ECS524
Link layer

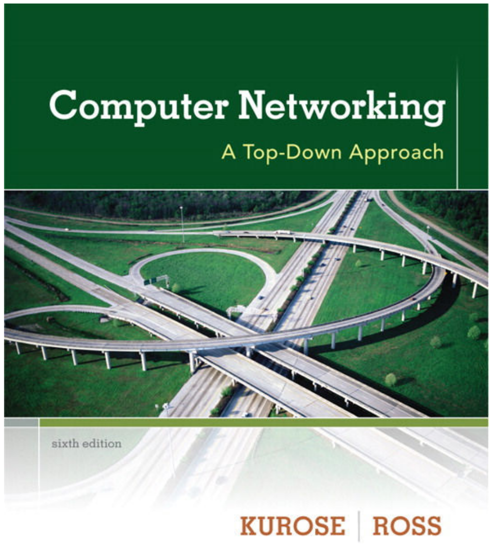
Prof. Steve Uhlig
steve.uhlig@qmul.ac.uk
Office: Eng 202

Dr. Felix Cuadrado
felix.cuadrado@qmul.ac.uk
Office: Eng 153a

Slides

Disclaimer:
Some of the slides' content is borrowed directly from those provided by the authors of the textbook. They are available from

http://www-net.cs.umass.edu/kurose-ross-ppt-6e



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

The Link layer

- **Link layer concepts**
- Link layer addressing
- Ethernet, switches

Link layer: introduction

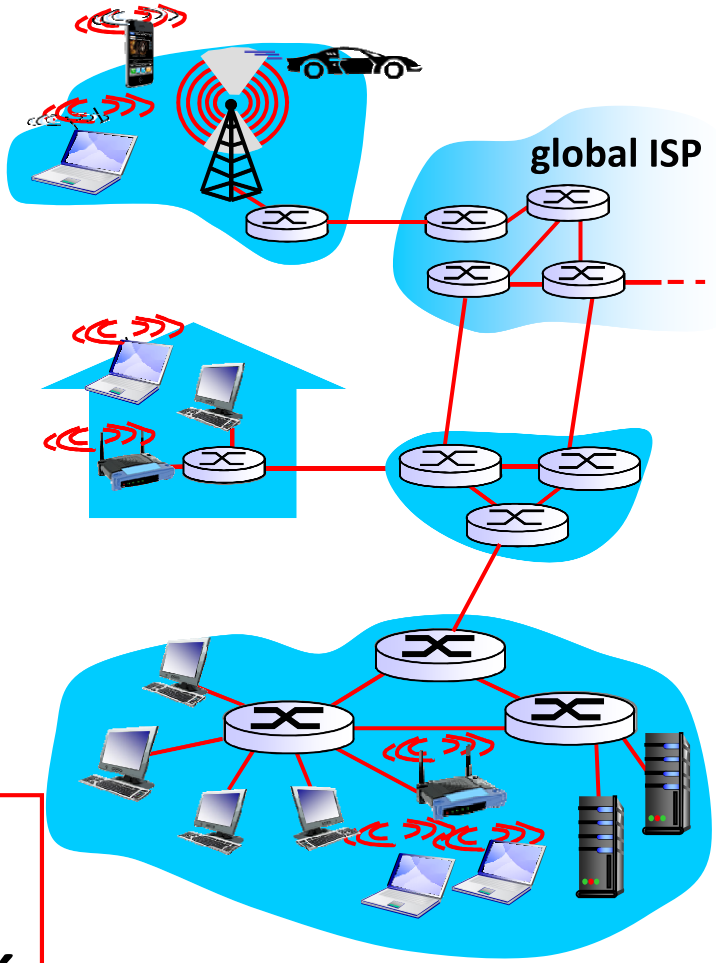
terminology:

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:

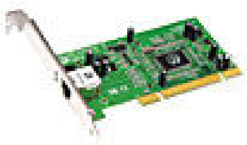
- encapsulate datagram into frame, adding header, trailer
- channel access if shared medium
- "MAC" addresses used in frame headers to identify source, dest
  - different from IP address!

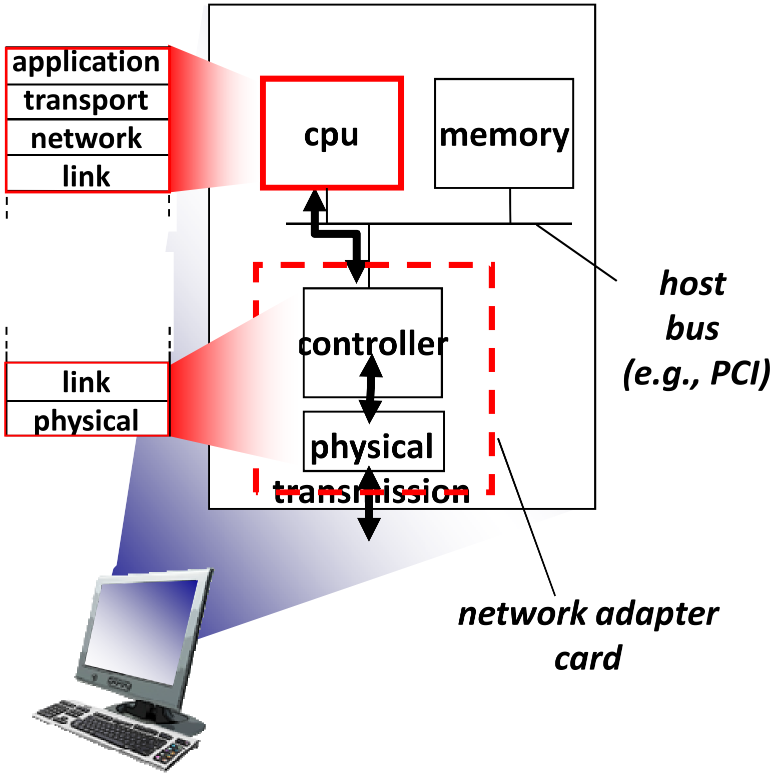
reliable delivery between adjacent nodes

- 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?

