# Course Summary; SDN; Datacenter

**Qiao Xiang**, Congming Gao

https://sngroup.org.cn/courses/cnns-xmuf23/index.shtml

12/14/2023

This deck of slides are heavily based on CPSC 433/533 at Yale University, by courtesy of Dr. Y. Richard Yang.

# Outline

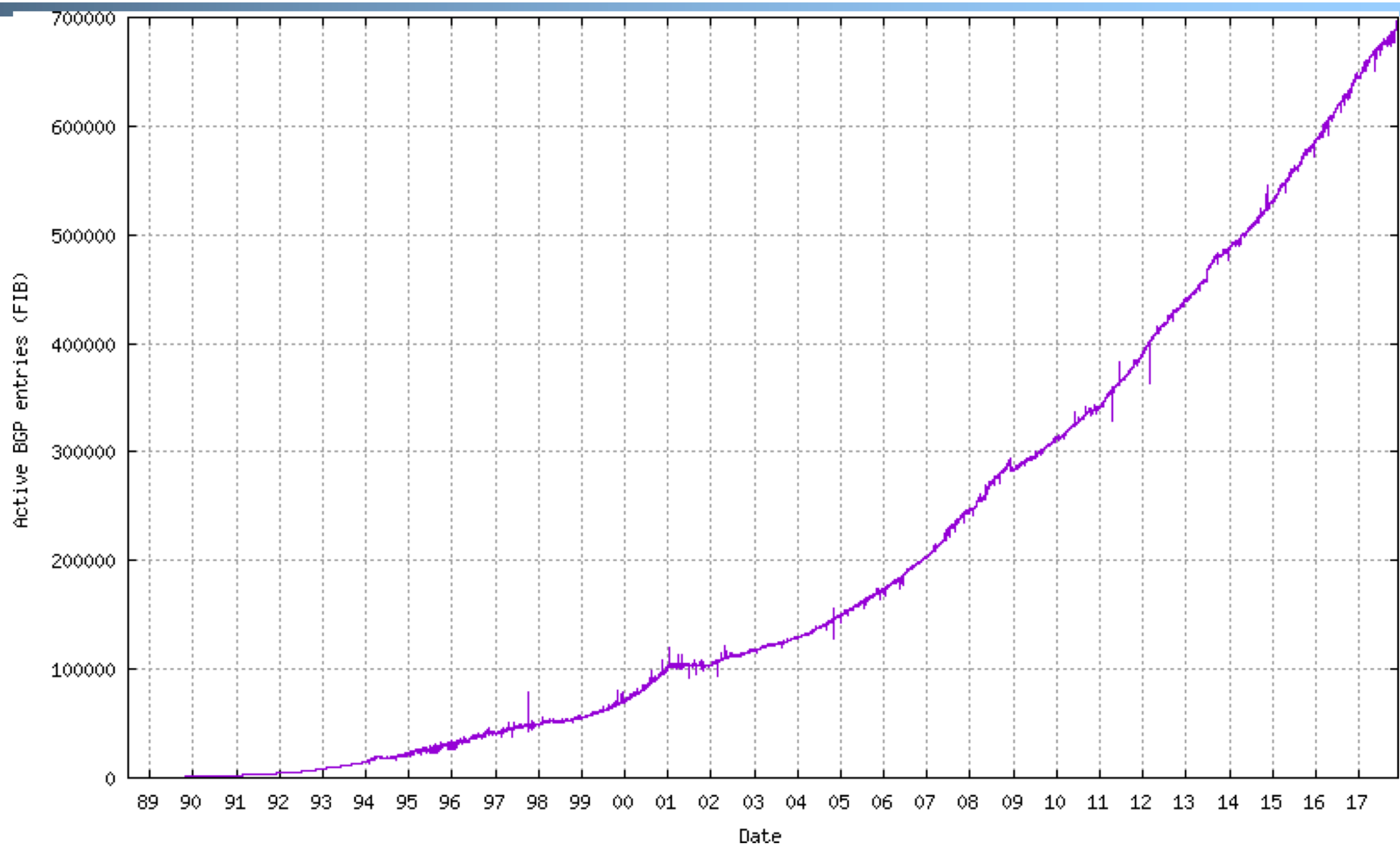
- Admin and recap
- Link layer
- Course Summary

# Admin

- Lab 4: Transport layer, implementing reliability data transfer, ~~due on Jan. 7~~ due on Jan. 10
- Lab 5: Network layer, questions on routing and forwarding, due on Dec. 24
- Class project: extend lab 4 to implement flow control and congestion control, due on Jan. 21.

- Final exam: Jan. 8 afternoon

# Routing Table Size of BGP (number of globally advertised, aggregated entries)




Active BGP Entries (http://bgp.potaroo.net/as1221/bgp-active.html)

Internet Growth (http://www.caida.org/research/topology/as_core_network/historical.xml)

# IP Addressing: How to Get One?

Q: How does an ISP get its block of addresses?

A: Local Internet Registry (LIR) or National Internet Registry (NIR)

https://www.iana.org/numbers

https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml

Use
%whois <IP address>
to check who is allocated the given address.

# IP addresses: How to Get One?

**Q:** How does a *host* get an IP address?

**A:**

- Static configured
  - unix:
    %/sbin/ifconfig eth0 inet 192.168.0.10 netmask 255.255.255.0
- DHCP: Dynamic Host Configuration Protocol (RFC2131): dynamically get address from a DHCP server

# DHCP Goal and History

- Goal: allow host to *dynamically* obtain its IP address from network server when it joins network
- History
  - 1984 Reverse ARP (RFC903): obtain IP address, but at link layer, and hence requires a server at each network link
  - 1985 Bootstrap Protocol (BOOTP; RFC951): introduces the concept of a relay agent to forward across networks
  - 1993 DHCP (RFC1531): based on BOOTP but can dynamically allocate and reclaim IP addresses in a pool, as well as delivery of other parameters
  - 1993 Errors in editorials led to immediate reissue as RFC1541
  - 1997 DHCP (RFC2131): add DHCPINFORM

# DHCP: Dynamic Host Configuration Protocol

The often used DORA model (4 messages)

- host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

# Outline

- Admin and recap
- Network layer
  - Overview
  - Routing
  - Forwarding (put it together)

# Network Forwarding: Putting it Together

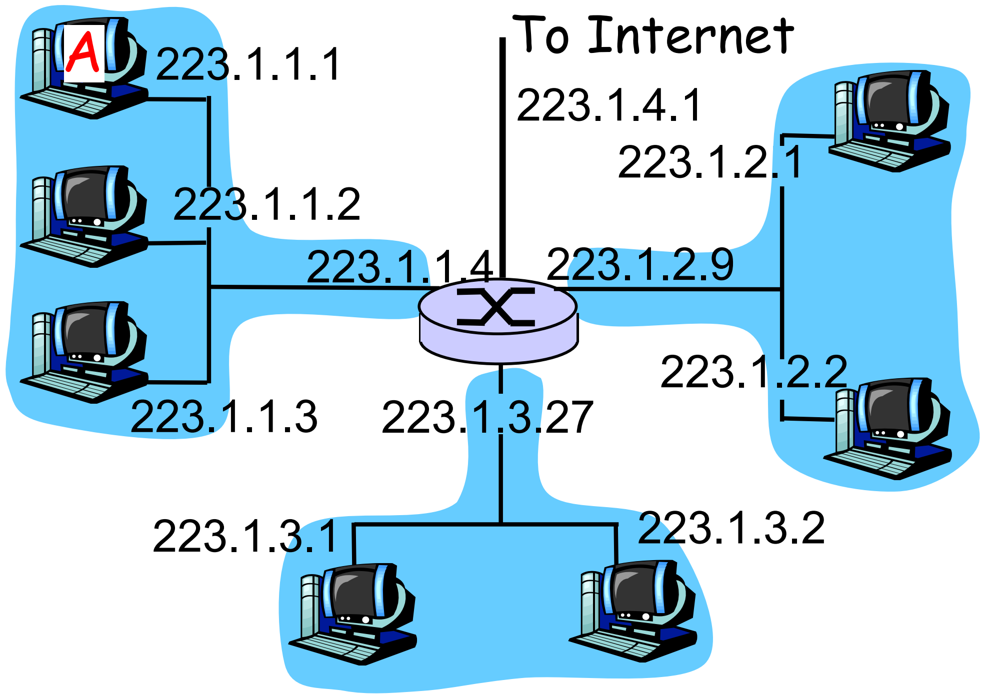
- Forwarding is also called the fast path (upon receiving each packet)

