one* network layer protocol: IP
 - *must* be implemented by every (billions) of Internet-connected devices



many protocols
in physical, link,
transport, and
application
layers

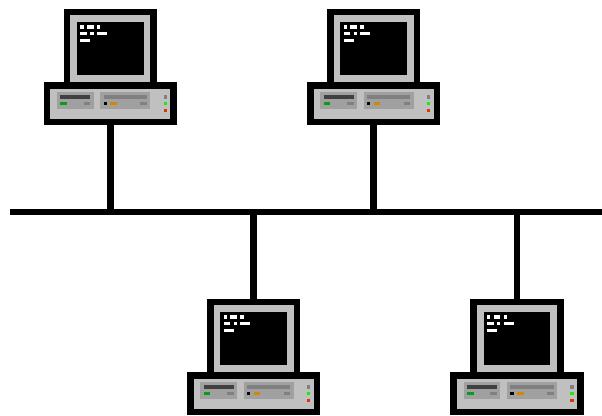
Institutional network



Hubs and Switches

How are multiple devices connected in the same subnet?

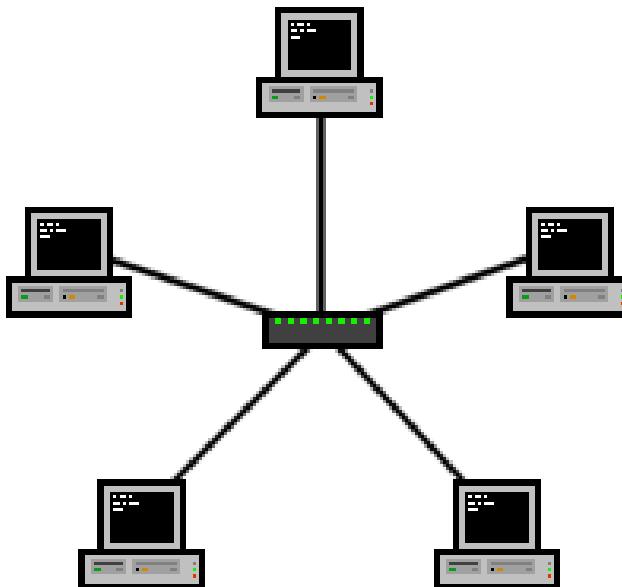
- **Hubs** operate on the physical layer. Each bit is simply recreated, and a copy is sent to all other interfaces



Hubs and Switches

How are multiple devices connected in the same subnet?

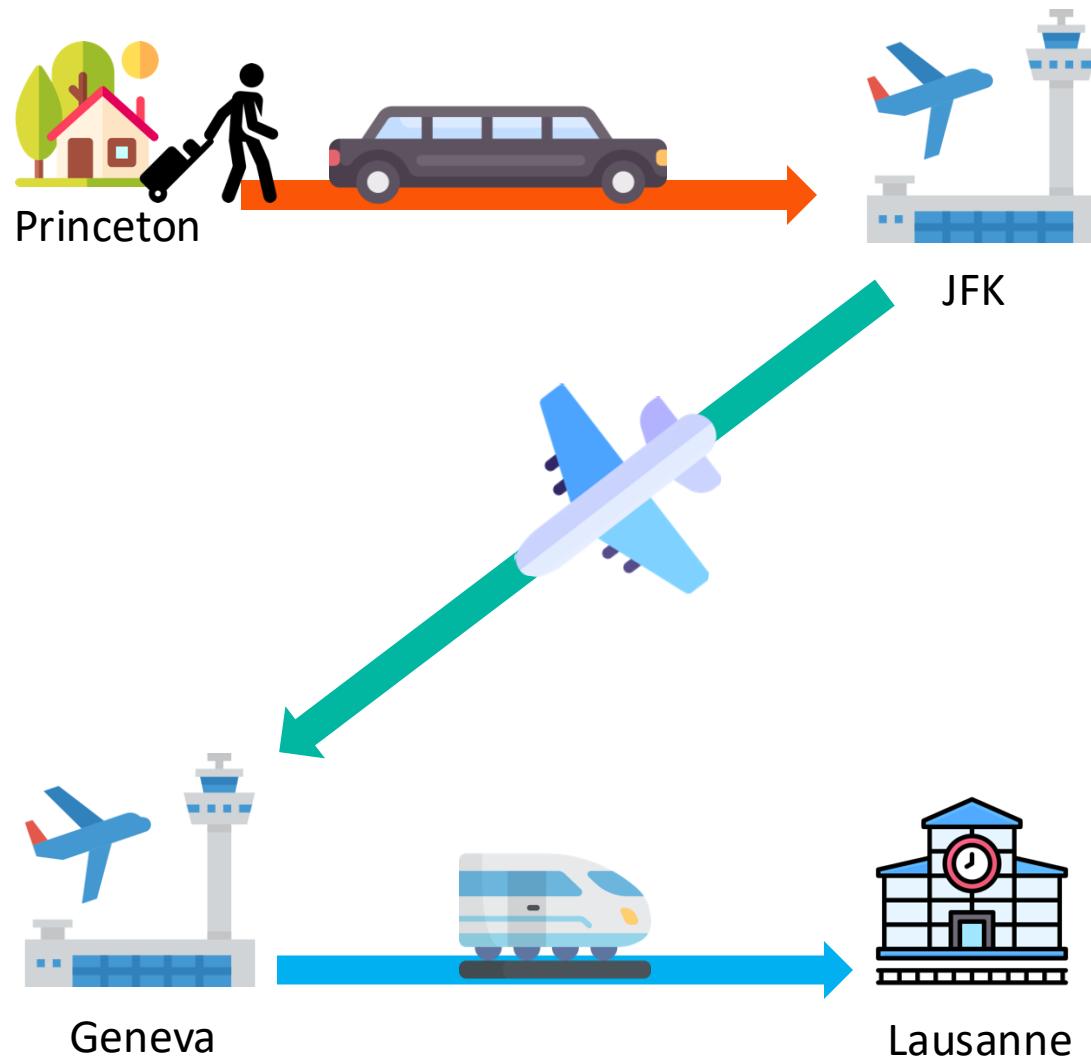
- **Switches** operate on the link layer. A switch can establish connections to send messages only to selected destinations.



Link layer

- The network layer provides a communication service between any two hosts
- Between the hosts, there are a series of communication links, some wired and some wireless, starting at the source, passing through a series of switches and routers and ending at the destination.
- We will next consider how packets are sent through the individual links on the communication path

Transportation analogy



transportation analogy:

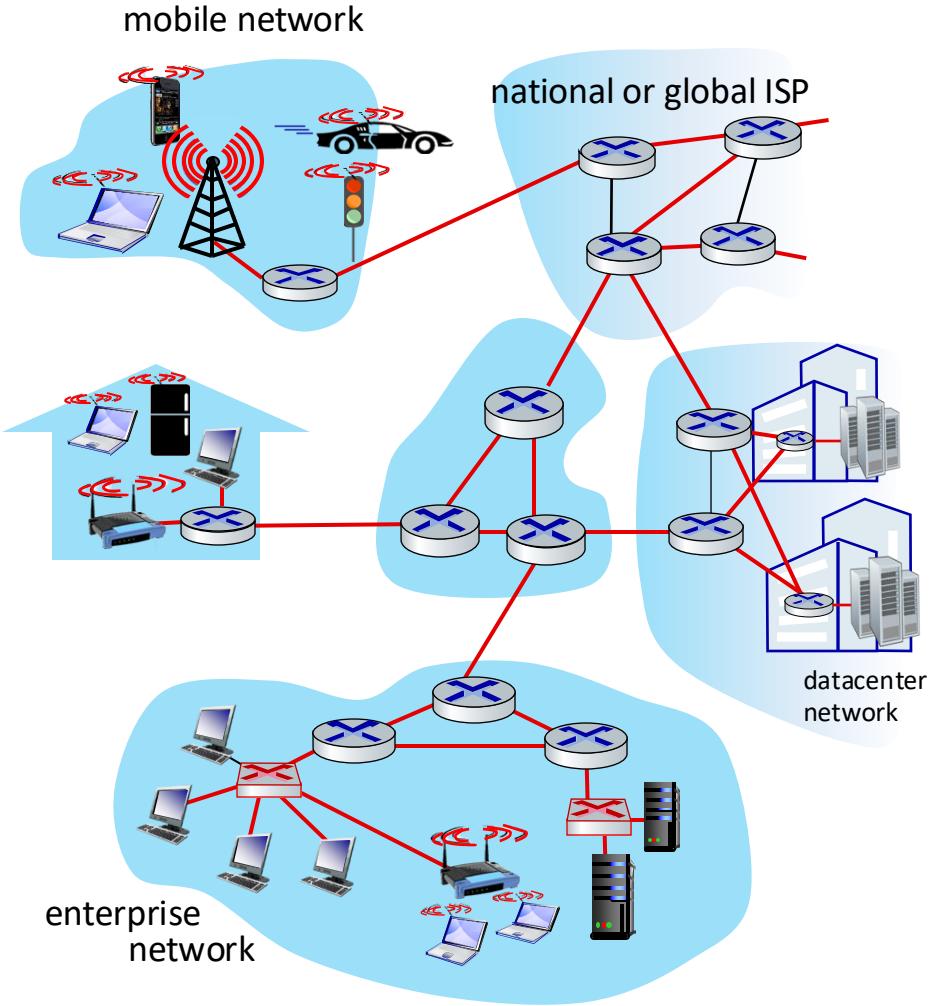
- trip from Princeton to Lausanne
 - limo: Princeton to JFK
 - plane: JFK to Geneva
 - train: Geneva to Lausanne
- tourist = **datagram**
- transport segment = **communication link**
- transportation mode = **link-layer protocol**
- travel agent = **routing algorithm**

Link layer: introduction

terminology:

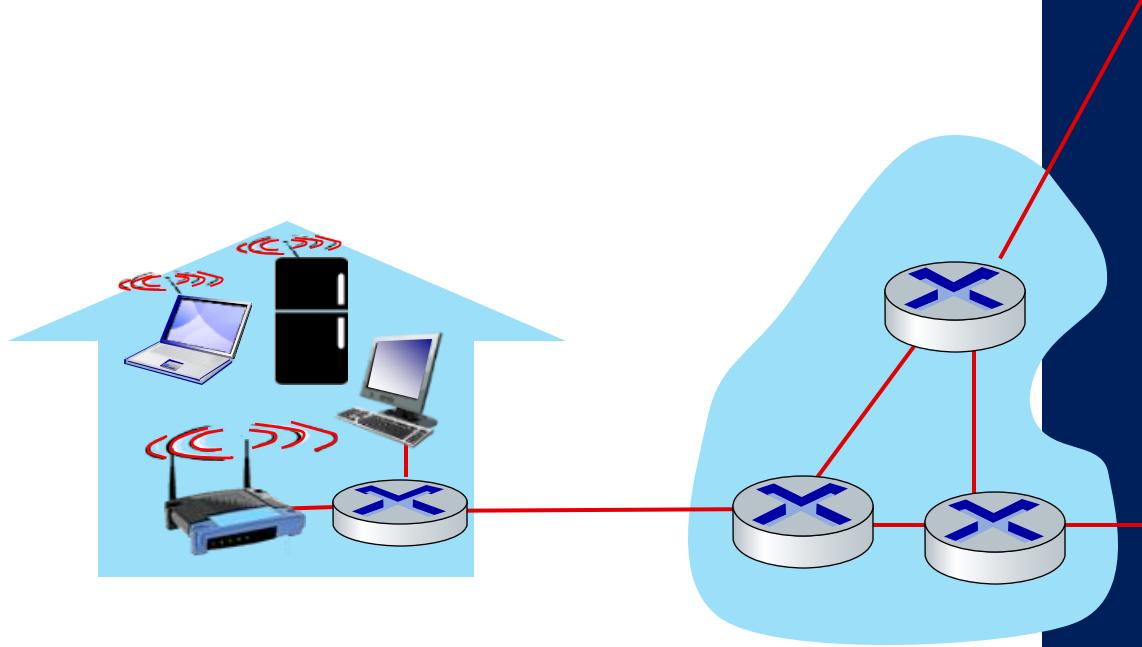
- hosts, routers: nodes
- communication channels that connect **adjacent** nodes along communication path: **links**
 - wired , wireless
- layer-2 packet: *frame*, encapsulates datagram

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



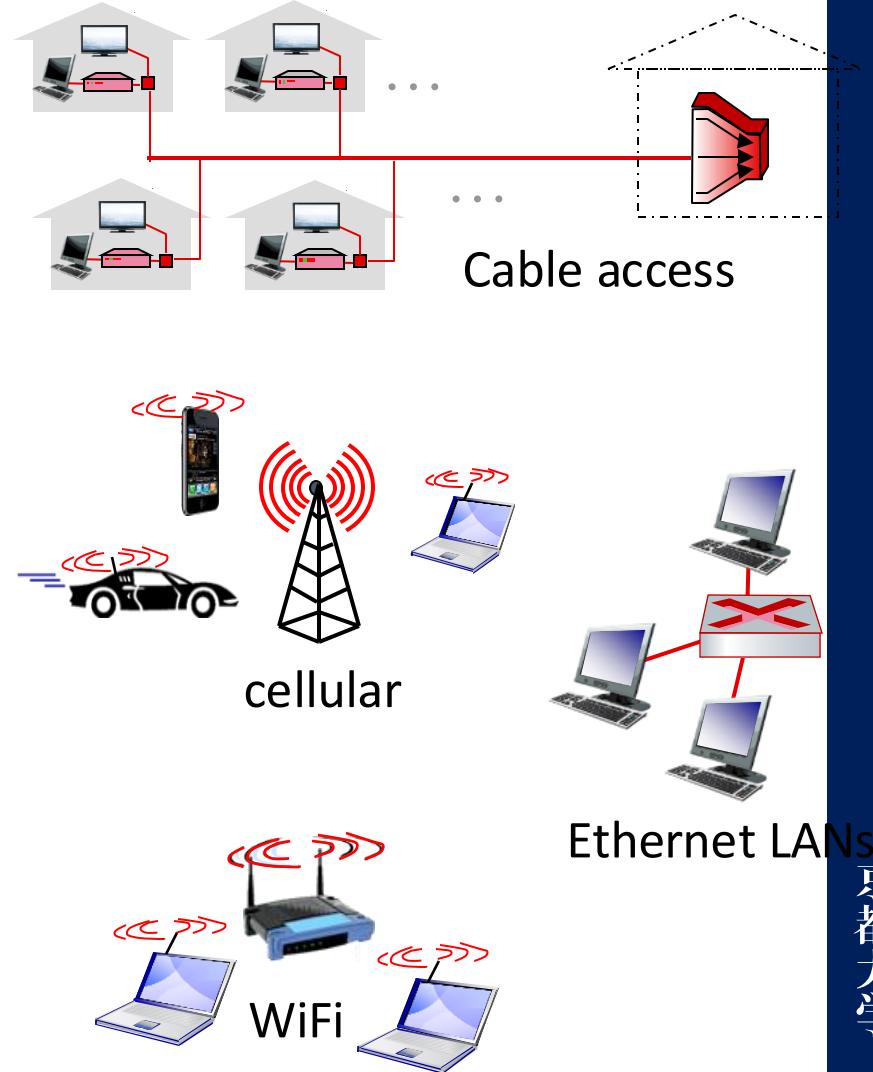
Link layer: context

- datagram transferred by **different link protocols** over different links:
 - e.g., WiFi on first link, Ethernet on next link
- each link protocol provides different services
 - e.g., **may or may not** provide reliable data transfer over link



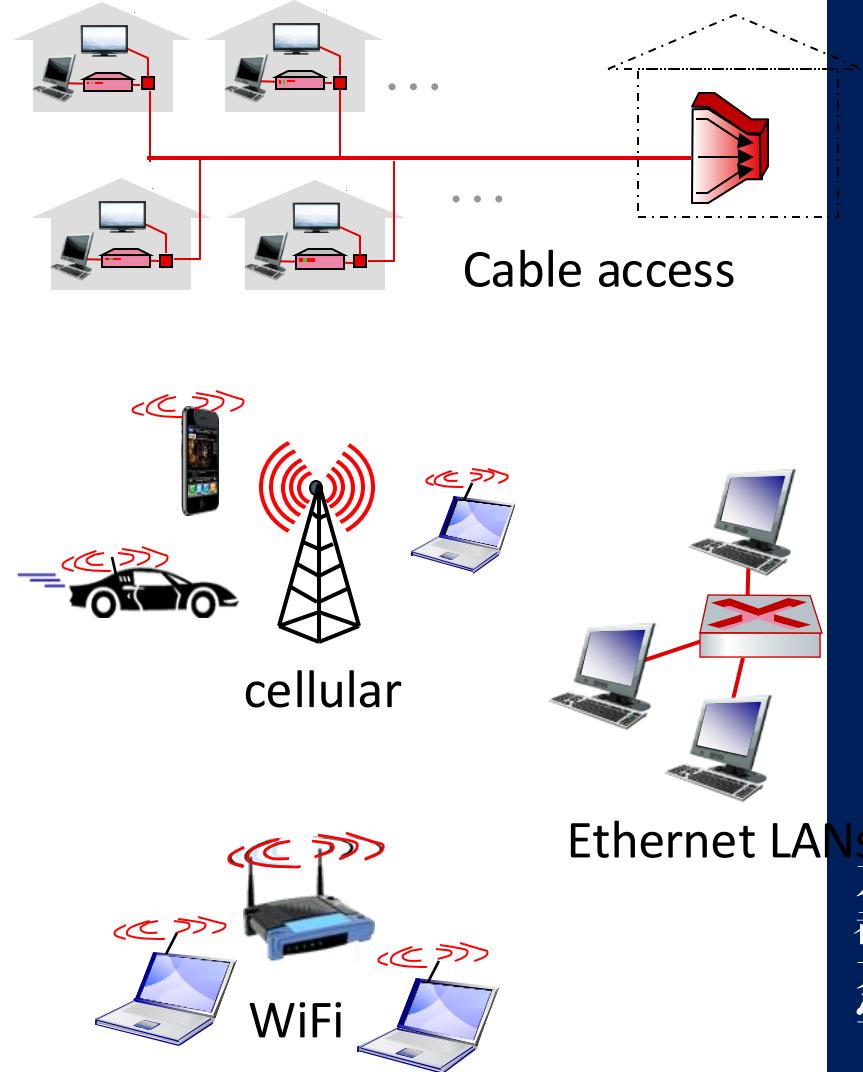
Link layer: services

- **framing, link access:**
 - encapsulate datagram into frame, adding header, trailer
 - channel access if shared medium
 - “MAC” addresses in frame headers identify source, destination (different from IP address!)
- **reliable delivery between adjacent nodes**
 - we already know how to do this!
 - seldom used on low bit-error links
 - wireless links: high error rates



Link layer: services (more)

- **flow control:**
 - pacing between adjacent sending and receiving nodes
- **error detection:**
 - errors caused by signal attenuation, noise.
 - receiver detects errors, signals retransmission, or drops frame
- **error correction:**
 - receiver identifies *and corrects* bit error(s) without retransmission
- **half-duplex and full-duplex:**
 - with half duplex, nodes at both ends of link can transmit, but not at same time



Link layer

6.1 introduction, services

6.2 multiple access
protocols

6.3 LANs

- addressing, ARP
- Ethernet
- switches

Multiple access links, protocols

two types of “links”:

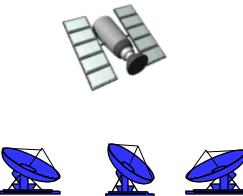
- point-to-point
 - point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
 - old-fashioned Ethernet
 - upstream HFC
 - 802.11 wireless LAN



shared wire (e.g.,
cabled Ethernet)



shared RF
(e.g., 802.11 WiFi)



shared RF
(satellite)