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





The Link layer

- Link layer concepts
- **Link layer addressing**
- Ethernet, switches

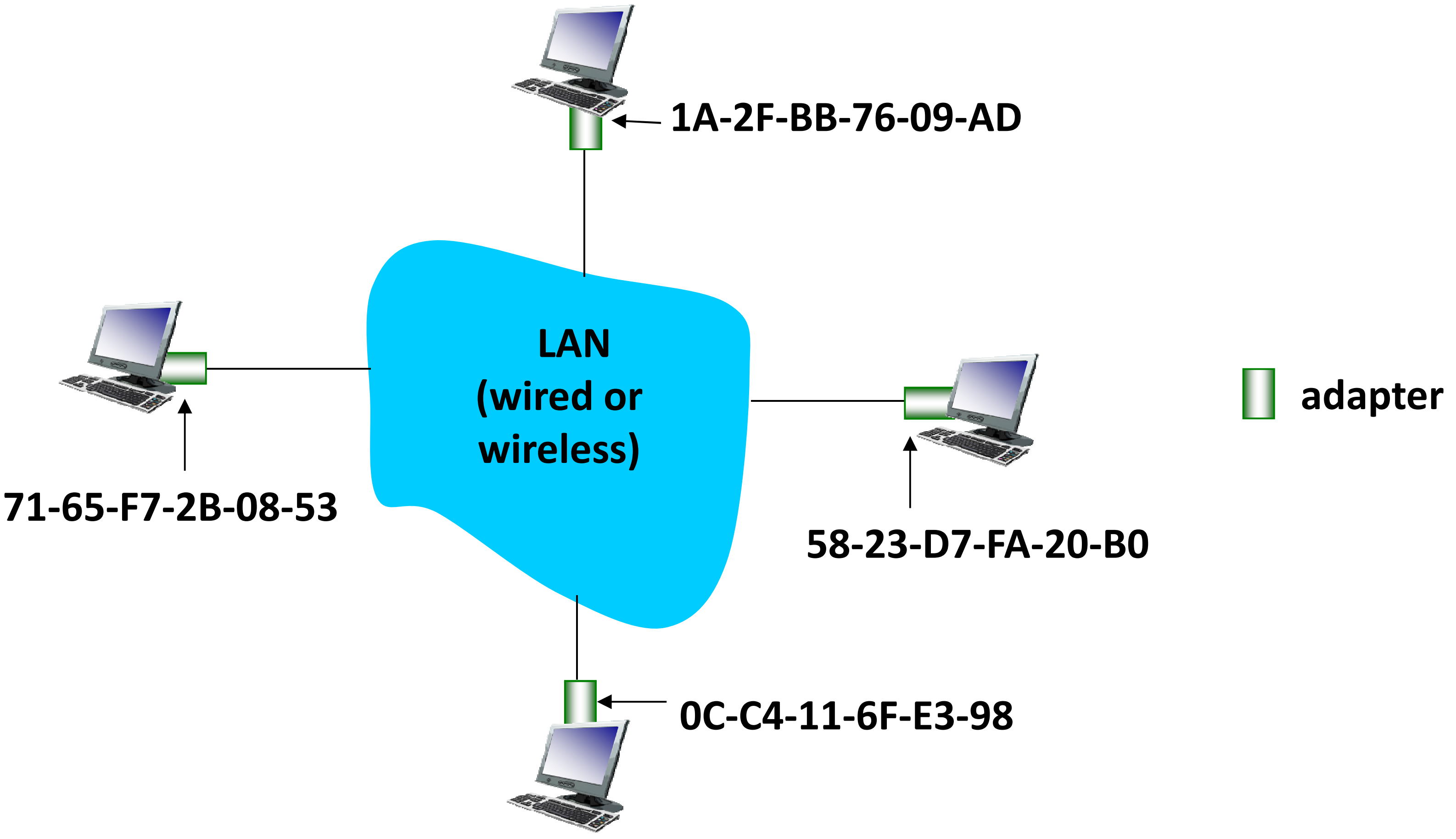
MAC addresses and ARP

- 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) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

hexadecimal (base 16) notation
(each “number” represents 4 bits)

LAN addresses and ARP

Each LAN adapter has **unique *LAN* address**



LAN addresses (more)

MAC address allocation administered by IEEE

manufacturer buys portion of MAC address space (to assure uniqueness)

analogy:

- MAC address: like Social Security Number
- IP address: like postal address

MAC flat address ➜ portability

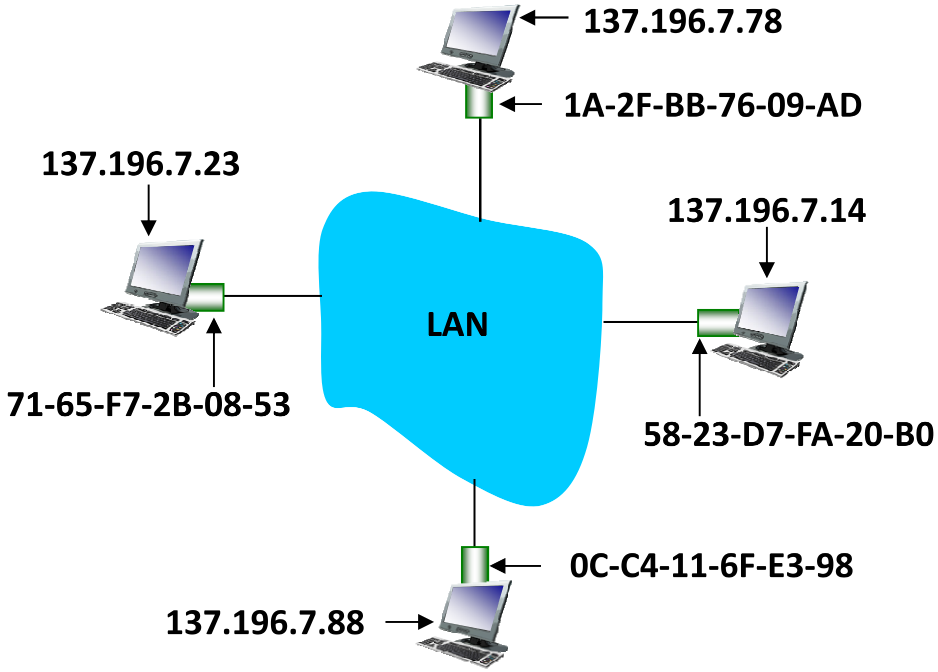
- can move LAN card from one LAN to another

IP hierarchical address ***not*** portable

- **address depends on IP subnet** to which node is attached

Address Resolution Protocol

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



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

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

1. A wants to send datagram to B

   B’s MAC address not in A’s ARP table.

