

Chapter 6

The Link Layer and LANs

A note on the use of these PowerPoint slides:

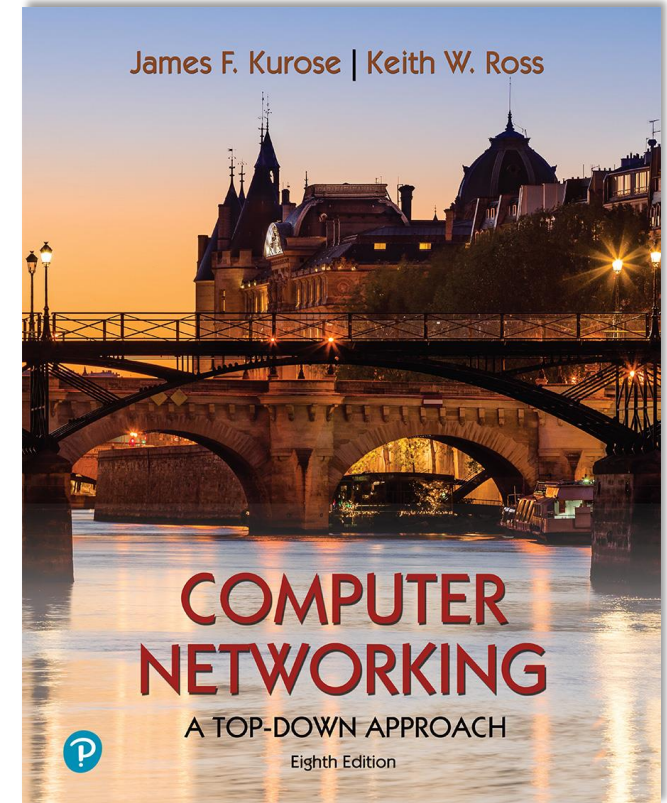
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2020
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer Networking: A
Top-Down Approach*

8th edition

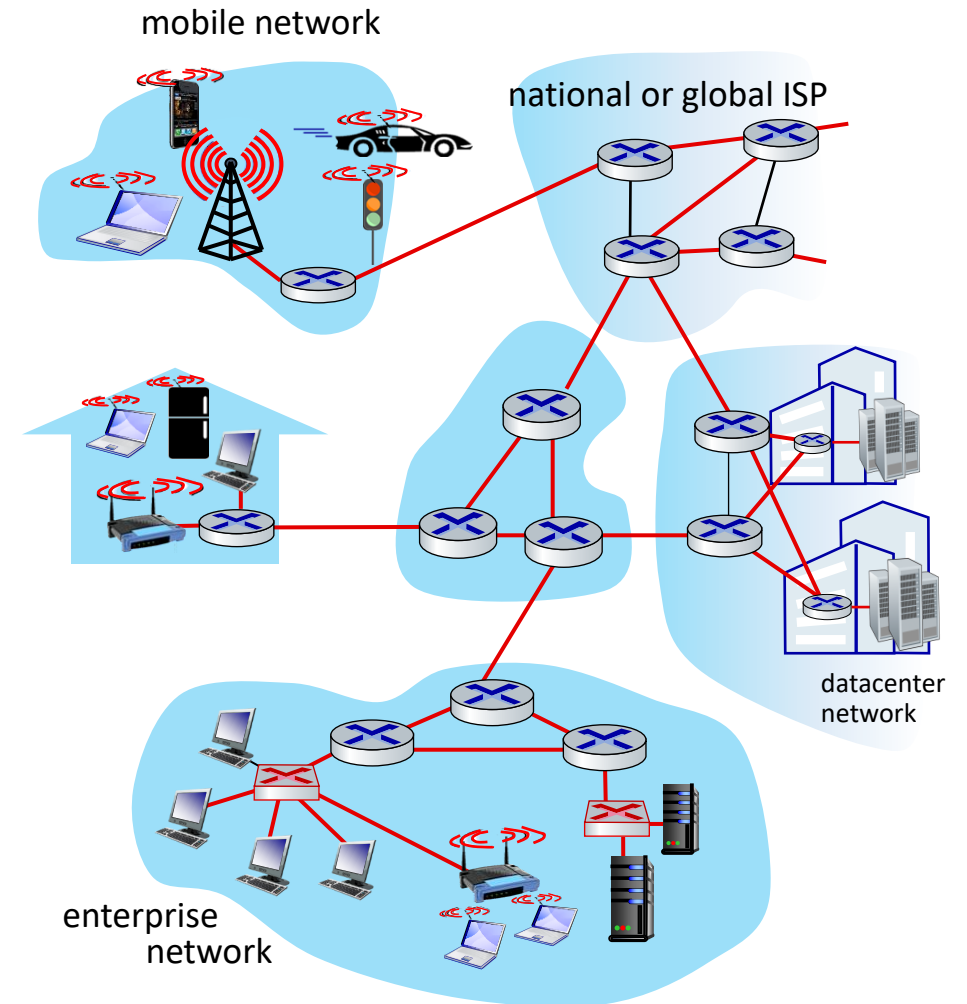
Jim Kurose, Keith Ross
Pearson, 2020

Link layer: introduction

terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
 - wired
 - wireless
 - LANs
- 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