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

Multiple Access protocols

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

Link layer

6.1 introduction, services

6.2 multiple access
protocols

6.3 LANs

- addressing, ARP
- Ethernet
- switches

Link layer addressing

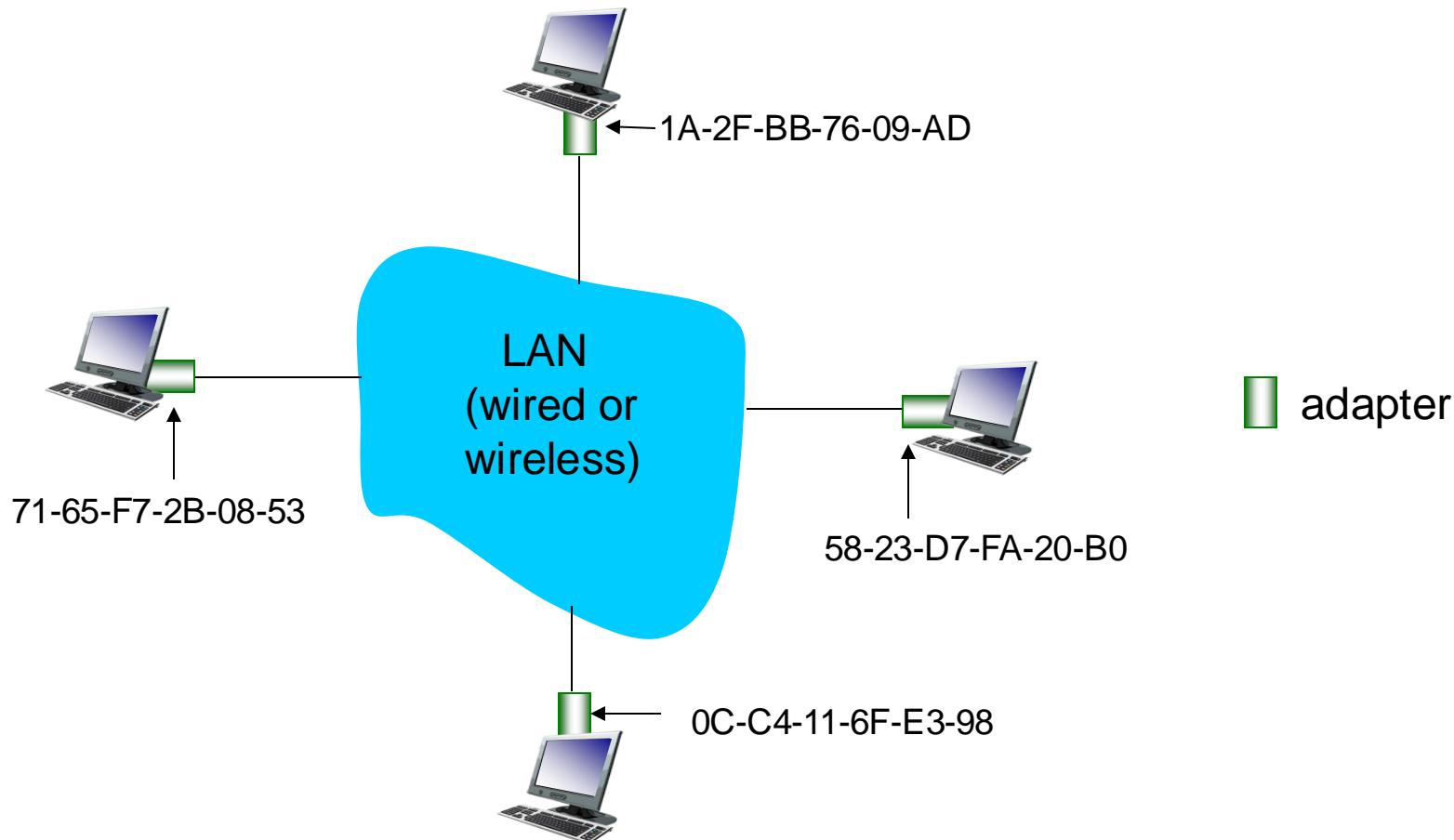
- IP Addresses are only used on the network layer. The Link layer does not know about IP addresses.
- Hosts and routers not only have network layer (IP) addresses but also link-layer addresses, which are very different from IP addresses.
- Link-layer addresses correspond to the adapters (network interfaces) instead of the host or router itself.
- A host or router with multiple interfaces will have multiple link-layer addresses.
- The link layer address is called LAN address, physical address or media access control (MAC) address.

MAC addresses and ARP

- 32-bit IP address:
 - *network-layer* address for interface
 - used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
 - function: *used ‘locally’ to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
 - 48 bit MAC address (for most LANs) typically burned into network device hardware
 - Designed to be permanent, no two adapters have the same address.
 - e.g.: IA-2F-BB-76-09-AD
 - hexadecimal (base 16) notation
 - (each “numeral” represents 4 bits)

MAC addresses and ARP

each adapter on LAN has unique **MAC** address



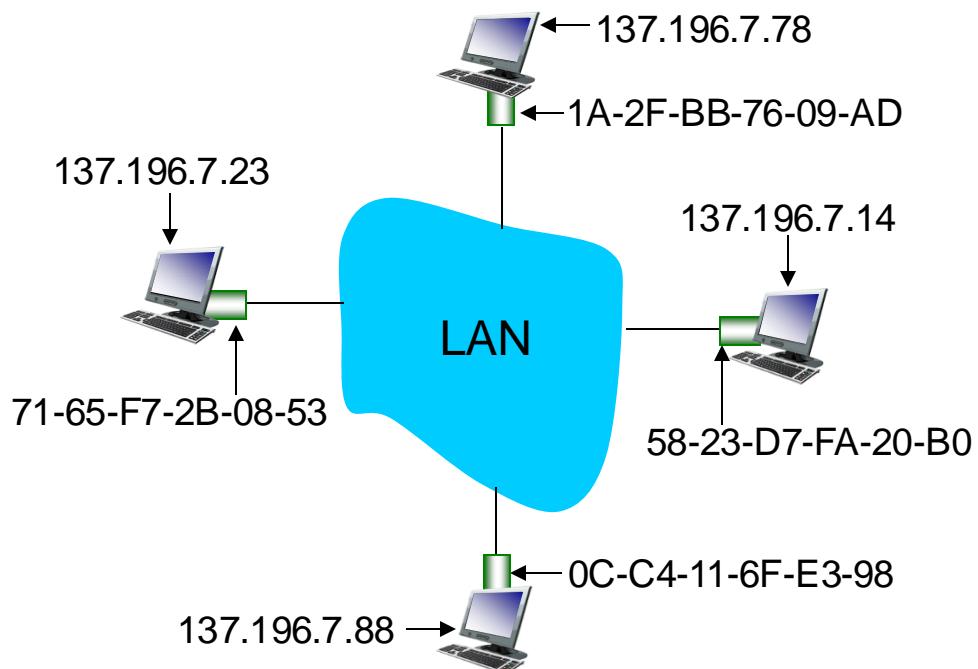
MAC addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- MAC flat address → portability
 - can move LAN card from one LAN to another
 - MAC address: like “My Number”
 - IP address: like postal address
- IP hierarchical address *not* portable
 - address depends on IP subnet to which node is attached

ARP: address resolution protocol



Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has 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: same LAN

- A wants to send datagram to B
 - B's MAC address not in A's ARP table.
- A **broadcasts** ARP query packet, containing B's IP address
 - destination MAC address = FF-FF-FF-FF-FF-FF
 - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
 - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
 - information that times out (goes away) unless refreshed
 - nodes create their ARP tables *without intervention from net administrator*

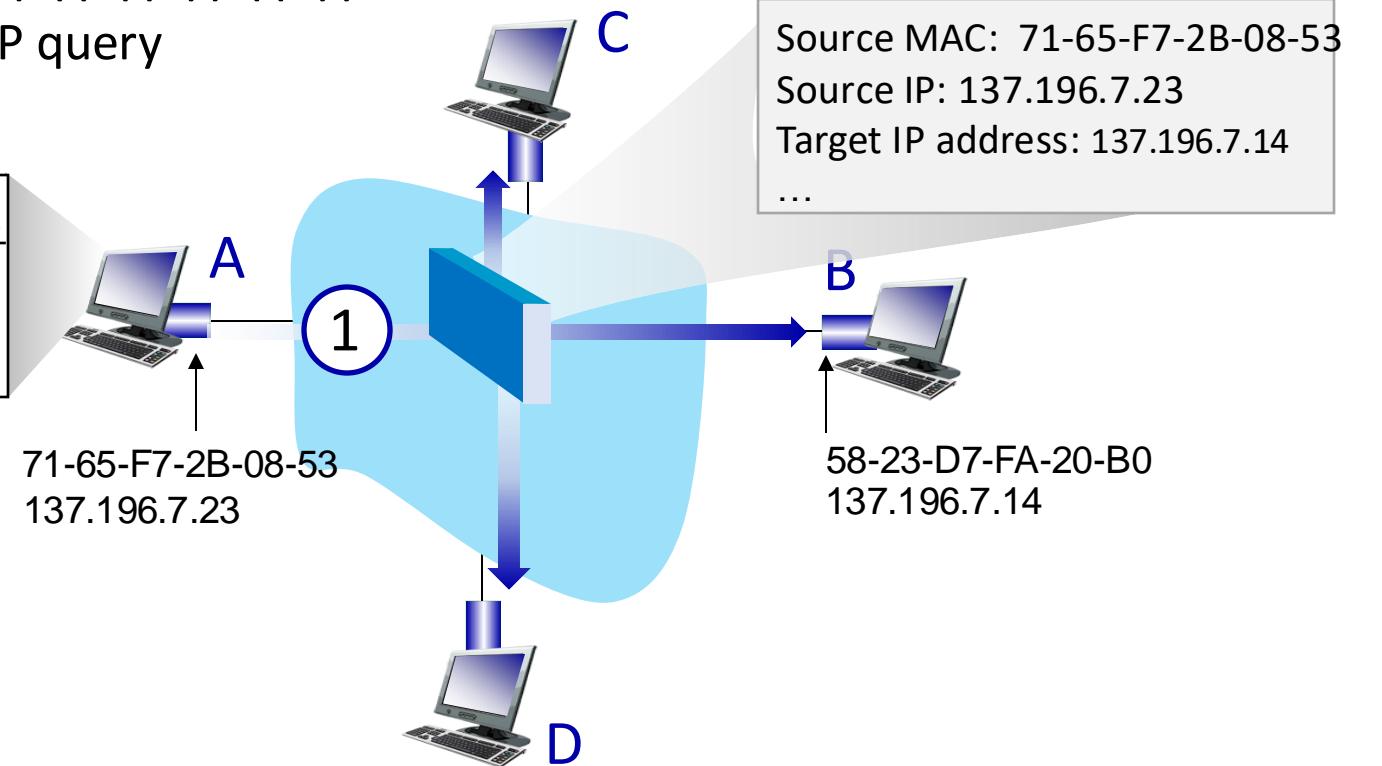
ARP protocol in action

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

- ① A broadcasts ARP query, containing B's IP addr
- destination MAC address = FF-FF-FF-FF-FF-FF
 - all nodes on LAN receive ARP query

