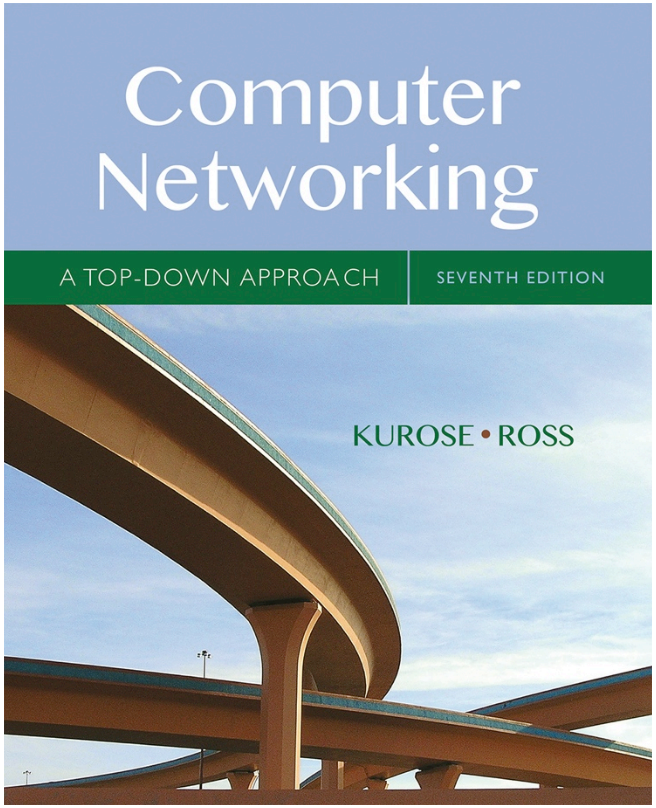
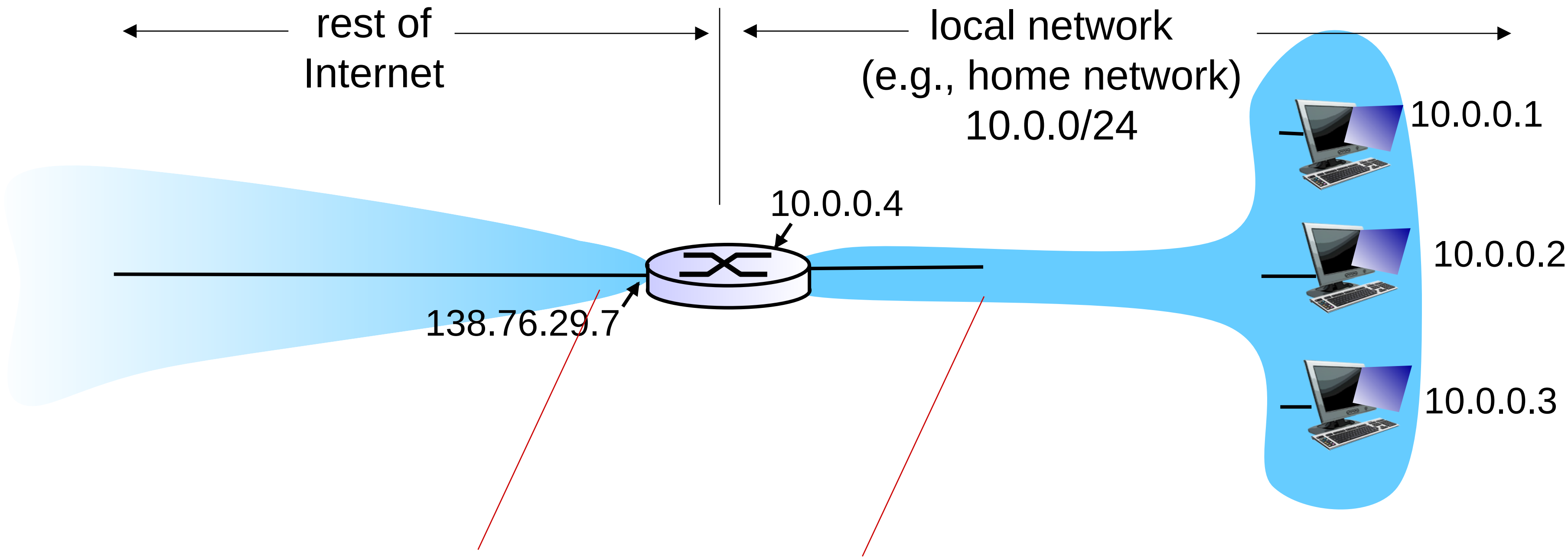
Introduction to TCP/IP III