2. A broadcasts ARP query packet, containing B's IP address

   dest MAC address = FF-FF-FF-FF-FF-FF

   all nodes on LAN receive ARP query

3. B receives ARP packet, replies to A with its (B's) MAC address

   frame sent to A’s MAC address (unicast)

4. A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)

   soft state: information times out (goes away) unless refreshed

ARP is “plug-and-play”:
nodes create their ARP tables without intervention from net administrator

Inter-LAN addressing/routing

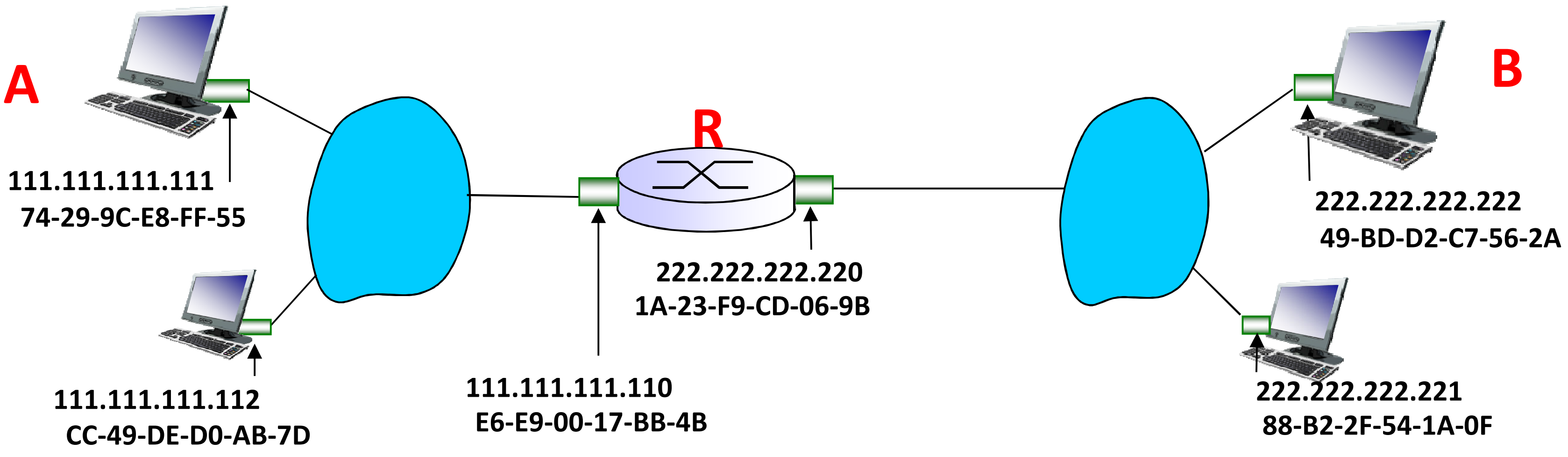
walkthrough: send datagram from A to B via R

focus on addressing – at IP (datagram) and MAC layer (frame)

assume A knows B' s IP address

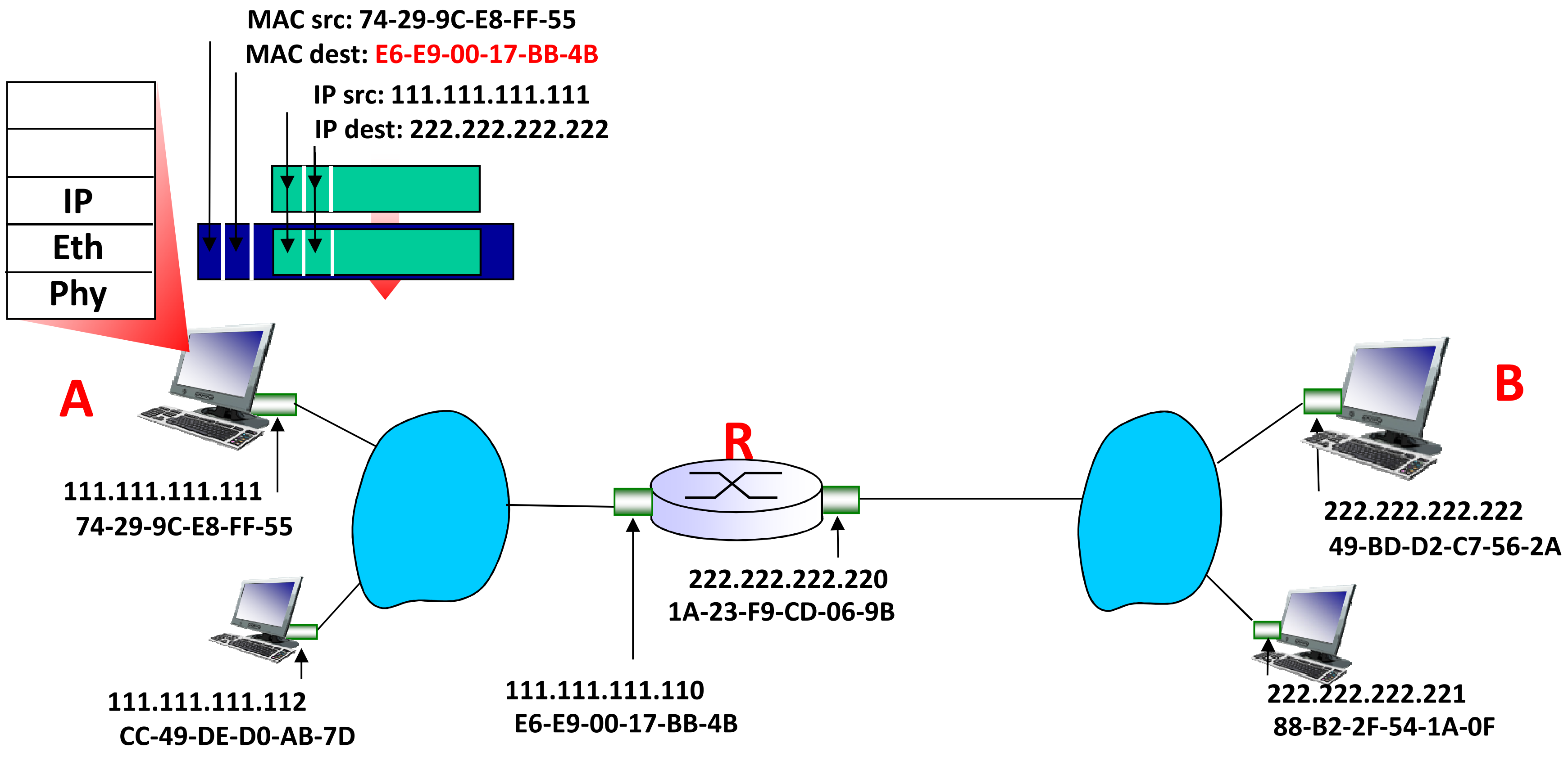
assume A knows IP address of first hop router, R (how?)

assume A knows R' s MAC address (how?)