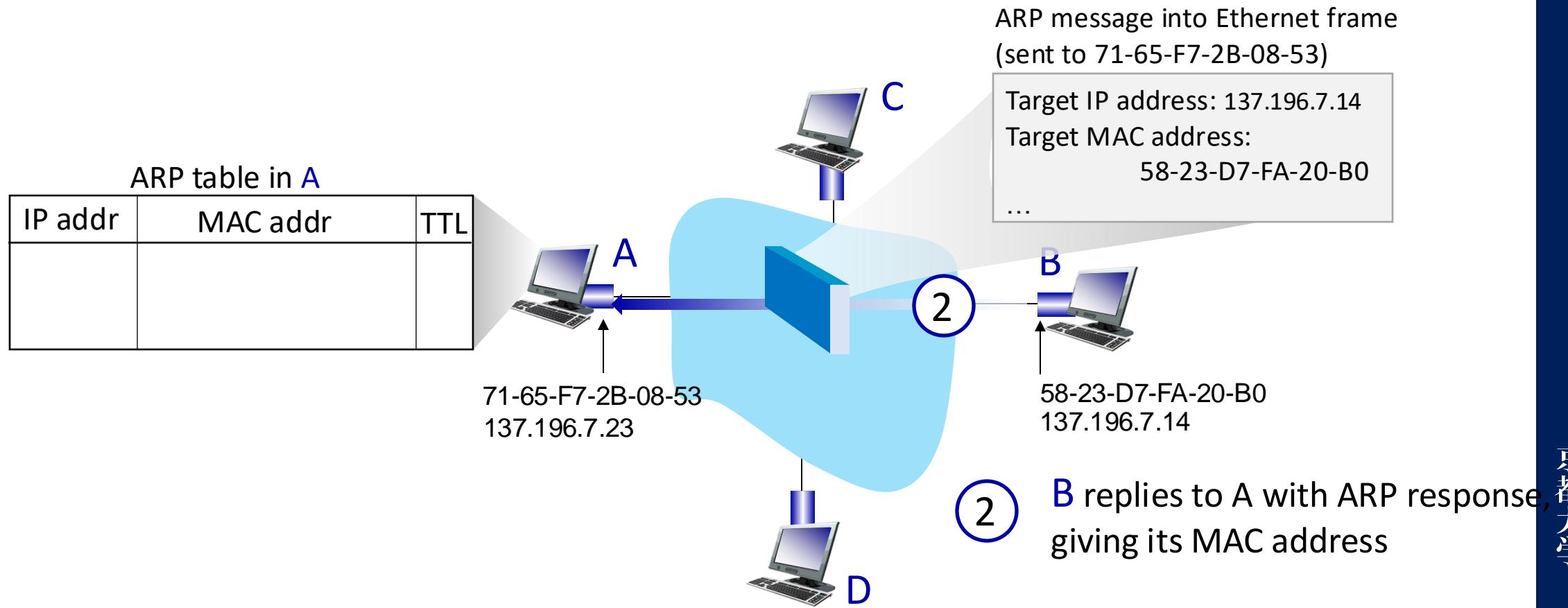
ARP table in A		
IP addr	MAC addr	TTL



ARP protocol in action

example: A wants to send datagram to B

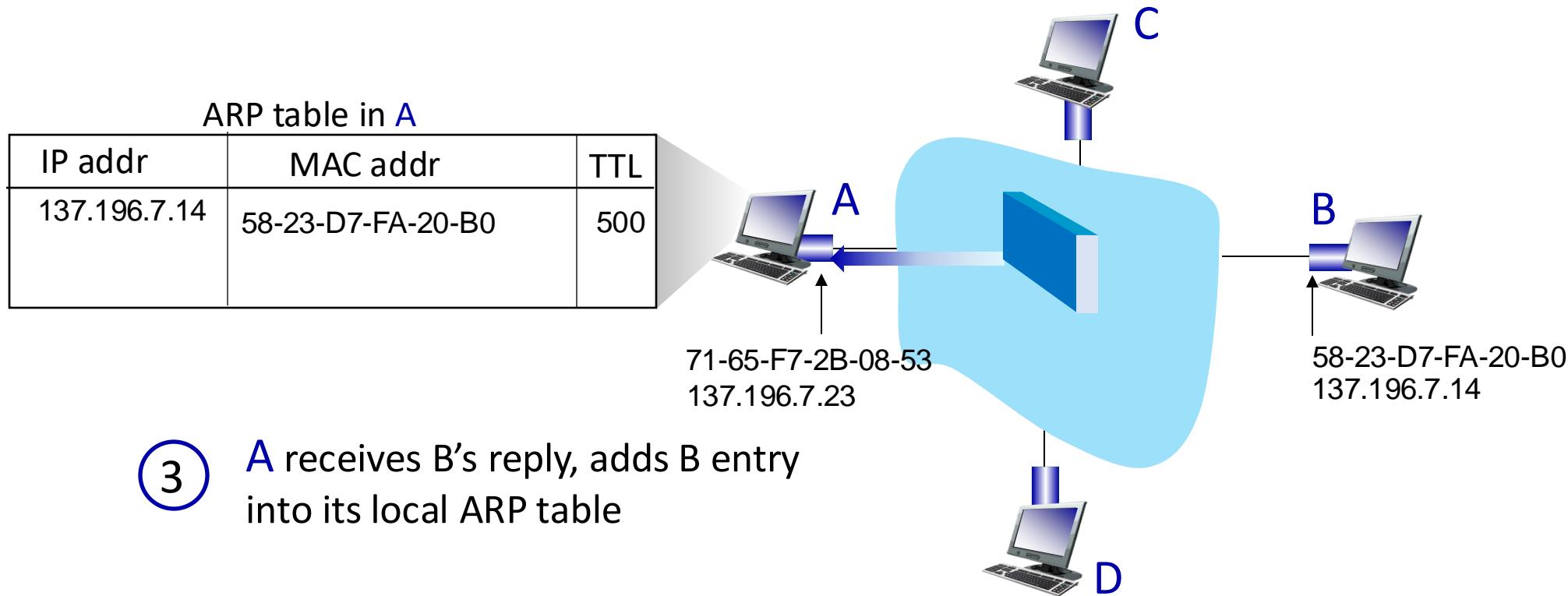
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



ARP protocol in action

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



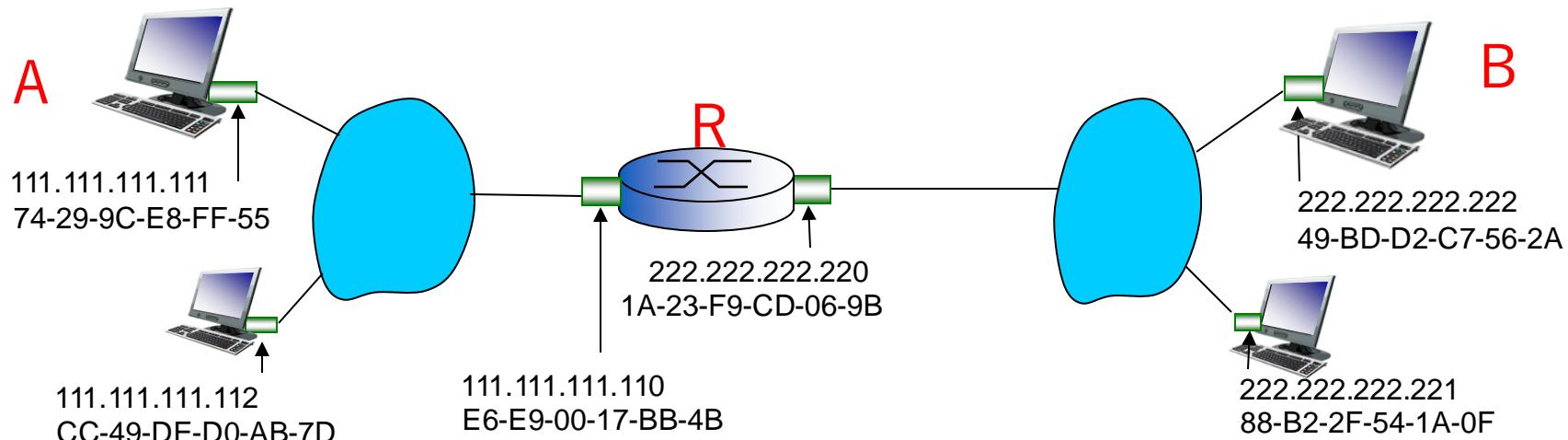
ARP protocol

- ARP operates when a host wants to send a datagram to another host on the same subnet.
- What if a host wants to send a network-layer datagram to a host across a router to another subnet?

Addressing: routing to another LAN

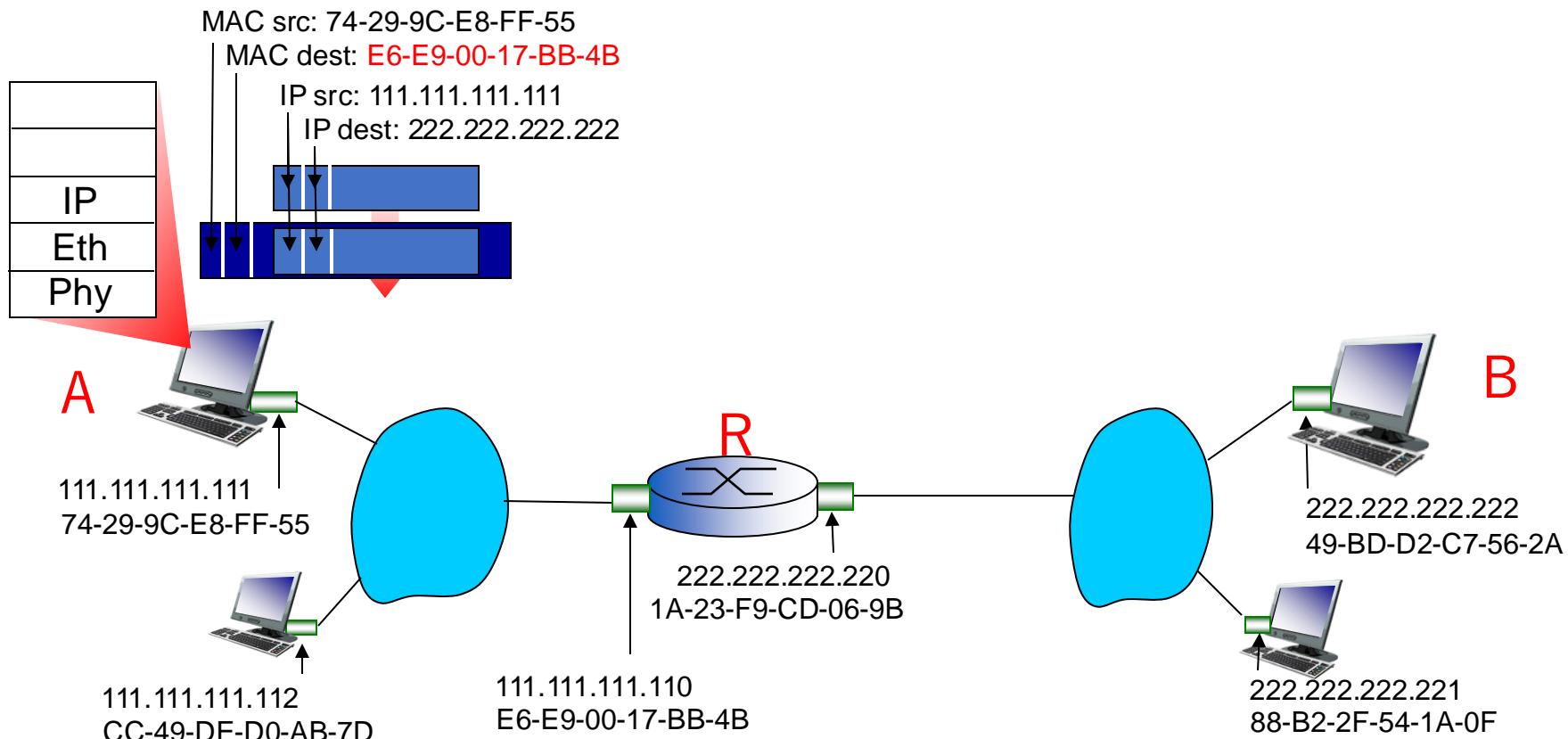
walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- assume A knows IP address of first hop router, R
- assume A knows R's MAC address (ARP)