7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

- essentially adapted from Kurose and Ross

NAT: network address translation

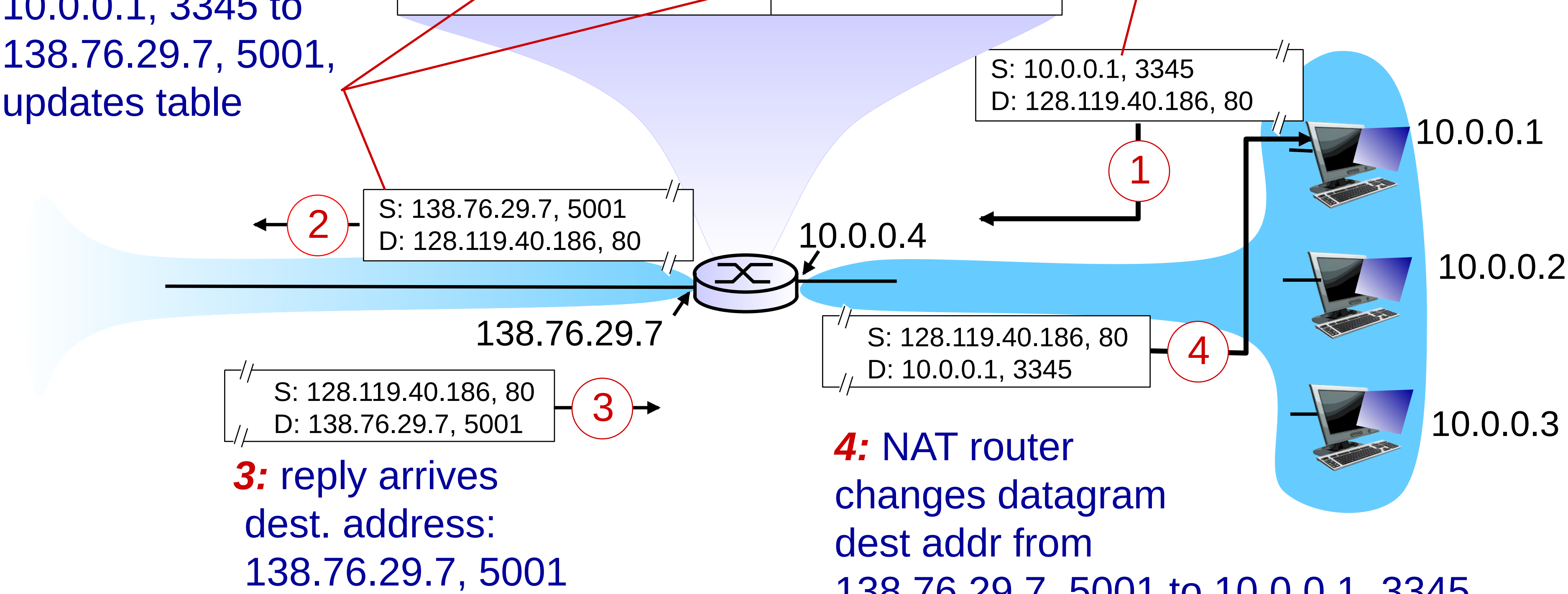


all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7 different

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address ~~translation~~

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……



IPv6: motivation

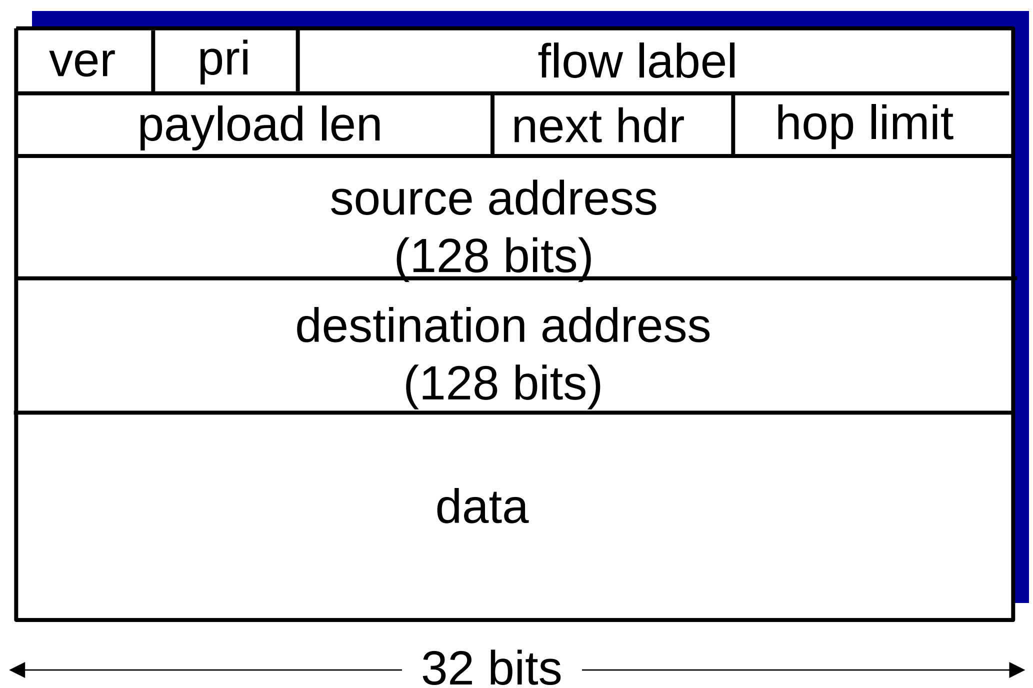
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

IPv6 datagram format

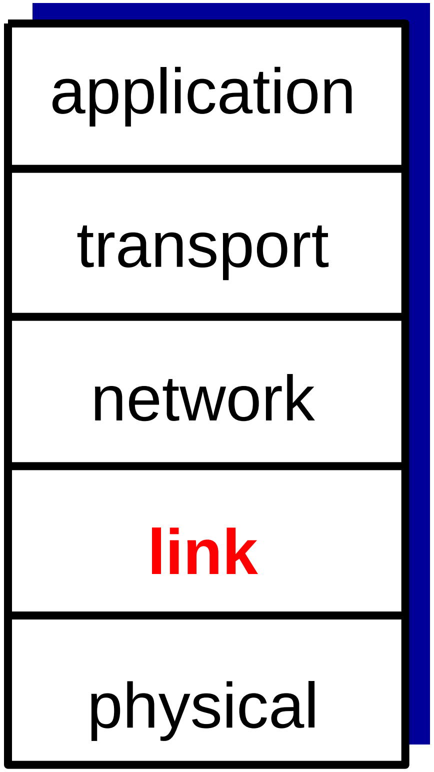
priority: identify priority among datagrams in flow
flow Label: identify datagrams in same "flow."
(concept of "flow" not well defined).
next header: identify upper layer protocol for data



Routing protocols

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

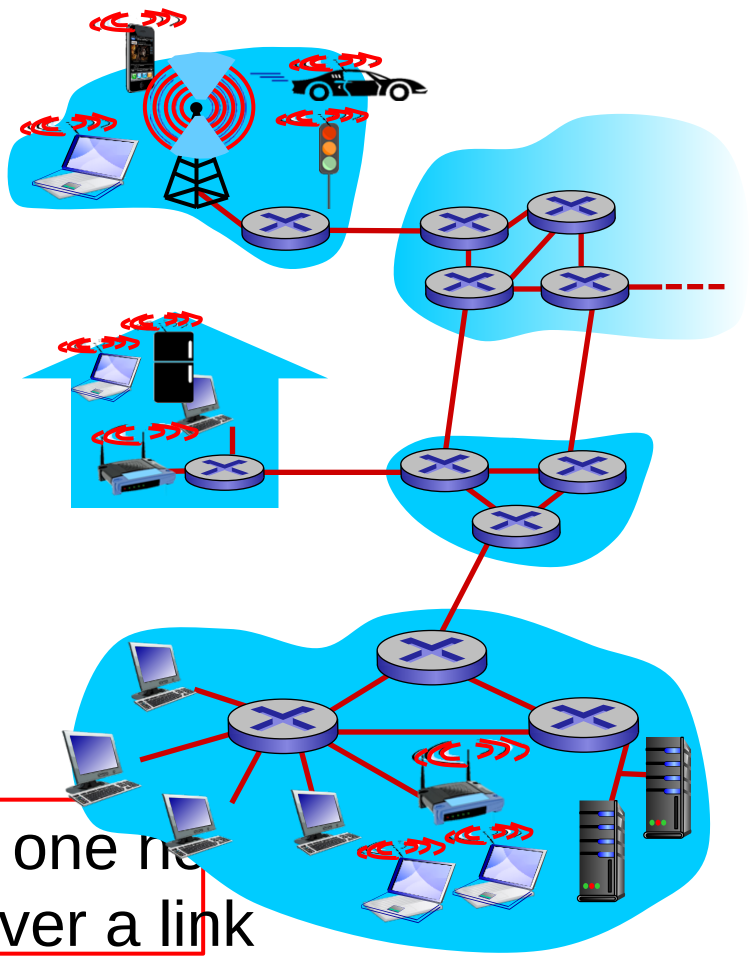
- “good”: least “cost”, “fastest”, “least congested”


application
transport
network
link
physical

Link layer: introduction

terminology:

- hosts and routers: nodes
- Links:
  - wired links
  - wireless links
- layer-2 packet: frame, encapsulates datagram



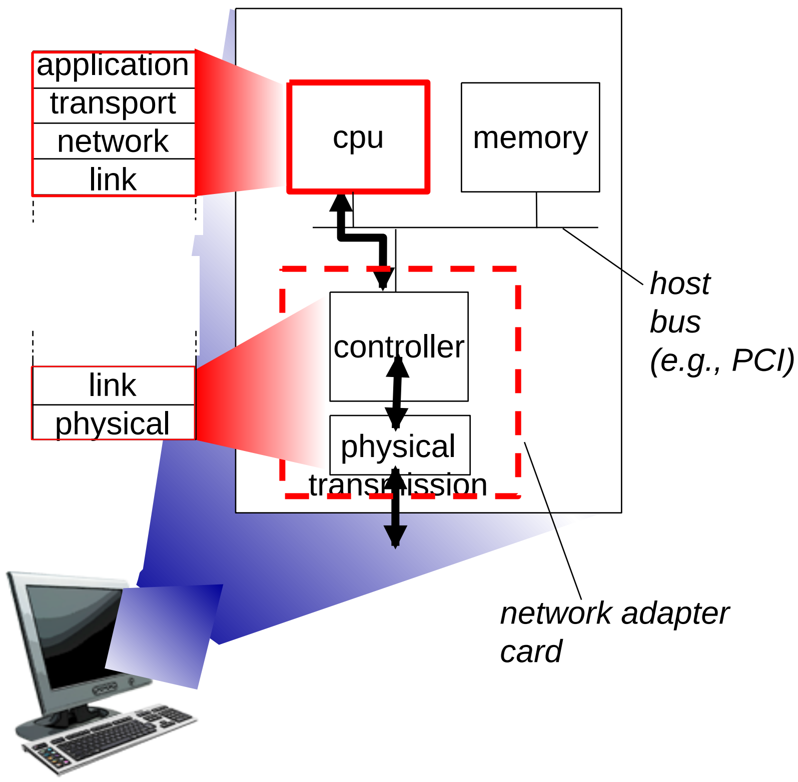
data-link layer: transfer frame from one n
to *physically connected neighbor* over a link

