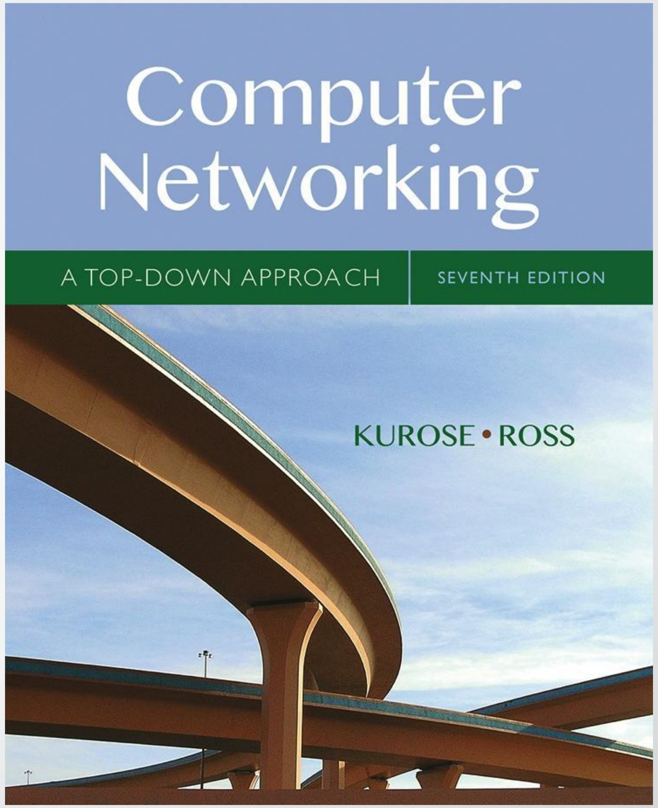
Data Link Layer

CSE421 –Computer Networks

Department of Computer Science and Engineering
School of Data & Science

Based on Chapter 6
The Link Layer and LANs



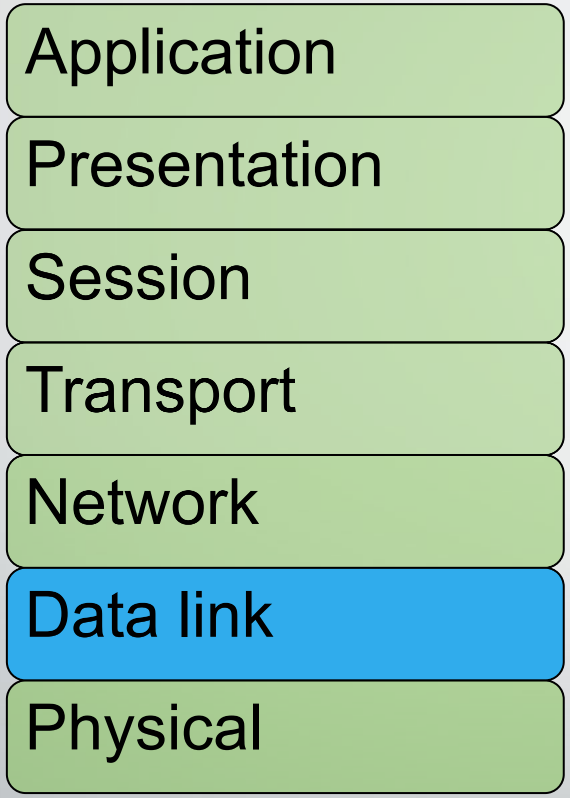
Computer Networking: A Top Down Approach
7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

- *The slides are adapted from Kurose and Ross, Computer Networks 7th edition, Kurose and Ross.*

Chapter 6: Link layer and LANs

Objectives:

- understand principles behind link layer services:
  - error detection, correction (done in CSE320)
  - sharing a broadcast channel: multiple access (done in CSE320)
  - Framing - link layer addressing
  - ARP
  - local area networks: Ethernet

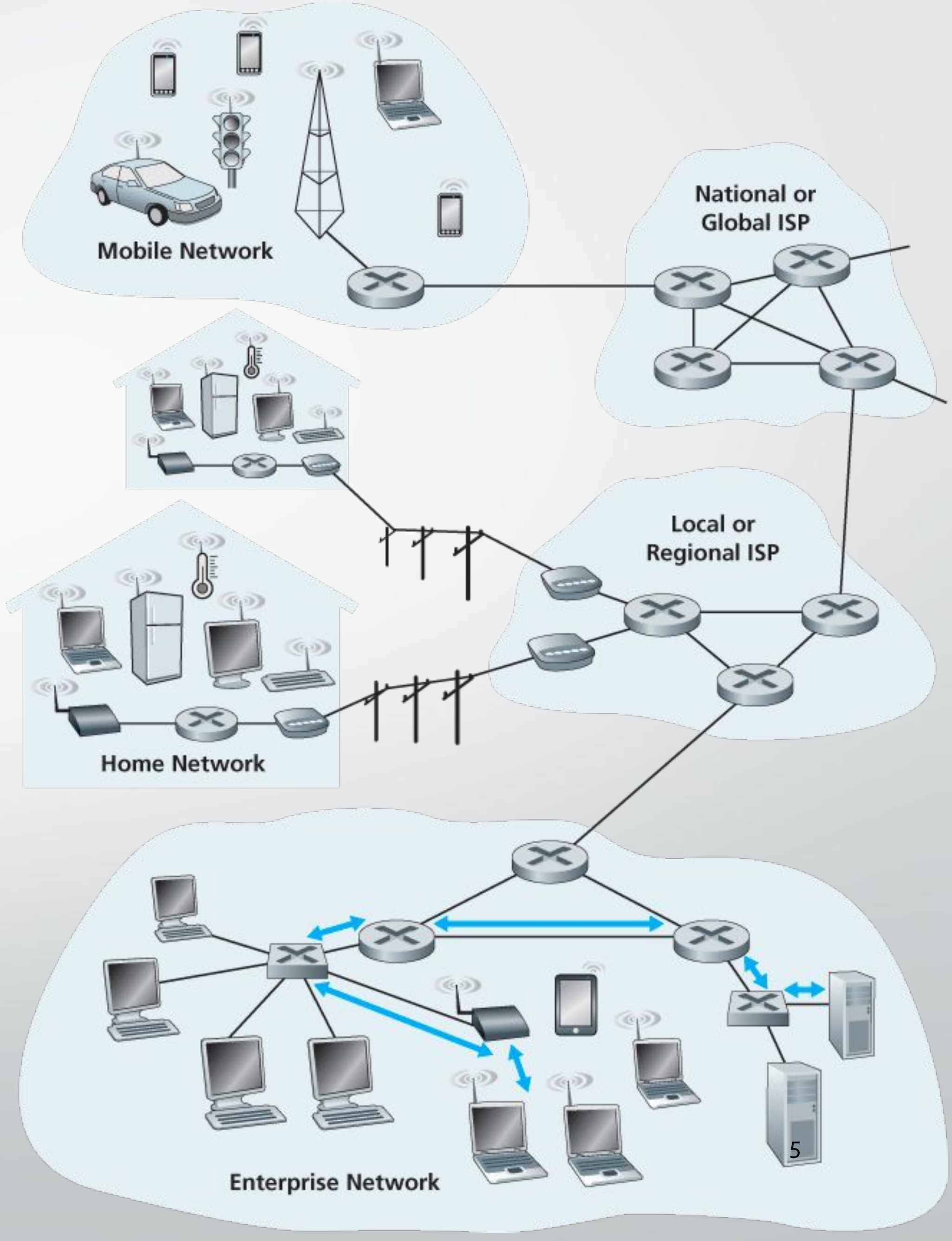


Introduction to Link Layer

Link Layer Terminology

- Nodes : hosts and routers
- Links:
  - wired links
  - wireless links
- Frame : layer-2 packet

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

Transportation analogy:

- trip from Home to Cox's Bazaar
  - Uber Car : Home to Dhaka Airport
  - Plane: Dhaka to Chittagong
  - Bus: Chittagong to Cox's Bazaar
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

Link layer functions/services

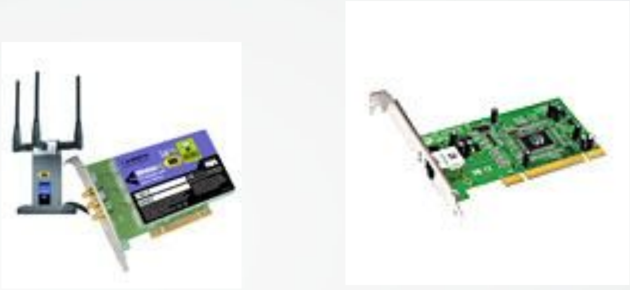
- *Framing*
  - encapsulate datagram into frame, adding header, trailer
    - Various information added such as the various protocols
  - "MAC" addresses used in frame headers to identify source, destination
    - different from IP address!
- *Link access:*
  - how to send a frame to the link
  - channel access if shared medium
    - Control/Avoid clashes in multi-access networks!
  - rules to follow when sending the link
- *Reliable delivery between adjacent nodes*
  - we learned how to do this already (Transport Layer)
  - 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?