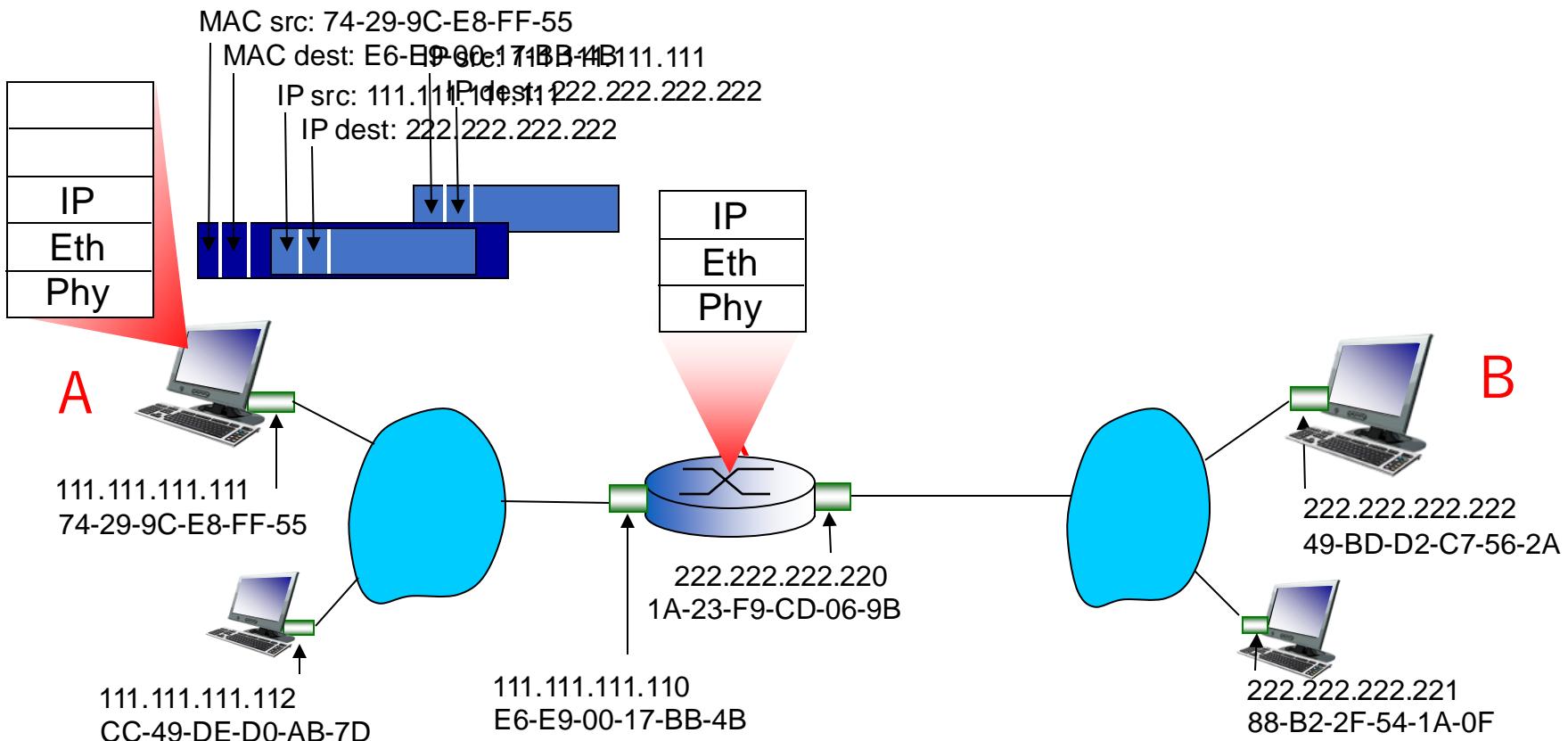
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as destination address, frame contains A-to-B IP datagram



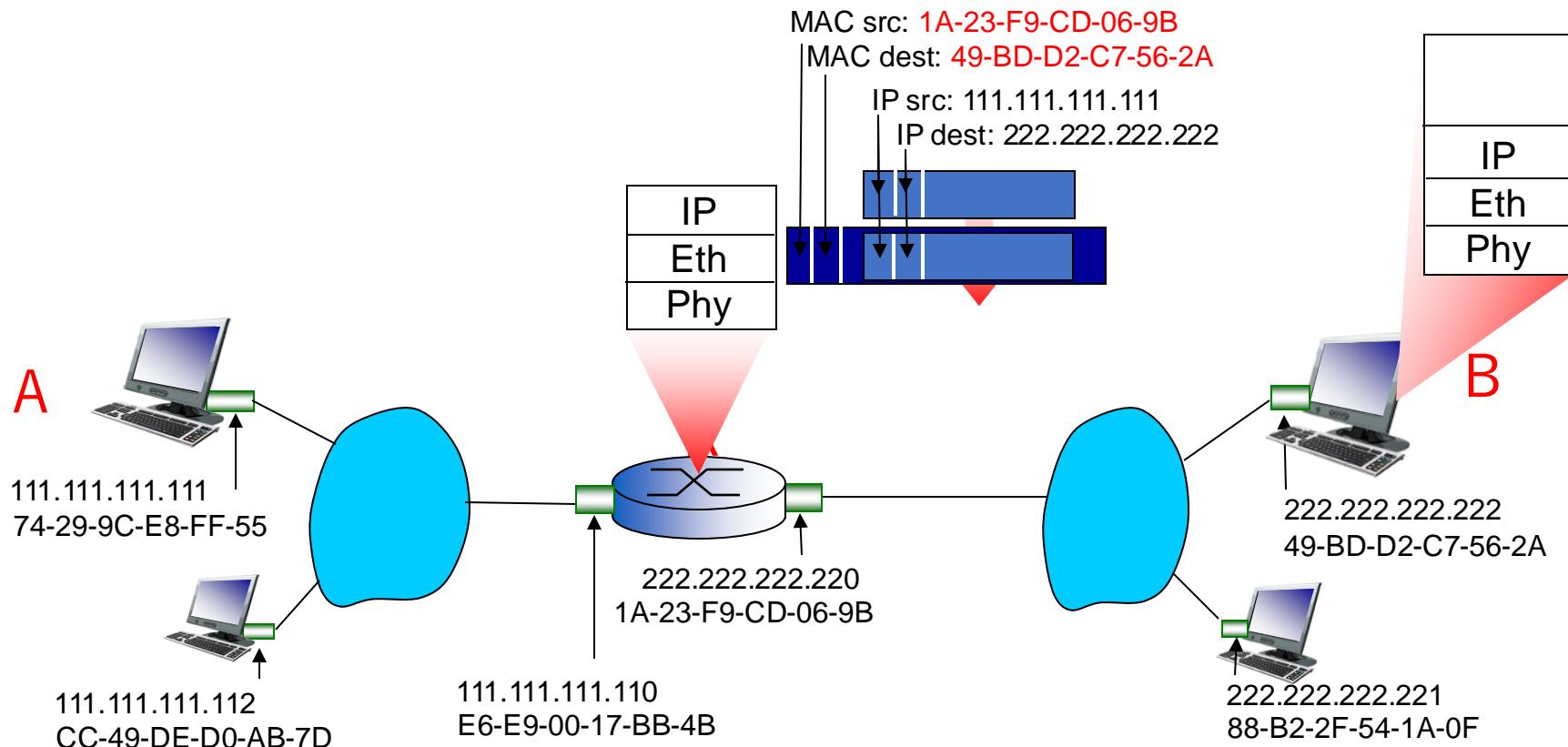
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