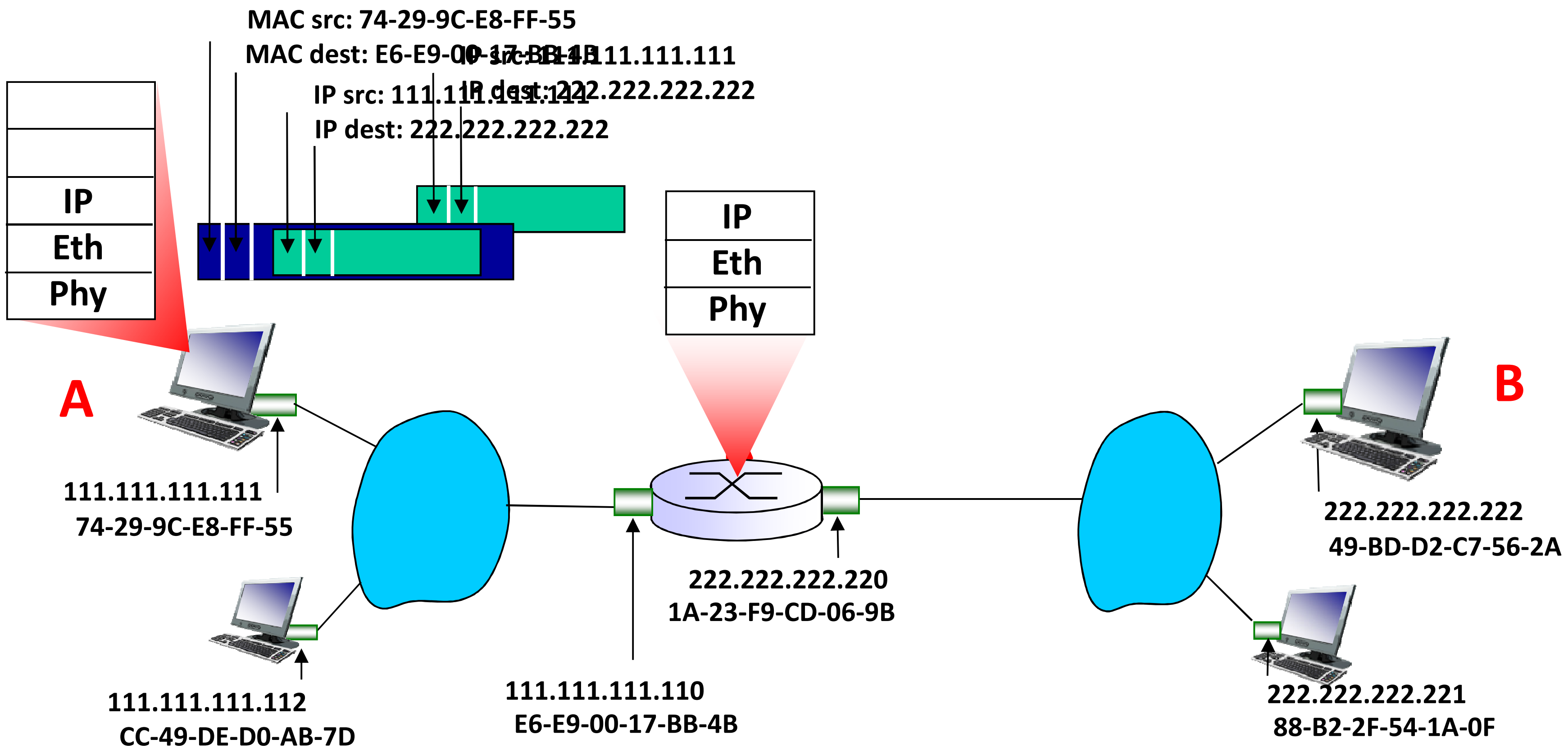
Inter-LAN addressing/routing

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



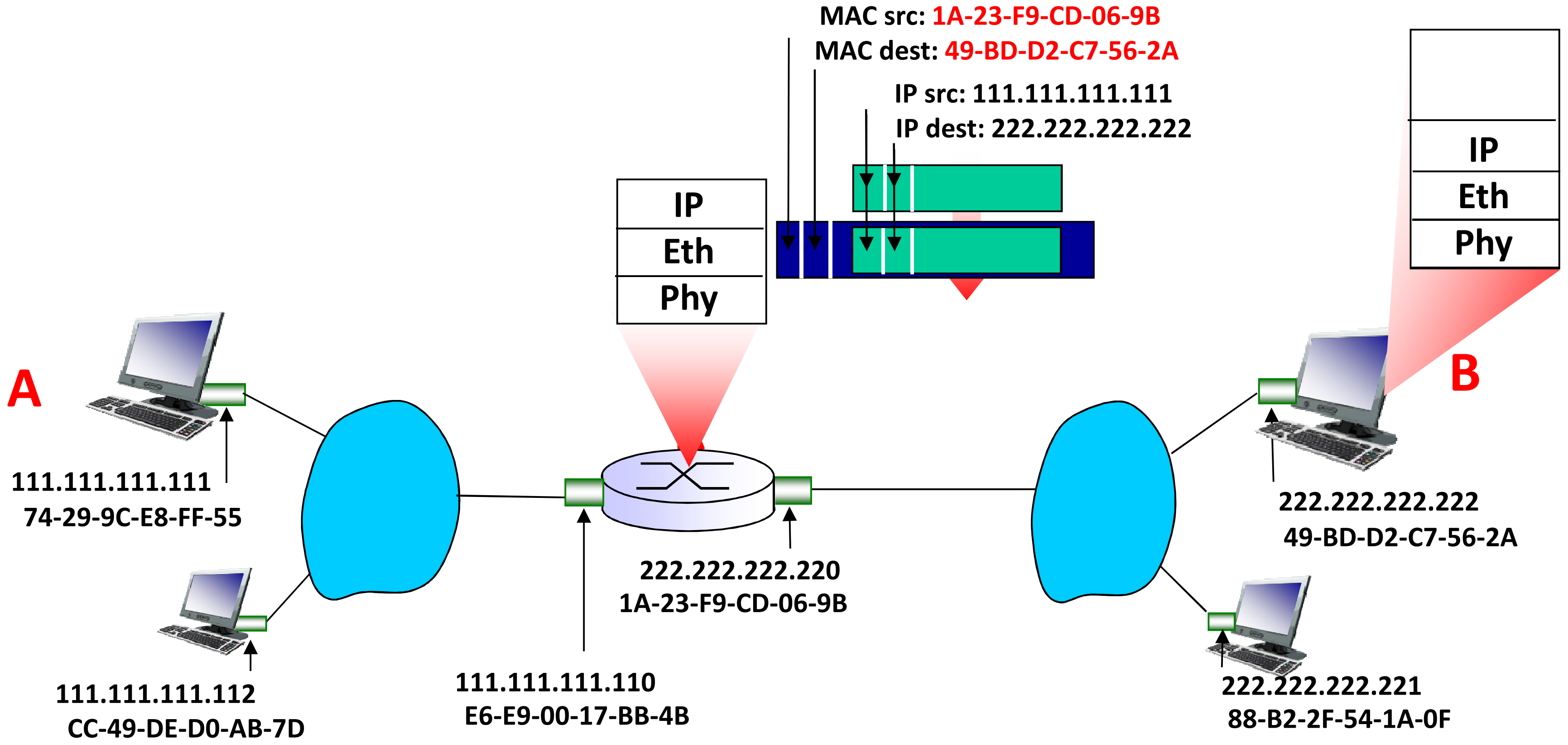
Inter-LAN addressing/routing

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



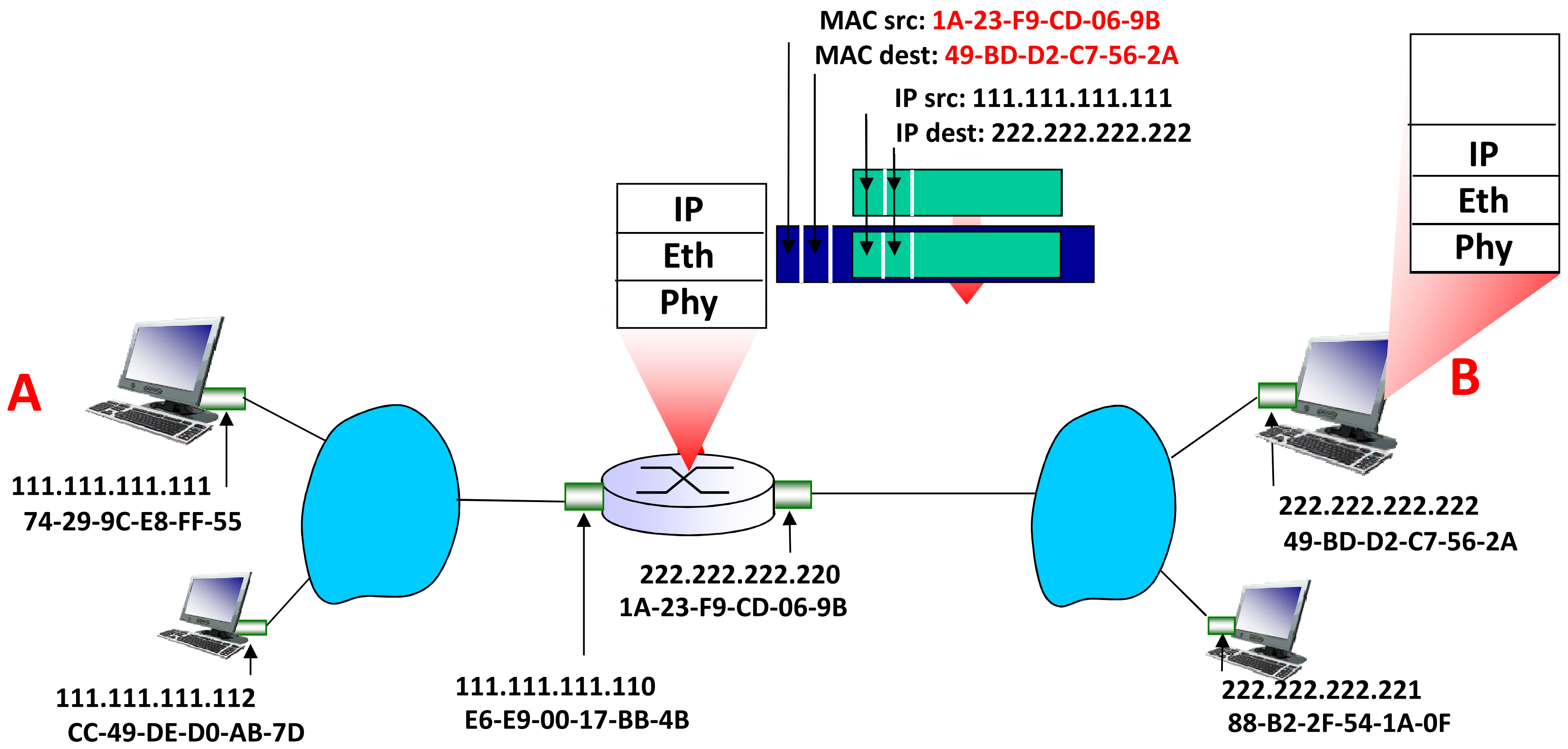
Inter-LAN addressing/routing

- 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