Link layer services

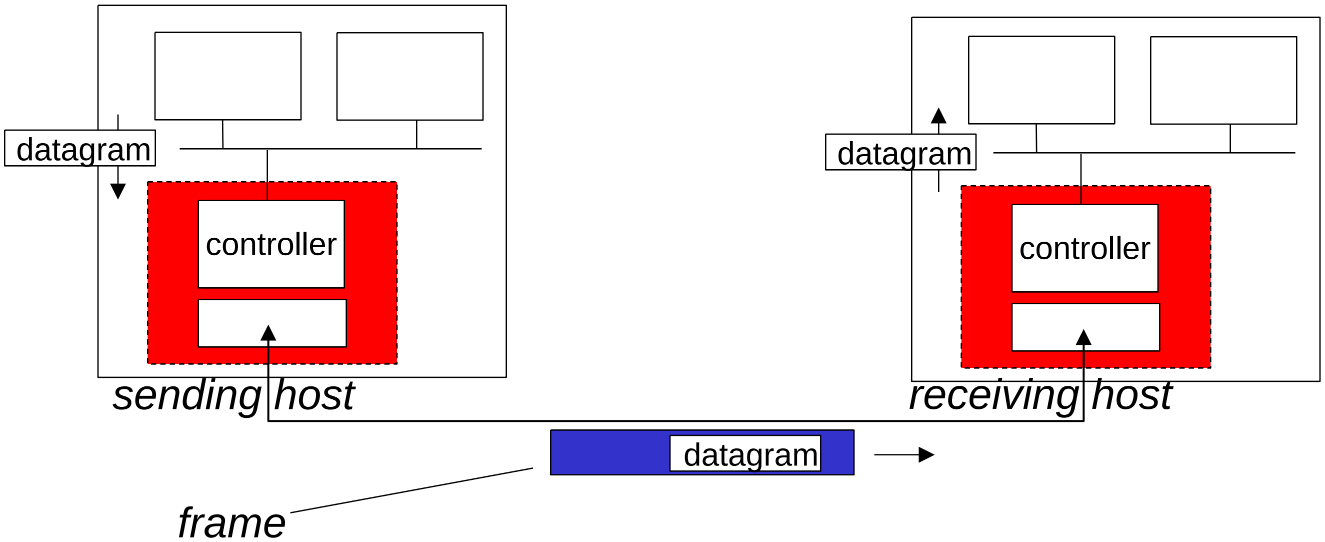
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - MAC addresses
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - Different from chapter 3!
  - solve high error rate and has error correction

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



Adaptors communicating

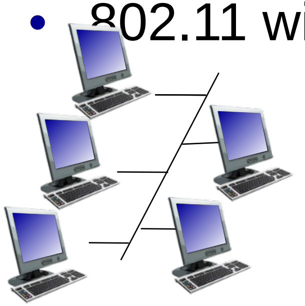


- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc.
  - extracts datagram, passes to upper layer at receiving side

Multiple access links, protocols

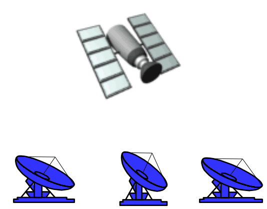
two types of "links":

- point-to-point
  - PPP for dial-up access
  - 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)

humans at a cocktail party (shared air, acoustical)

Multiple access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: *collision*

multiple access protocol: to avoid collision

- determine whose turn to use the channel

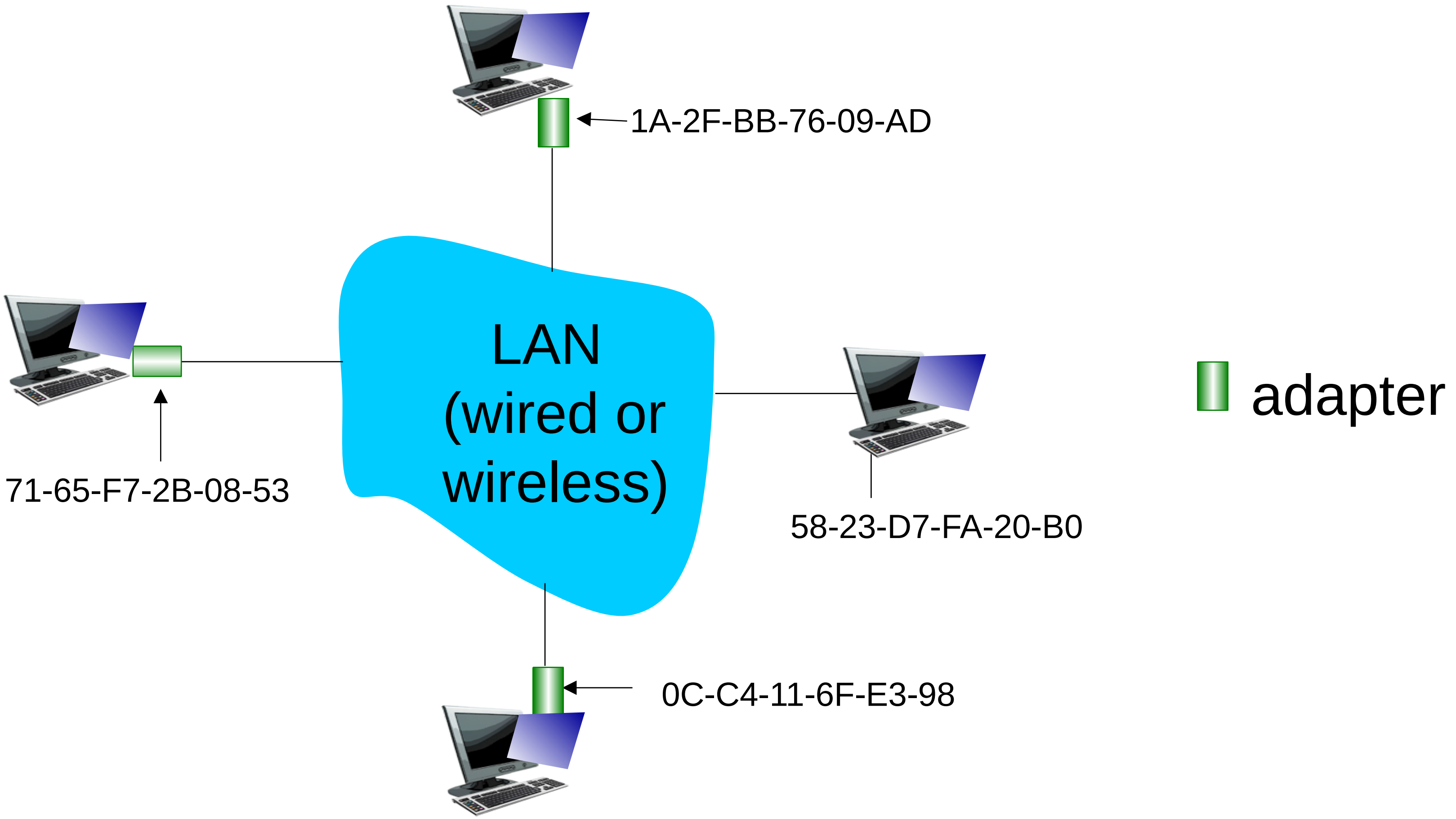
MAC addresses and ARP

- 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 to identify an interface*
  - 48 bit MAC address
  - e.g.: 1A-2F-BB-76-09-AD
  - MAC address portable while IP not portable

hexadecimal (base 16) notation
(each "numeral" represents 4 bits)