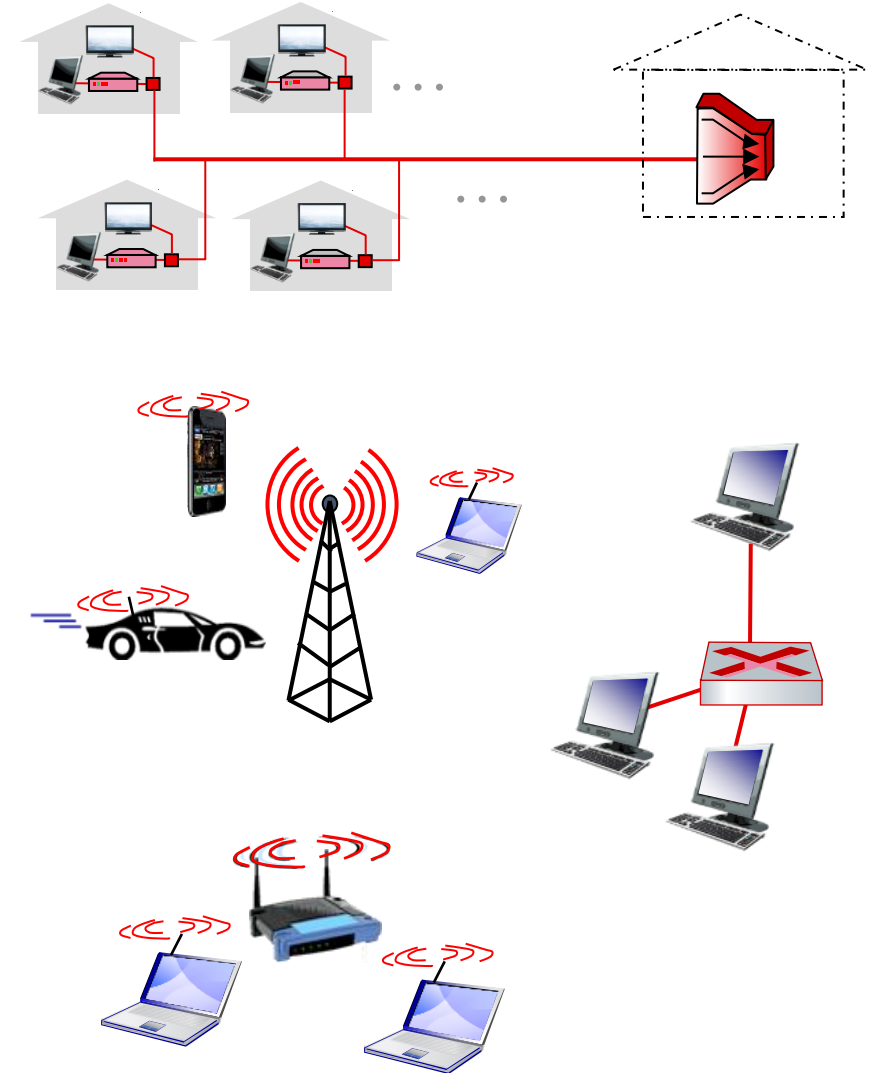
- datagram transferred by different link protocols over different links:
 - e.g., WiFi on first link, Ethernet on next link
- each link protocol provides different services
 - e.g., may or may not provide reliable data transfer over link