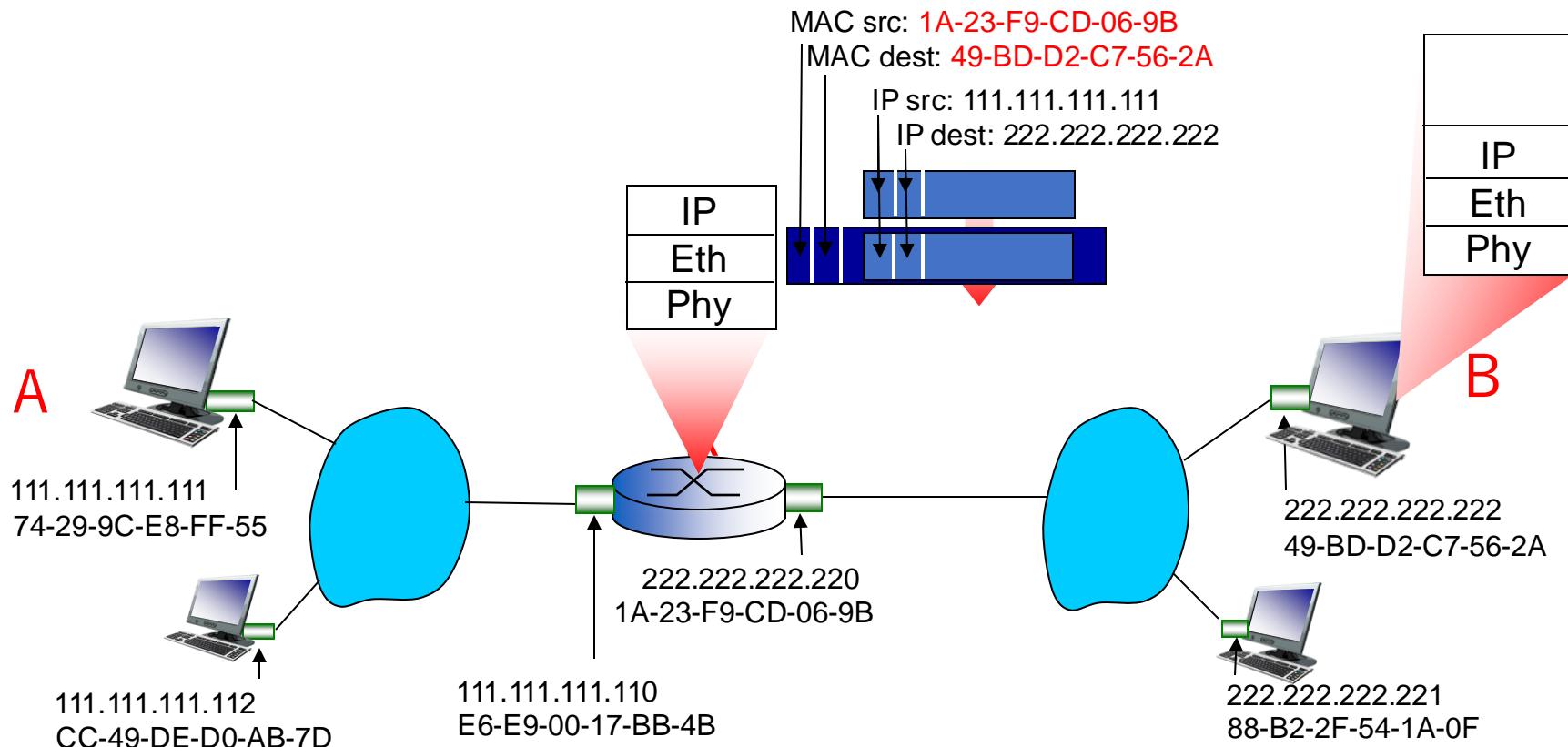
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as destination address, frame contains A-to-B IP datagram



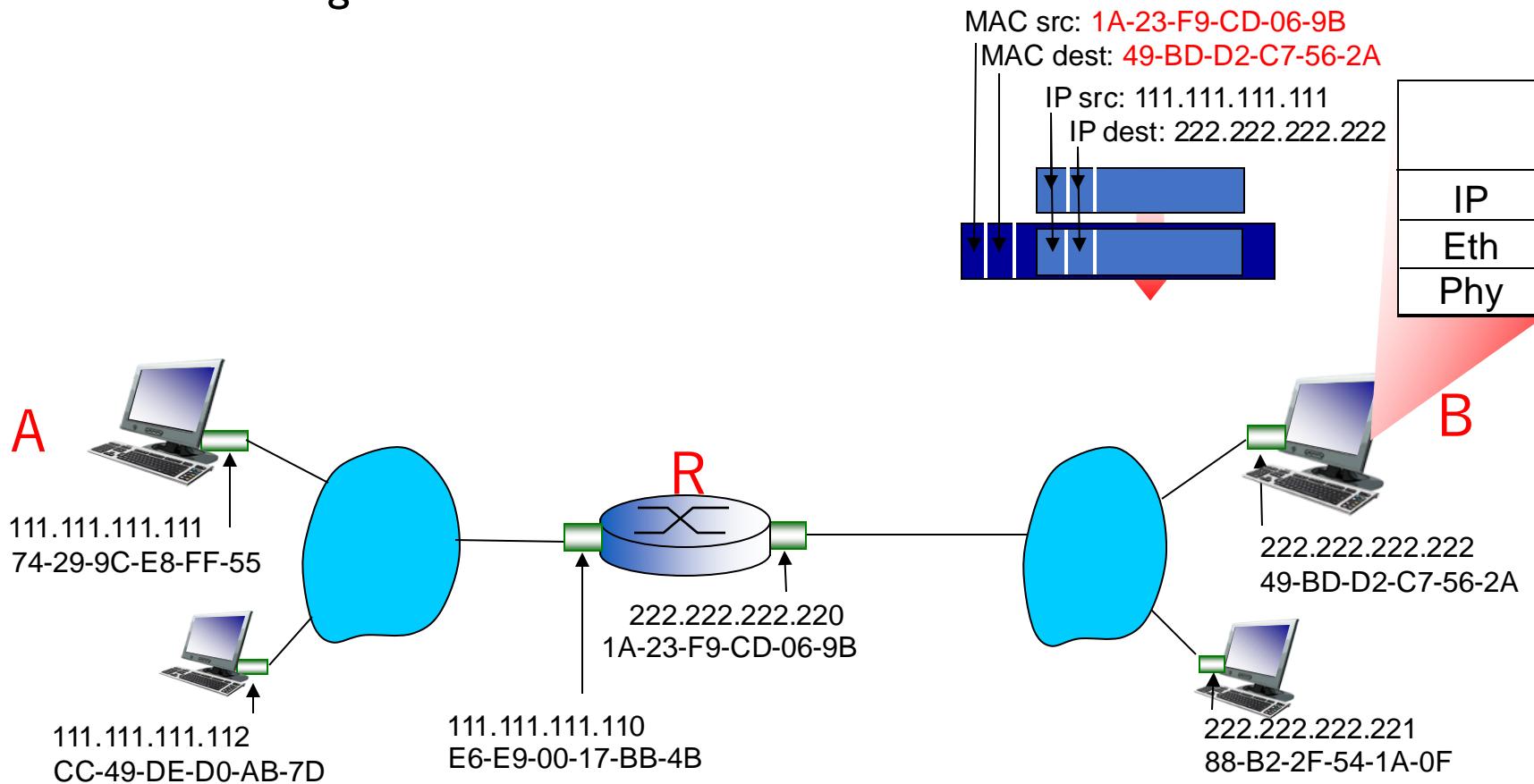
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as destination address, frame contains A-to-B IP datagram



Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Link layer

6.1 introduction, services

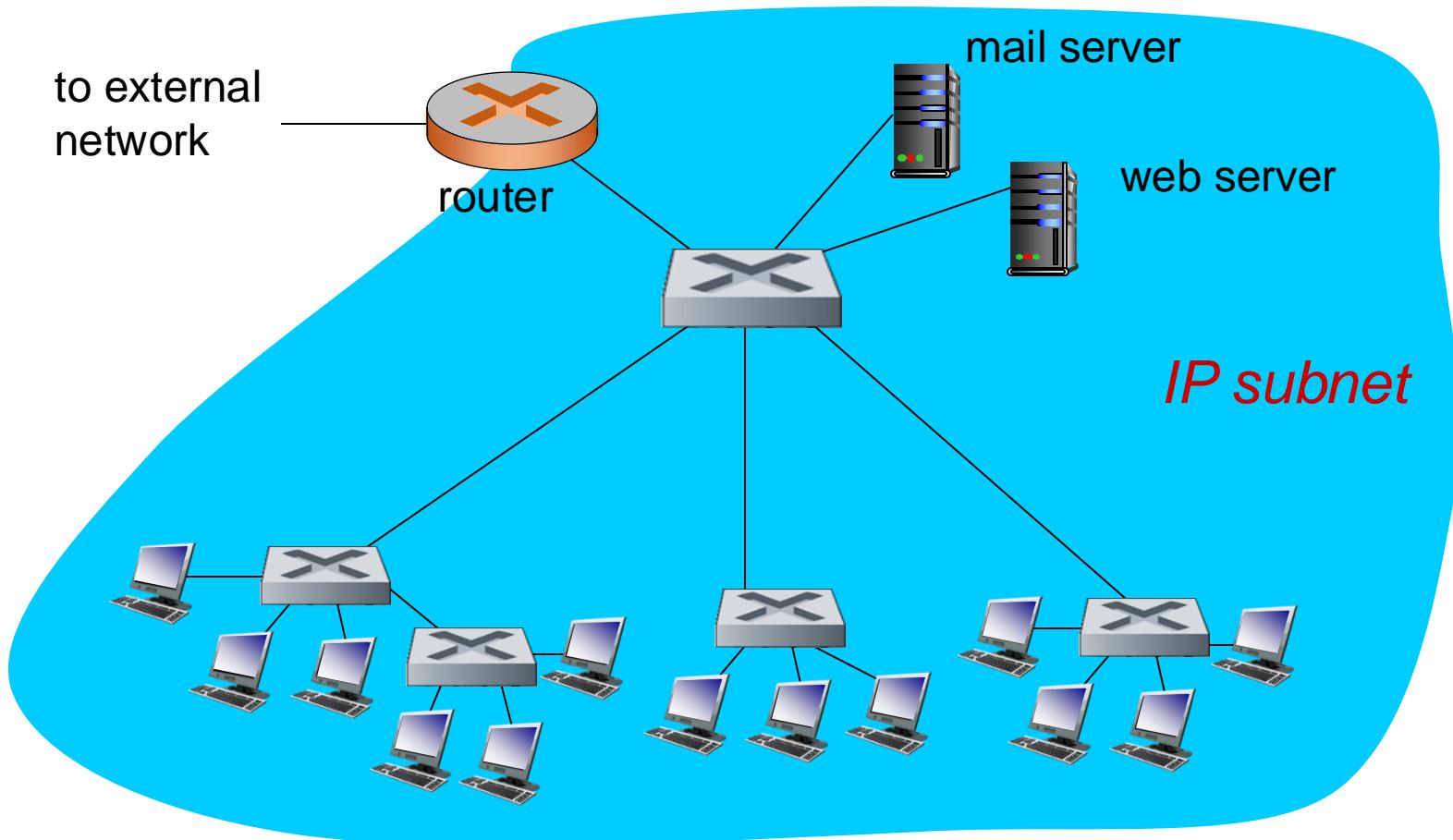
6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches

Institutional network



Ethernet

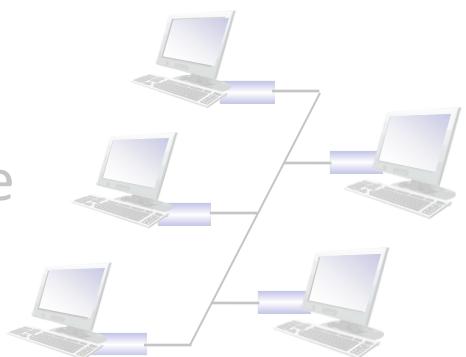
“dominant” wired LAN technology:

- first widely used LAN technology
- simple, cheap
- kept up with speed race: 10 Mbps – 400 Gbps
- single chip, multiple speeds