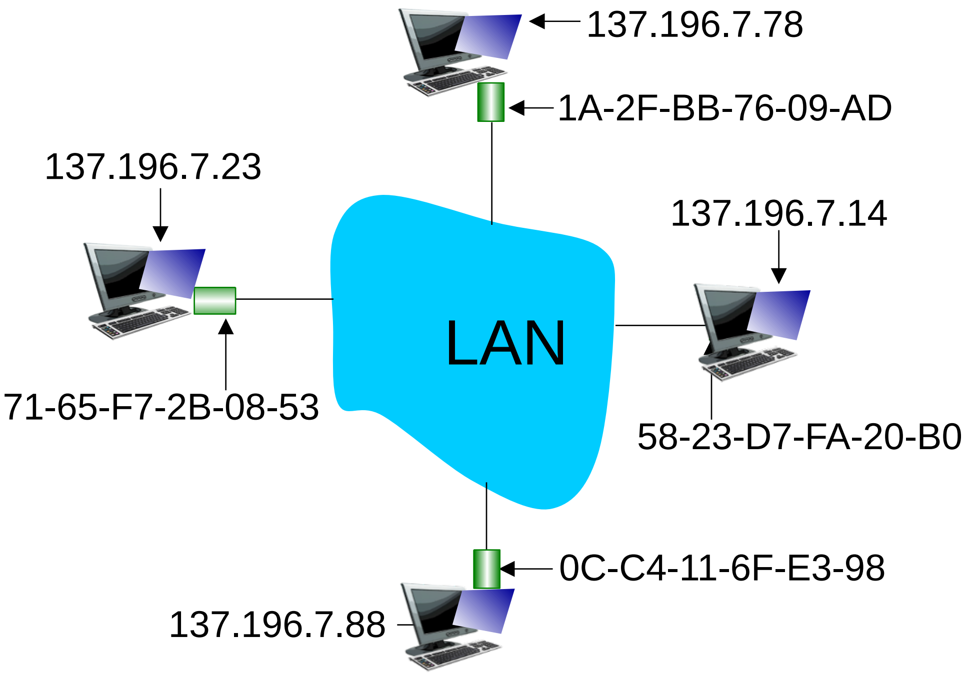
MAC address

each adapter on LAN has unique *MAC* address



ARP: address resolution protocol

Question: how to obtain the MAC address of an IP address?
Answer: ARP protocol



each node builds *ARP table*:

- < IP address; MAC address; TTL>
- TTL (Time To Live): expired record will be deleted (e.g. 20 min)

ARP protocol: same LAN

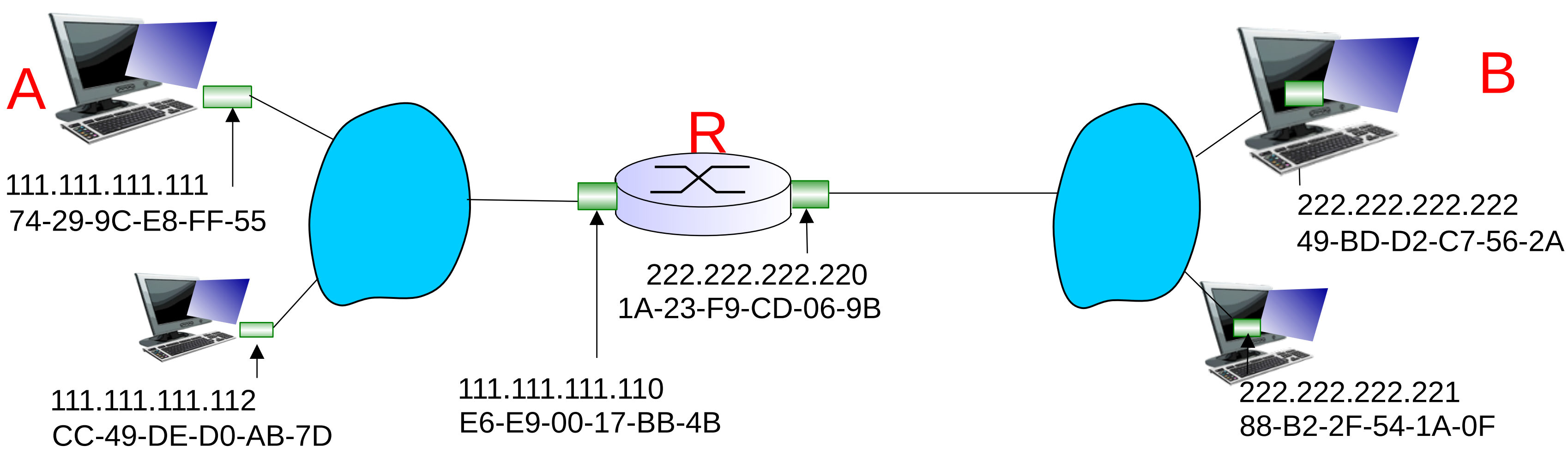
- A wants the MAC address of B.
- A broadcasts ARP query packet, containing B's IP address
  - destination MAC address = FF-FF-FF-FF-FF-FF
- Receiving ARP packet, B replies to A with its (B's) MAC address, by unicast