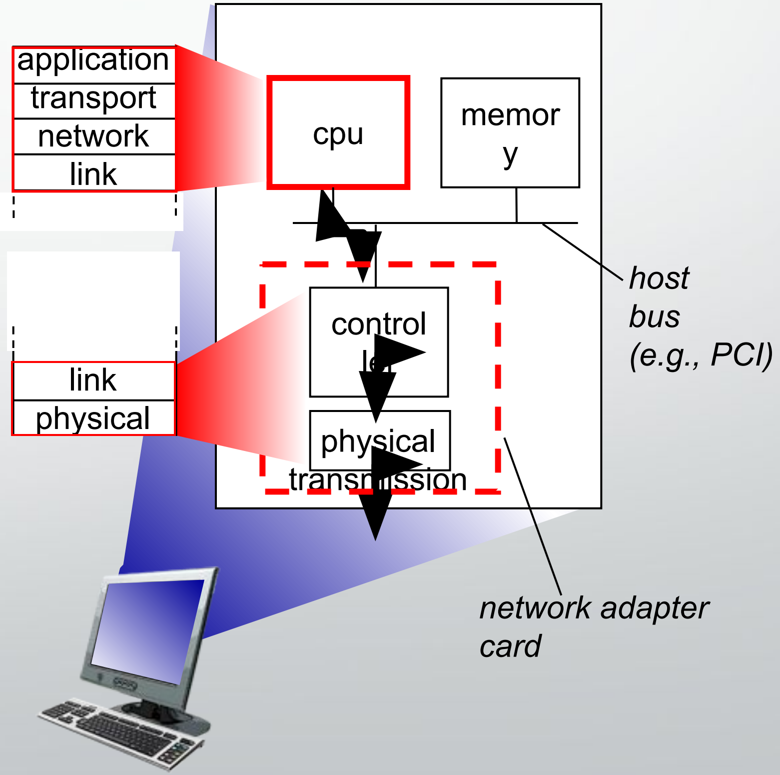
Link layer services (more)

- *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 (there are various protocols)
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

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





Our objectives

Objectives – Part 1

- Link Layer Addressing
  - MAC Address
  - Types of MAC Addresses
- ARP
- ARP within LAN
- LAN Switch

Link Layer Addressing

IP Address vs MAC Address

IP address

- 32 bits
- Dotted decimal notation
  - Example : 192.168.10.1
- *Network-layer* address for interface
- Hierarchal
  - Not portable
- Function

MAC address

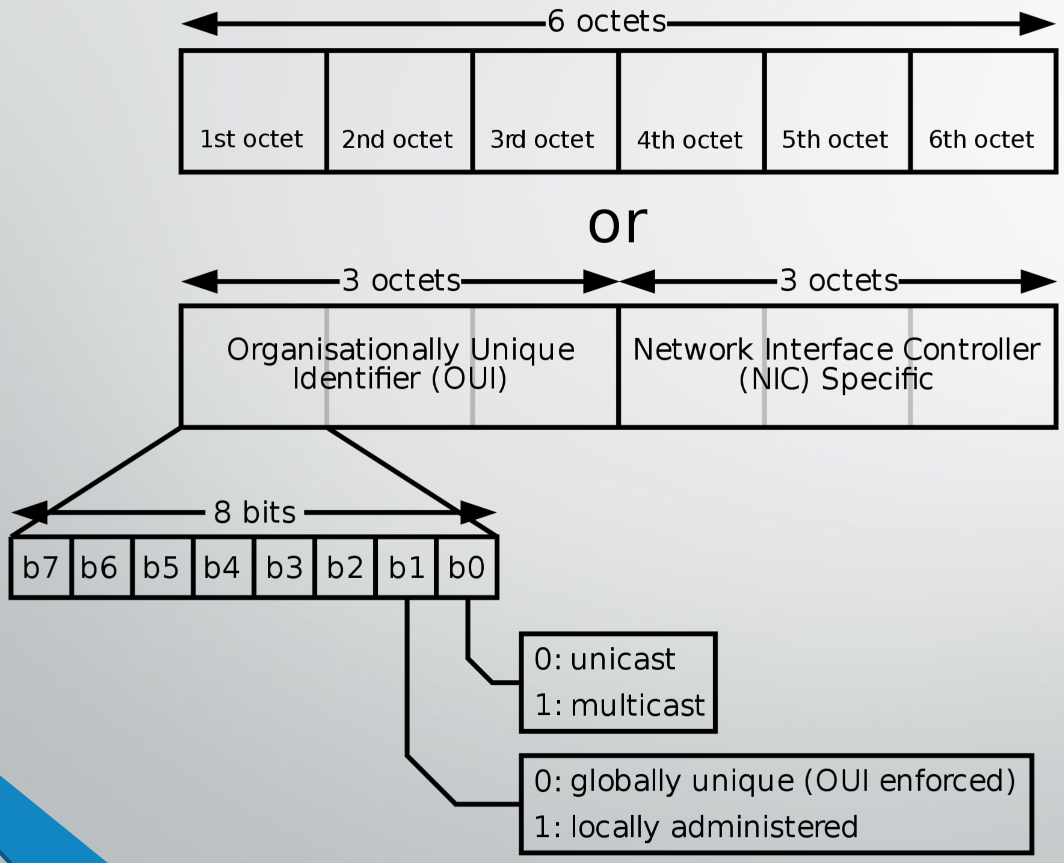
- 48 bits
- 12 Hexadecimal digits
  - Example : 1A-2F-BB-76-09-AD
- *Data Link-layer* address for interface
- Flat
  - portable
- Function

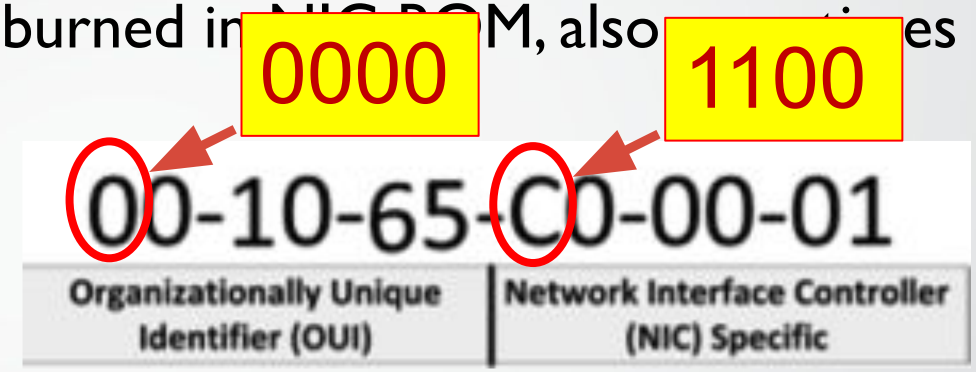
MAC or LAN or Physical or Ethernet addresses (more)

- 48 bits MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
- MAC address allocation administered by IEEE
- Manufacturer buys portion of MAC address space (to assure uniqueness)
- Analogy:
  - MAC address: like National ID
  - IP address: like Postal Address

MAC Address

- 48 bits MAC address (for most LANs) burned in NIC ROM, also sometimes software settable





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

Different display formats:

- 0000.0c43.2e08
- 00:00:0c:43:2e:08
- 00-00-0C-43-2E-08

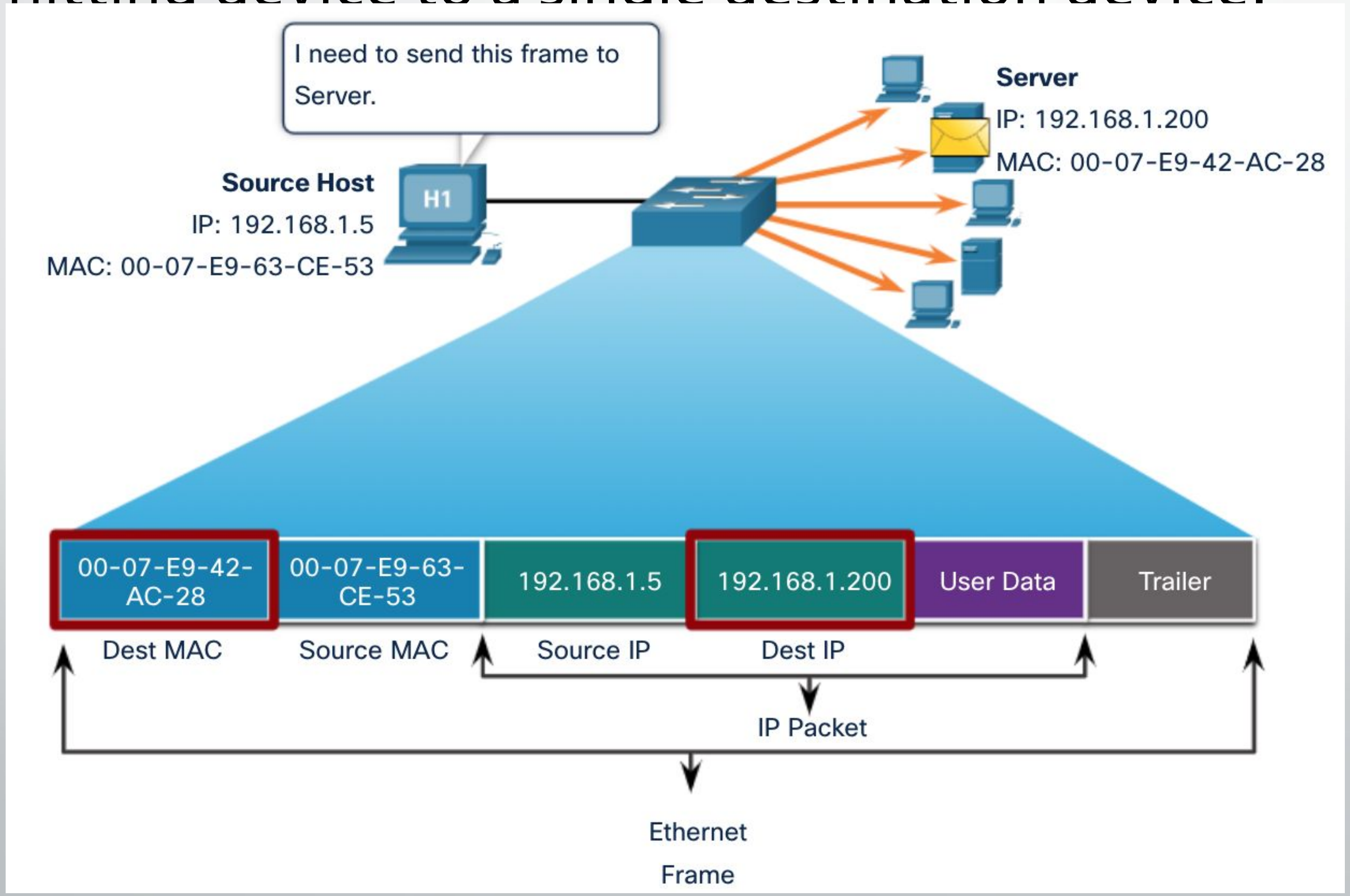
Types of MAC Address

MAC Address

- Unicast
- Multicast
- Broadcast

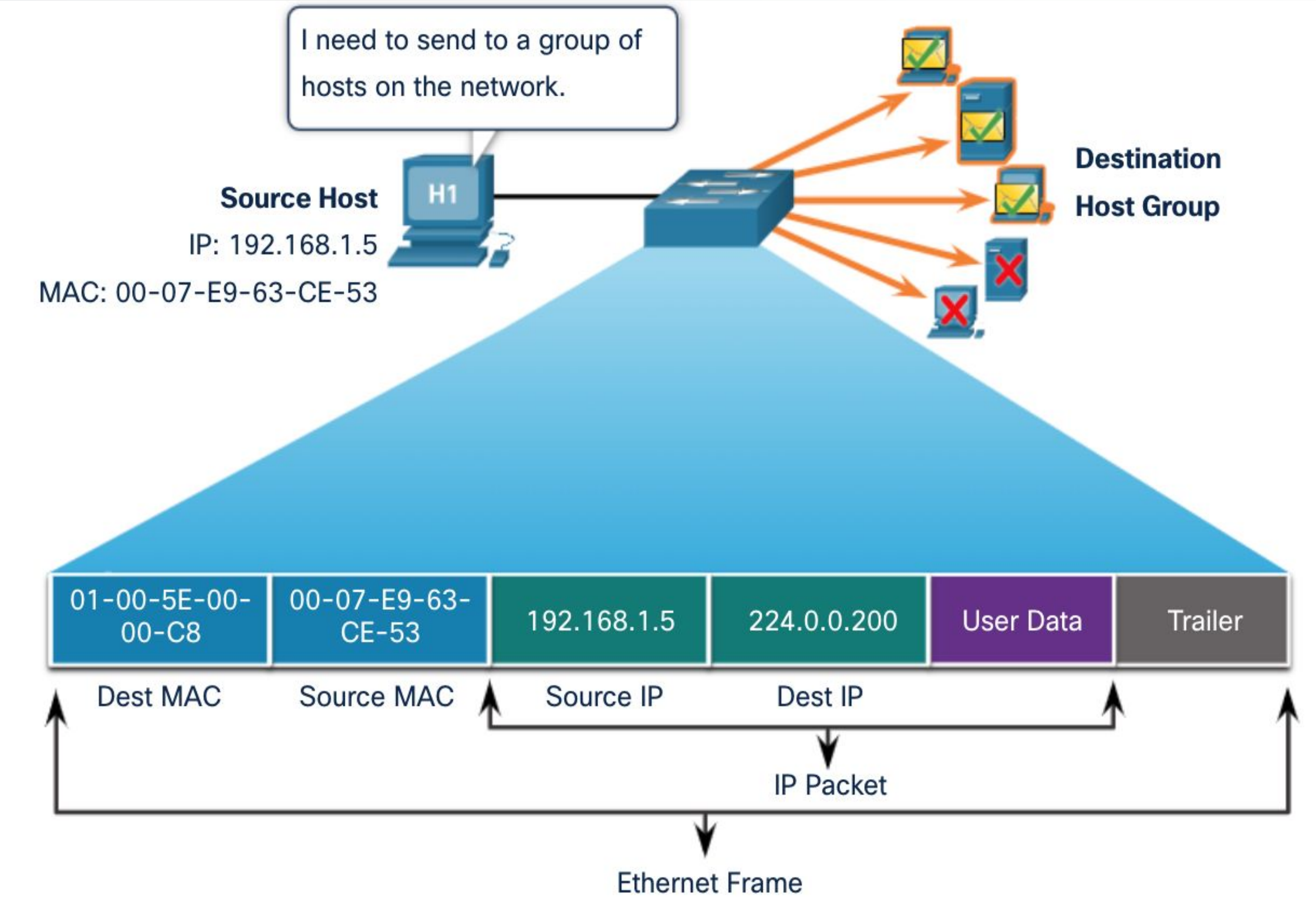
Unicast MAC Addresses

- The unique address used when a frame is sent from a single transmitting device to a single destination device.



