Link Layer-Systems

Addressing

- How do we know where to send messages?
- Every node needs some kind of address
- This allows each message to go to its destination
- Human speech equivalent: name

MAC addresses

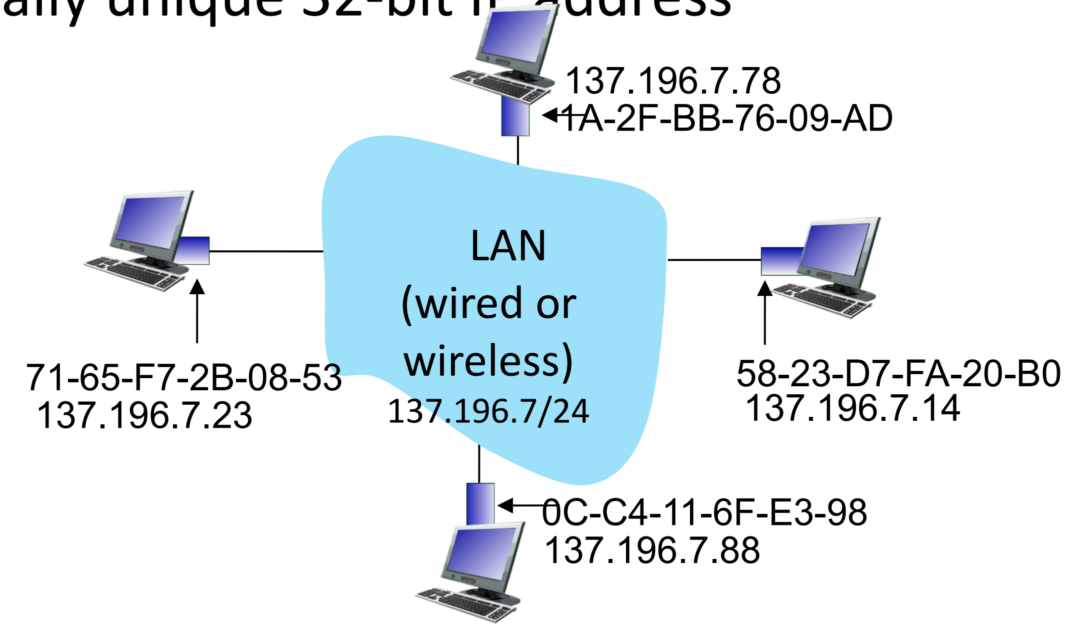
- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
  - e.g.: 128.119.40.136
- MAC (or LAN or physical or Ethernet) address:
  - function: used "locally" to get frame from one interface to another physically-connected interface (same subnet,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



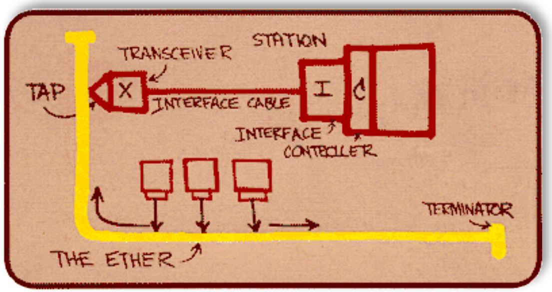
MAC addresses

- 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

"dominant" wired LAN technology:

- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 400 Gbps
- single chip, multiple speeds (e.g., Broadcom BCM5761)



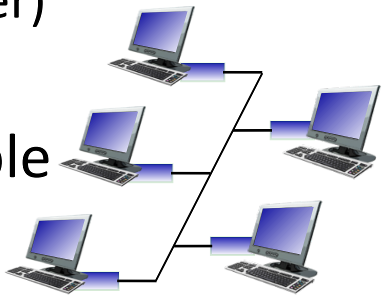
Metcalfe's Ethernet sketch

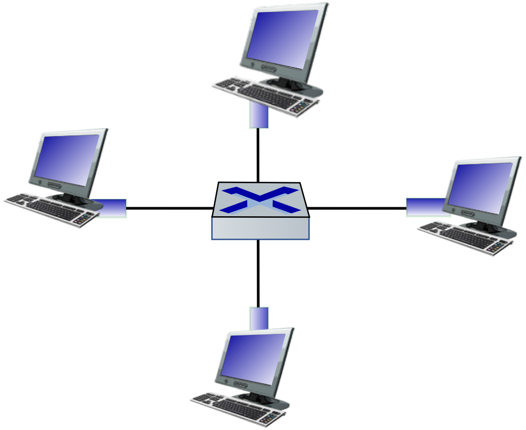
https://www.uspto.gov/learning-and-resources/journeys-innovation/audio-stories/defying-doubters

Ethernet

- 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
  - each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)

bus: coaxial cable





switched

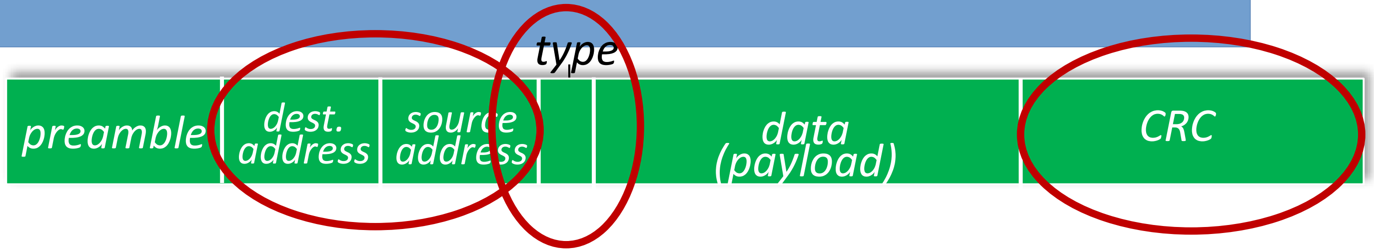
Ethernet

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

preamble:

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

Ethernet



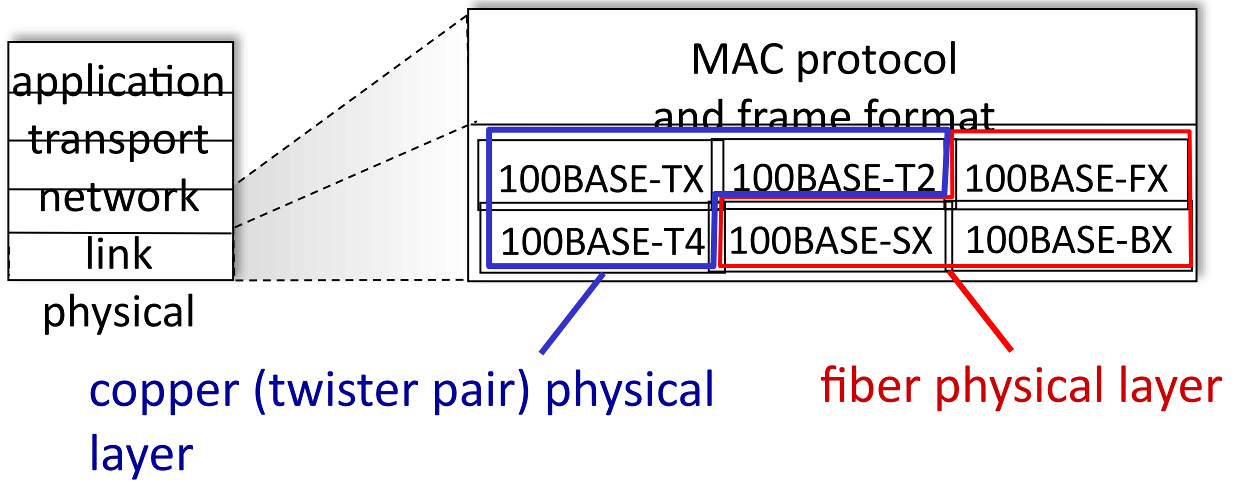
- addresses: 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g., ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- type: indicates higher layer protocol
  - mostly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

Ethernet

- connectionless: no handshaking between sending and receiving NICs
- unreliable: receiving NIC doesn't send ACKs or NAKs to sending NIC
  - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- Ethernet's MAC protocol: unslotted CSMA/CD with binary backoff

Ethernet

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



Ethernet

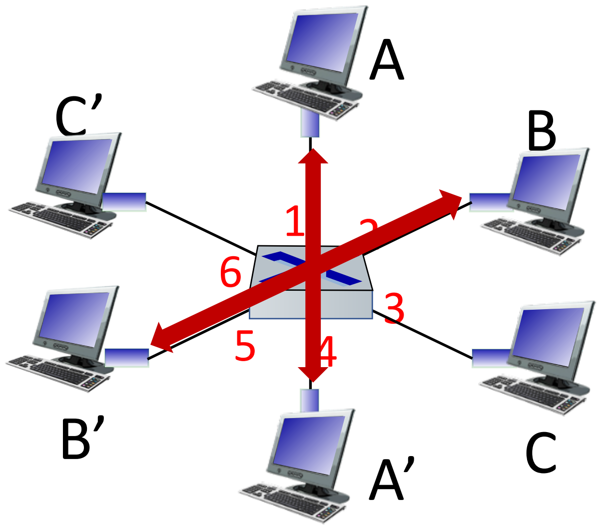
- Lets look at a protocol document: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9844436