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

Addressing: routing to another ~~LAN~~

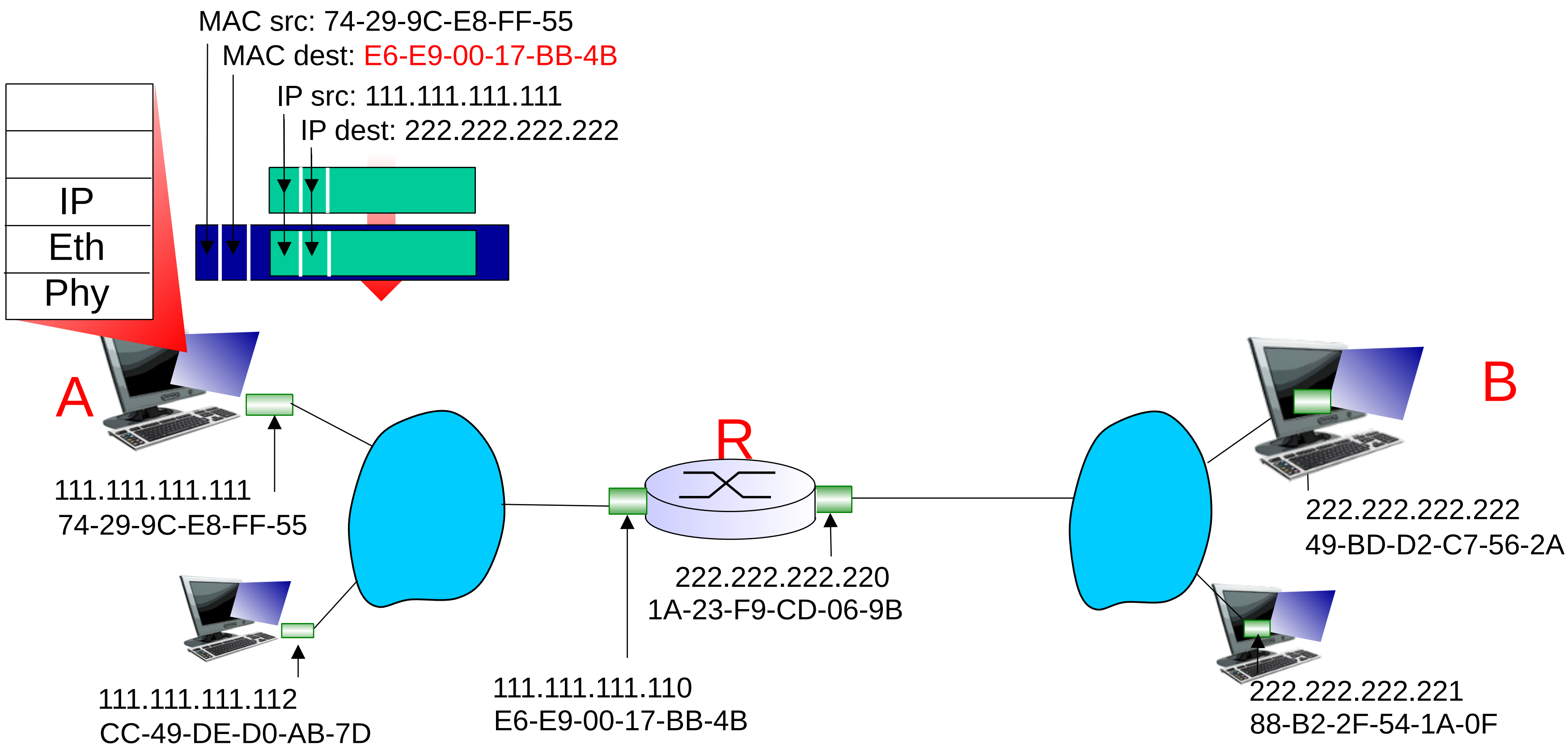
walkthrough: send datagram from A to B via R