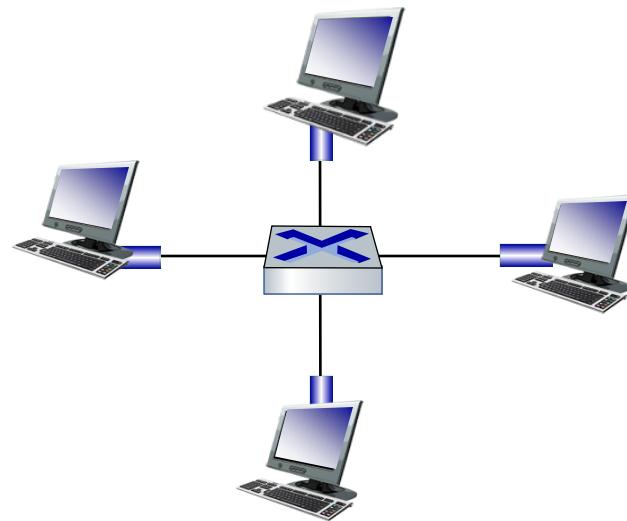
Ethernet: physical topology

- **bus:** popular through mid 90s
 - all nodes in same collision domain (can collide with each other)
- **switched:** prevails today
 - active link-layer 2 *switch* in center
 - nodes do not collide with each other

bus: coaxial cable



switched



Ethernet frame structure

sending adapter encapsulates IP datagram (or other network layer protocol packet) in **Ethernet frame**



preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- The first 7 bytes used to wake up the receiving adapters and synchronize the clock to sender's clock (the size of an interval used to transmit one bit as electronic voltage)

Ethernet frame structure (more)

- *addresses*: 6 byte source, destination MAC addresses
 - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
 - otherwise, adapter discards frame
- *type*: indicates higher layer protocol (mostly IP but others possible, e.g., ARP, Novell IPX, AppleTalk)
- *CRC*: cyclic redundancy check at receiver used for error detection
 - error detected: frame is dropped

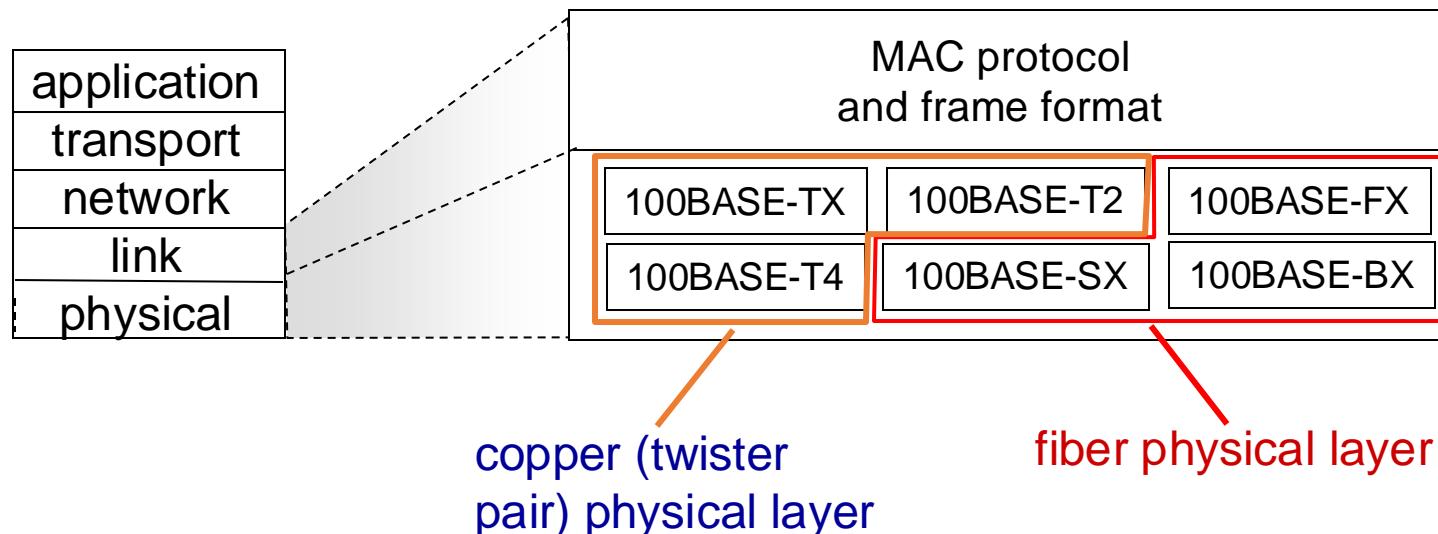


Ethernet: unreliable, connectionless

- *connectionless*: no handshaking between sending and receiving network interface controllers
- *unreliable*: receiver doesn't send acks to sender
 - data in dropped frames recovered only if initial sender uses higher layer reliable protocol (e.g., TCP), otherwise dropped data lost

802.3 Ethernet standards: link & physical layers

- *many* different Ethernet standards
 - common MAC protocol and frame format
 - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1 Gbps, 10 Gbps, 40 Gbps
 - different physical layer media: fiber, cable



Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches

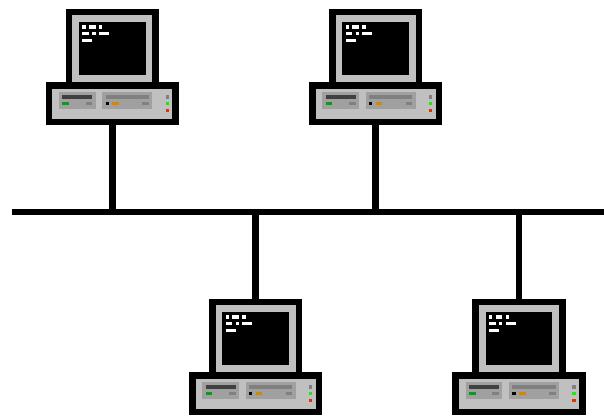
Switches

- Up until now we have not discussed what a switch actually does
- The role of a switch is to receive incoming link-layer frames and forward them onto outgoing links
- We will study forwarding in more detail in this section.

Hubs and Switches

How are multiple devices connected in the same subnet?

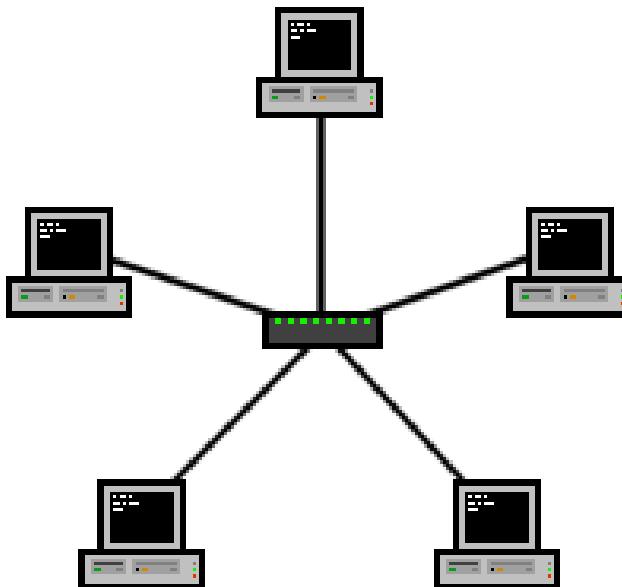
- **Hubs** operate on the physical layer. Each bit is simply recreated, and a copy is sent to all other interfaces



Hubs and Switches

How are multiple devices connected in the same subnet?

- **Switches** operate on the link layer. A switch can establish connections to send messages only to selected destinations.

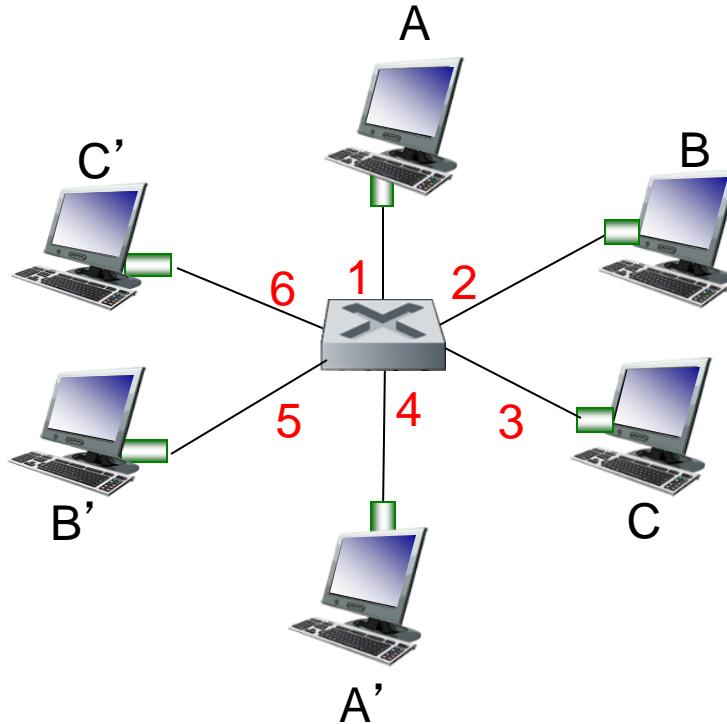


Ethernet switch

- Switch is a **link-layer** device: takes an *active* role
 - store, forward Ethernet (or other type of) frames
 - examine incoming frame's MAC address, *selectively* forward frame to one-or-more outgoing links when frame is to be forwarded on segment
- **transparent**: hosts *unaware* of presence of switches
- **plug-and-play, self-learning**
 - switches do not need to be configured

Switch: *multiple simultaneous transmissions*

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on each incoming link, but no collisions
- **switching**: A-to-A' and B-to-B' can transmit simultaneously, without collisions

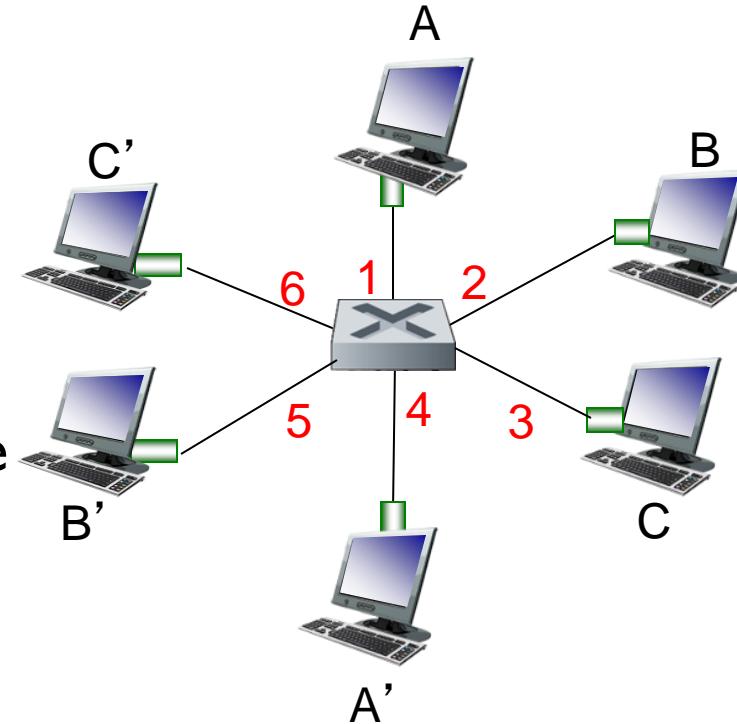


*switch with six interfaces
(1,2,3,4,5,6)*

Switch forwarding table

Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

- A: each switch has a **switch table**, each entry:
 - (MAC address of host, interface to reach host, time stamp)
 - looks like a routing table!