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

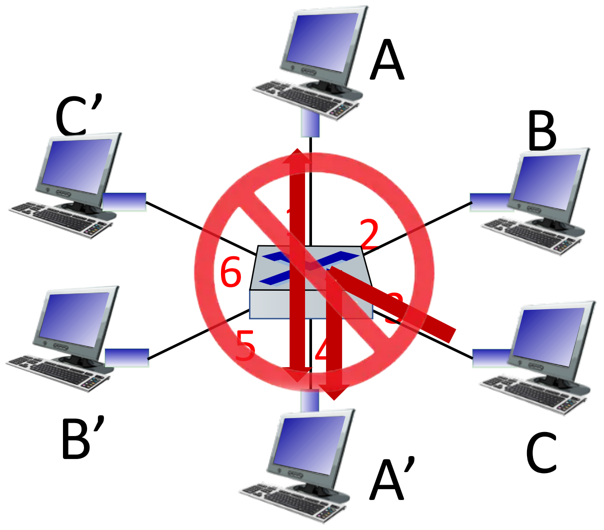
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - no collisions; full duplex
  - each link is its own collision domain
- switching: A-to-A’ and B-to-B’ can transmit simultaneously, without collisions



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

Switch

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

Switch

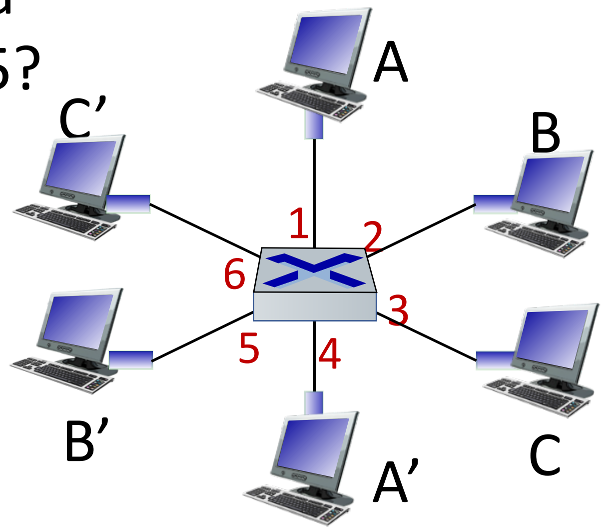
Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

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

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

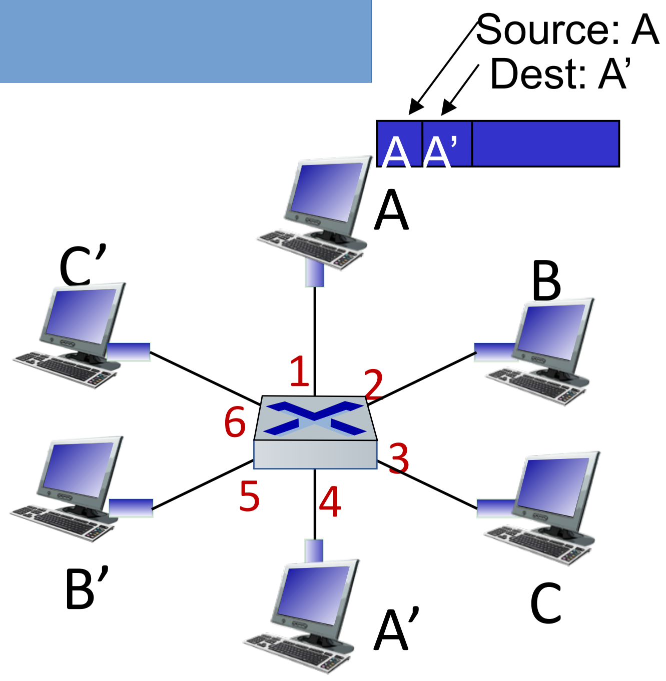
Q: how are entries created, maintained in switch table?

- something like a routing protocol?



Switch

- 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

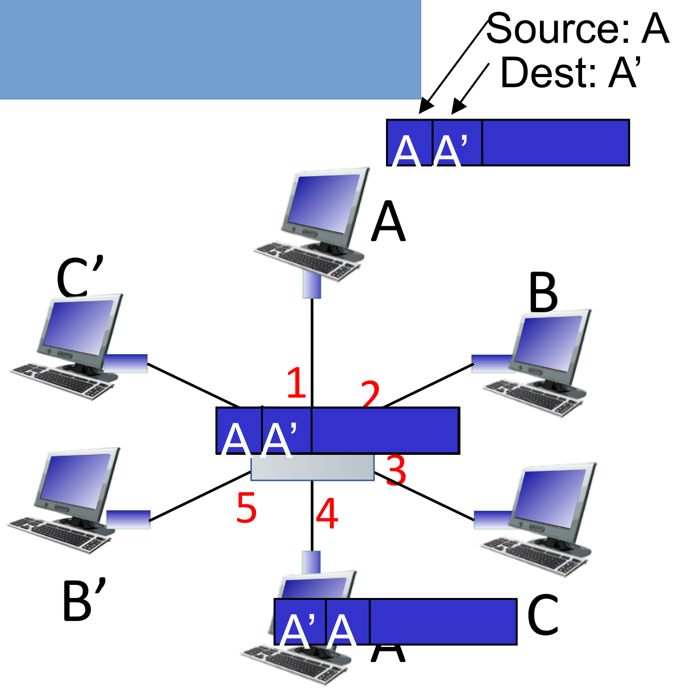
when frame received at switch:

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 */
```

Switch

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

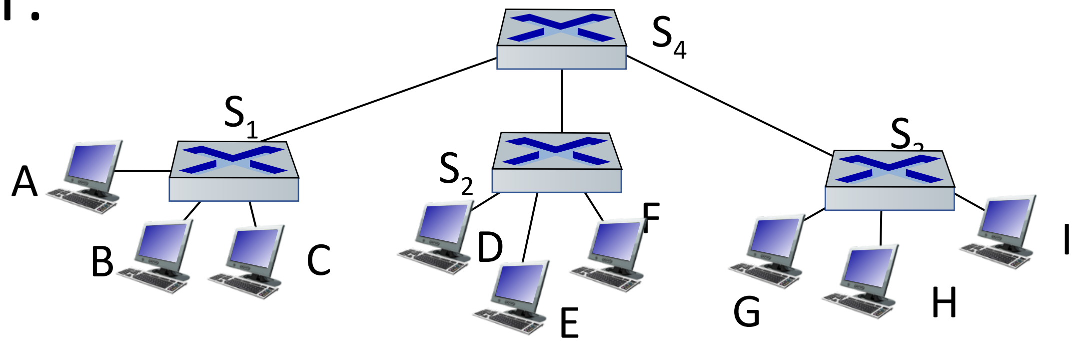


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Switch

self-learning switches can be connected together:

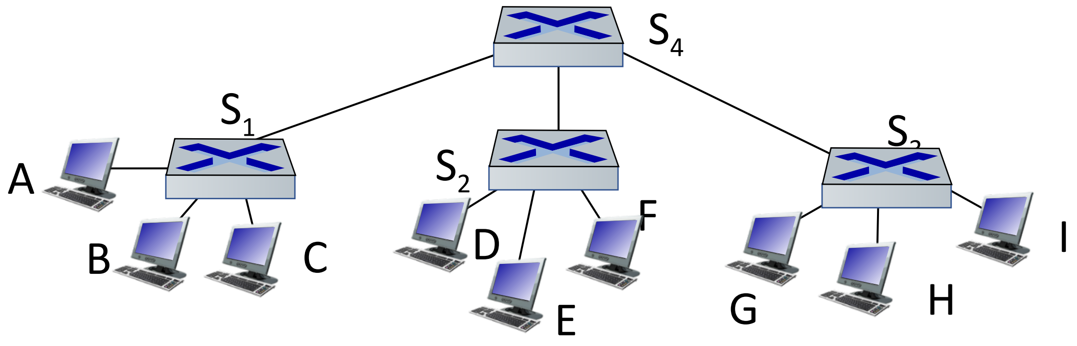


Q: sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?

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

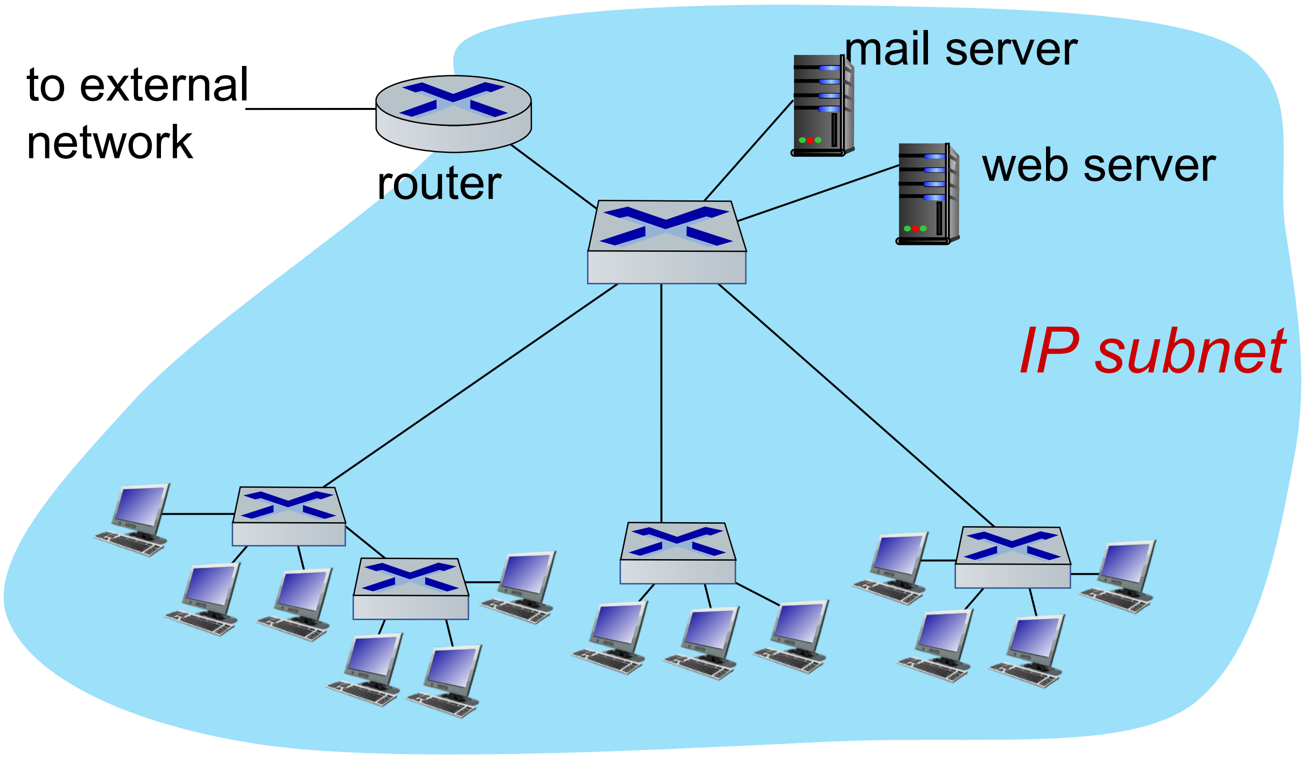
Switch

Suppose C sends frame to I, I responds to C



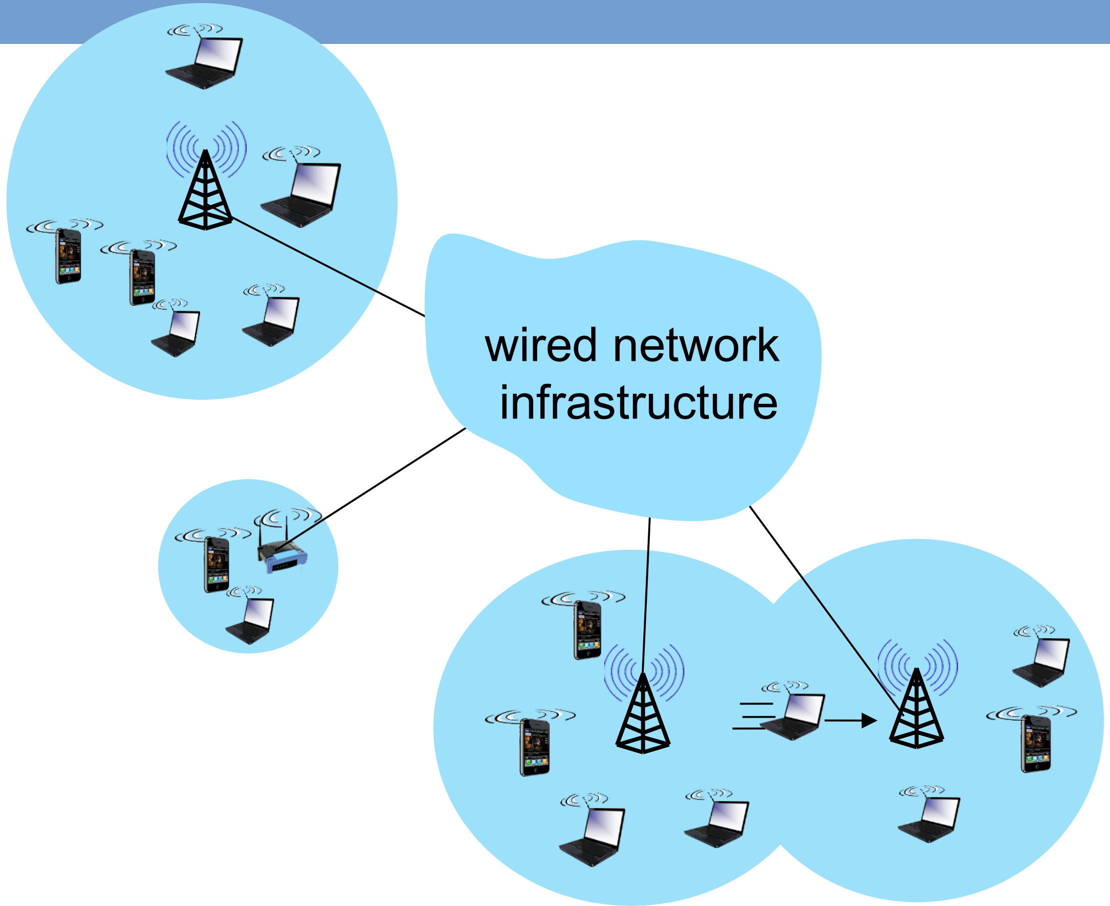
Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Switch

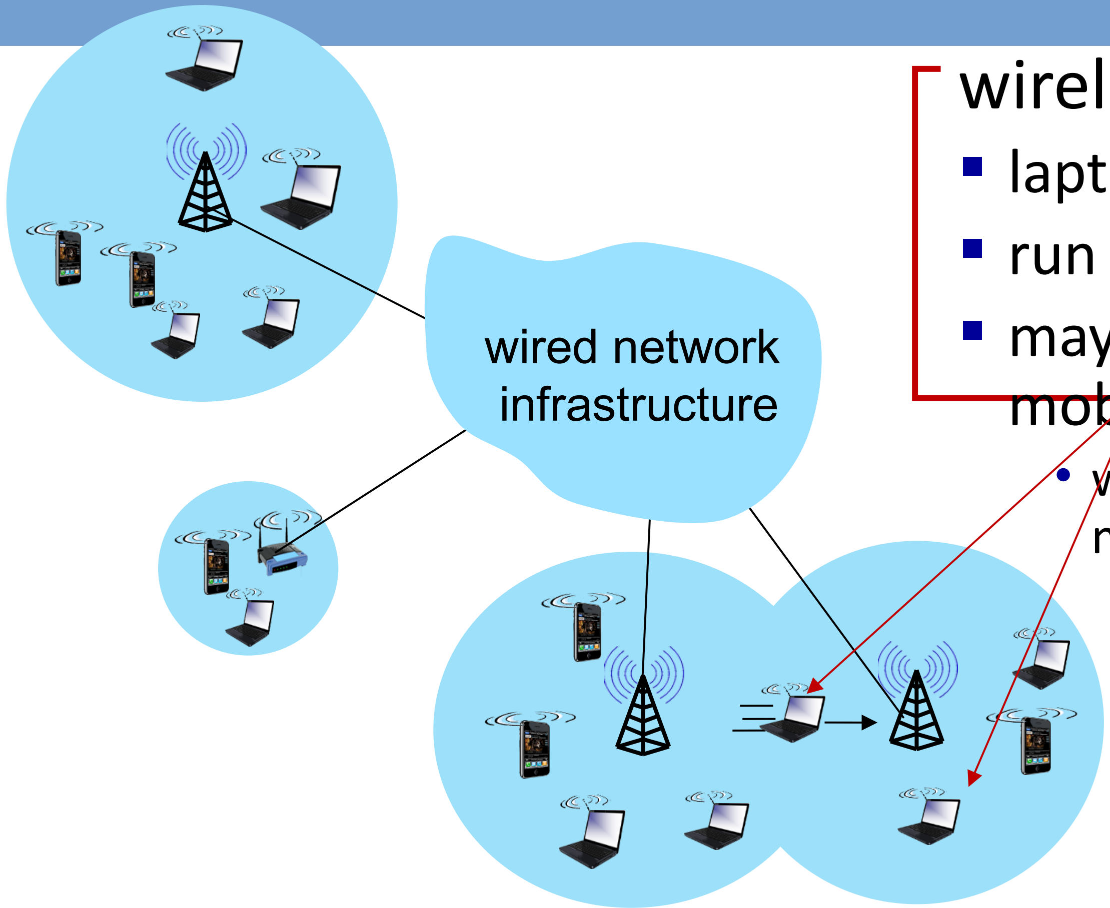

to external network
router
mail server
web server
IP subnet

Wireless Networks

Wireless Networks



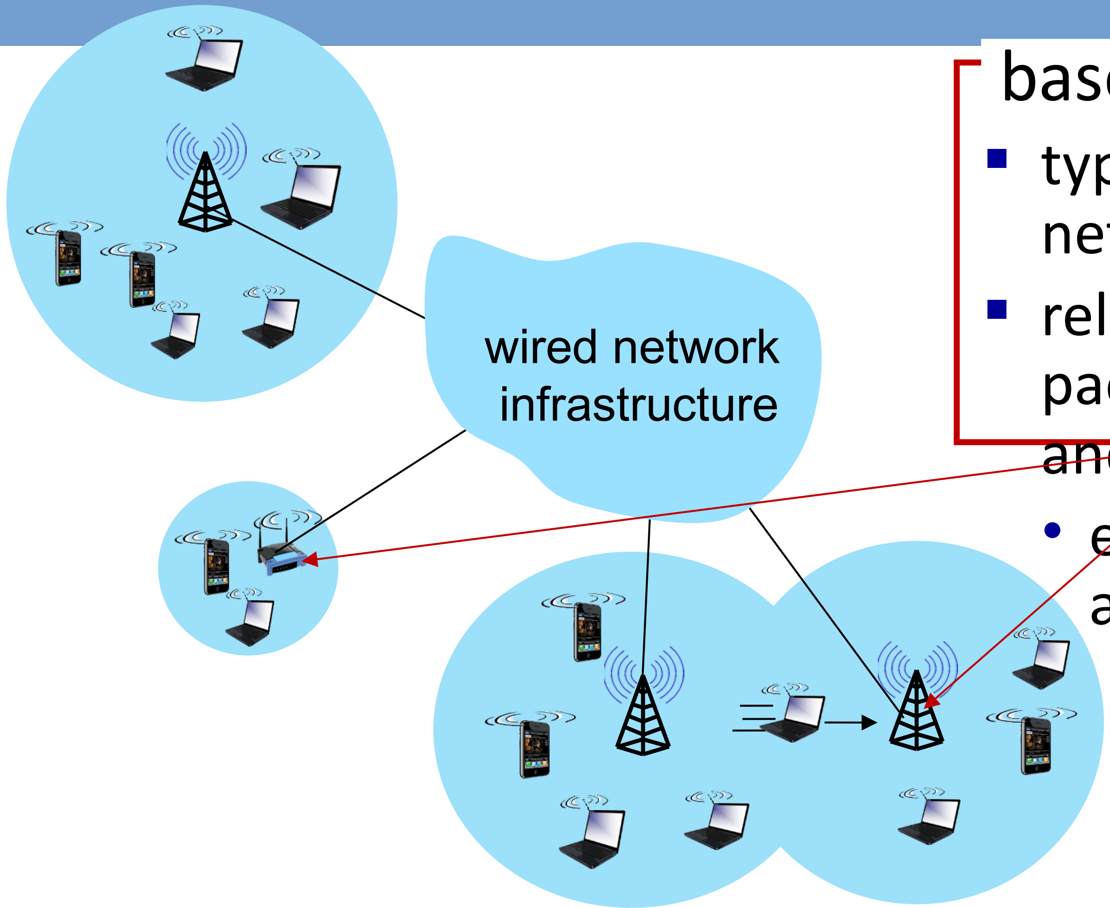
Wireless Networks



wireless hosts

- laptop, smartphone,
- run applications
- may be stationary (non-mobile) or mobile
  - wireless does *not* always mean mobility!

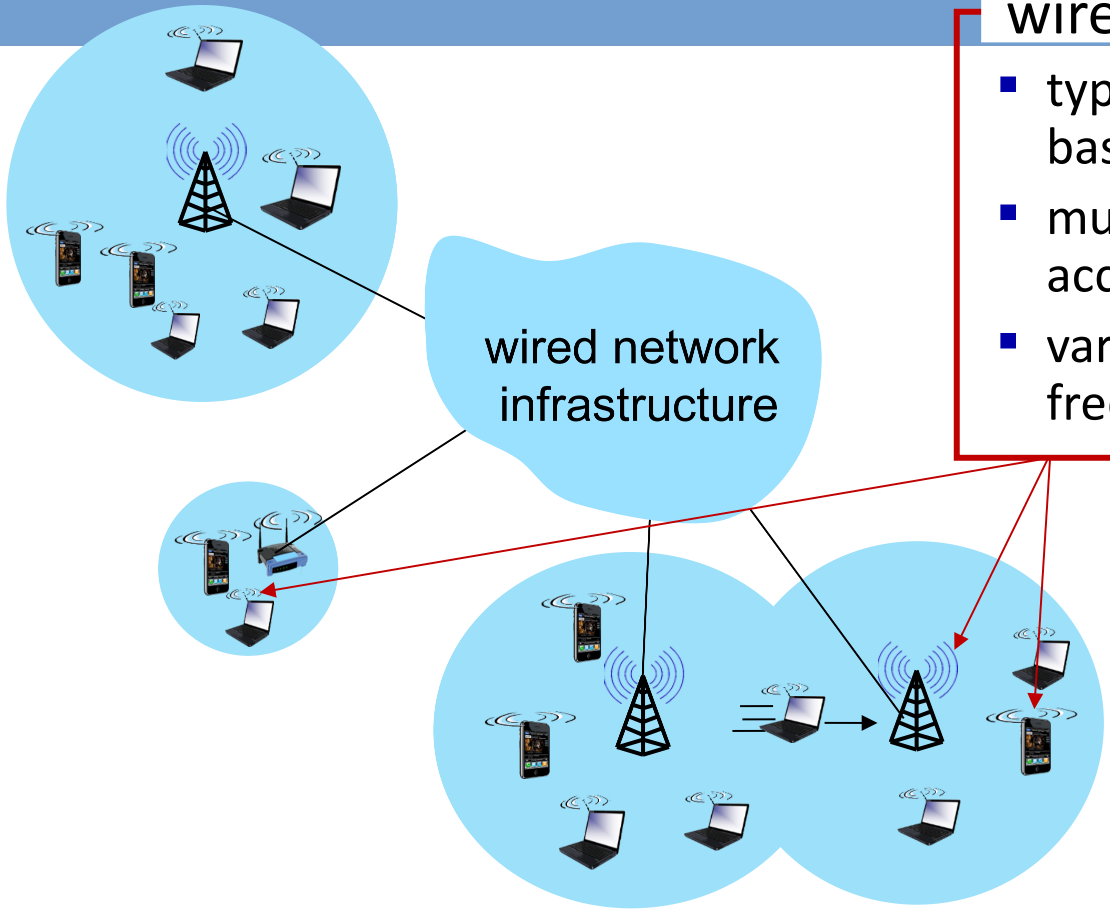
Wireless Networks



base station

- typically connected to wired network