- Slow path: not per packet
  - Get IP address (DHCP, or static)
  - Setup/compute routing table

# Forwarding: Example 1

| | src | dst | |
|---|---|---|---|
| misc fields | 223.1.1.1 | 223.1.1.3 | data |

- Setting: Host A network layer receives a packet above.
- Action:
  - Host A looks up destination in routing table
    - Exercise: Suppose A uses DHCP to obtain its address, how can A construct its routing table (routing information base, RIB)?



# Host Routing Table Example: my Mac

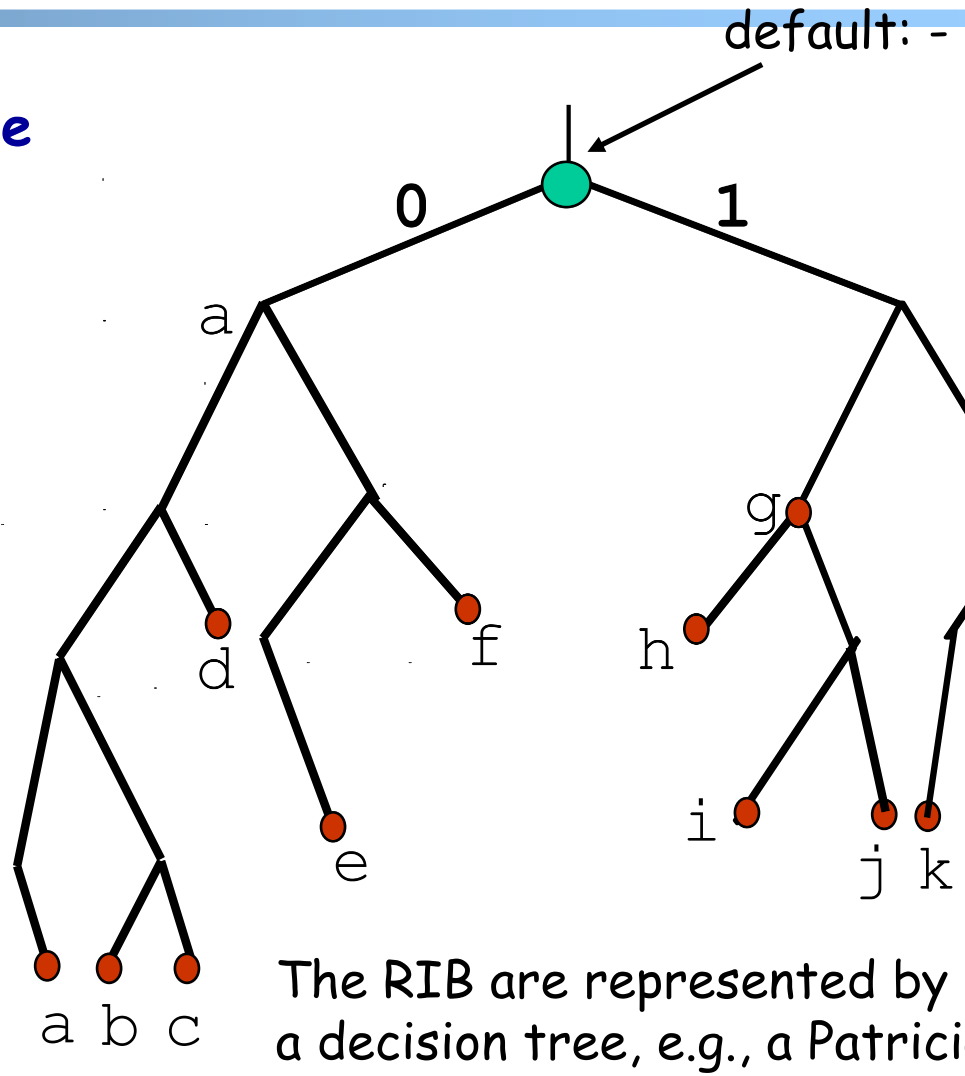
- Mac
  - ifconfig -a
  - netstat -rn (man netstat to see description)

```
Routing tables

Internet:
Destination        Gateway            Flags        Refs      Use   Netif Expire
default            172.27.16.1        UGSc         1470        0     en0
127                127.0.0.1          UCS             1        0     lo0
127.0.0.1          127.0.0.1          UH              4     3788     lo0
169.254            link#4             UCS           106        0     en0
169.254.1.229      link#4             UHLSW           1        0     en0
169.254.5.209      f0:99:bf:1e:6f:de  UHLSW           1        0     en0    989
169.254.8.254      link#4             UHLSW           1        0     en0
169.254.11.96      0:cd:fe:75:59:75   UHLSW           1        0     en0   1009
169.254.13.89      64:9a:be:af:34:53  UHLSW           1        0     en0   1145
169.254.16.49      link#4             UHLSW           1        0     en0
169.254.19.58      link#4             UHLSW           1        0     en0
169.254.19.82      link#4             UHLSW           1        0     en0
169.254.21.198     link#4             UHLSW           1        0     en0
169.254.22.67      0:23:12:12:bc:39   UHLSW           1        0     en0     31
169.254.23.4       link#4             UHLSW           1        0     en0
...
```

# CIDR Forwarding Look Up: Software

| # | prefix | interface |
|---|---|---|
| a) | 00001 | |
| b) | 00010 | |
| c) | 00011 | |
| d) | 001 | |
| e) | 0101 | |
| f) | 011 | |
| g) | 10 | |
| h) | 100 | |
| i) | 1010 | |
| j) | 1011 | |
| k) | 1100 | |




The RIB are represented by a decision tree, e.g., a Patricia Trie to look for the longest match of the destination address