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



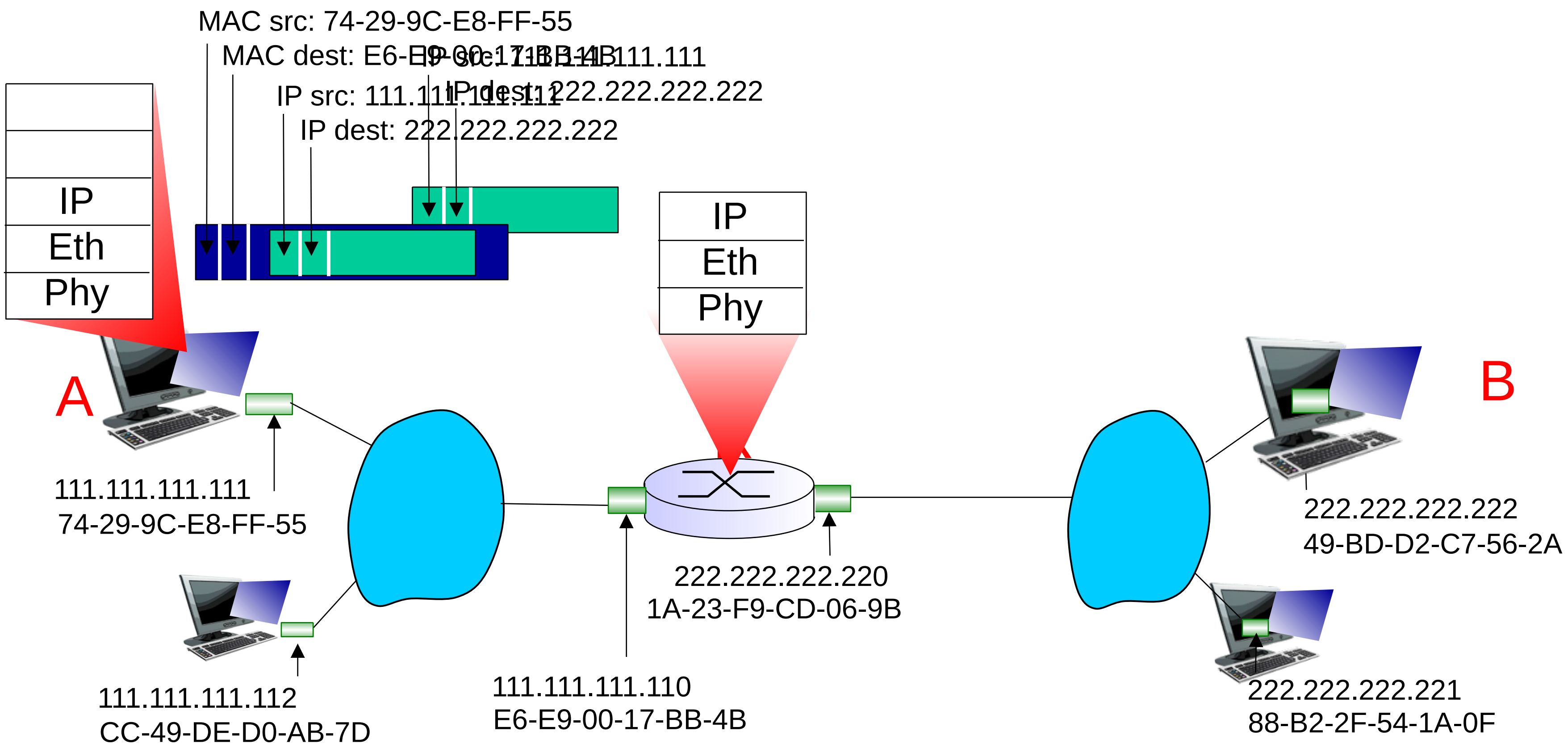
Addressing: routing to another ~~LAN~~

- 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



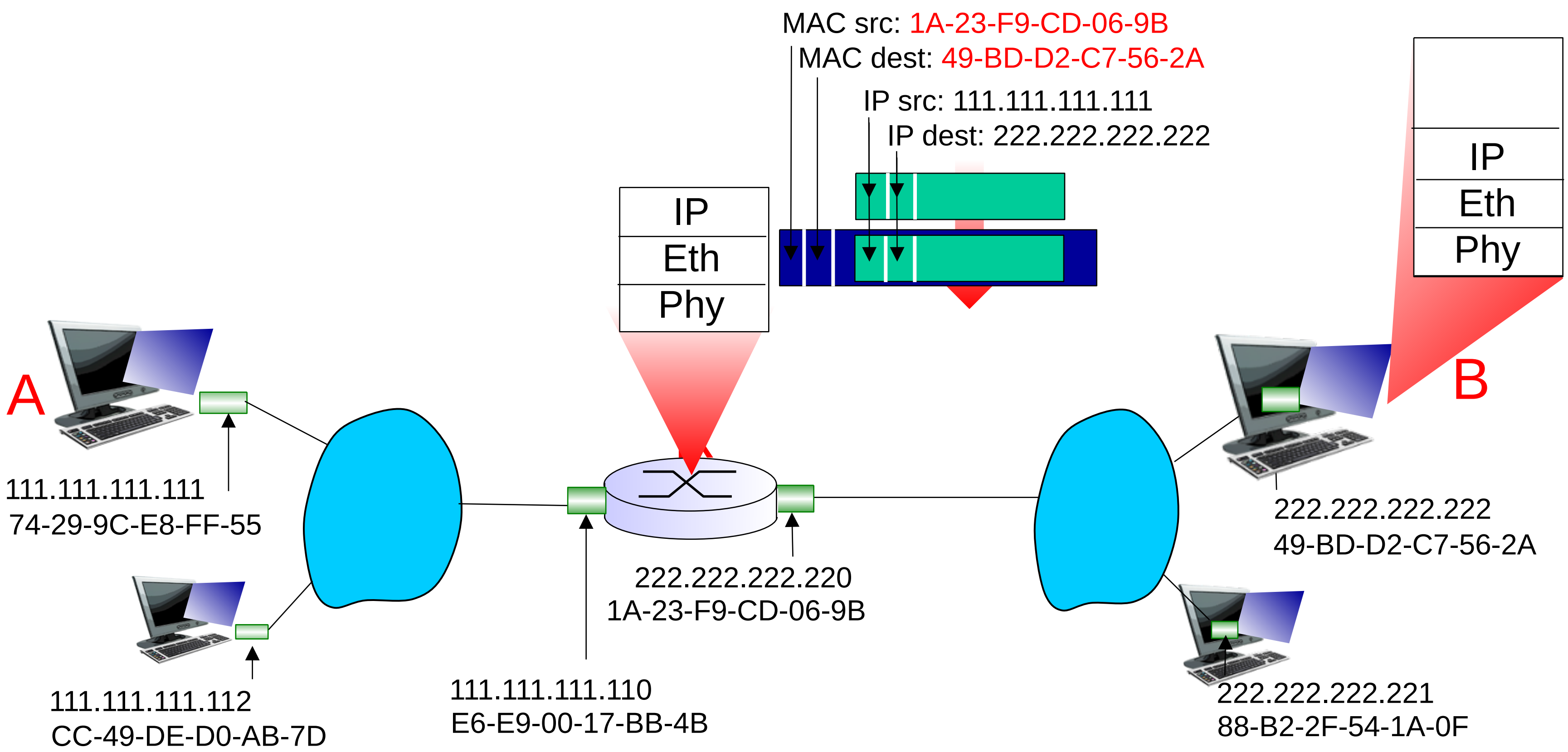
Addressing: routing to another LAN

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



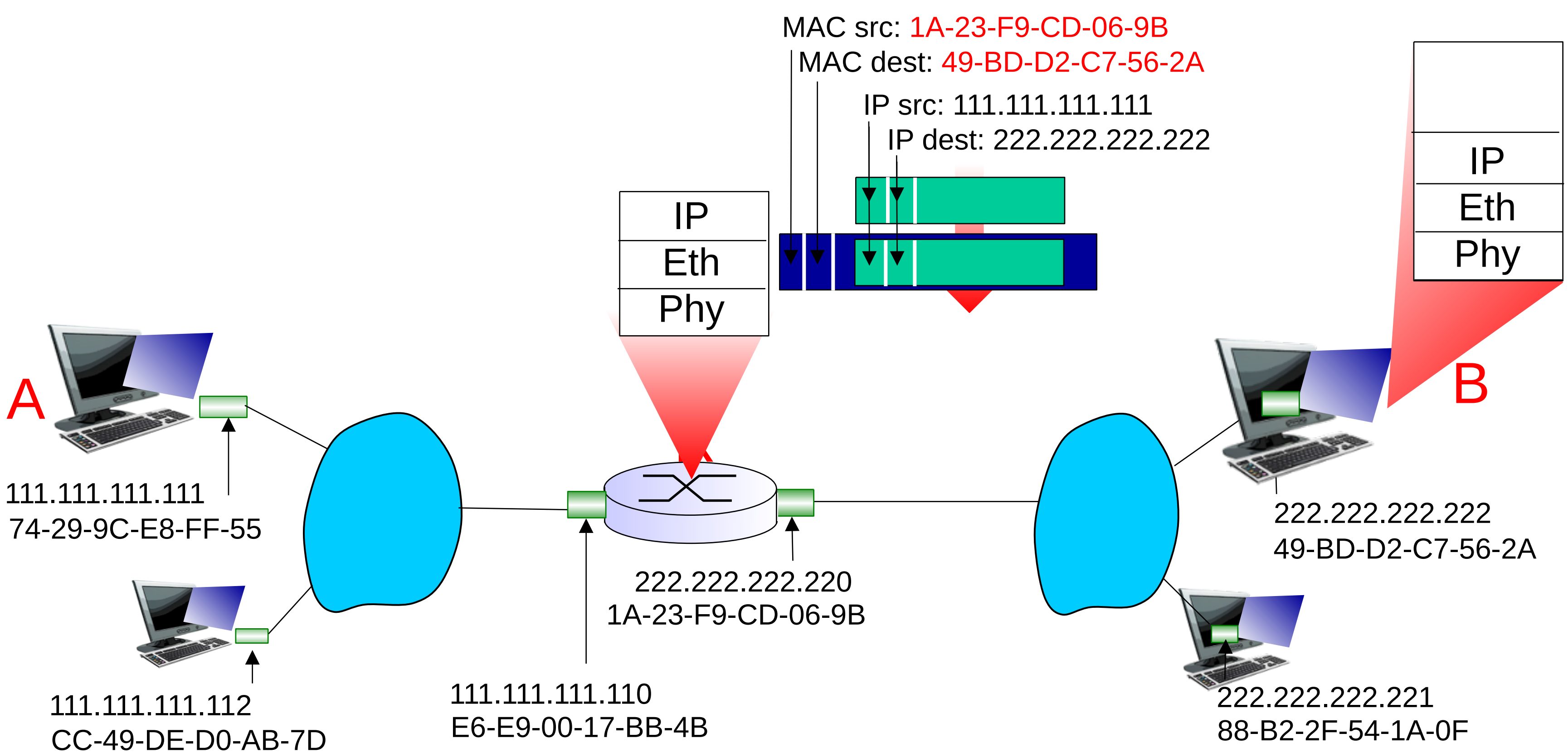
Addressing: routing to another ~~LAN~~

- 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



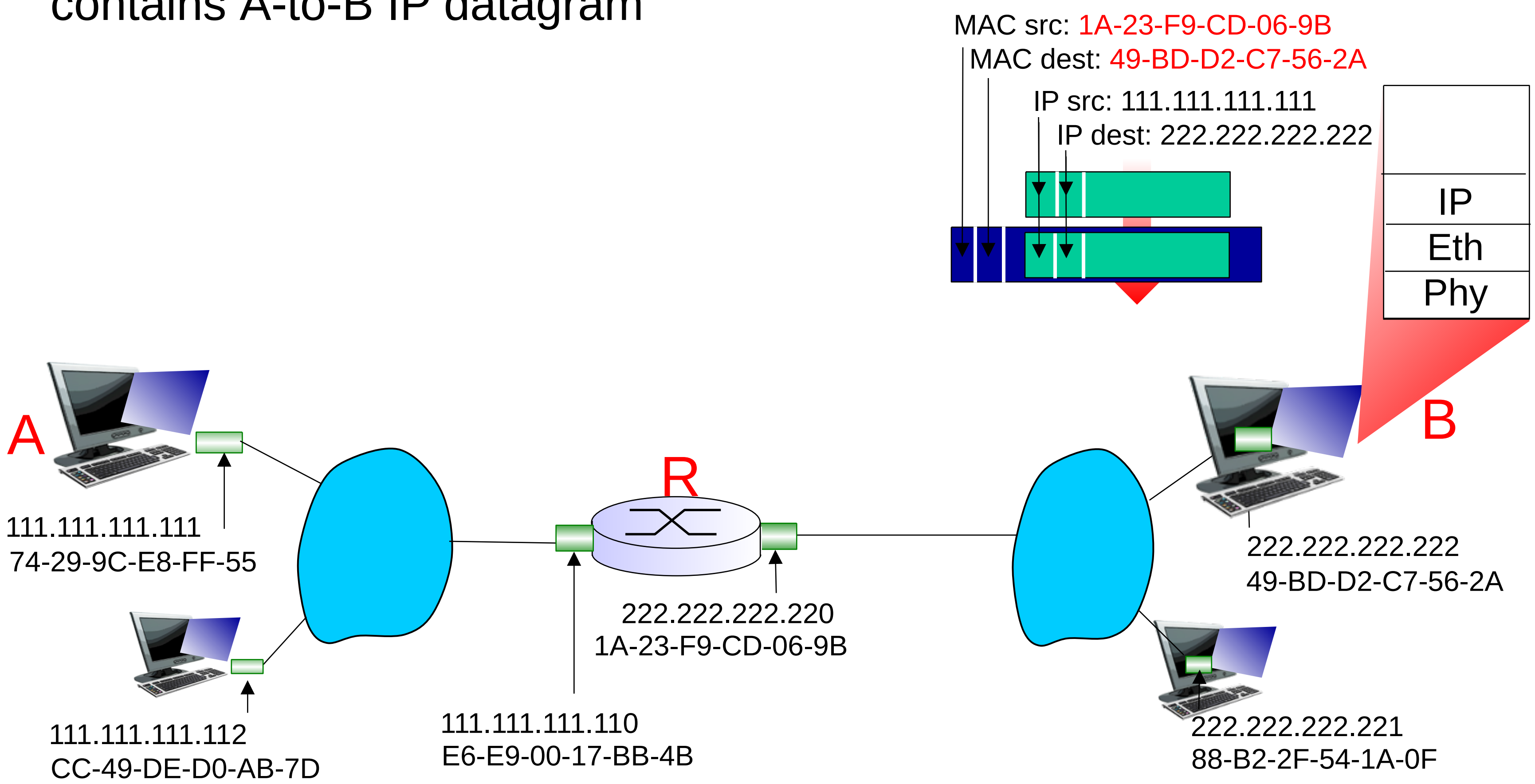
Addressing: routing to another LAN

- 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



frame structure for Ethernet ~~link~~

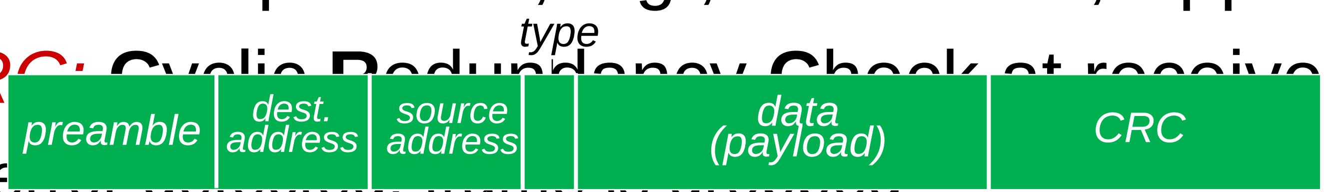
sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