- relay - responsible for sending packets between wired network and wireless host(s) in its “area”
  - e.g., cell towers, 802.11 access points

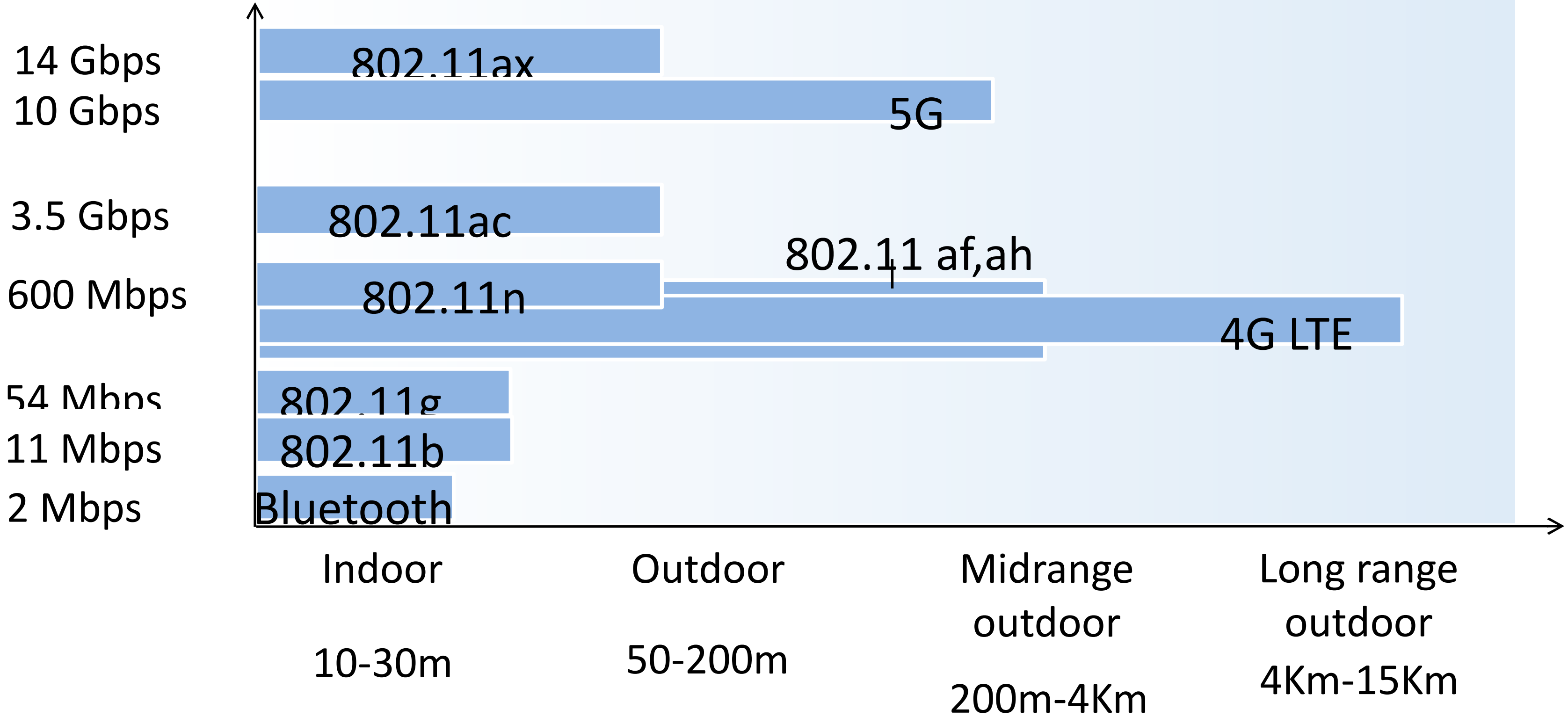
Wireless Networks



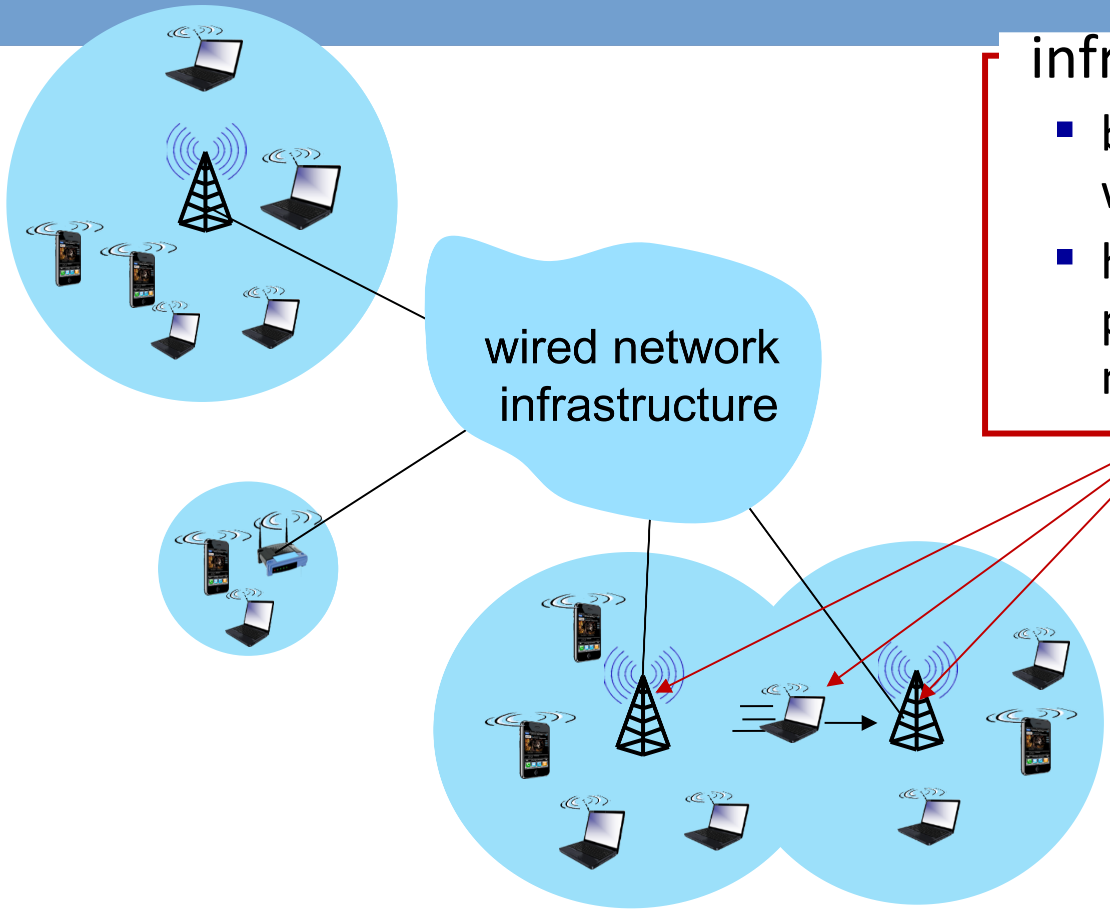
wireless link

- typically used to connect mobile(s) to base station, also used as backbone link
- multiple access protocol coordinates link access
- various transmission rates and distances, frequency bands

Wireless Networks



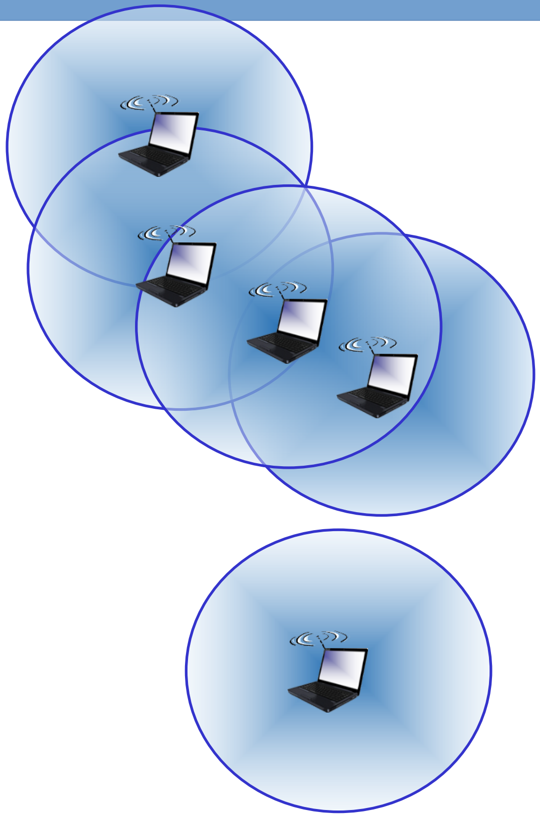
Wireless Networks



infrastructure mode

- base station connects mobiles into wired network
- handoff: mobile changes base station providing connection into wired network

Wireless Networks



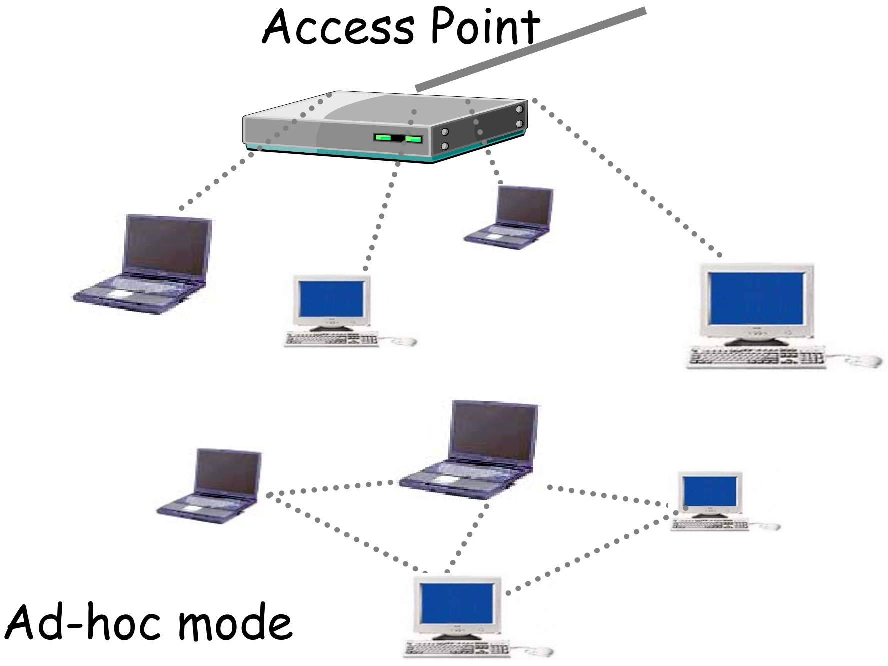
ad hoc mode

- no base stations
- nodes can only transmit to other nodes within link coverage
- nodes organize themselves into a network: route among themselves

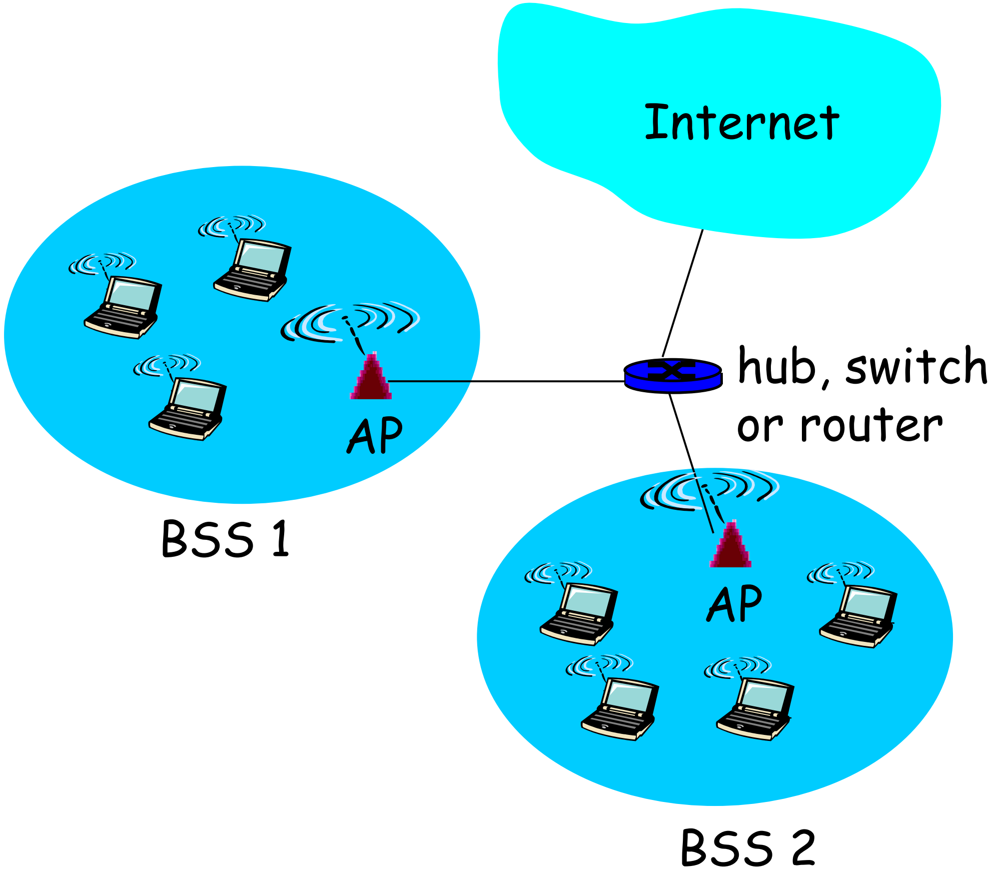
802.11 (Wifi)

802.11

- Basically wireless Ethernet
- Connects a number of computers in a wireless LAN
- Ad-hoc mode (AHM) as well as Access Point mode (APM) supported
- AHM - Only direct communication, no routing functionality