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**

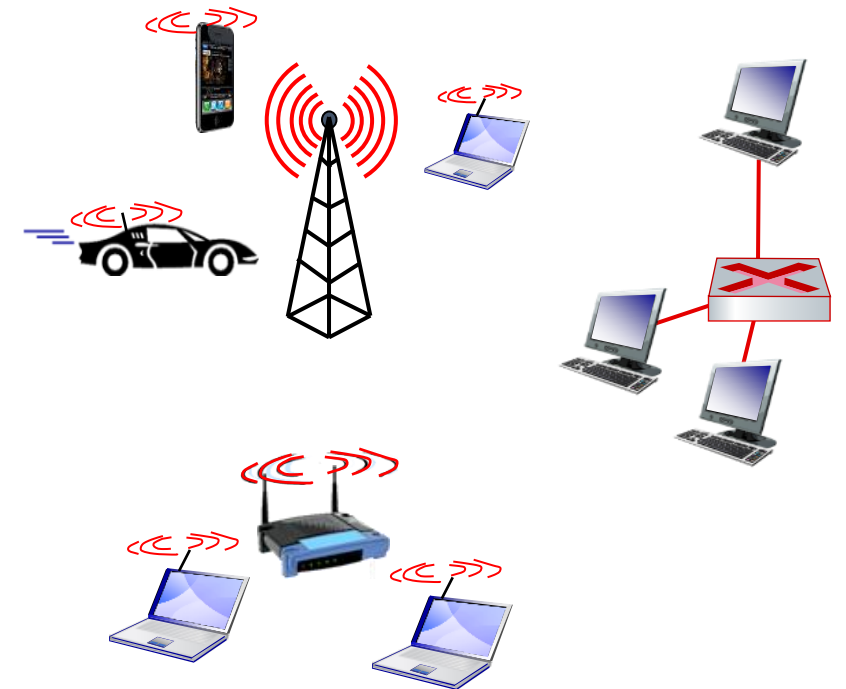
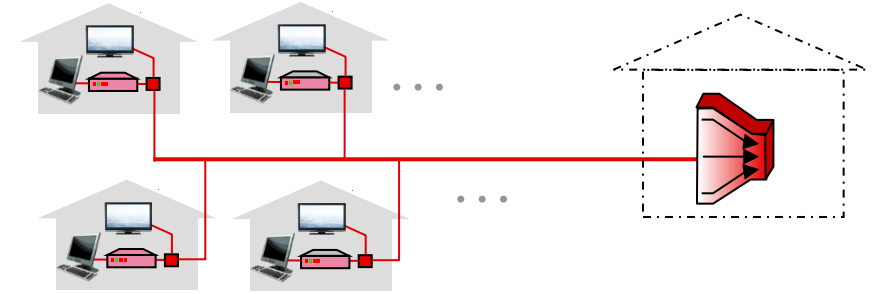
Link layer: services

- **framing, link access:**
 - encapsulate datagram into frame, adding header, trailer
 - channel access if shared medium
 - “MAC” addresses 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



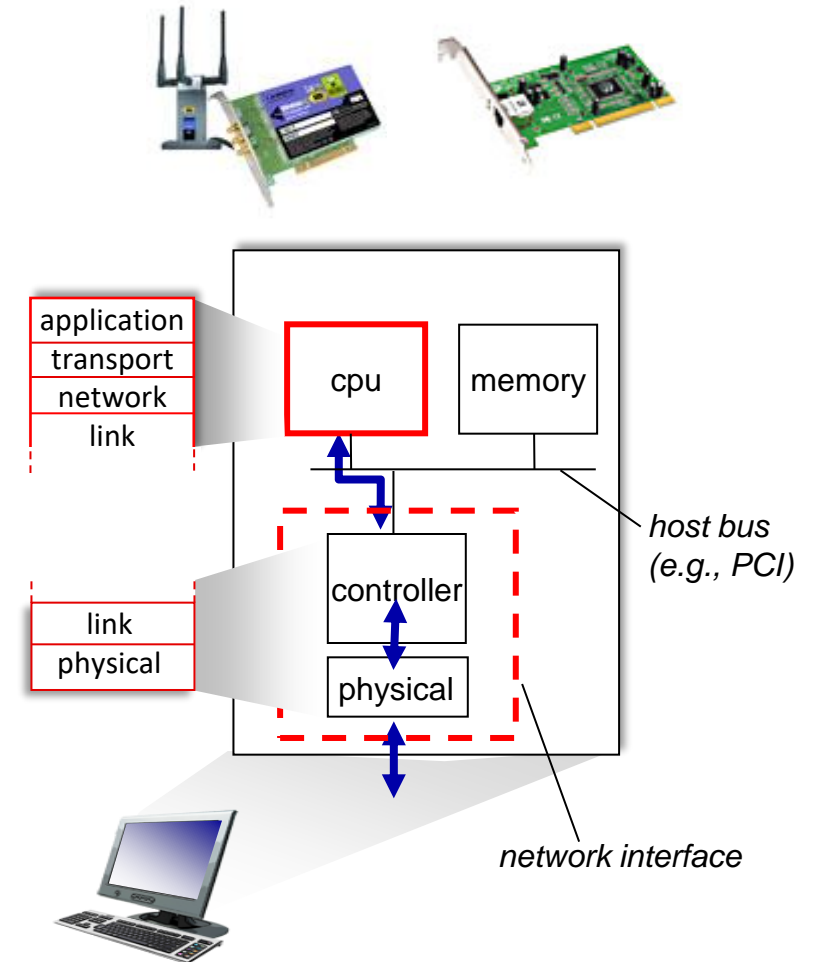
Link layer: services (more)