Ethernet frame structure (more)

- *addresses:* 6 byte source, destination MAC addresses
  - if adapter sees its MAC address as the destination address, or sees 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., Novell IPX, AppleTalk)
- *CRC:* **C**yclic **R**edundancy **C**heck at receiver
  - error detected: frame is dropped

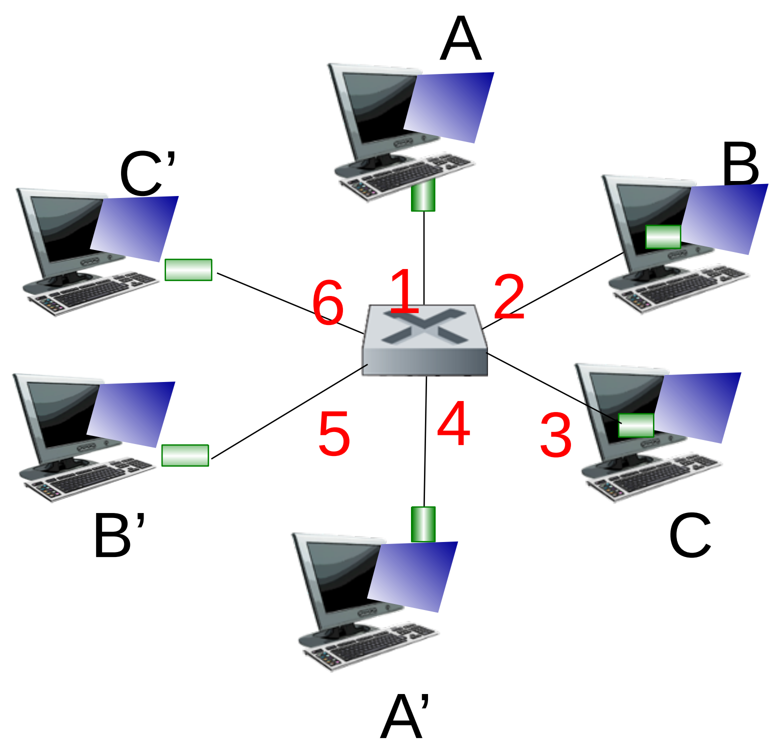


Ethernet switch

- link-layer device:
  - store, forward Ethernet frames
  - examine and forward the incoming frame to its destination MAC address
- *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
- *switching:* A-to-A’ and B-to-B’ can transmit simultaneously, without collisions

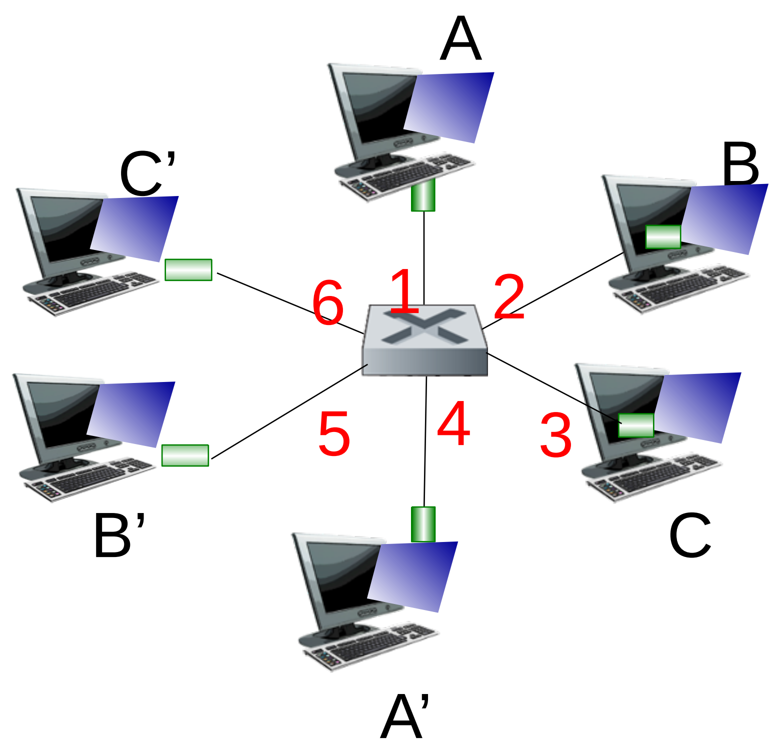


switch with six interfaces (1,2,3,4,5,6)