- APM - Computers connected to the Internet via an AP
  - Typical mode of operation
- Access point name refers to a channel; a host connected to an AP tunes to the same channel as the AP



802.11



- wireless host communicates with base station
  - base station = access point (AP)
- Basic Service Set (BSS) (aka "cell") in infrastructure mode contains:
  - wireless hosts
  - access point (AP): base station
  - ad hoc mode: hosts only

802.11

802.11 sender

1 if sense channel idle for **DIFS** then

transmit entire frame (no CD)

2 if sense channel busy then

start random backoff time

timer counts down while channel idle

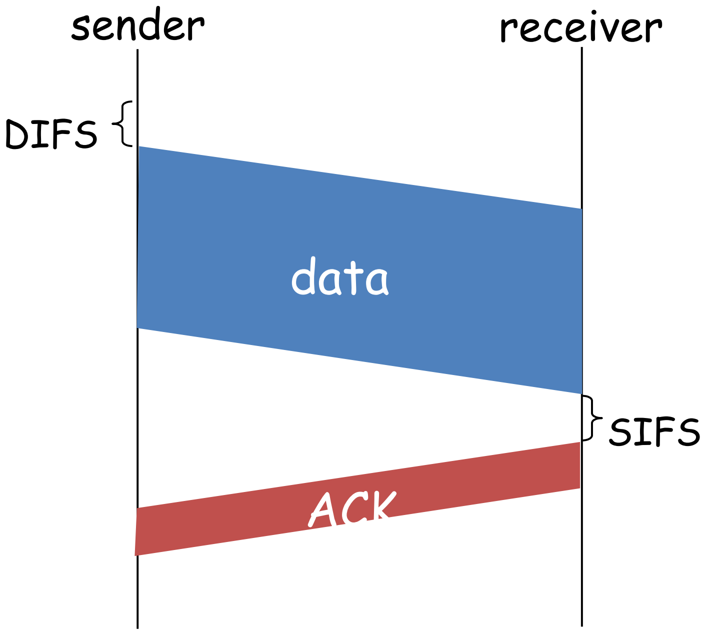
transmit when timer expires

if no ACK, increase random backoff interval, repeat 2

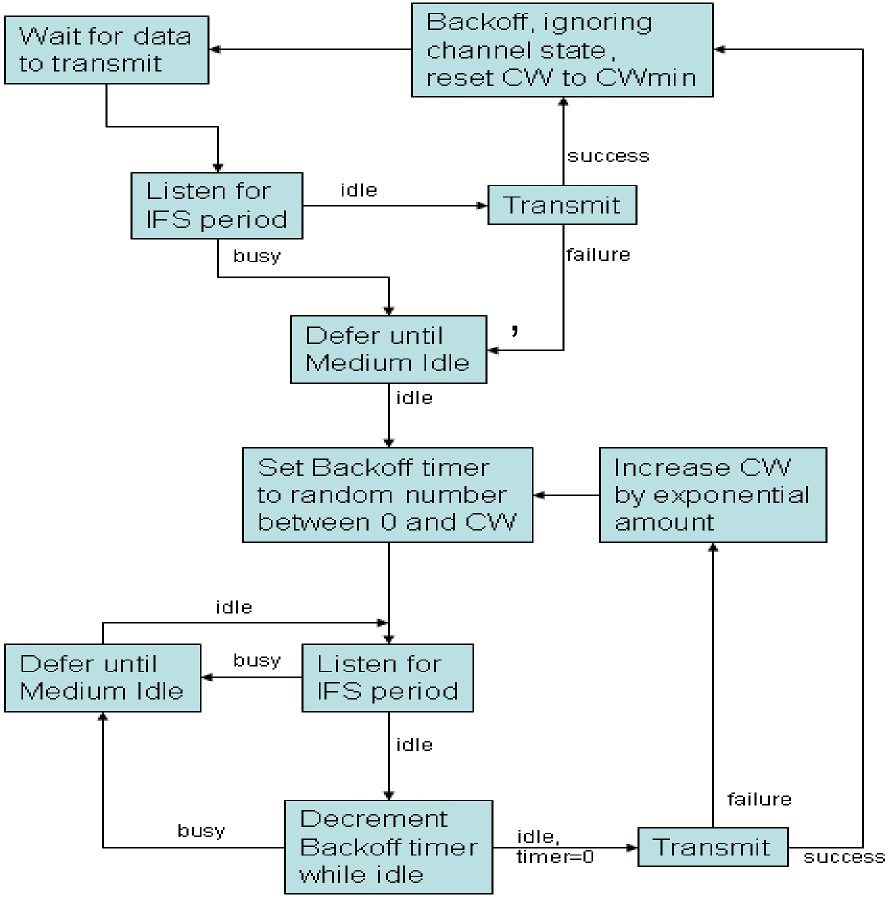
802.11 receiver

- if frame received OK

return ACK after **SIFS** (ACK needed due to hidden terminal problem)



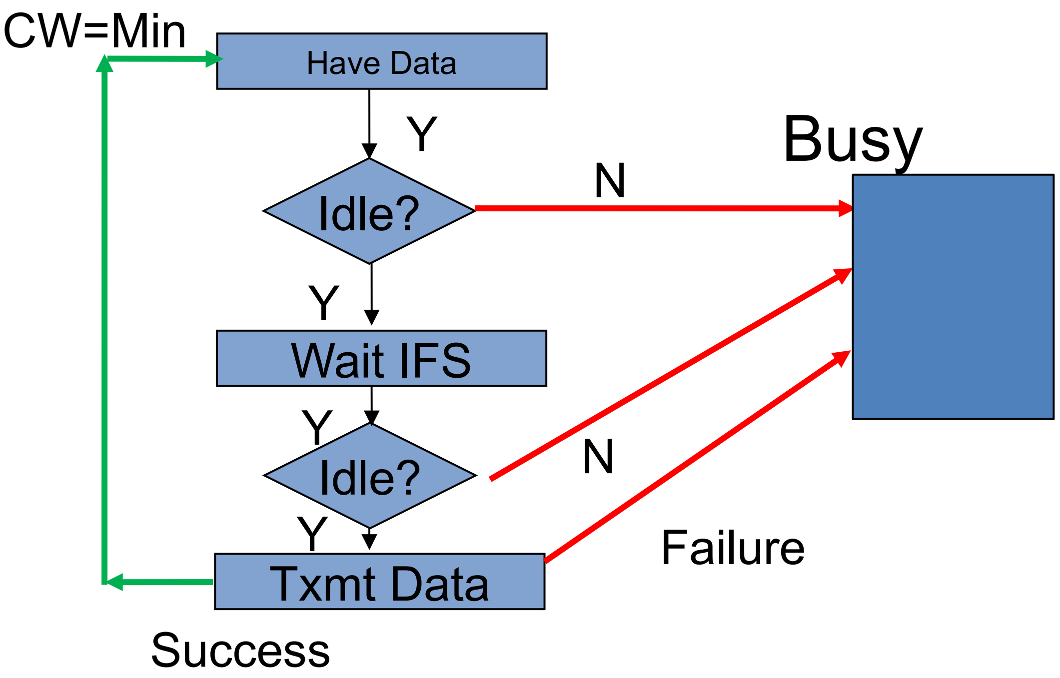
802.11



Wait for data to transmit
Backoff, ignoring channel state, reset CW to CWmin
success
Listen for IFS period
idle
Transmit
busy
failure
Defer until Medium Idle
idle