- **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

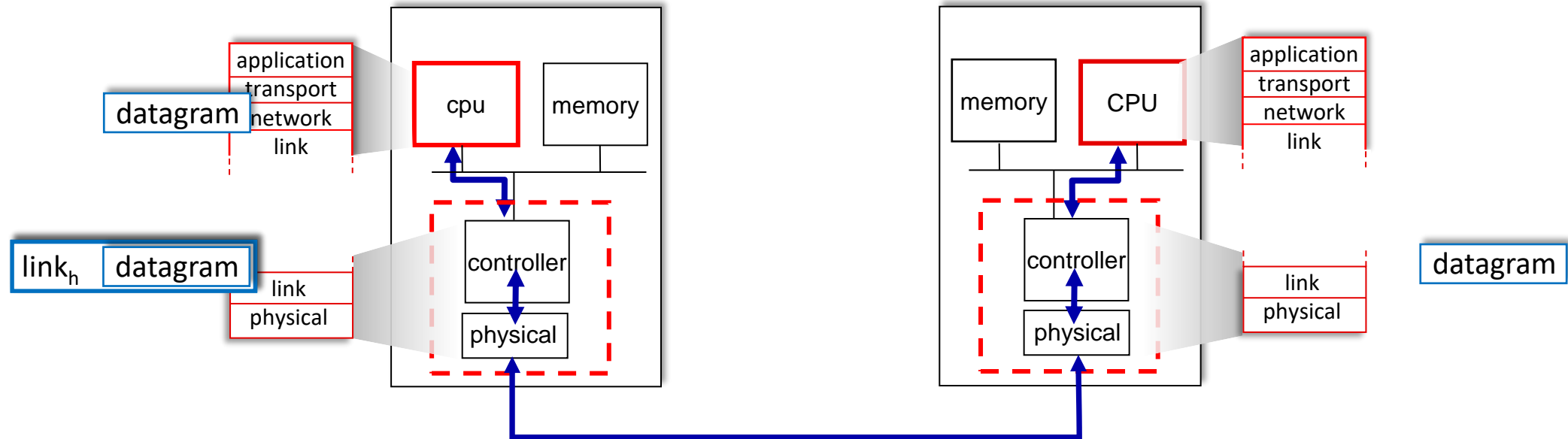


Where is the link layer implemented?

- in each-and-every host
- link layer implemented in *network interface card* (NIC) or on a chip
 - Ethernet, WiFi card or chip
 - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Interfaces communicating



sending side:

- encapsulates datagram in frame
- adds error checking bits, reliable data transfer, flow control, etc.

receiving side:

- looks for errors, reliable data transfer, flow control, etc.
- extracts datagram, passes to upper layer at receiving side