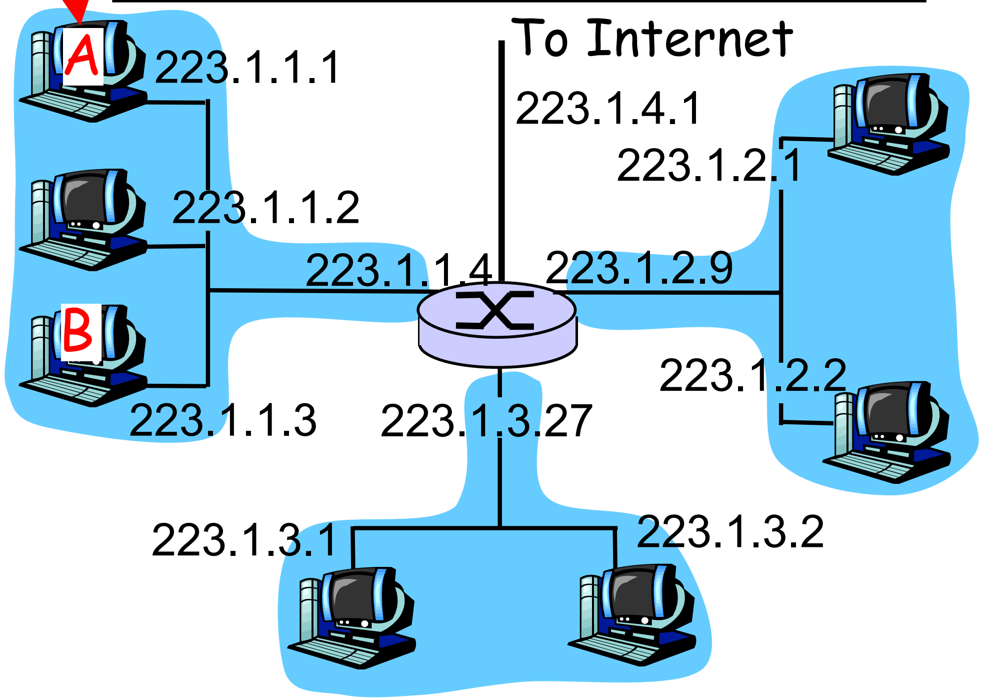
# Putting it Together: Example 1: A->B

| misc fields | src 223.1.1.1 | dst 223.1.1.3 | data |
|---|---|---|---|

- Setting: Host A network layer receives a packet above.
- Action:
  - Host A looks up destination in routing table (on same subnet)
  - Hand datagram to link layer to send inside a link-layer frame
    - Key step: need to map B's IP address 223.1.1.3 to B's MAC address

**forwarding table in A**

| Dest. Net. | next router | Nhops |
|---|---|---|
| 223.1.1/24 | | 1 |
| 223.1.2/24 | 223.1.1.4 | 2 |
| 223.1.3/24 | 223.1.1.4 | 2 |
| 0.0.0.0/0 | 223.1.1.4 | - |



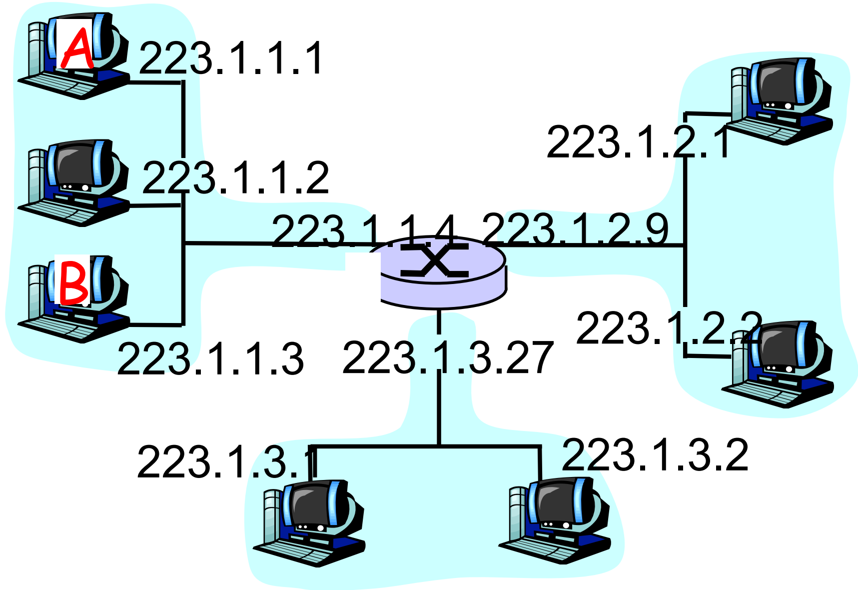
# Comparison of IP address and MAC Address

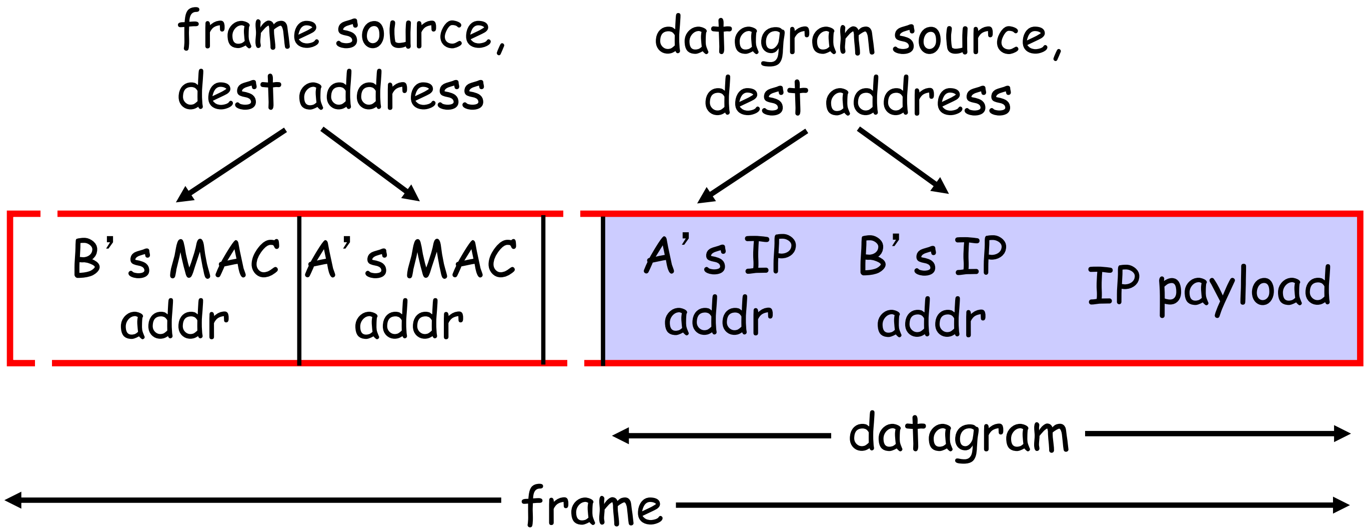
- IP address is locator
  - address depends on network to which an interface is attached
    - NOT portable
  - introduces features (e.g., CIDR) for routing scalability
- MAC address is an identifier
  - dedicated to a device
    - portable
  - flat

- IP address needs to be globally unique (if no NAT)
- MAC address does not need to be globally unique, but the current assignment ensures uniqueness