Set Backoff timer to random number between 0 and CW
Increase CW by exponential amount
idle
Defer until Medium Idle
busy
Listen for IFS period
idle
failure
busy
Decrement Backoff timer while idle
idle, timer=0
Transmit
success

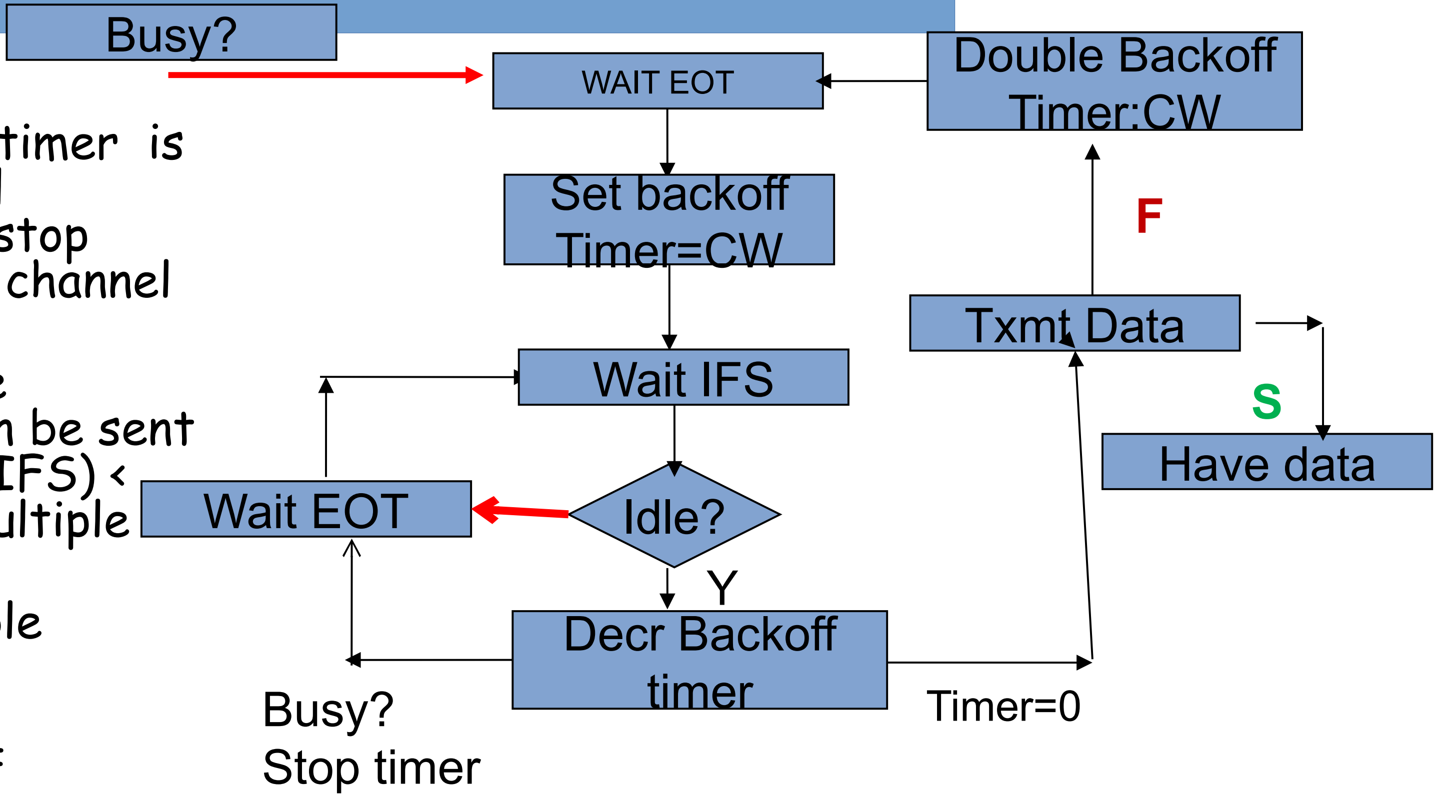
802.11

- Carrier sensing
- Is the medium idle? → Wait for an amount of time (IFS), if still idle transmit
  - IFS = inter frame spacing
- Is the medium busy? → Wait until current txm ends, wait (IFS), if idle wait for random amount of time, else wait until current txm ends and repeat
  - (exponential backoff for collisions)
- If channel is found to be busy wait and start a backoff timer (min = 15 slots)

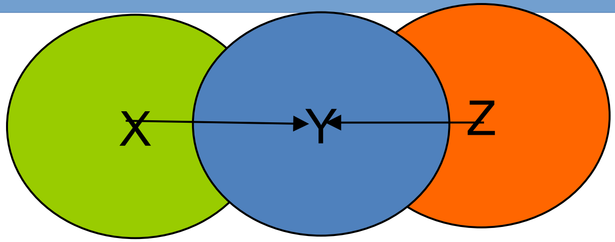


802.11

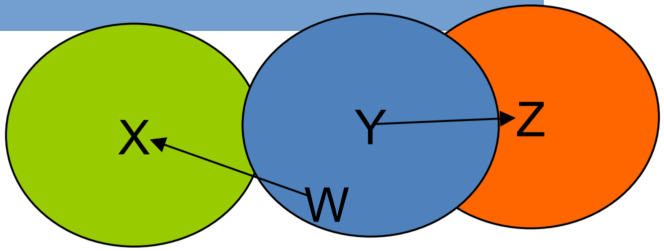
- While the back off timer is running and channel becomes busy then stop timer and wait until channel becomes Idle
- ACKs and immediate response actions can be sent after SIFS (Short IFS) < IFS value used in multiple access control
- On a collision ,Double backoff timer value CWMIN++;
- Exponential backoff



802.11



Hidden Terminal Problem



Exposed Terminal Problem