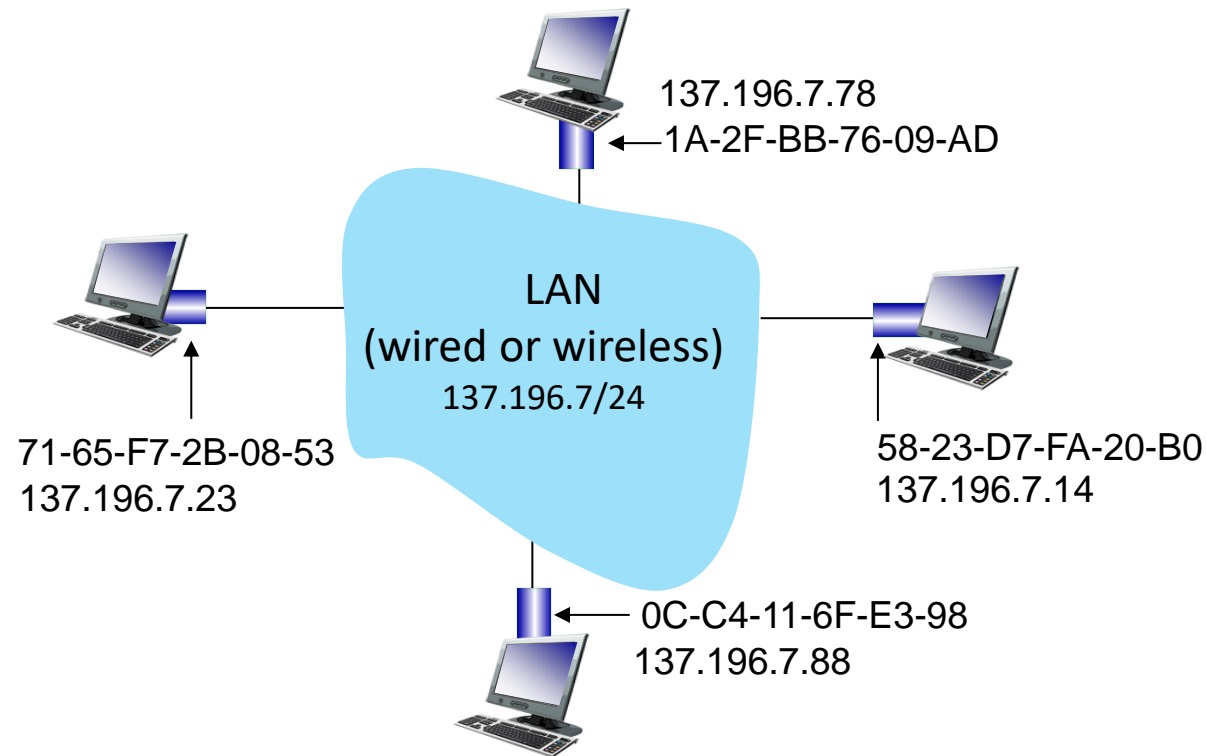
MAC addresses

- 32-bit IP address:
 - *network-layer* address for interface
 - used for layer 3 (network layer) forwarding
 - e.g.: 128.119.40.136
- MAC (or LAN or physical or Ethernet) address:
 - function: used “locally” to get frame from one interface to another physically-connected interface (same subnet, 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 “numeral” represents 4 bits)

MAC addresses

each interface on LAN

- has unique 48-bit **MAC** address
- has a locally unique 32-bit IP address (as we've seen)



MAC addresses

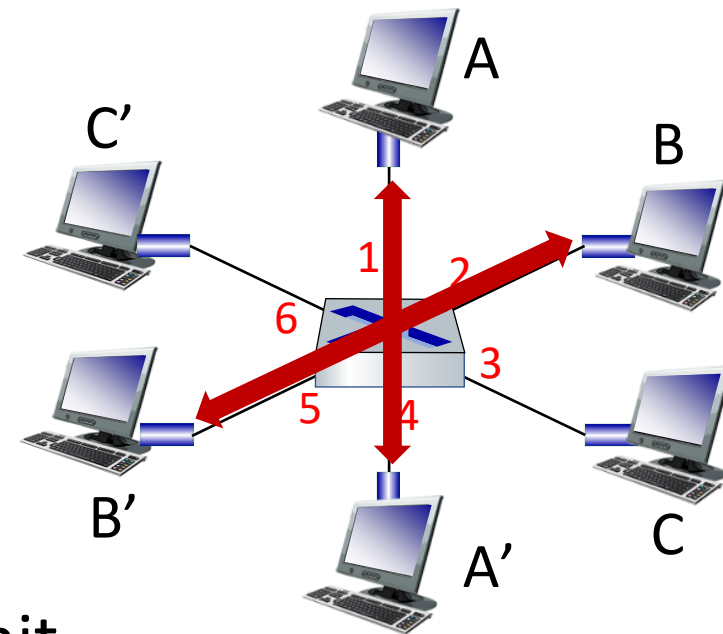
- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
 - MAC address: like Social Security Number
 - IP address: like postal address
- MAC flat address: portability
 - can move interface from one LAN to another
 - recall IP address *not* portable: depends on IP subnet to which node is attached

Ethernet switch

- Switch is a **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 *unaware* of presence of switches
- **plug-and-play, self-learning**
 - switches do not need to be configured

Switch: multiple simultaneous transmissions

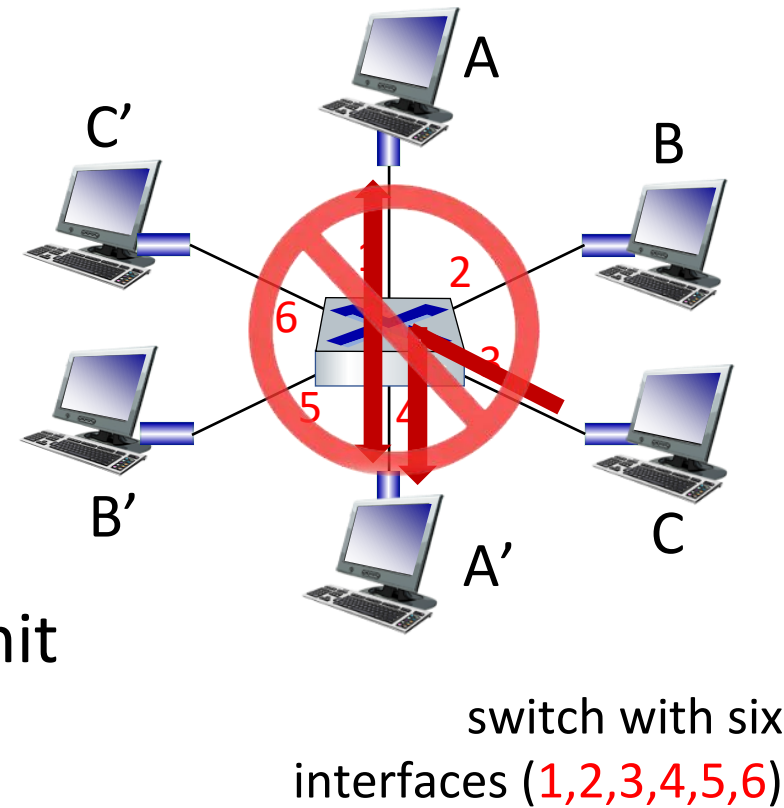
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
 - no collisions; full duplex
 - each link is its own collision domain
- **switching**: A-to-A' and B-to-B' can transmit simultaneously, without collisions