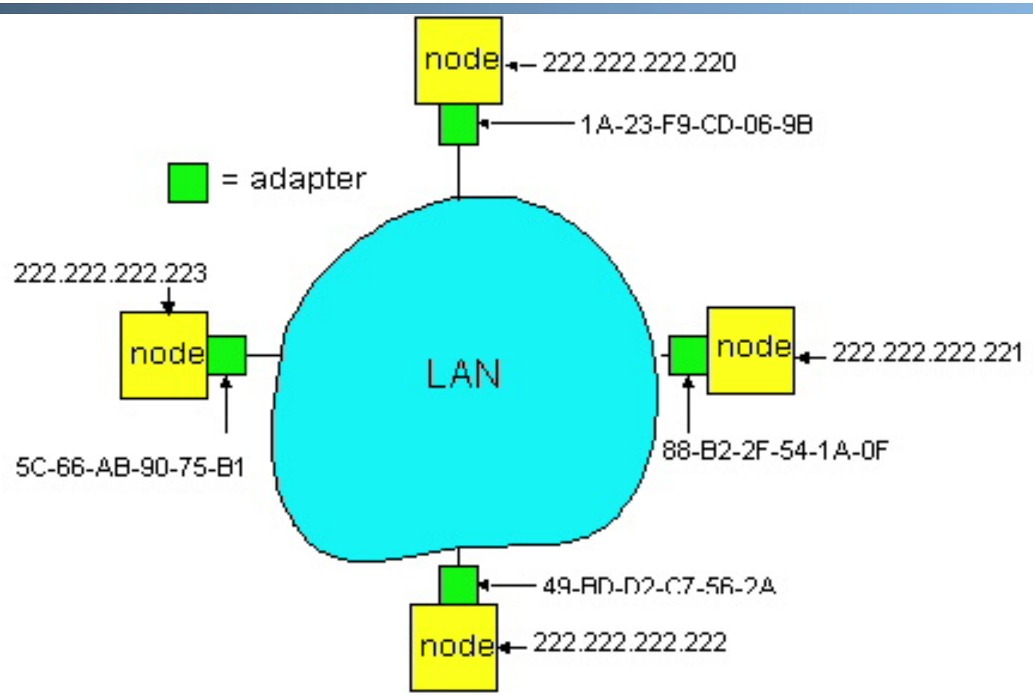
# Issue

A finds the MAC address of B to construct



# Recall: Address Resolution Table




- Each IP node (Host, Router) on LAN has ARP table
- ARP Table: 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)

```
[yry3@cicada yry3]$ /sbin/arp
Address                  HWtype  HWaddress           Flags Mask            Iface
zoo-gatew.cs.yale.edu    ether   AA:00:04:00:20:D4   C                     eth0
artemis.zoo.cs.yale.edu  ether   00:06:5B:3F:6E:21   C                     eth0
lab.zoo.cs.yale.edu      ether   00:B0:D0:F3:C7:A5   C                     eth0
```

# Recall: ARP Protocol

- ARP table by the ARP Protocol, which is a "plug-and-play" protocol
  - nodes create their ARP tables without intervention from net administrator

- A broadcast protocol:
  - source broadcasts query frame, containing queried IP address
    - all machines on LAN receive ARP query
  - destination D receives ARP frame, replies
    - frame sent to A's MAC address (unicast)

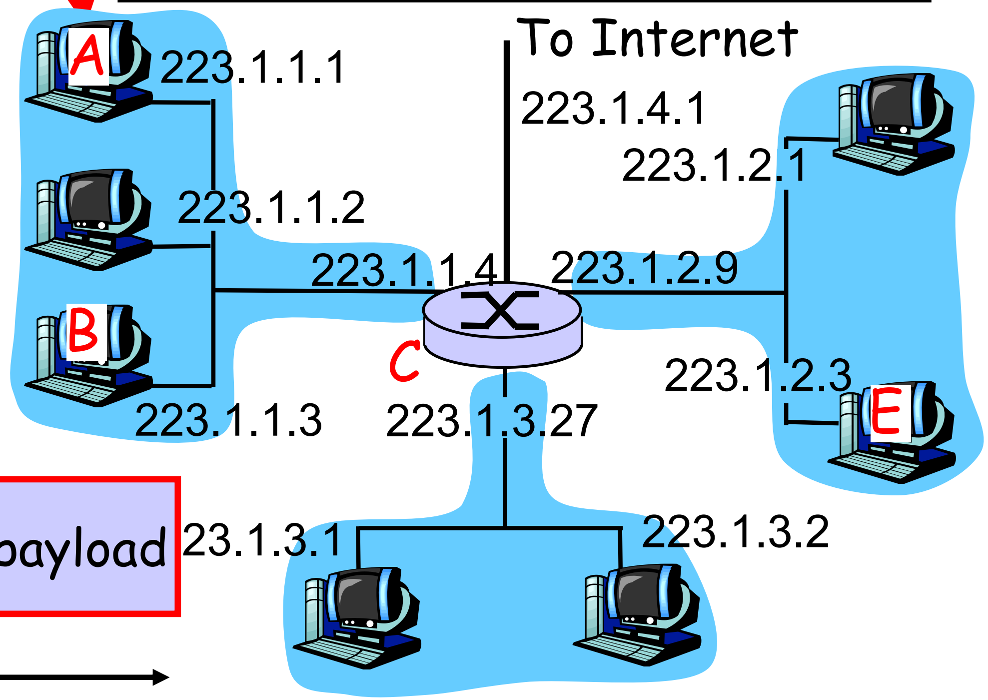
# Putting it Together: Example 2 (Different Networks): A-> E

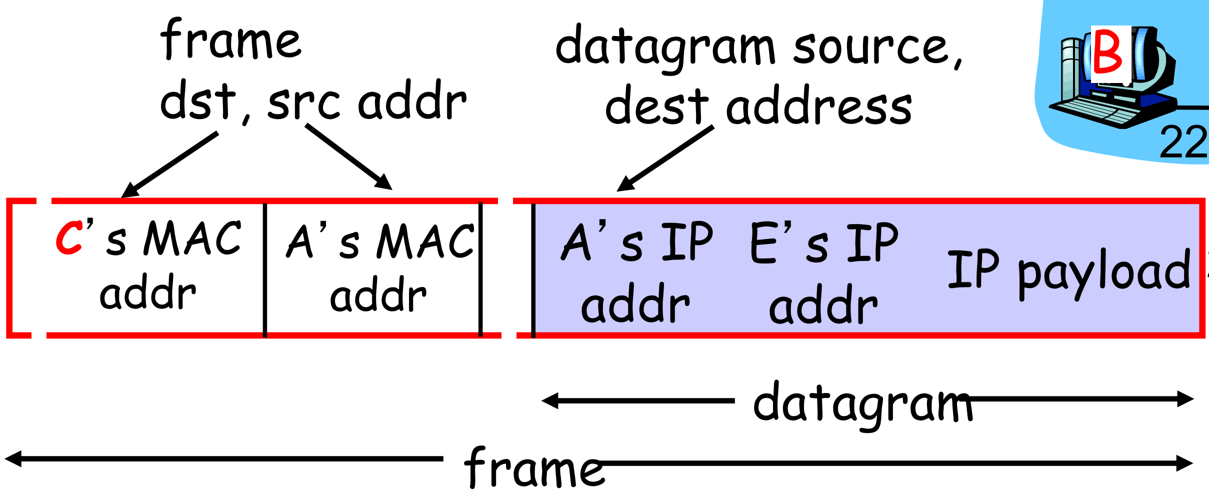
| | | | |
|---|---|---|---|
| misc fields | 223.1.1.1 | 223.1.2.3 | data |

**forwarding table in A**

| Dest. Net. | next router | Nhops |
|---|---|---|
| 223.1.1/24 | | 1 |
| 223.1.2/24 | 223.1.1.4 | 2 |
| 223.1.3/24 | 223.1.1.4 | 2 |
| 0.0.0.0/0 | 223.1.1.4 | - |

- Setting: Host A network layer receives a packet above.
- Action:
  - Host A looks up destination in routing table
    - Find next hop should be 223.1.1.4
  - Hand datagram to link layer to send inside a link-layer frame