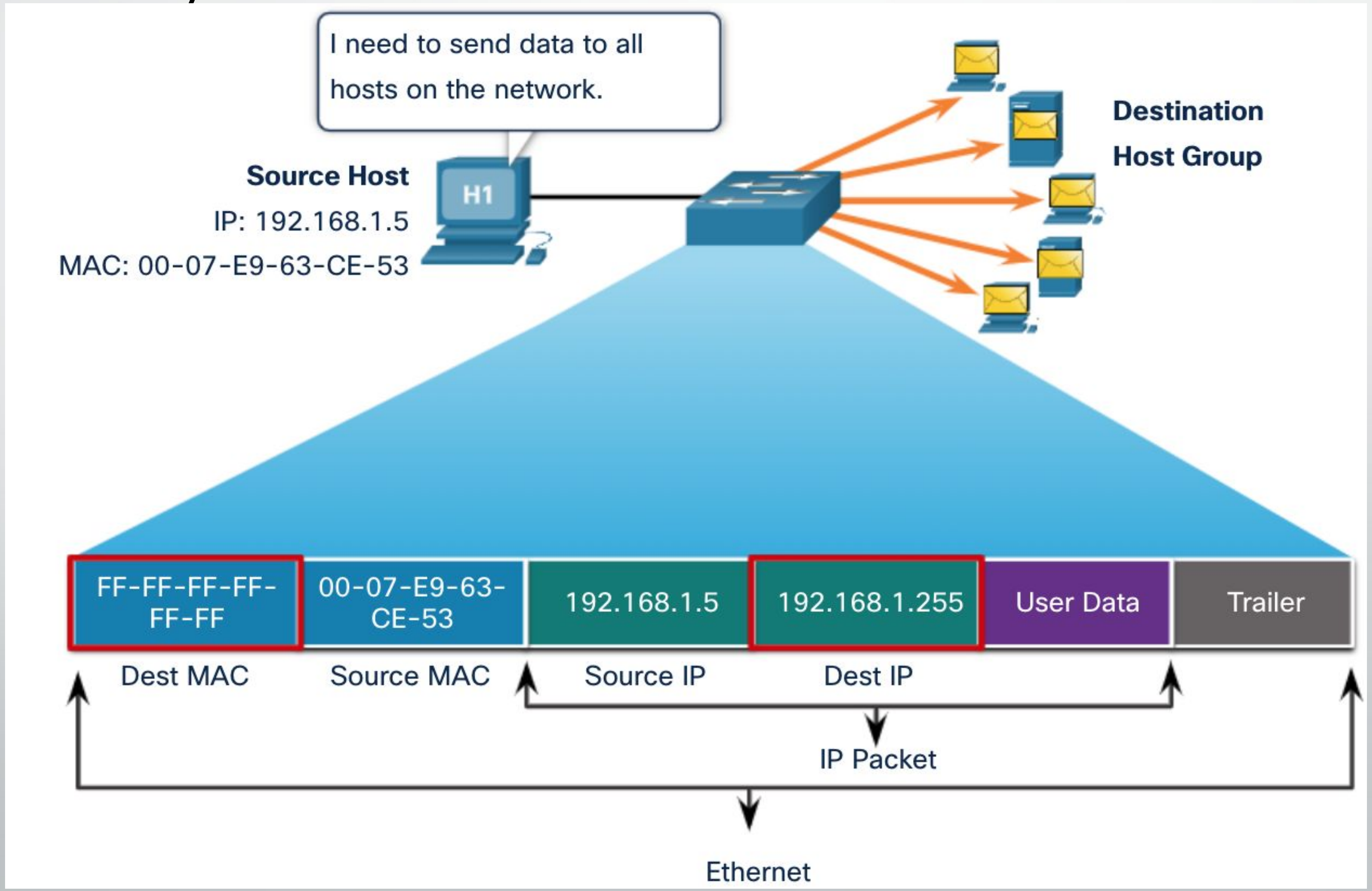
Multicast MAC Addresses

- " 01-00-5E" in an IPv4 multicast packet



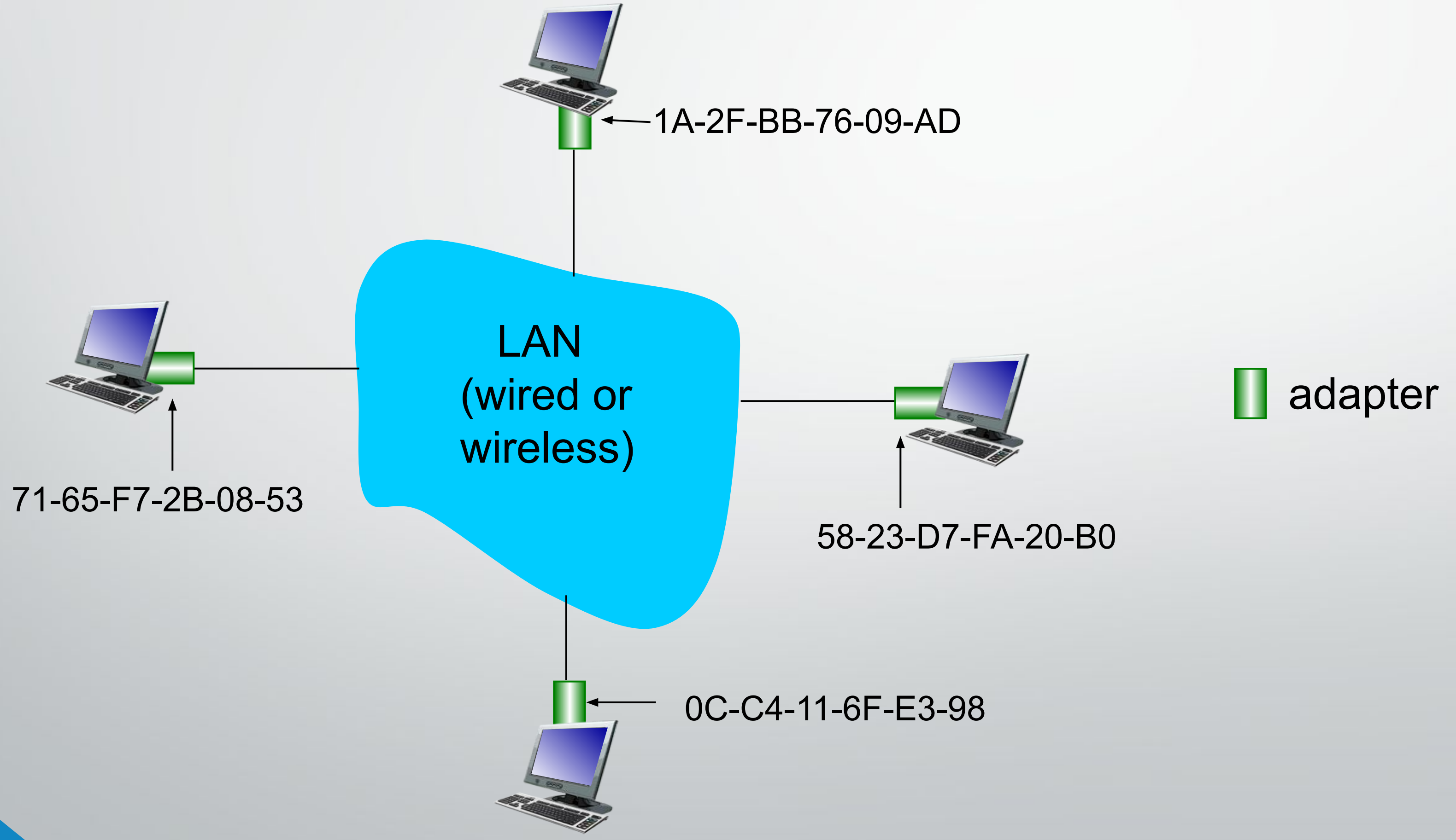
Broadcast MAC Address

- A destination MAC address of FF-FF-FF-FF-FF-FF
- To be processed by all devices in the network



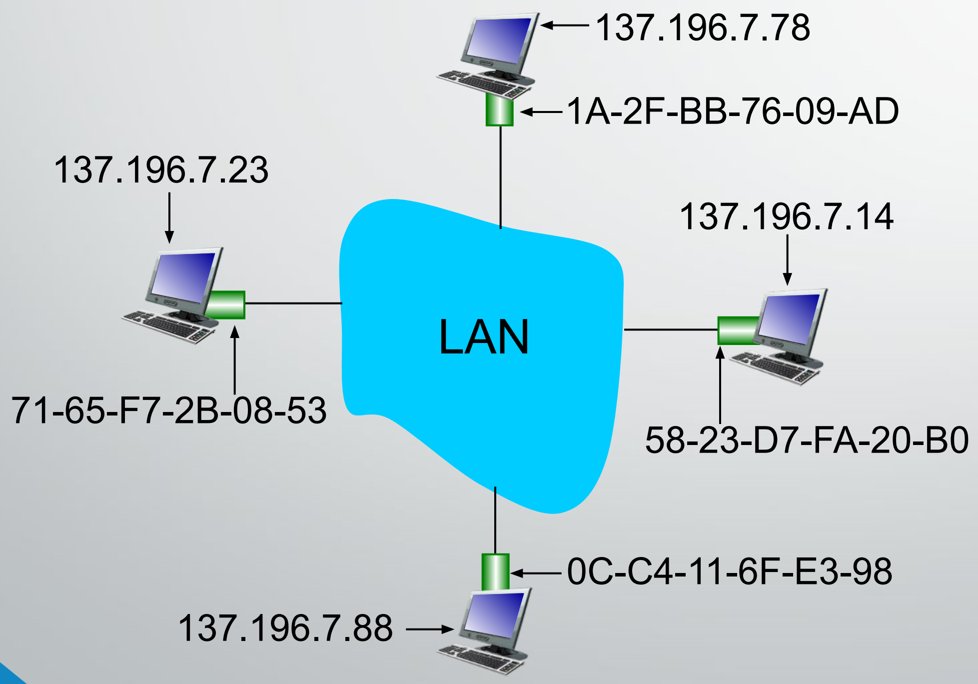
LAN addresses and ARP

each adapter on LAN has unique *LAN* address



ARP: address resolution protocol

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



- *ARP*
- *Mapping _IP Add___ to _MAC Add____*
- *ARP table*
  - IP
  - MAC address
  - TTL (Time To Live) Or Age
    - time after which address mapping will be forgotten (typically 20 min)

ARP Tables

```
C:\WINDOWS\system32\cmd.exe

C:\>arp -a

Interface: 192.168.0.2 --- 0x2
  Internet Address      Physical Address      Type
  192.168.0.1           00-0a-cd-00-0d-1d     dynamic

C:\>
```

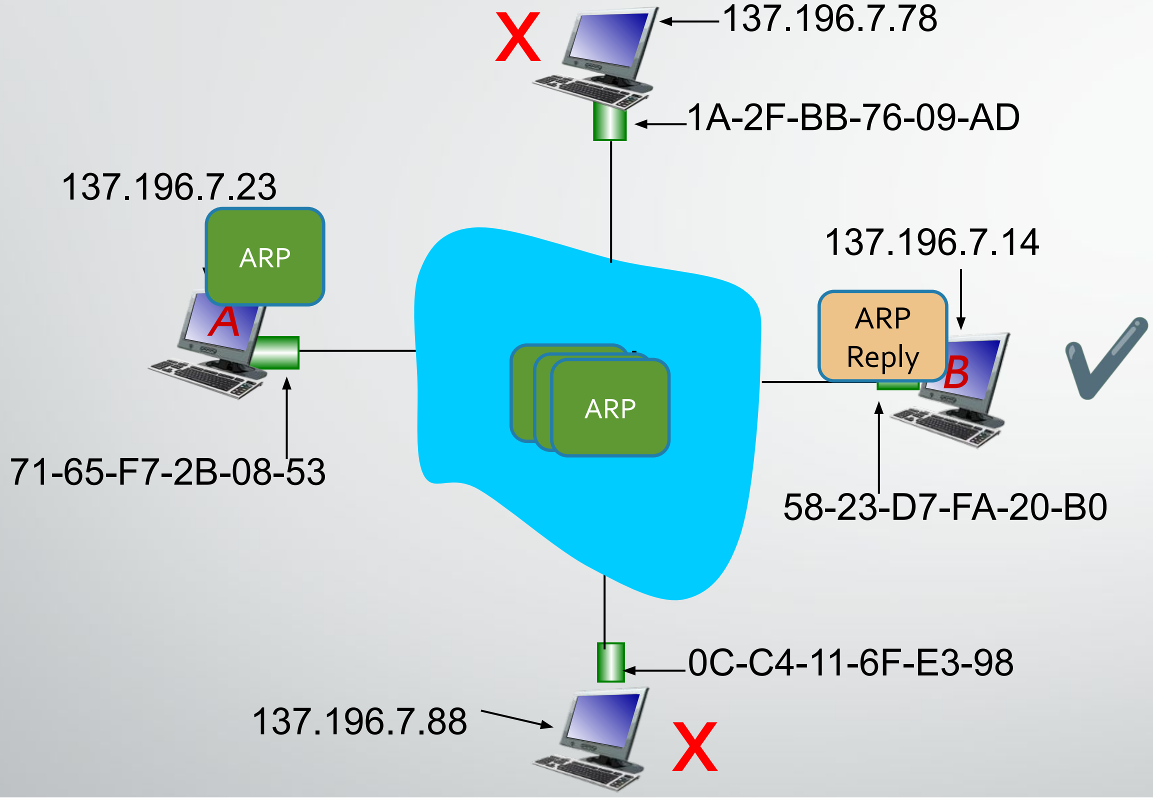
IP Adress MAC Adress ARP Type

Host or PC

```
R1#sh ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  192.168.1.1             -   ca02.238f.0008  ARPA   FastEthernet0/0
Internet  192.168.1.2             6   0050.7966.6800  ARPA   FastEthernet0/0
Internet  192.168.2.1             -   ca02.238f.0006  ARPA   FastEthernet0/1
Internet  192.168.2.2             6   0050.7966.6801  ARPA   FastEthernet0/1
```

Router

ARP: address resolution protocol



A wants to send datagram to B

71-65-F7-2B-08-53	58-23-D7-FA-20-B0	137.196.7.23	137.196.7.14	ARP Reply
FF-FF-FF-FF-FF-FF	71-65-F7-2B-08-53	137.196.7.14	137.196.7.23	ARP Request
Dest MAC	Source MAC	Dest IP Add	Source IP Add	ARP Req/Reply

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
  - Soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

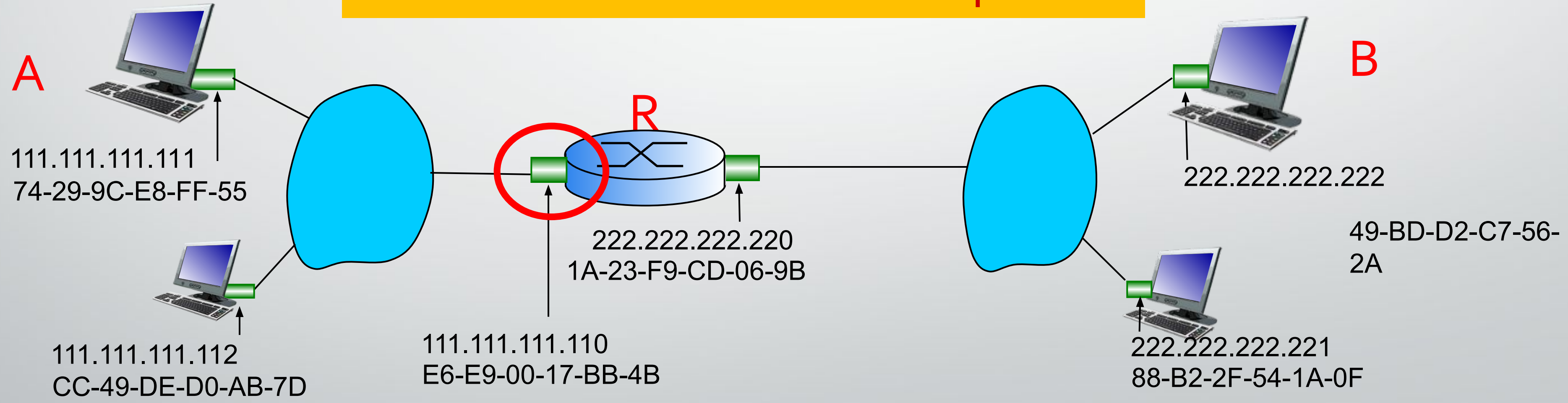
Addressing: routing to another LAN

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
- What will be the destination MAC Address?