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

Switch: multiple simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
 - no collisions; full duplex
 - each link is its own collision domain
- **switching:** A-to-A' and B-to-B' can transmit simultaneously, without collisions
 - but A-to-A' and C to A' can *not* happen simultaneously



Switch forwarding table

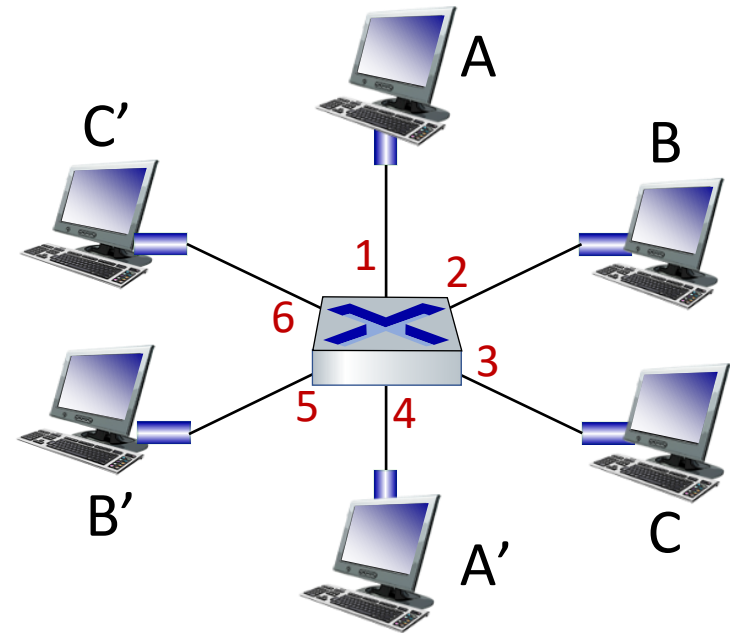
Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

A: each switch has a **switch table**, each entry:

- (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!

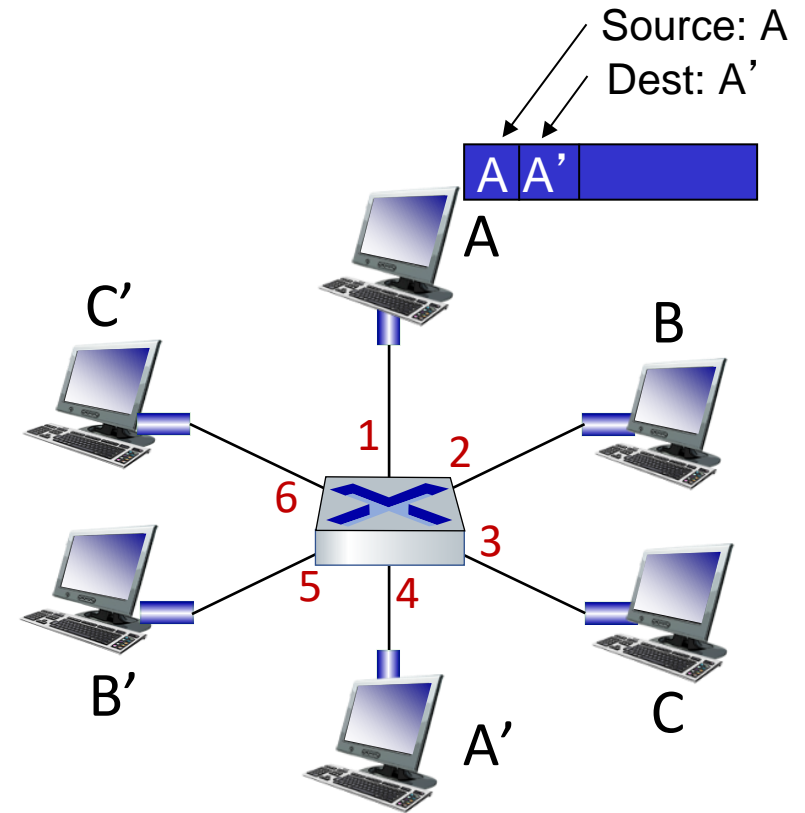
Q: how are entries created, maintained in switch table?

- something like a routing protocol?