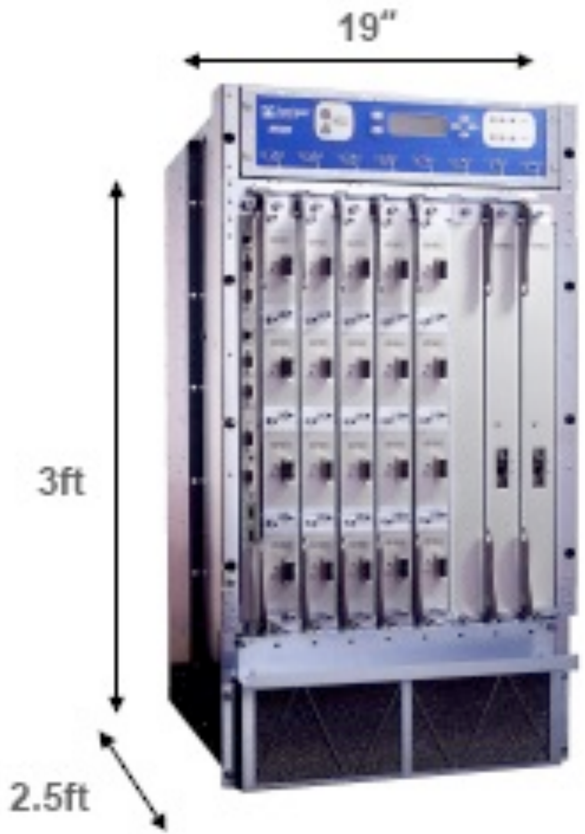



# What A Router Looks Like: Outside

### Cisco GSR 12416

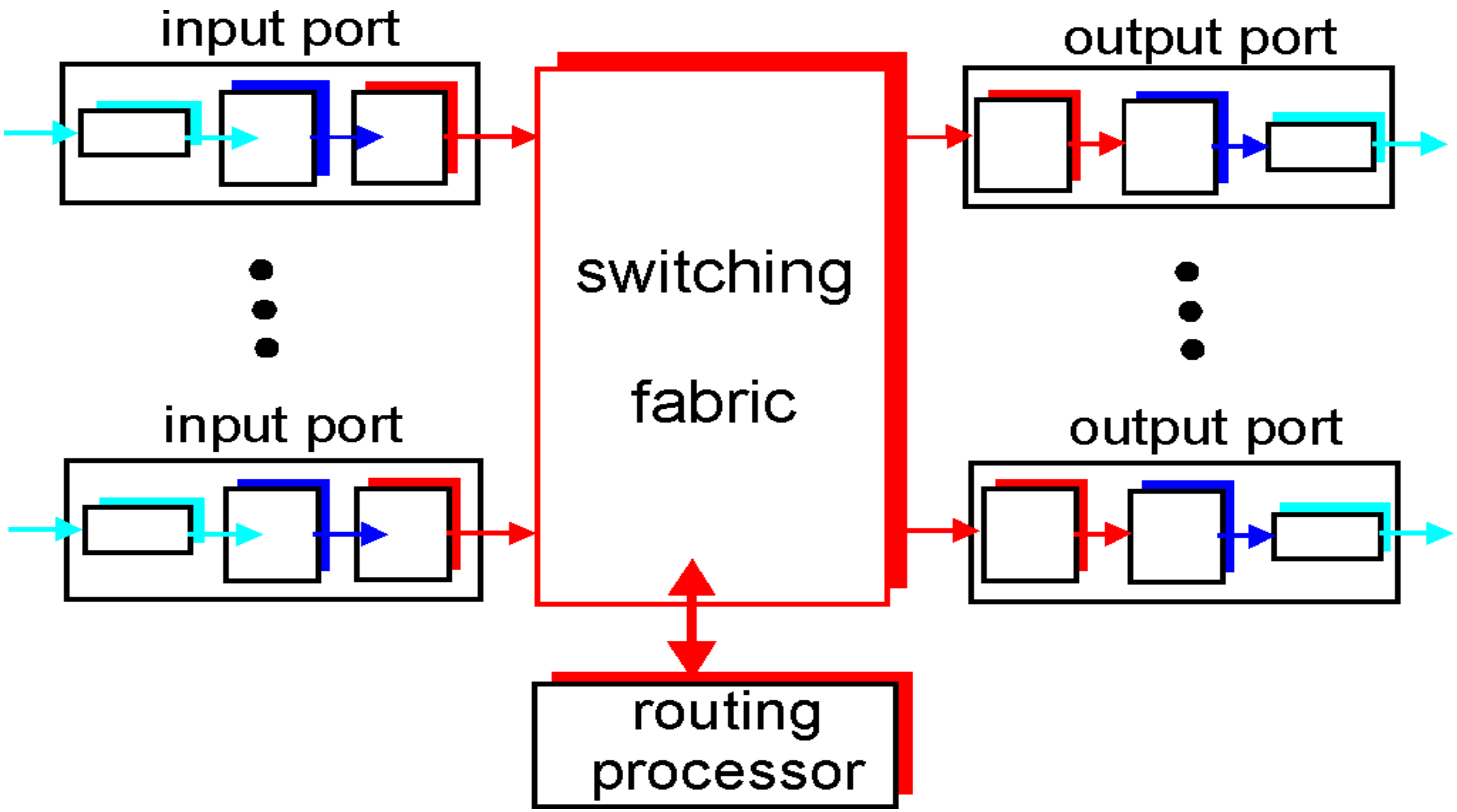

Capacity: 160Gb/s
Power: 4.2kW

### Juniper M160

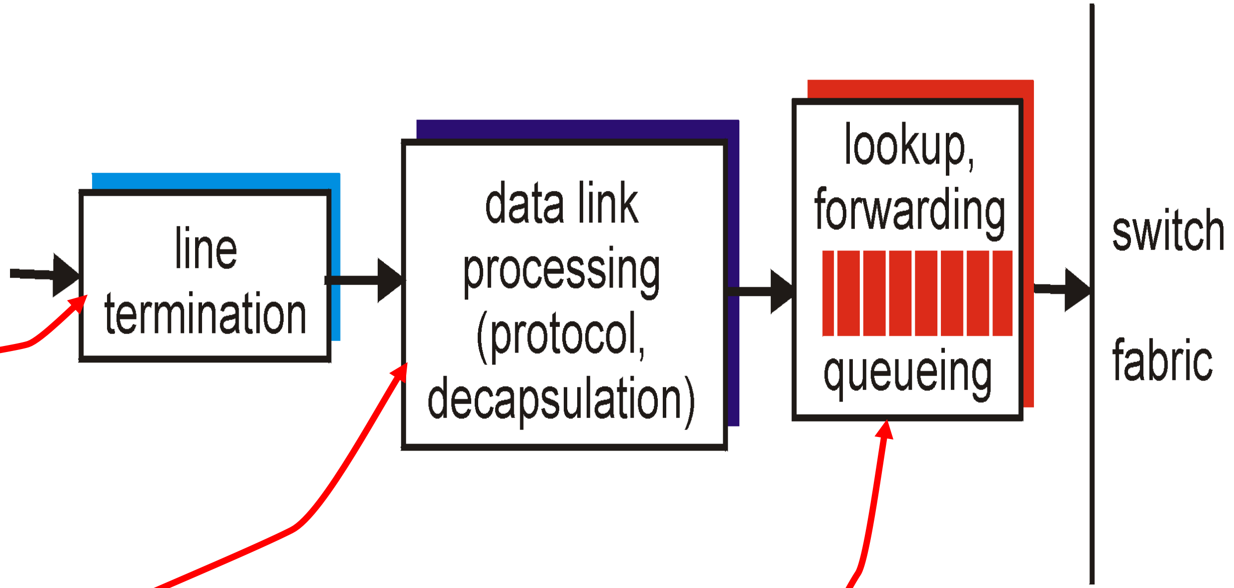



Capacity: 80Gb/s
Power: 2.6kW

# Look Inside a Router




# Look Inside a Router: Input Port



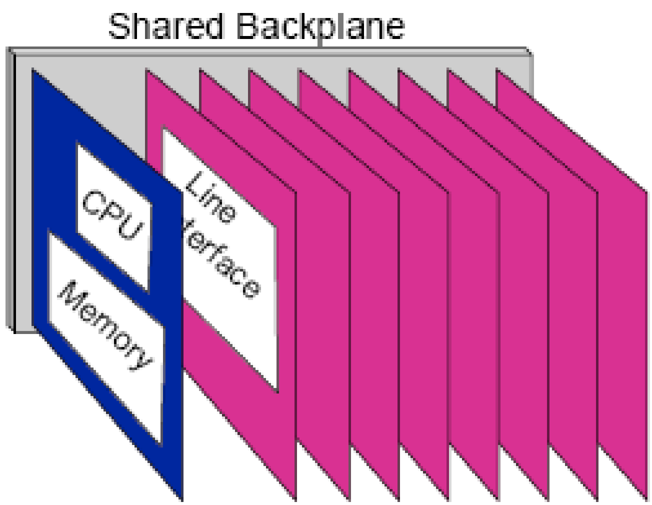

physical layer:
bit-level reception

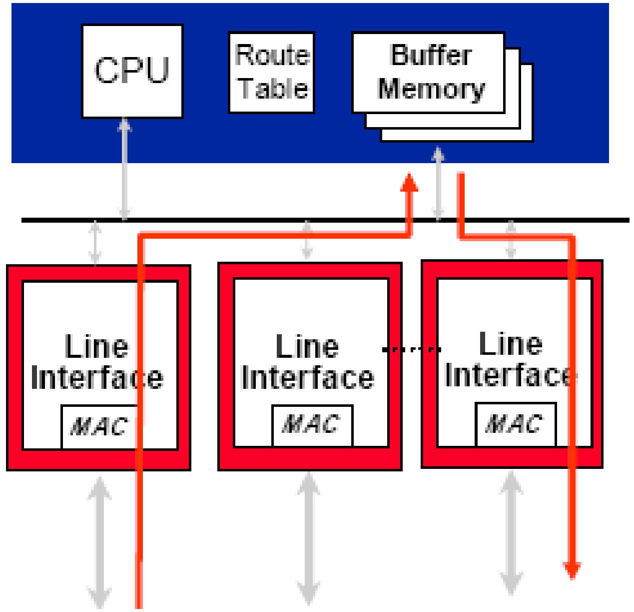
data link layer:
e.g., Ethernet

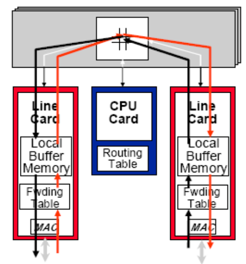
network layer:
lookup output port using forwarding table

# Look Inside a Router: Switching Fabric

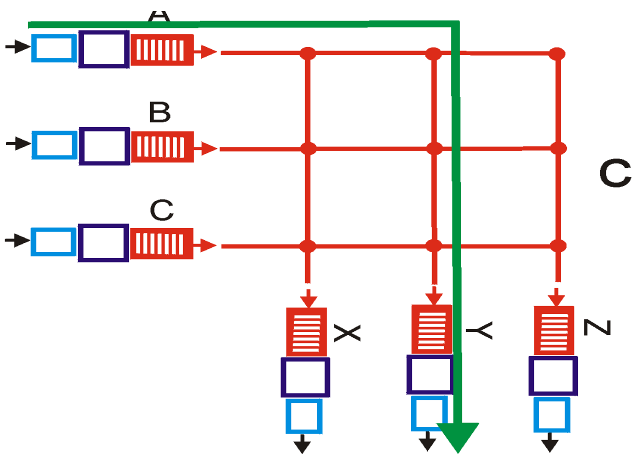
Low End



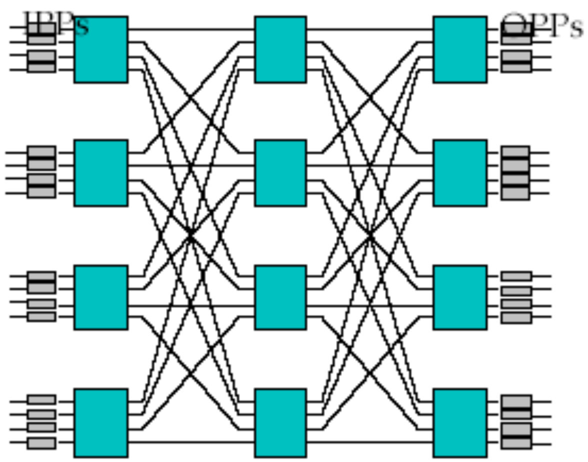




High End

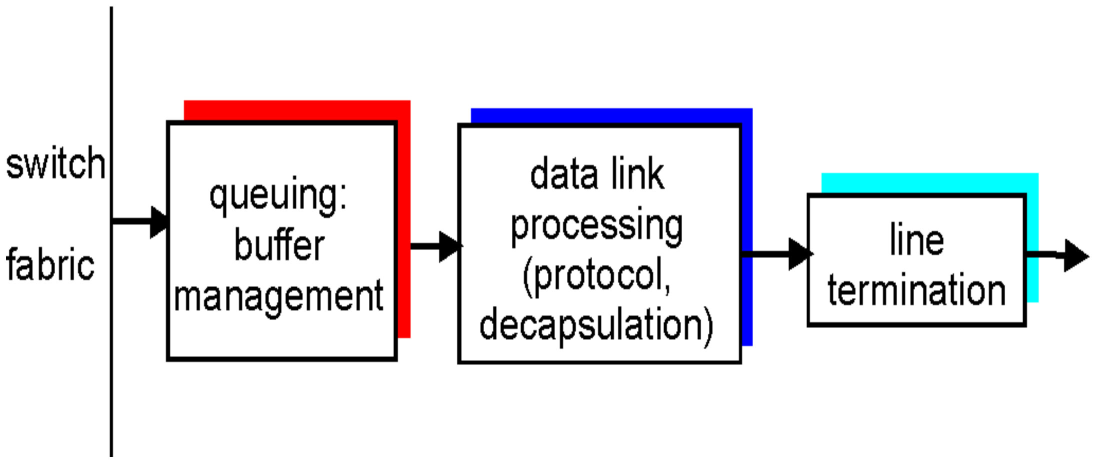



crossbar



Banyan

# Look Inside a Router: Output Port




- *Buffering* required when datagrams arrive from fabric faster than the transmission rate
- *Queueing (delay) and loss due to output port buffer overflow !*
- *Scheduling and queue/buffer management* choose among queued datagrams for transmission