ARP- To know B's MAC address as B's IP address is known

Routers never forward broadcast packets!

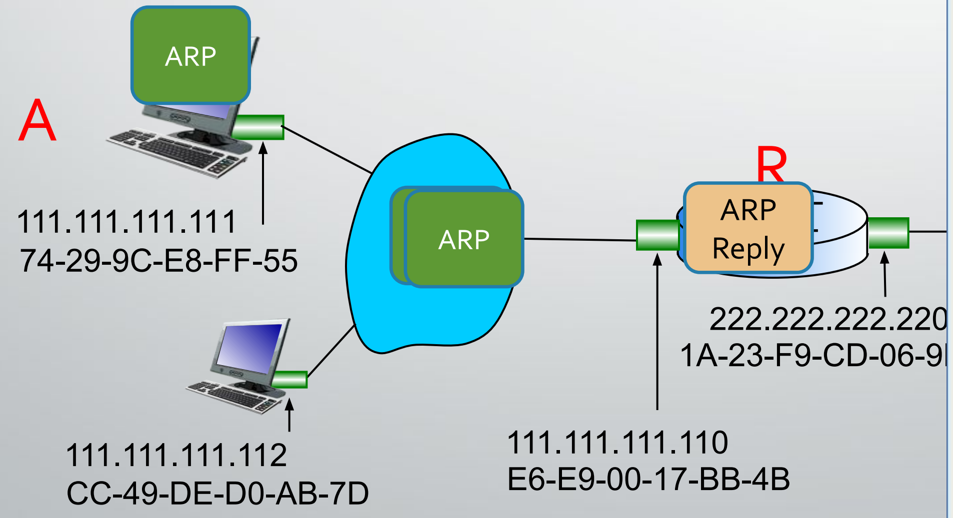


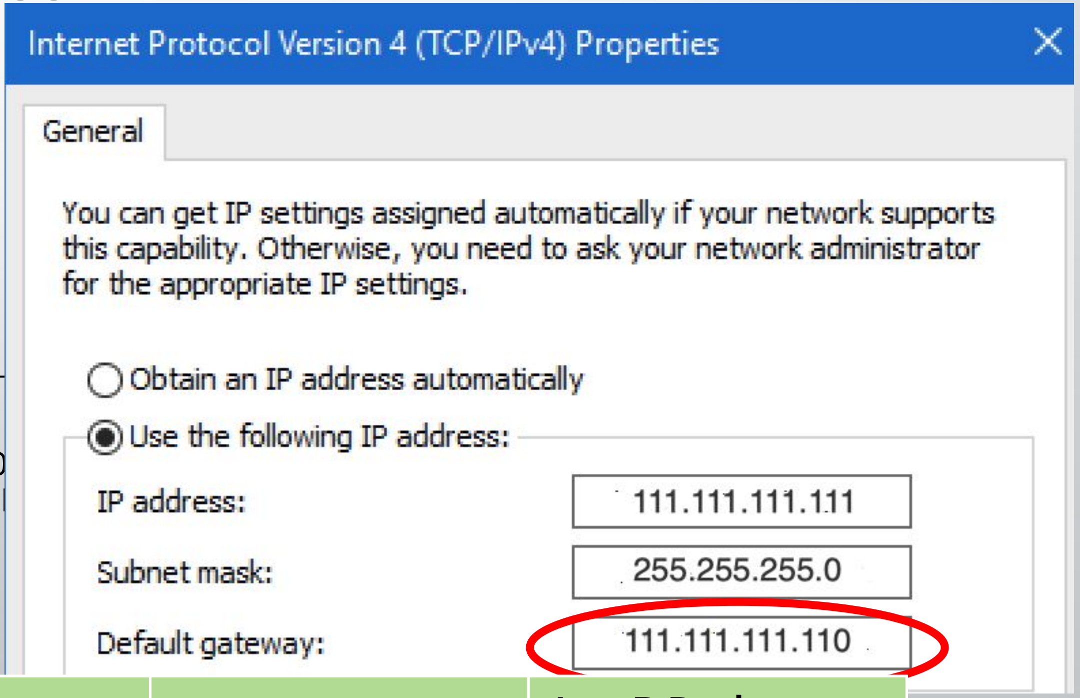
Dest MAC	Source MAC	Dest IP Add	Source IP Add	Packet Type
??? Default Gateway's MAC Address	74-29-9C-E8-FF-55	222.222.222.222	111.111.111.111	A to B Packet

Addressing: routing to another LAN

Send datagram from A to B via R

- Does A know the IP address of first hop router, R which is also known as **Default Gateway**? (how?)
- Will A know R's MAC address?

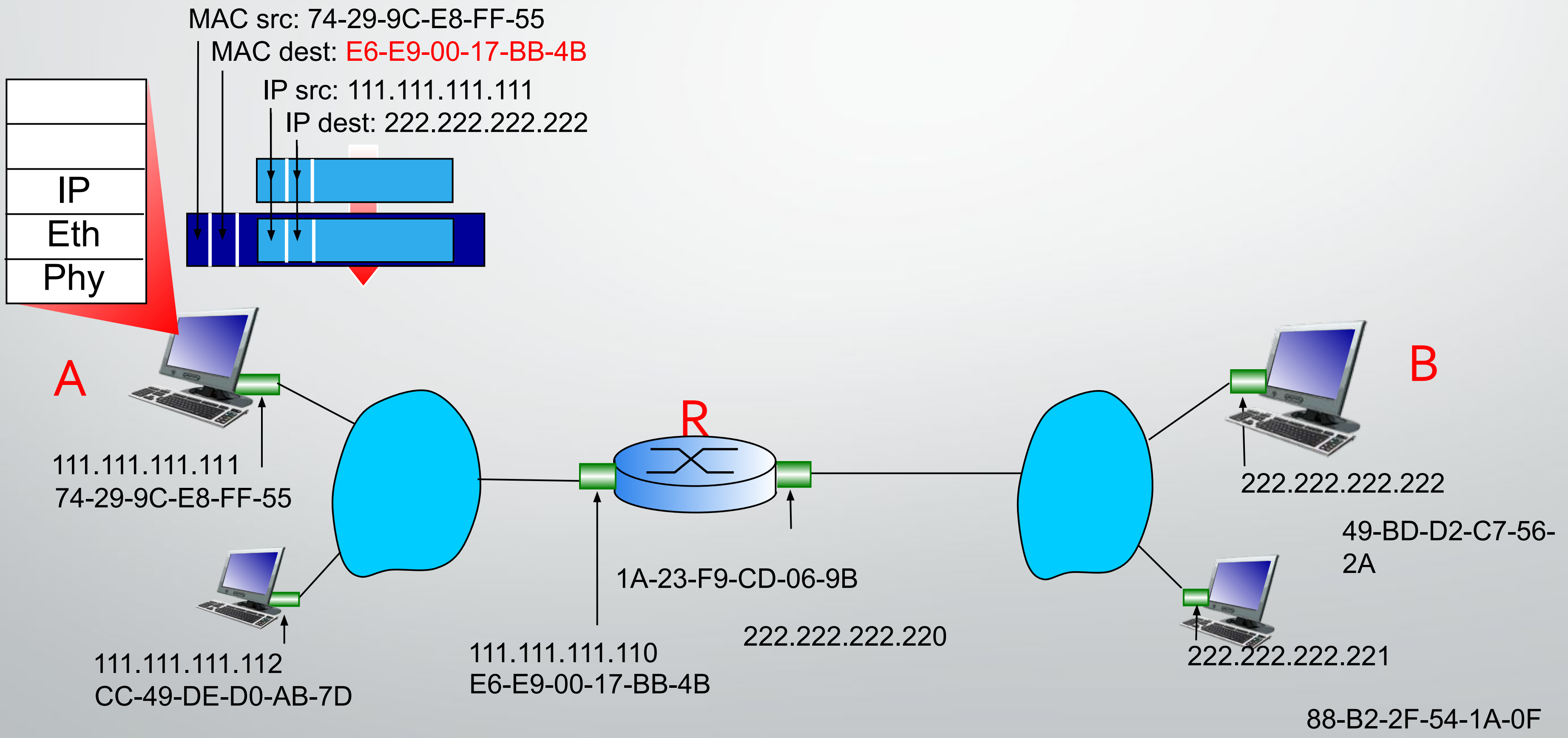




E6-E9-00-17-BB-4B	74-29-9C-E8-FF-55	222.222.222.222	111.111.111.111	A to B Packet