Switch: 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



MAC addr	interface	TTL
A	1	60

*Switch table
(initially empty)*

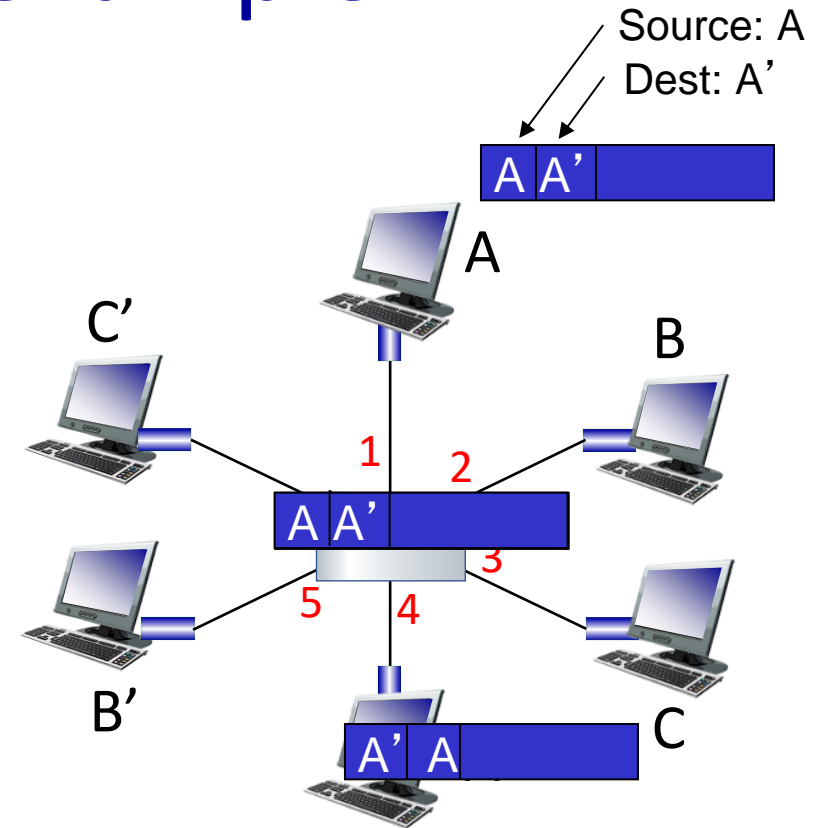
Switch: frame filtering/forwarding

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

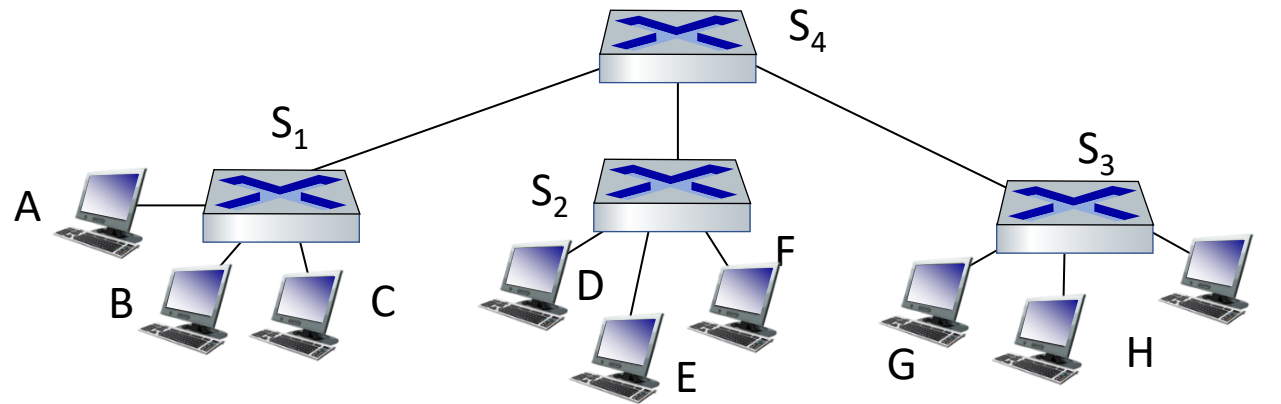


MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table
(initially empty)*

Interconnecting switches

self-learning switches can be connected together:

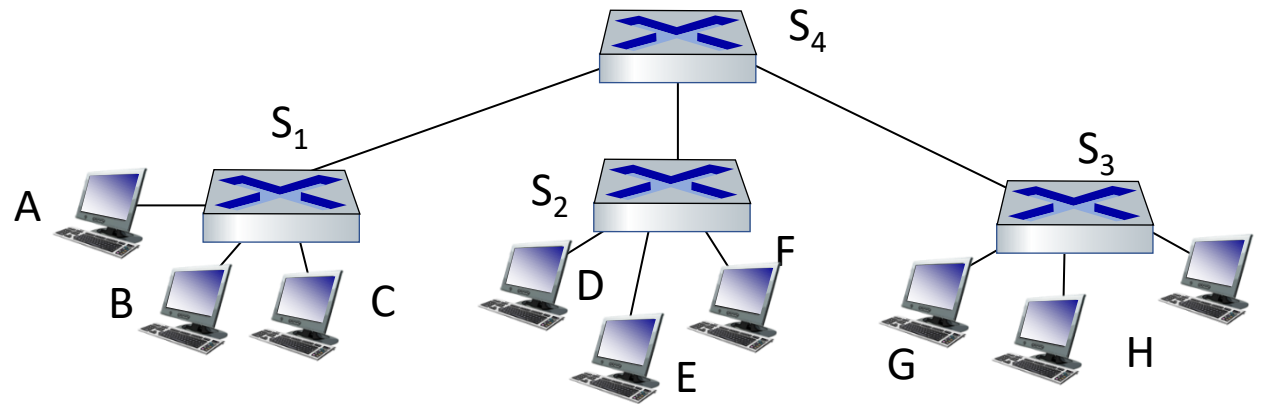


Q: sending from A to G - how does S₁ know to forward frame destined to G via S₄ and S₃?

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

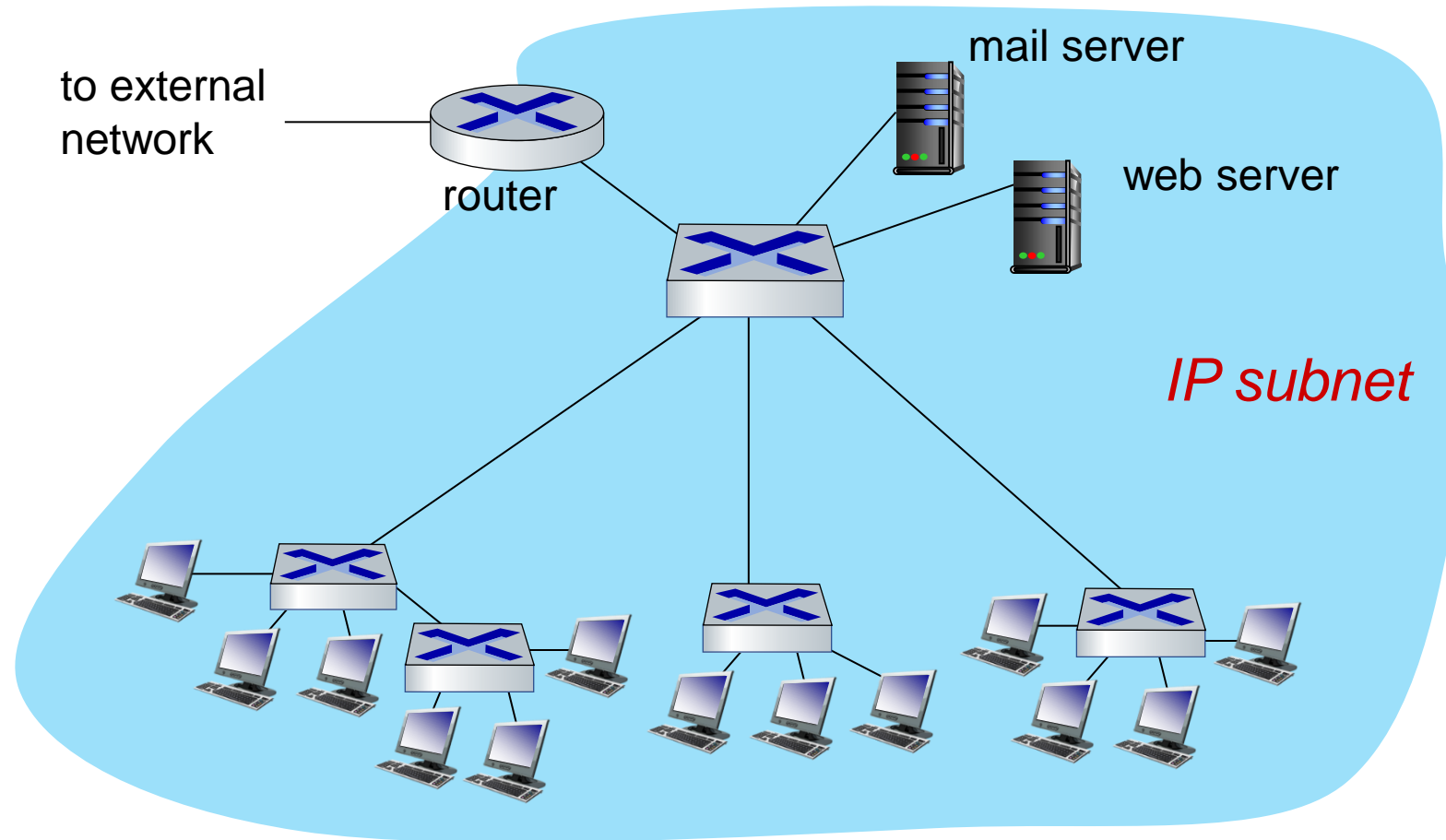
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



Q: show switch tables and packet forwarding in S₁, S₂, S₃, S₄

Small institutional network



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