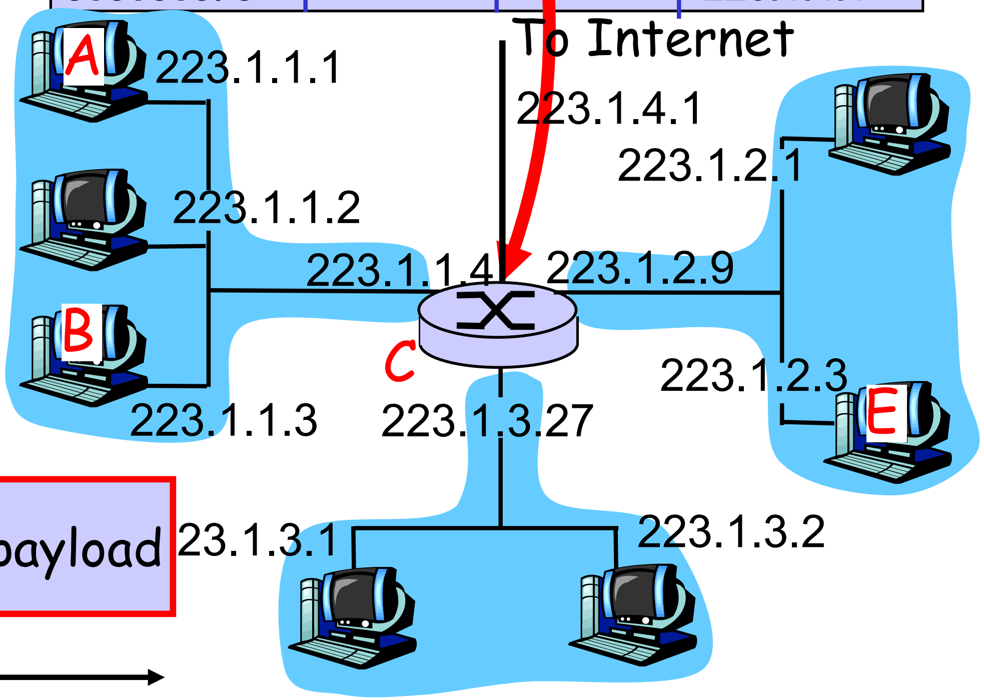
# Putting it Together: Example 2 (Different Networks): A-> E

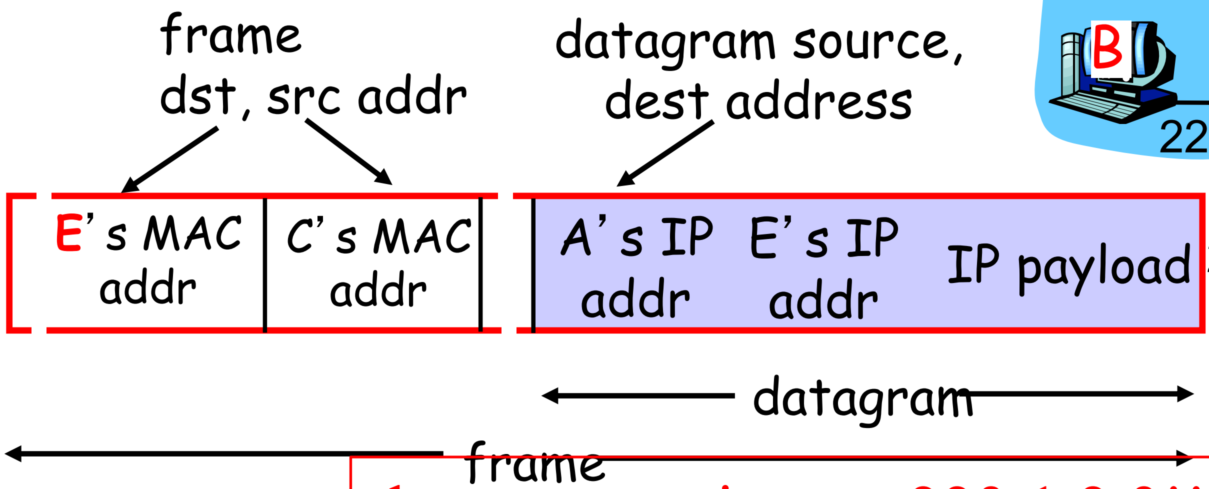
| misc fields | 223.1.1.1 | 223.1.2.3 | data |
|---|---|---|---|

- Setting: Packet above arrives at Router C's network layer.
- Action:
  - Router C conducts standard router actions
    - Assume packet correct, find next hop should be 223.1.2.9
  - Hand datagram to link layer to send inside a link-layer frame

**forwarding table in router**

| Dest. Net | router | Nhops | interface |
|---|---|---|---|
| 223.1.1/24 | - | 1 | 223.1.1.4 |
| 223.1.2/24 | - | 1 | 223.1.2.9 |
| 223.1.3/24 | - | 1 | 223.1.3.27 |
| 0.0.0.0/0 | - | - | 223.1.4.1 |



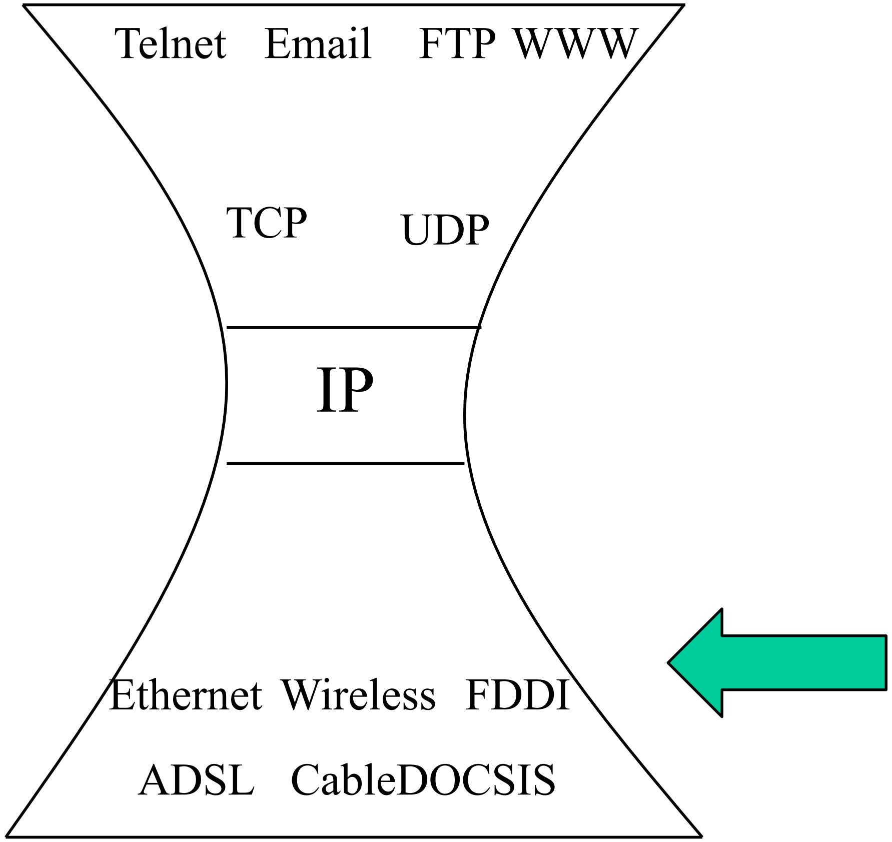

datagram arrives at 223.1.2.3!! (hooray!)