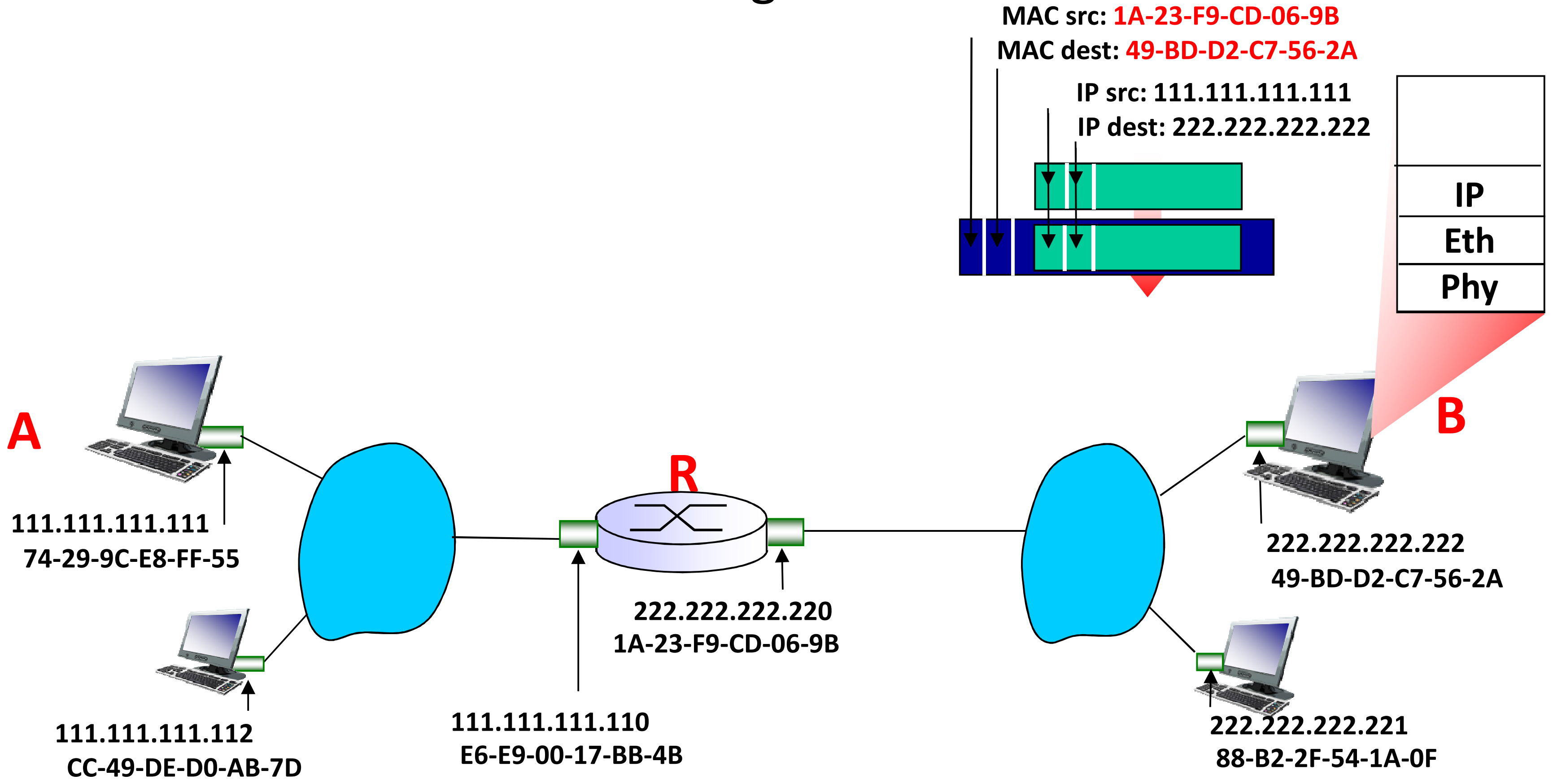
Inter-LAN addressing/routing

- 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



Inter-LAN addressing/routing

- 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



The Link layer

- Link layer concepts
- Link layer addressing
- **Ethernet, switches**

Ethernet

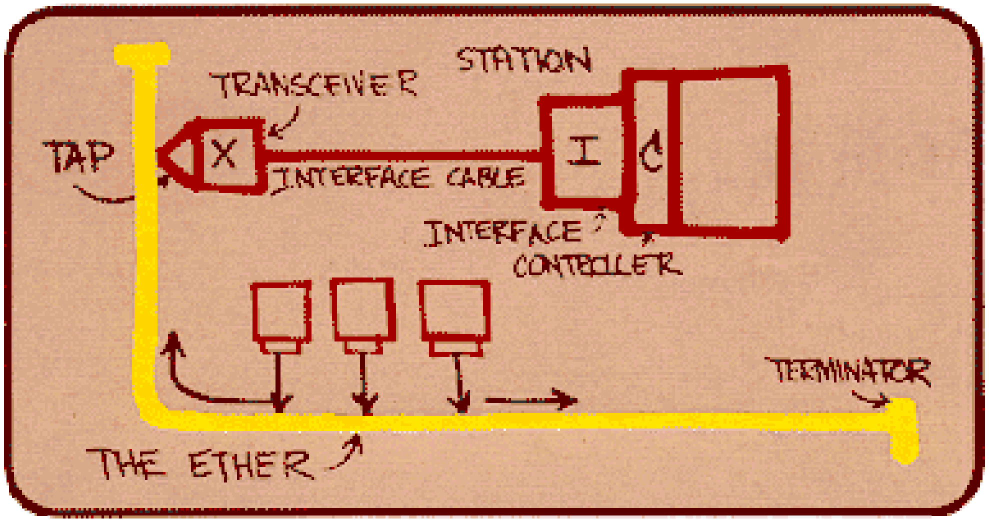
"dominant" wired LAN technology:

cheap $20 for NIC

first widely used LAN technology

simpler, cheaper than token LANs and ATM

kept up with speed race: 10 Mbps – 10 Gbps



Metcalfe's Ethernet sketch

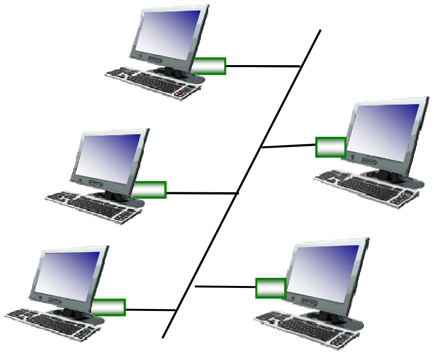
Ethernet: physical topology

bus: popular through mid 90s

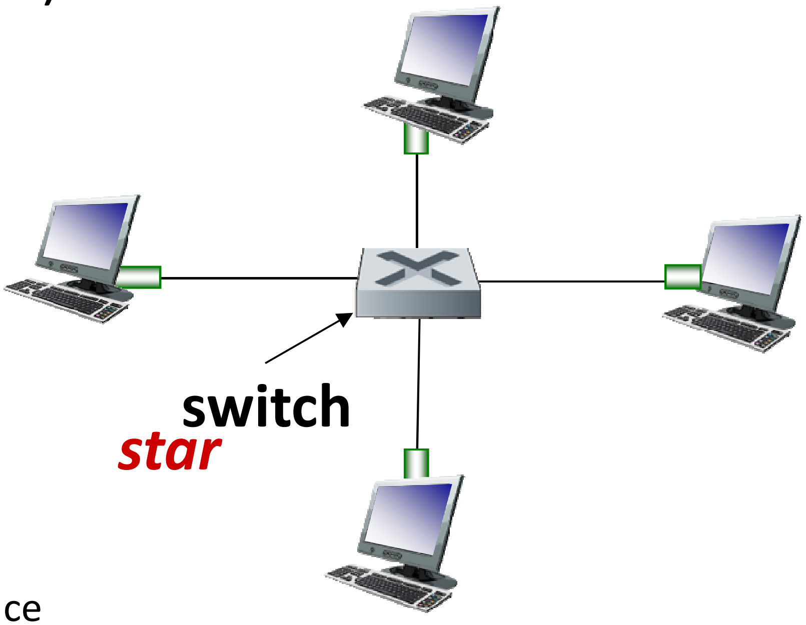
- all nodes in same collision domain (can collide with each other)