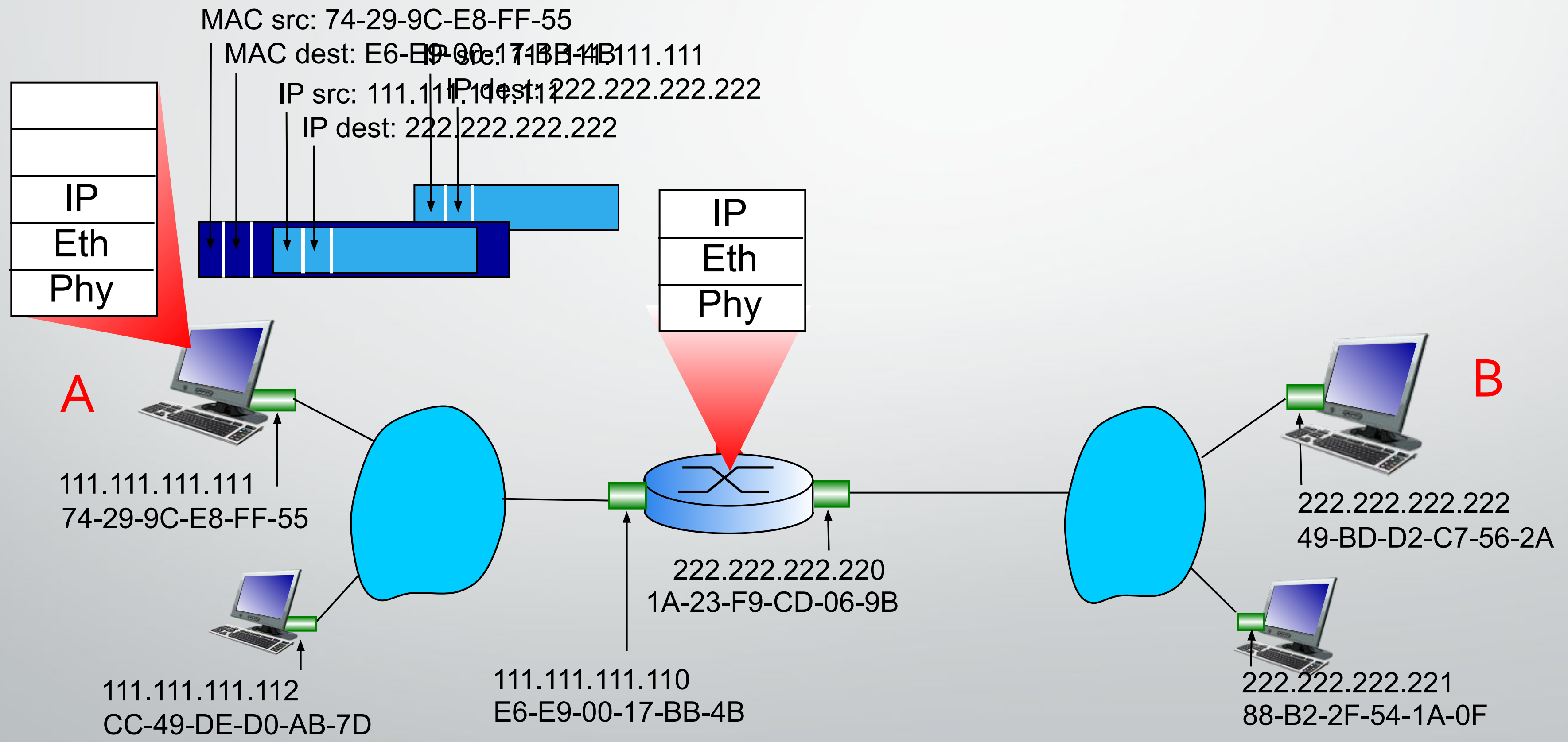
Addressing: routing to another LAN

- 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



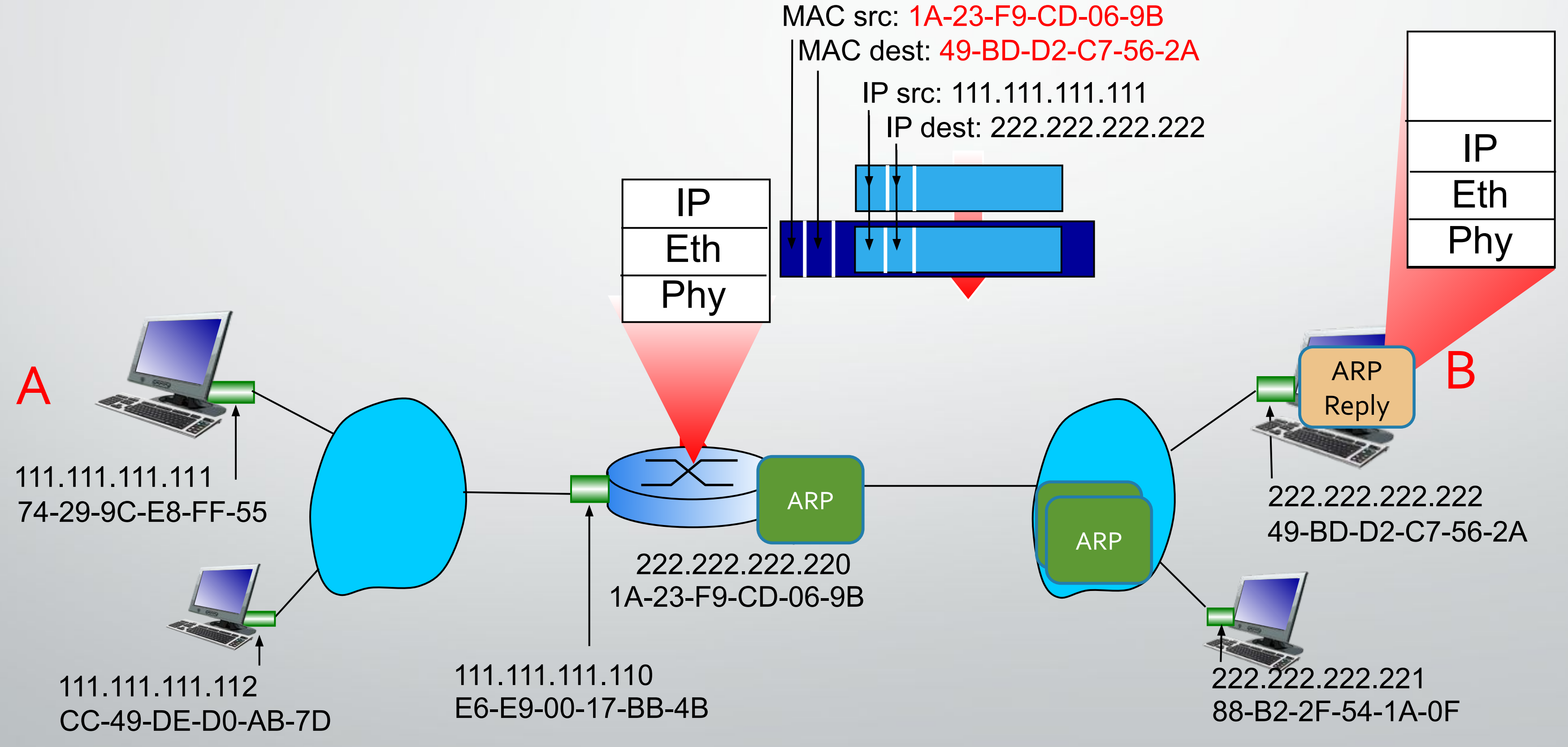
Addressing: routing to another LAN

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



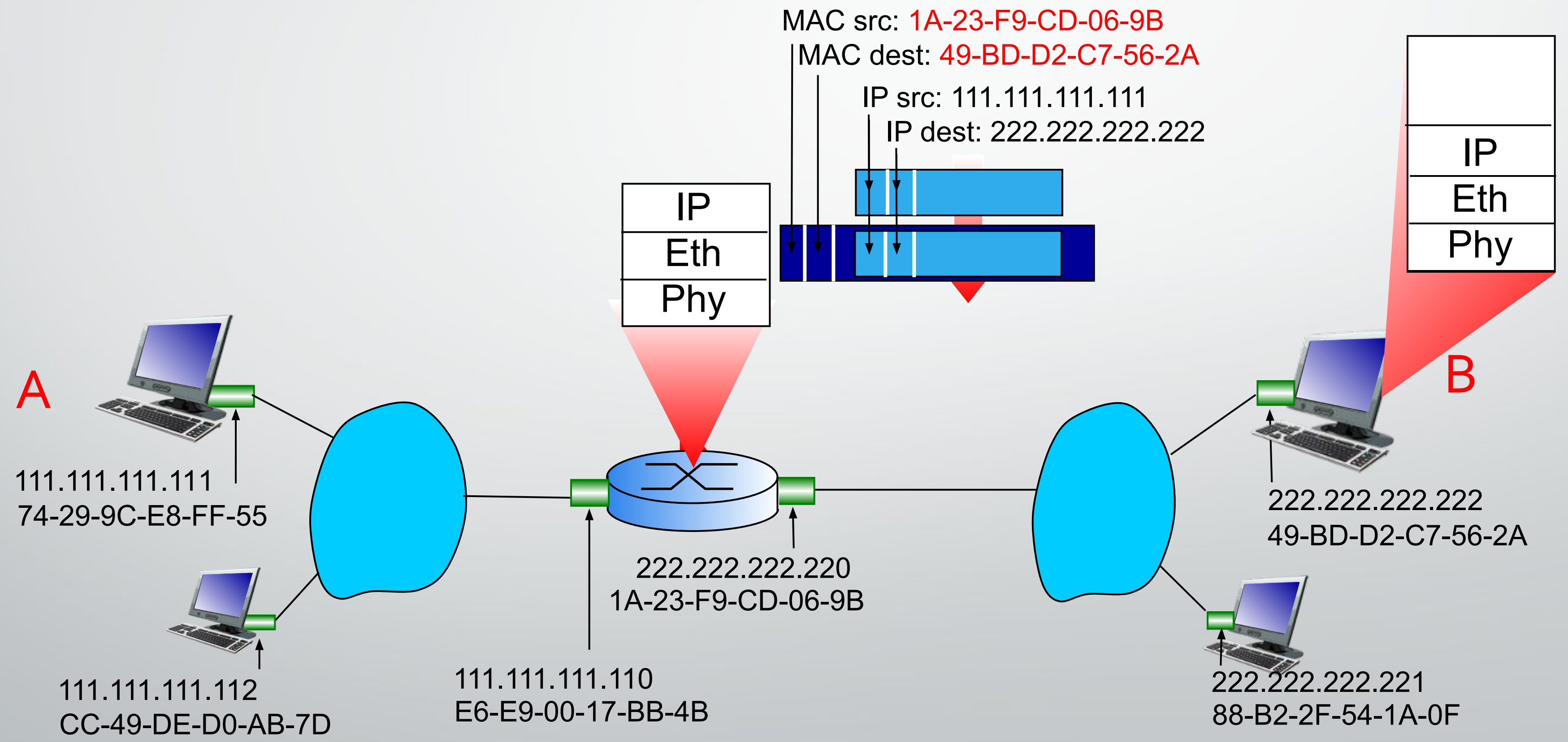
Addressing: routing to another LAN

- 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



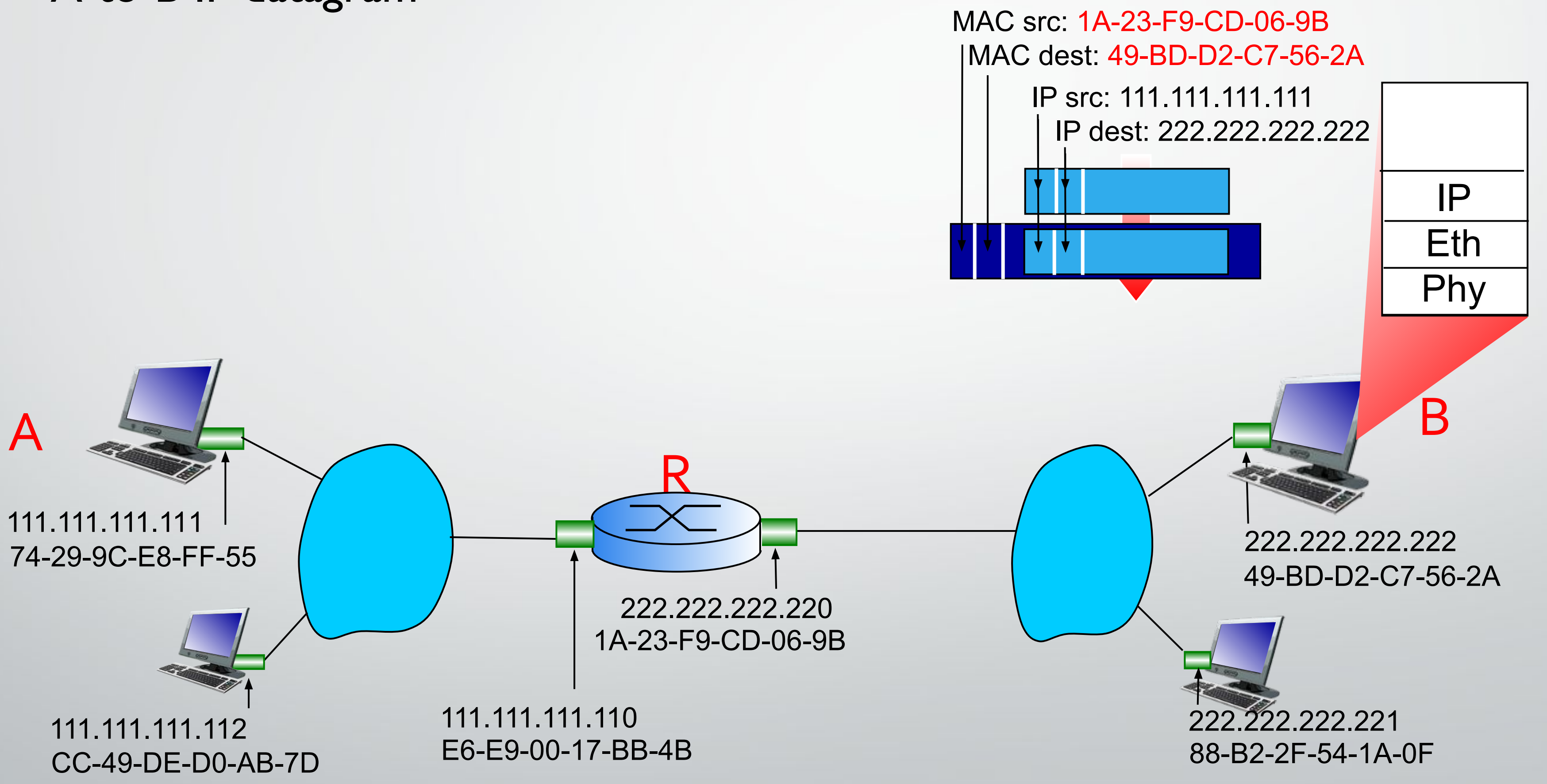
Addressing: routing to another LAN