- Hidden terminal problem
  - Z does not hear X; hence transmits to Y and collides with transmission from X
  - No carrier does not imply send
- Exposed terminal problem
  - W hears Y but can safely transmit to X
  - Carrier may not imply don't send

802.11

- Sender sends a small packet RTS (request to send) before sending data
- Receiver sends CTS (clear to send)
- All potential senders hearing RTS waits until a CTS is heard from some receiver
- If no CTS, transmit
- If CTS, wait for a time for sender to send data
- Hear RTS, but no CTS, then send
  - Exposed terminal case
- Don't hear RTS, but CTS receiver is close, don't send
  - Hidden terminal case

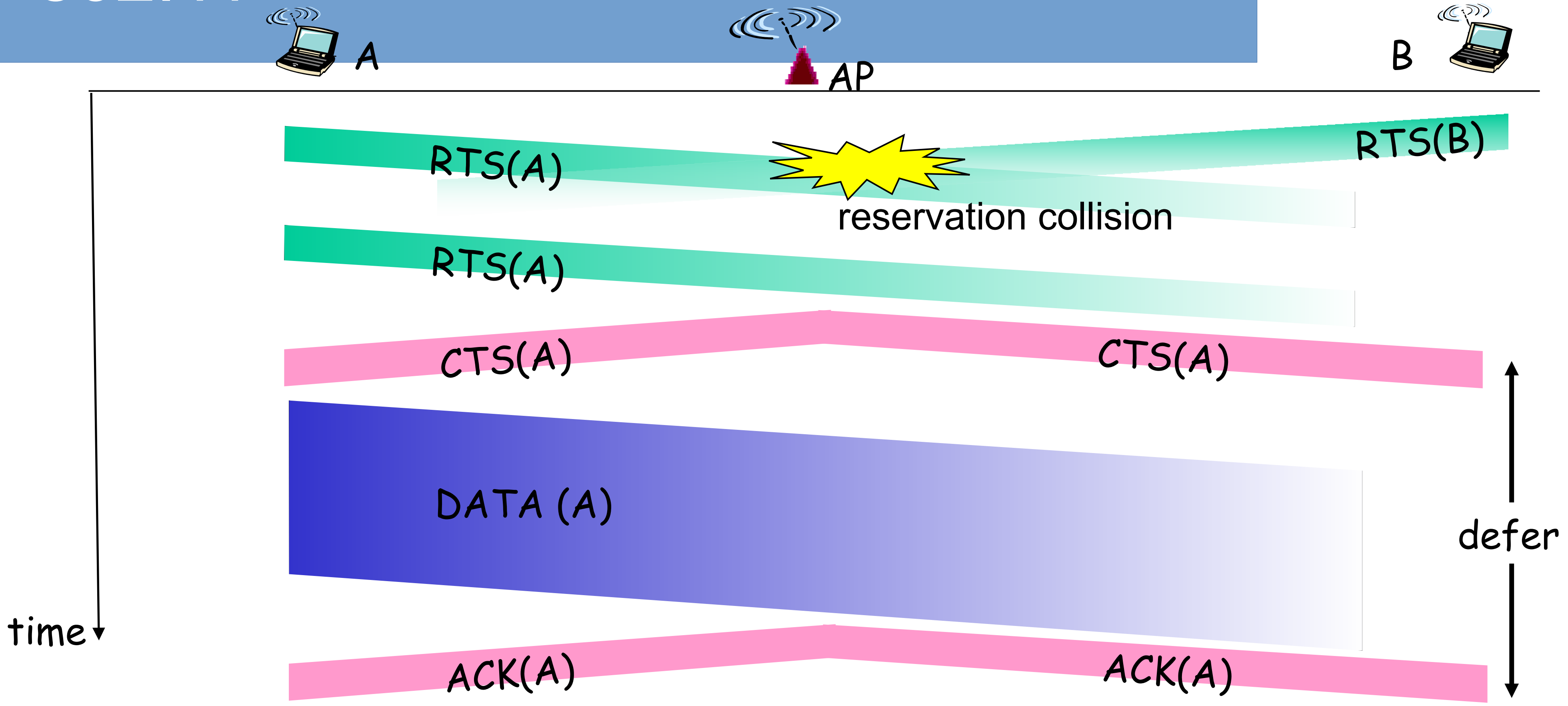
802.11

idea: allow sender to "reserve" channel rather than random access of data frames: avoid collisions of long data frames

- sender first transmits *small* request-to-send (RTS) packets to BS using CSMA
  - RTSs may still collide with each other (but they're short)
- BS broadcasts clear-to-send CTS in response to RTS
- CTS heard by all nodes
  - sender transmits data frame
  - other stations defer transmissions