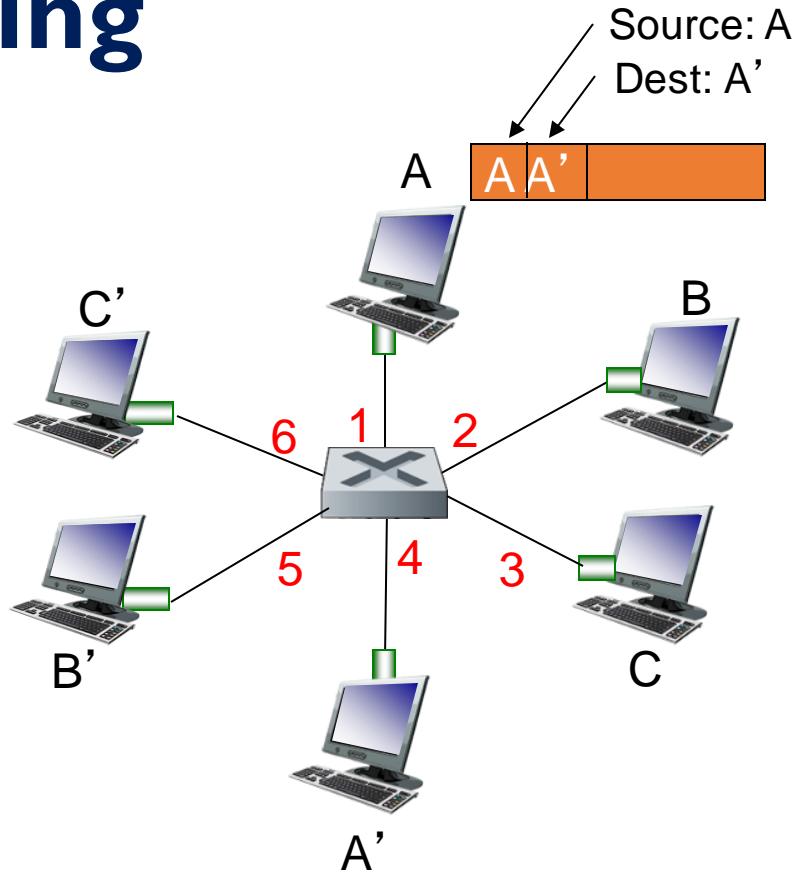
Q: how are entries created, maintained in switch table?

- something like a routing protocol?

*switch with six interfaces
(1,2,3,4,5,6)*

Switch: self-learning

- switch **learns** which hosts can be reached through which interfaces
 - when frame received, switch “learns” location of sender: incoming LAN segment
 - records sender/location pair in switch table

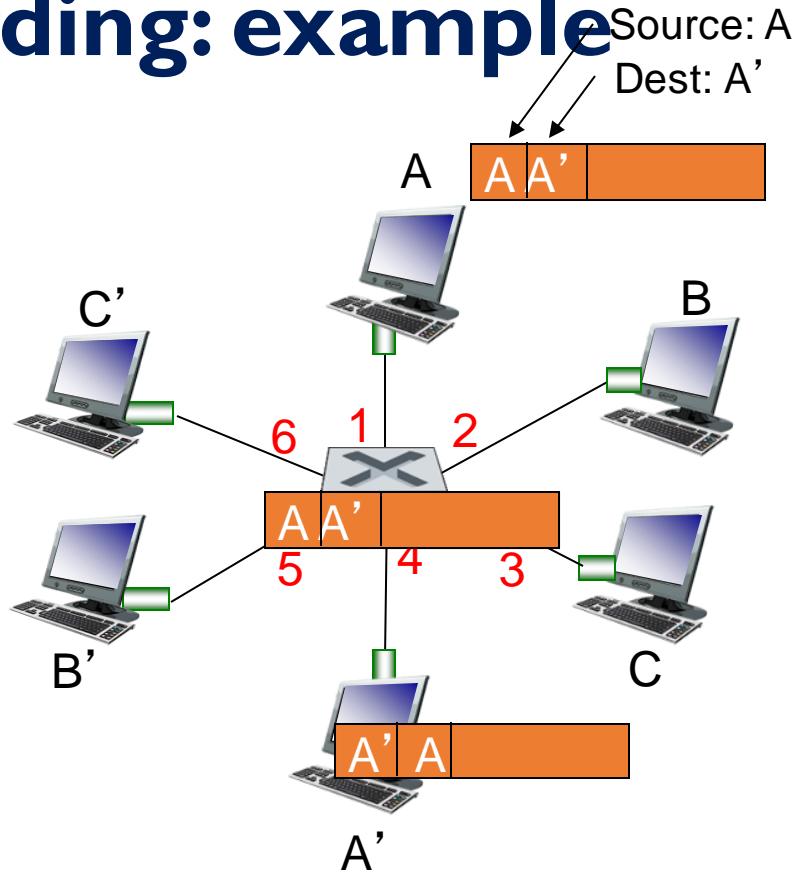


*Switch table
(initially empty)*

MAC addr	interface	TTL
A	1	60

Self-learning, forwarding: example

- frame destination, A', location unknown: *flood*
- destination A location known: *selectively send on just one link*



*switch table
(initially empty)*

MAC addr	interface	TTL
A	1	60
A'	4	60

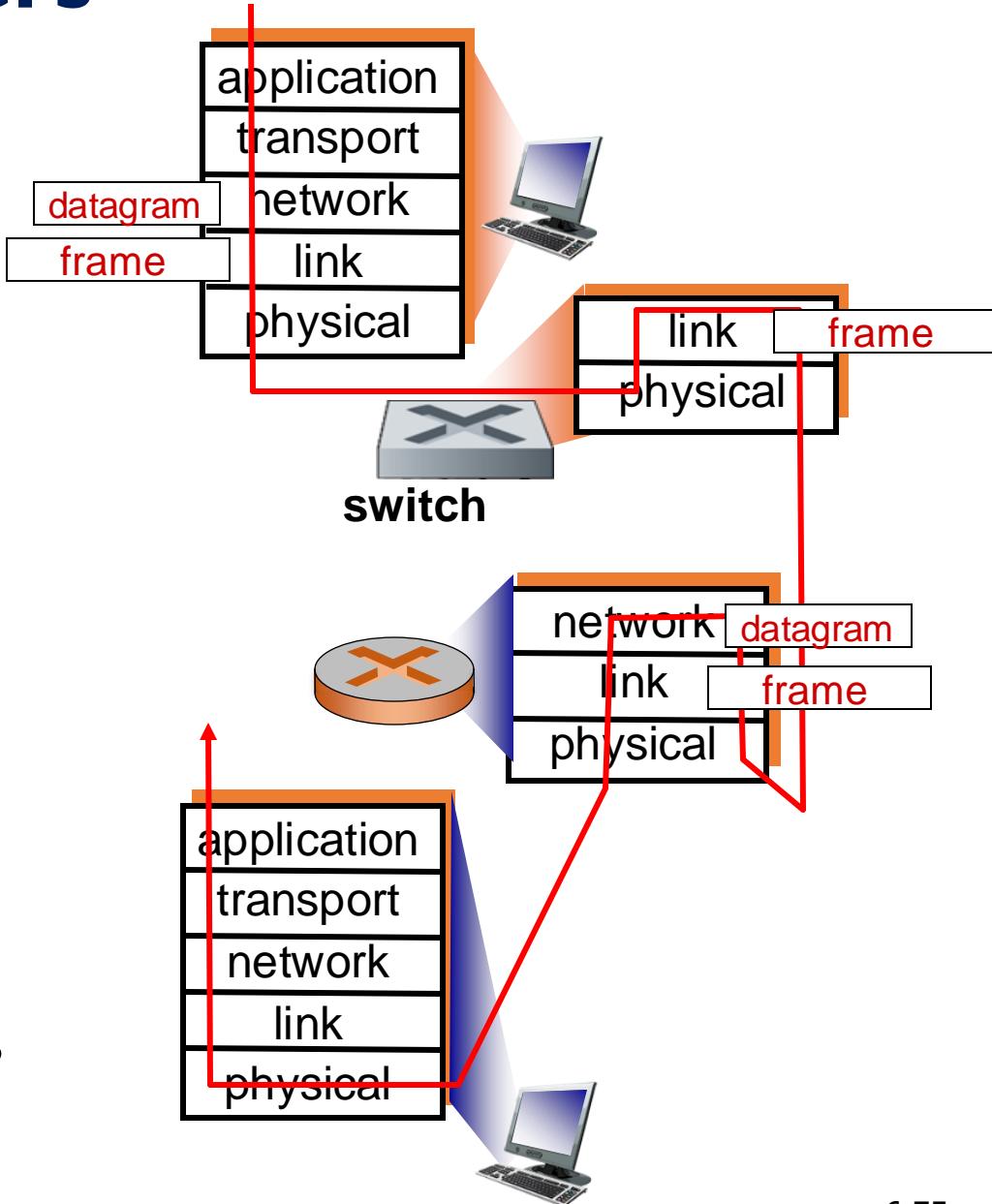
Switches vs. routers

both are store-and-forward:

- *routers*: network-layer devices (examine network-layer headers)
- *switches*: link-layer devices (examine link-layer headers)

both have forwarding tables:

- *routers*: compute tables using routing algorithms, IP addresses
- *switches*: learn forwarding table using flooding, learning, MAC addresses



Link Layer Summary

- principles behind data link layer services:
 - sharing a broadcast channel: multiple access
 - link layer addressing
- instantiation, implementation of layer technologies
 - Ethernet
 - Switches