- 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



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Objectives

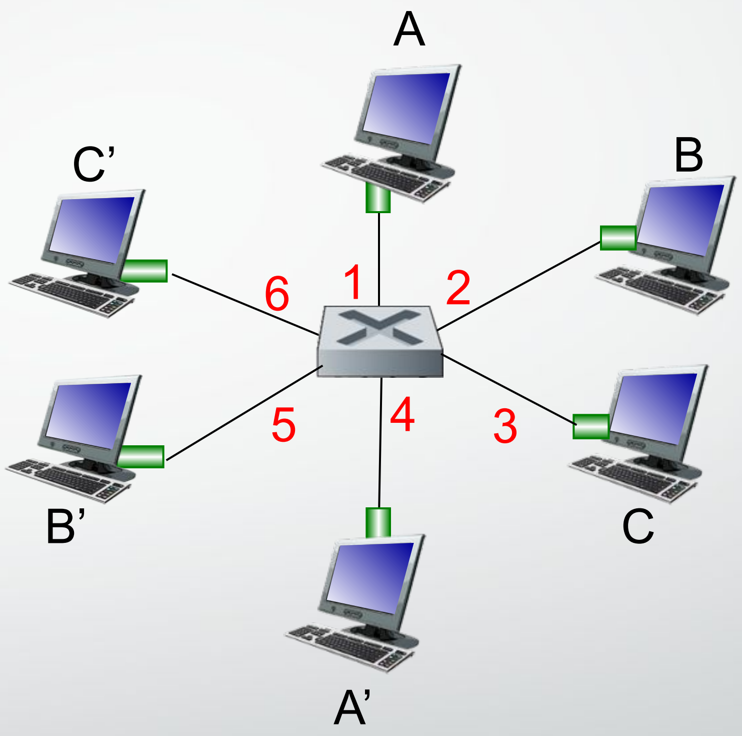
- Switch
  - Characteristics of a switch
  - Role of switch in a LAN

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)

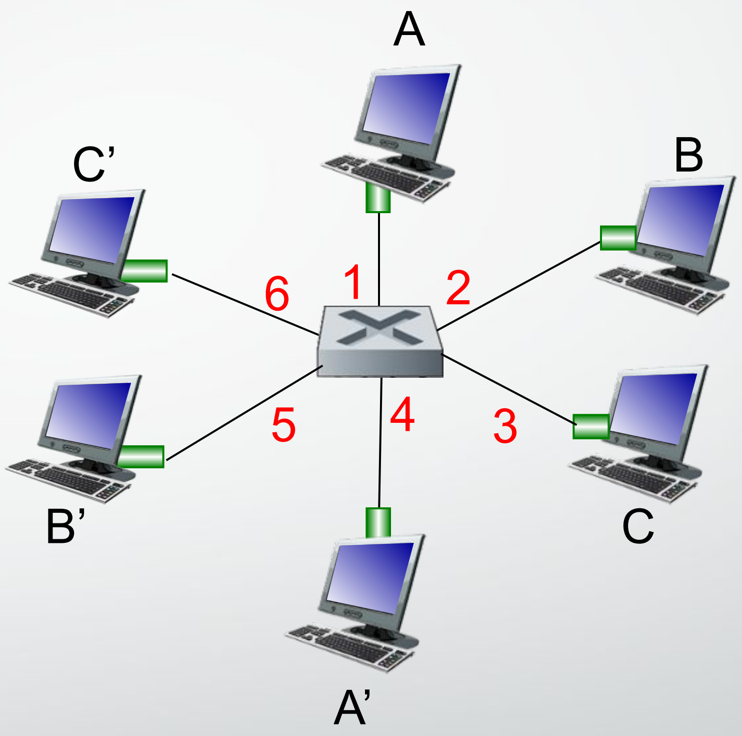
Switch forwarding table

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?

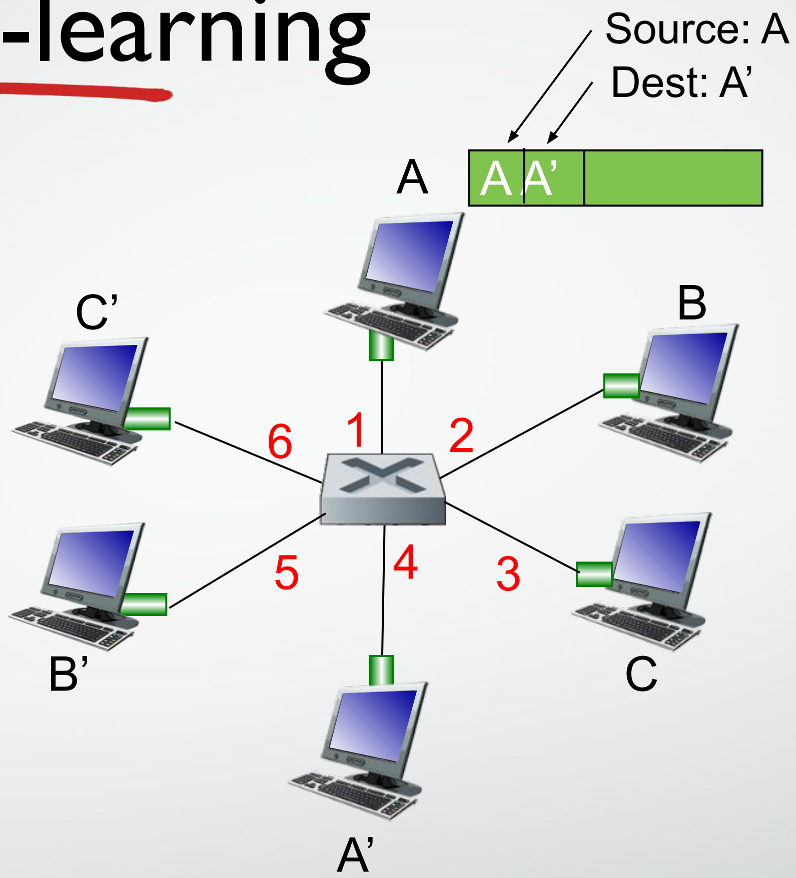
- something like a routing protocol?