star: prevails today

- active *switch* in center
- each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)



bus: **coaxial cable**



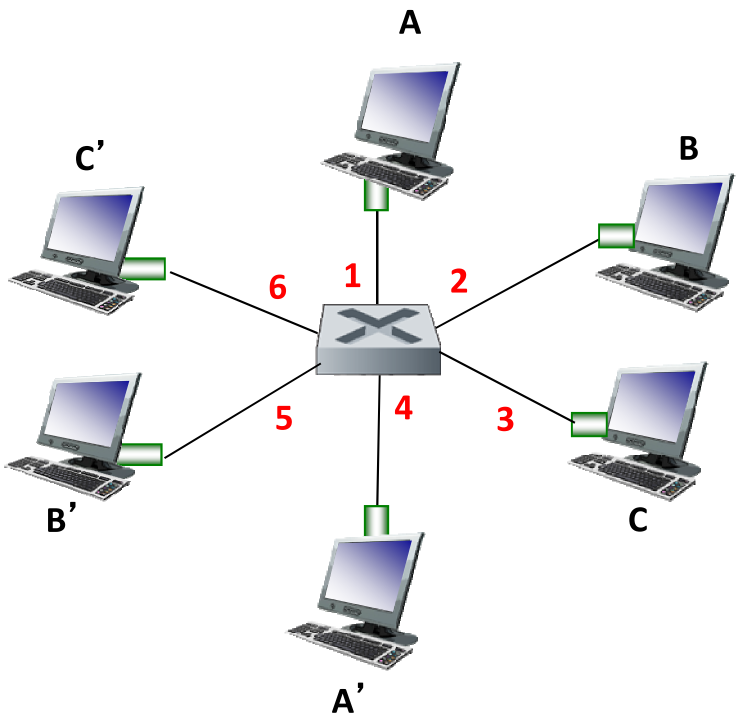
star

Ethernet switch

- link-layer device: takes an *active* role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- *transparent*
  - hosts are 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; full duplex
  - each link is its own collision domain
- *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

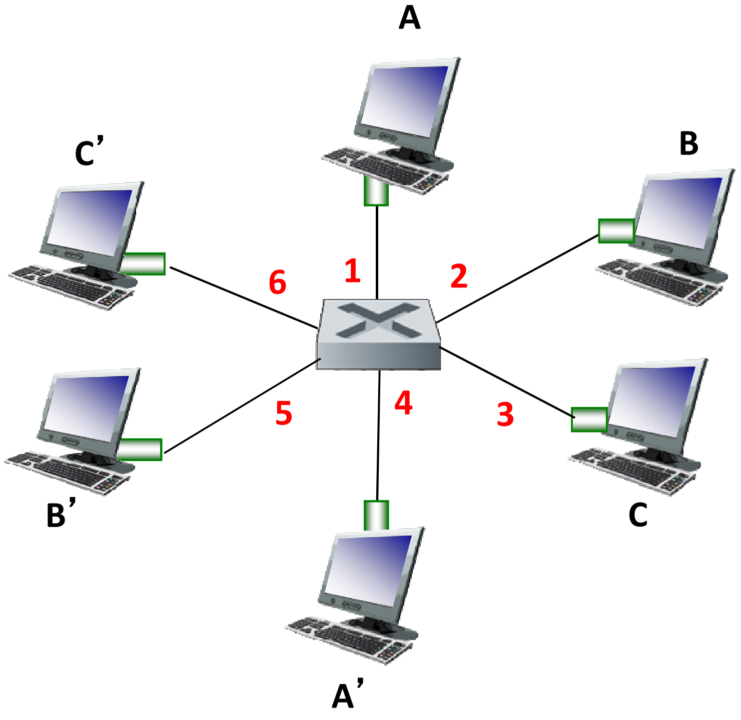
Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

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?

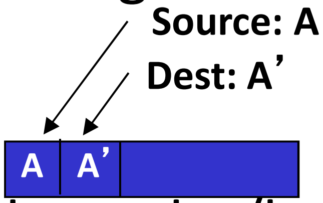


switch with six interfaces (1,2,3,4,5,6)

Switch table: self-learning

switch *learns* which hosts can be reached through which interfaces

- when frame received, switch "learns" location of sender: incoming LAN segment



Source: A

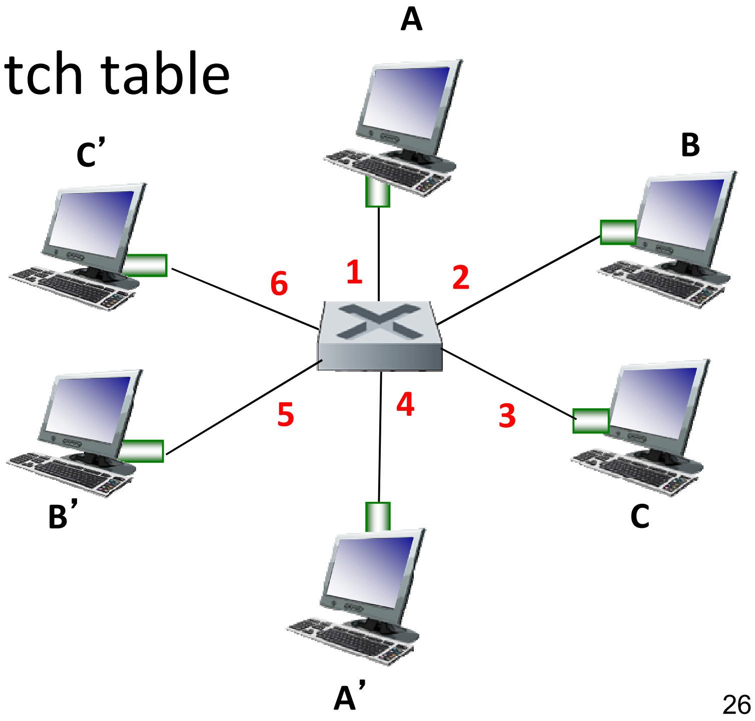
Dest: A'