Switch forwarding table

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

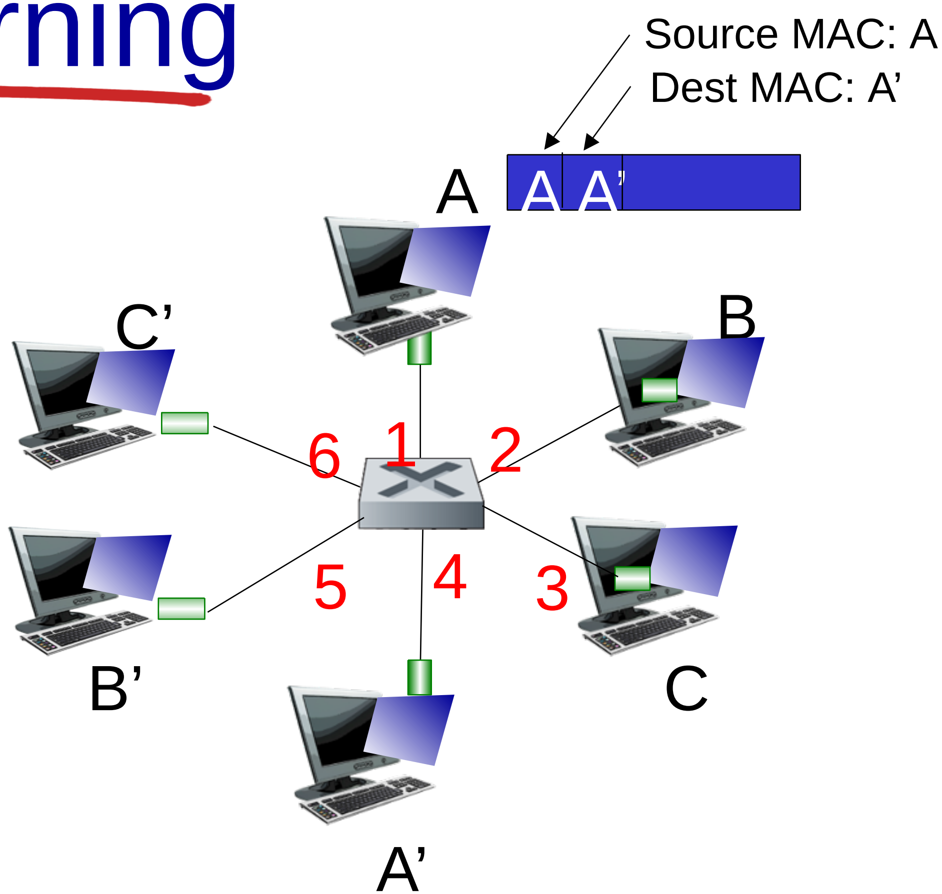
- *A:* each switch has a switch table, each entry:
  - (MAC addr, interface, TTL)
  - looks like a routing table!



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

Switch: self-learning

- switch *learns* which hosts can be reached through
  - an incoming frame from the sending host
  - records (mac address, interface) pair in switch table



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

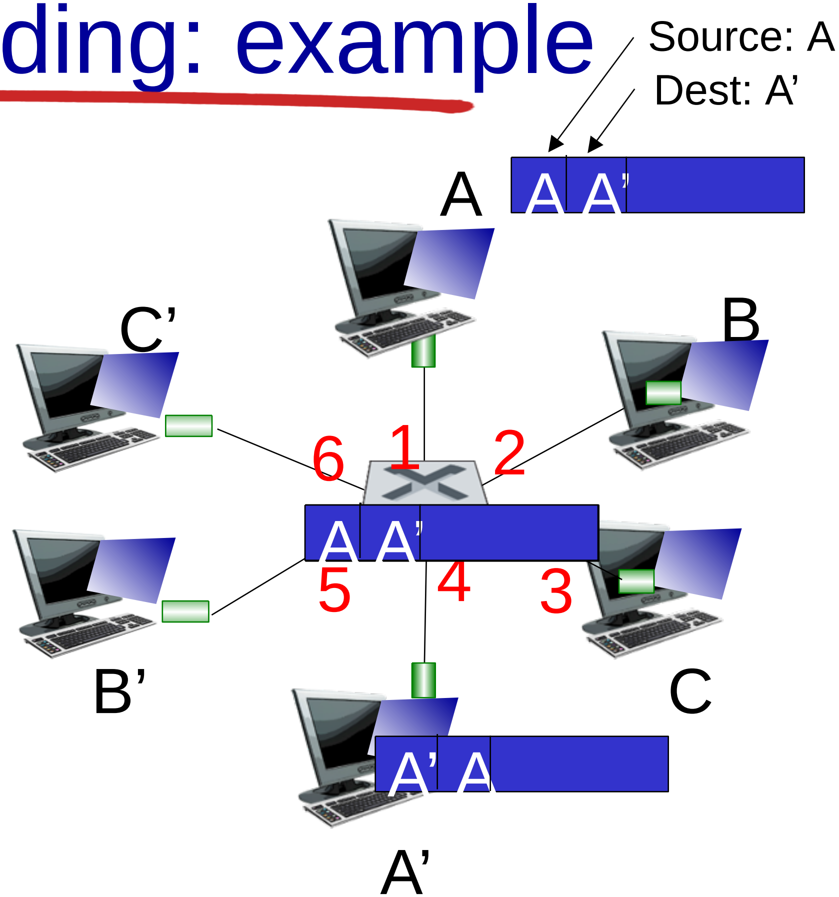
Switch: frame filtering/forwarding

when **frame** received at switch:

1. record (MAC addr, incoming link) of **sender** in switch table
2. if entry for **destination** MAC of **frame** found in the table
   then {
   if destination is on the incoming link
   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: *flood*
- destination A location known: selectively send on just one link

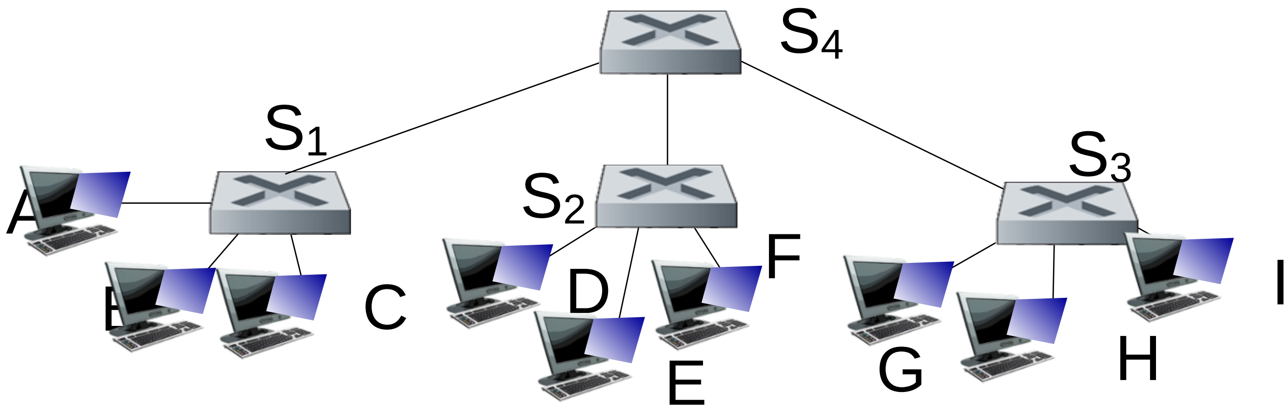


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

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

Institutional network