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

Switch: self-learning

- The table is empty initially
- 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



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

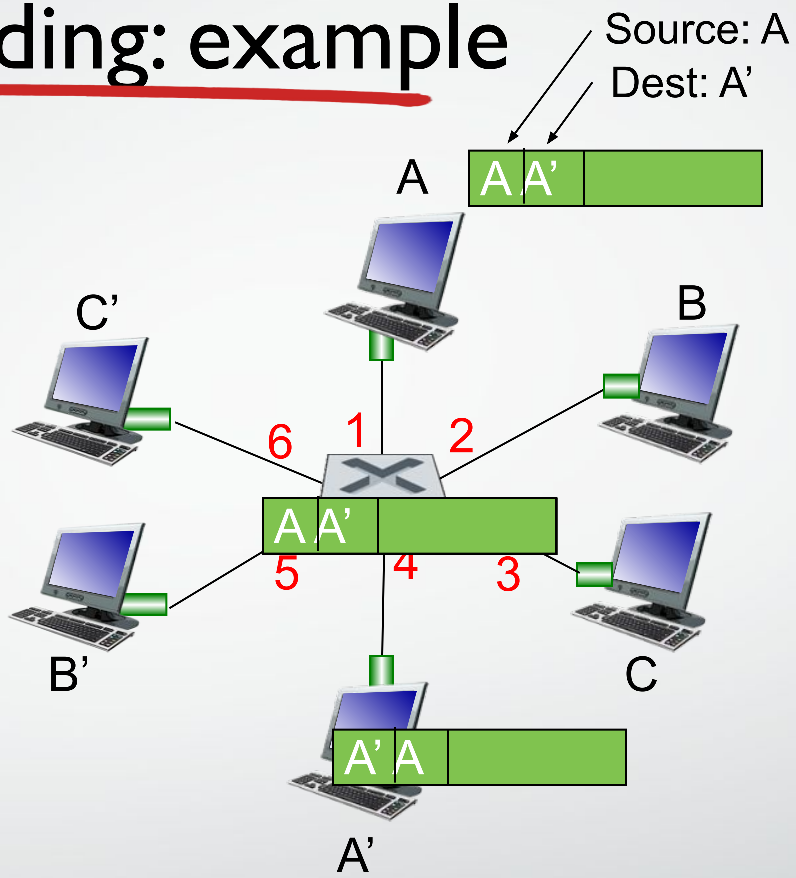
Switch: frame filtering/forwarding

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: *flood*
- destination A location known: selectively send on just one link

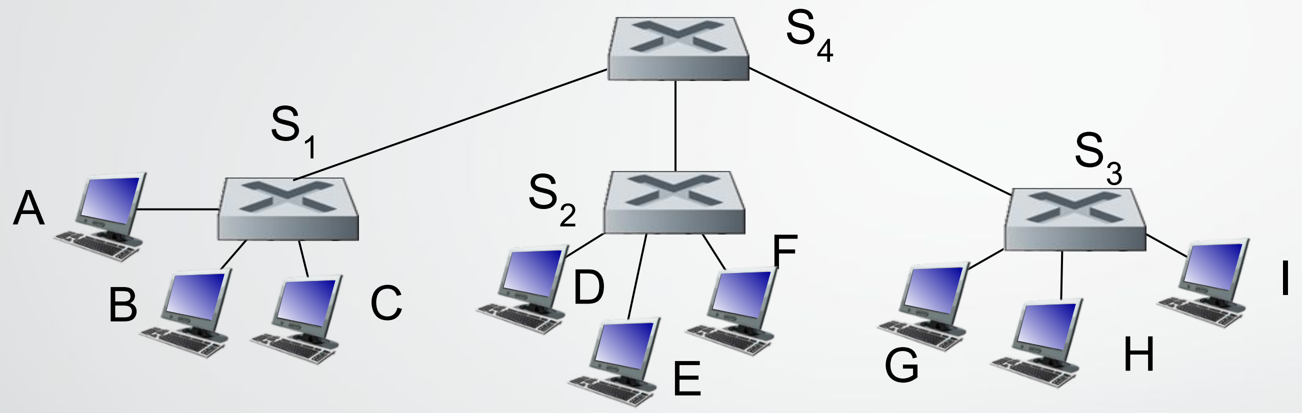


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Interconnecting switches

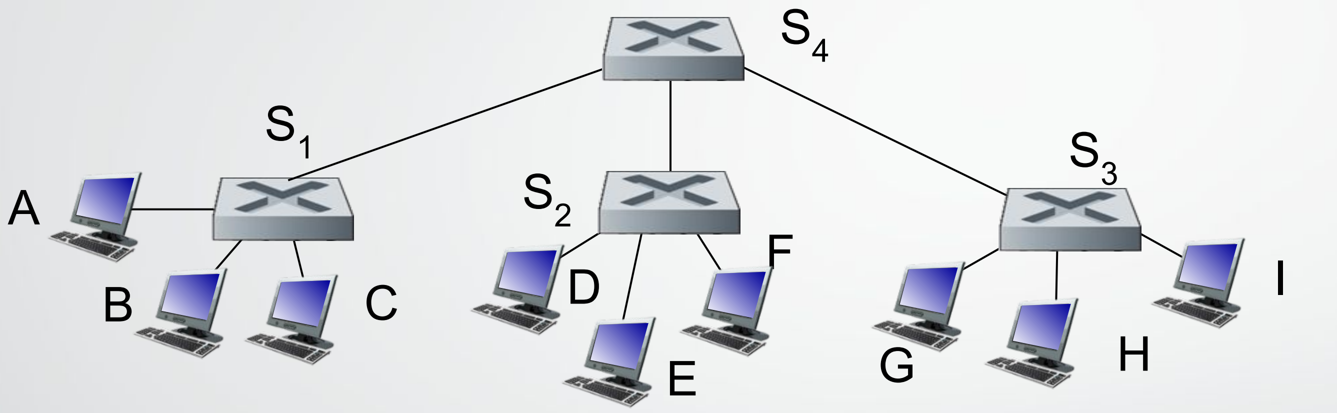
self-learning switches can be connected together:



Q: sending from A to G - how does S_1 know to forward frame destined to G 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

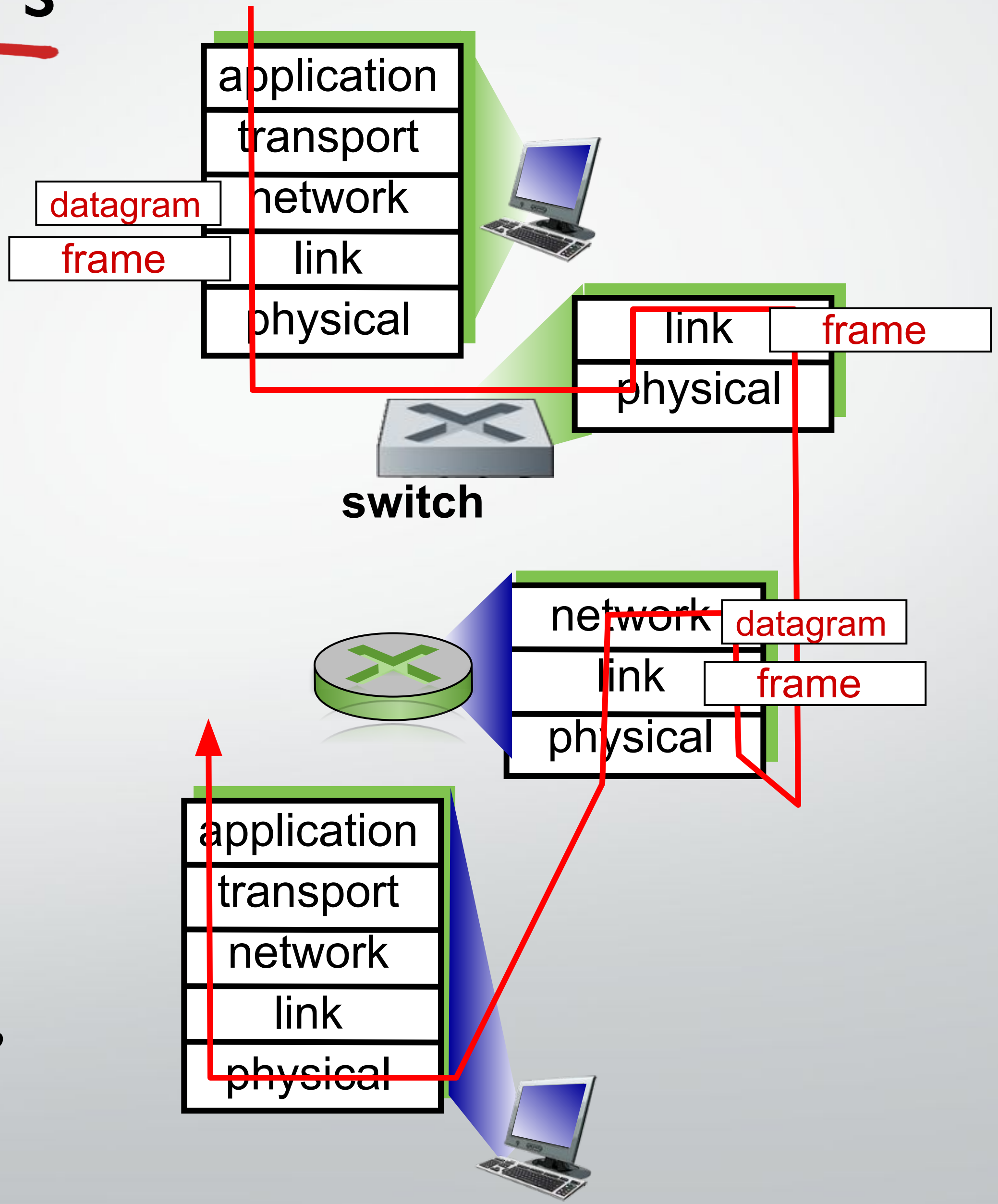
Switches vs. routers

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



THE END!