A | A'

- records sender/location pair in switch table

MAC Addr	Interface	TTL
A	1	60

Switch table (initially empty)



Switch: frame filtering/forwarding

when frame received at switch:

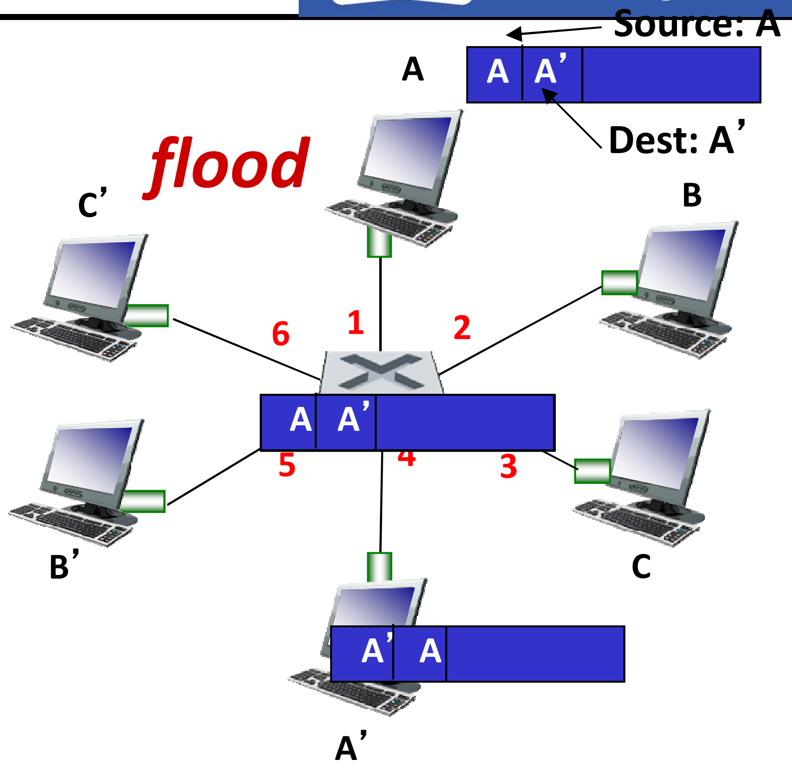
```
1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination then {
     if destination on segment from which frame arrived
         then drop frame
     else forward frame on interface indicated by entry
   }
   else flood  /* forward on all interfaces except
                  arriving interface */
```

Self-learning, forwarding: example

frame destination, A’, location unknown:

destination A location known:

selectively send on just one link

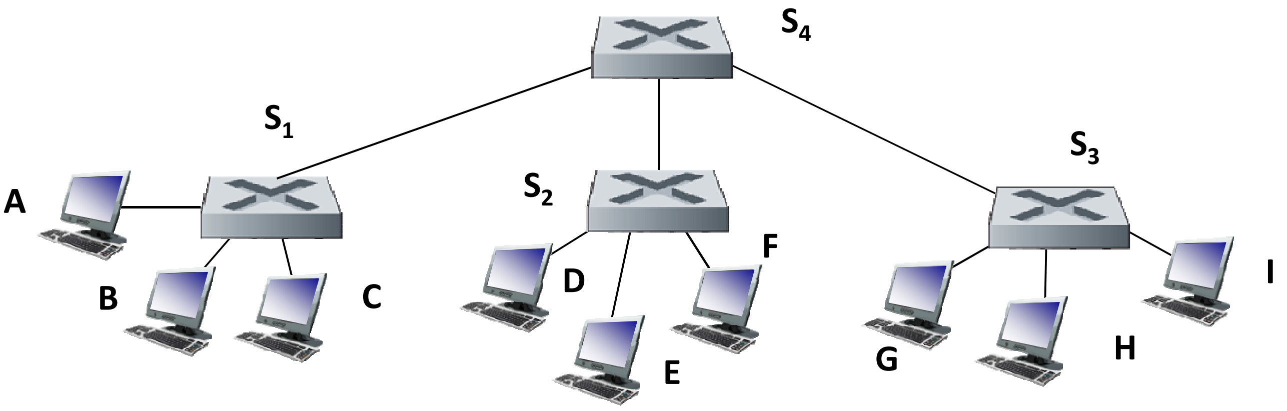


MAC addr	interface	TTL
A	*1*	*60*
A’	*4*	*60*

switch table (initially empty)

Interconnecting switches

switches can be connected together

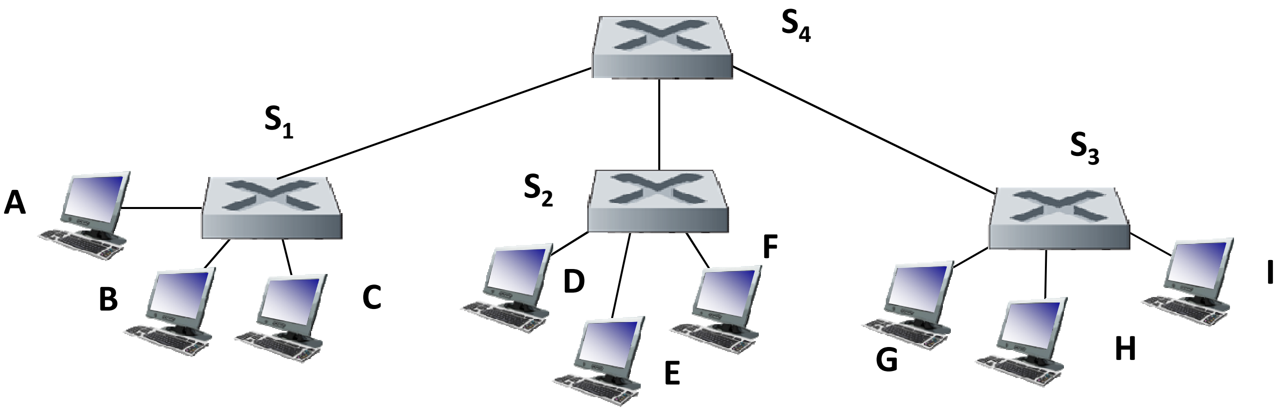


Q: sending from A to G - how does S_1 know to forward frame destined to F via S_4 and S_3?

A: self learning! (works exactly the same as in single-switch case!)

Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4