avoid data frame collisions completely using small reservation packets!

802.11



A
AP
B
RTS(A)
RTS(B)
reservation collision
RTS(A)
CTS(A)
CTS(A)
DATA (A)
defer
time
ACK(A)
ACK(A)

802.11

2	2	6	6	6	2	6	0 - 2312	4
frame control	duration	address 1	address 2	address 3	seq control	address 4	payload	CRC

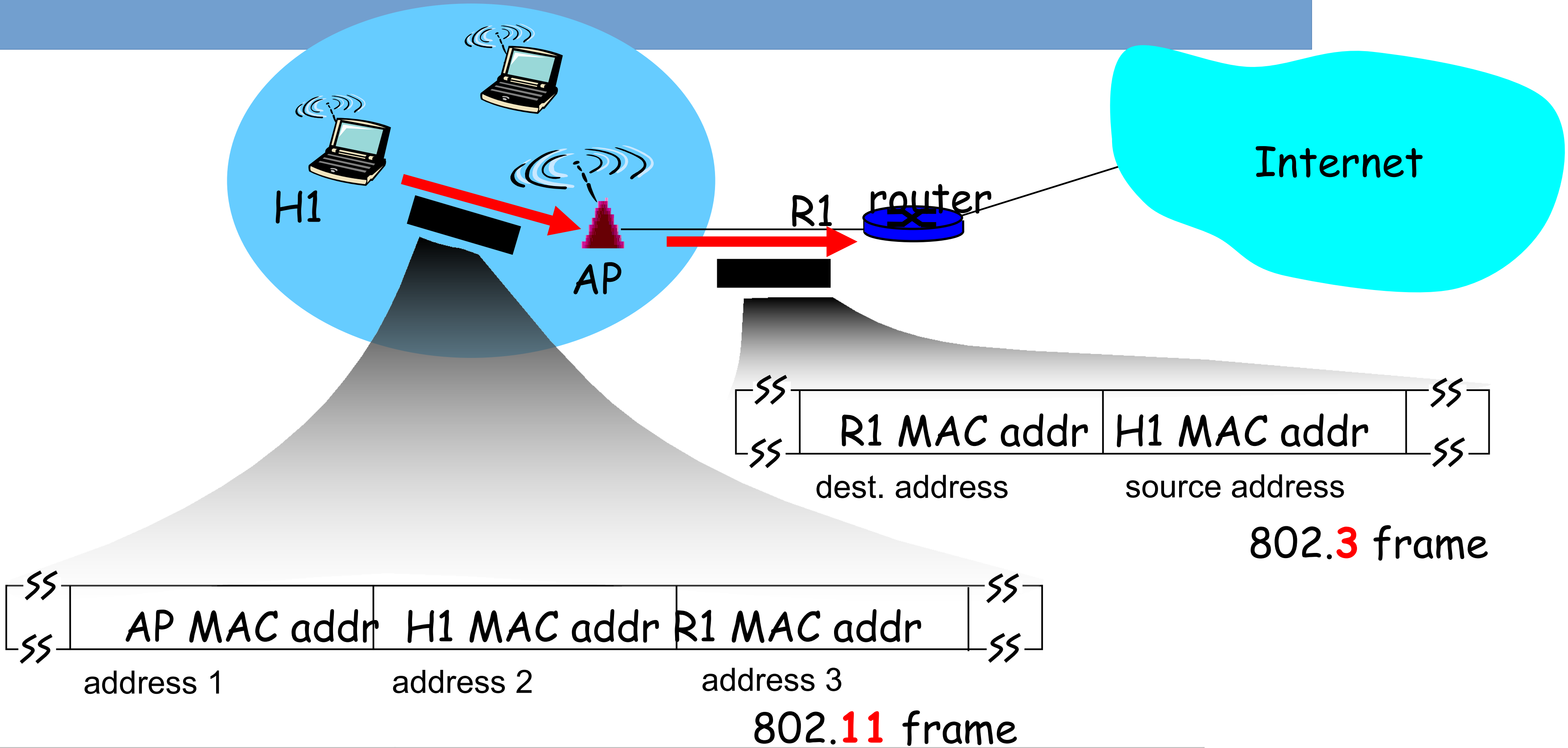
Address 4: used only in ad hoc mode

Address 1: MAC address of wireless host or AP to receive this frame

Address 3: MAC address of router interface to which AP is attached

Address 2: MAC address of wireless host or AP transmitting this frame

802.11



R1 MAC addr	AP MAC addr	H1 MAC addr

← From AP

802.11

duration of reserved transmission time (RTS/CTS)

frame sequence # (for reliable data transfer)

2	2	6	6	6	2	6	0 - 2312	4
frame control	duration	address 1	address 2	address 3	seq control	address 4	payload	CRC

2	2	4	1	1	1	1	1	1	1	1
protocol version	type	subtype	to AP	from AP	more frag	retry	power mgt	more data	WEP	rsvd

frame type (RTS, CTS, ACK, data)

802.11

- Very popular in buildings, public spaces
- Free/unlicensed spectrum interference issues
- Security, privacy, authentication being added