# Summary of Network Layer

- We have covered the very basics of the network layer
  - routing and basic forwarding

- There are multiple other topics that we did not cover
  - Multicast/anycast
  - QoS
  - slides as backup
    just in case you need
    reading in the winter

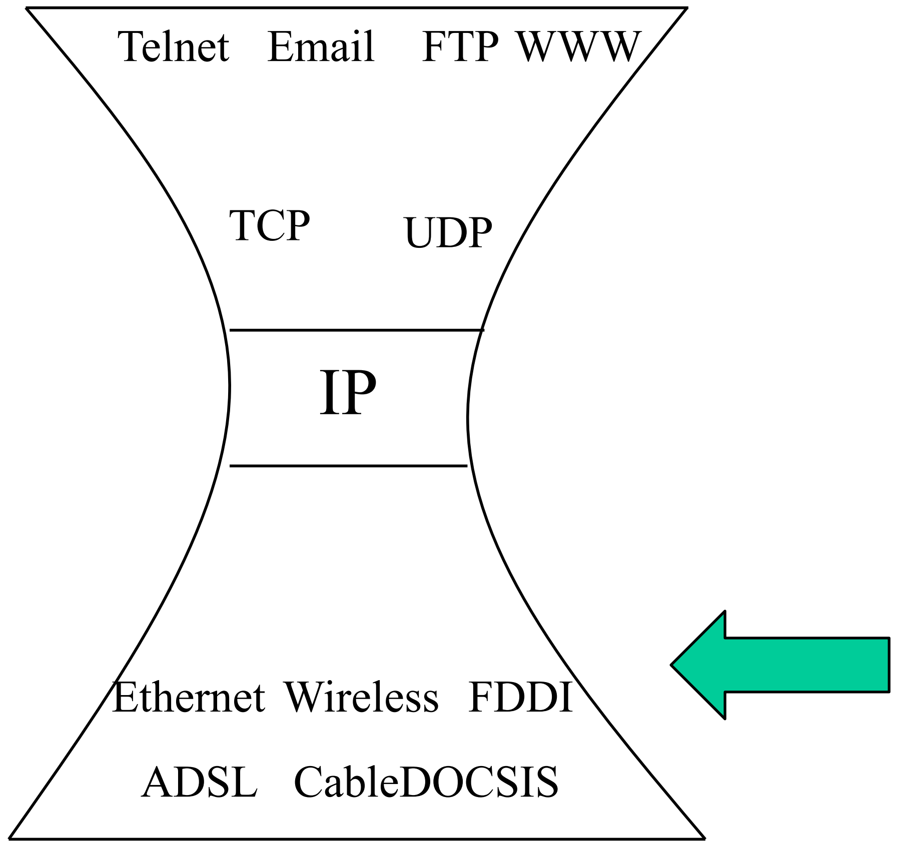
# Roadmap: The Hourglass Architecture of the Internet



# Exercise

- DHCP lease renew

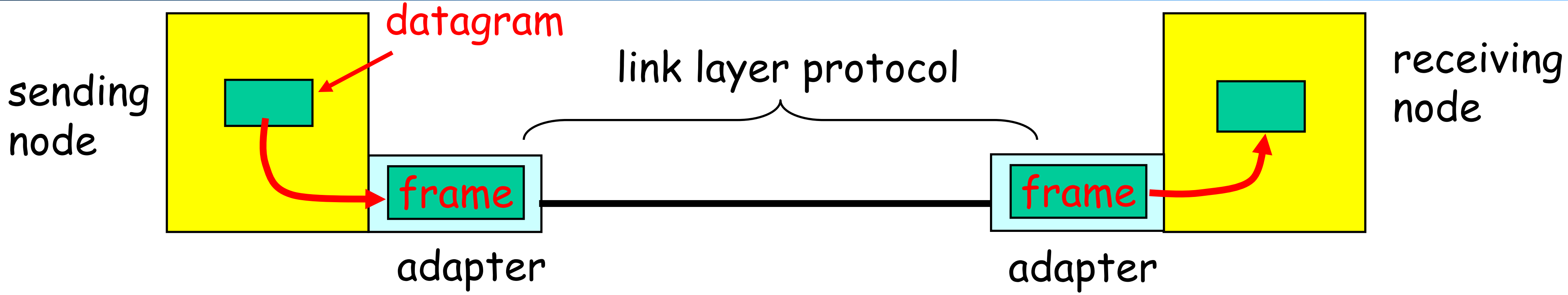
# Roadmap: The Hourglass Architecture of the Internet



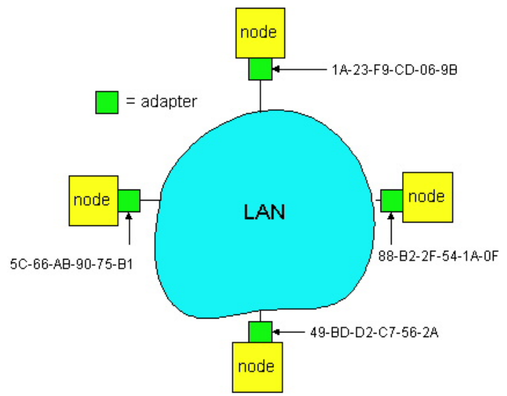
# Link Layer Services

- Framing
  - encapsulate datagram into frame, adding header, trailer and error detection/correction
- Multiplexing/demultiplexing
  - frame headers to identify src, dest
- Reliable delivery between adjacent nodes
  - we learned how to do this already !
  - seldom used on low bit error link (fiber, some twisted pair)
  - common for wireless links: high error rates
- Media access control
- Forwarding/switching with a link-layer (Layer 2) domain

# Adaptors Communicating




- link layer typically implemented in "adaptor" (aka NIC)
  - Ethernet card, modem, 802.11 card, cloud virtual switch
- adapter is semi-autonomous, implementing link & physical layers
- in most link-layer, each adapter has a unique link layer address (also called MAC address)

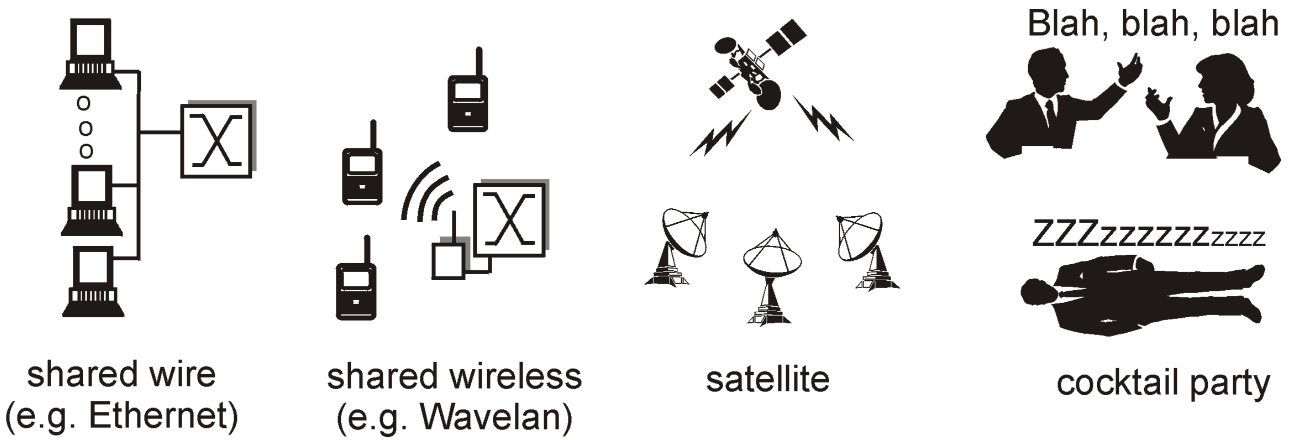


# Outline

- Admin and recap
- Network layer
- Link layer
  - Overview
  - Media access
  - Link layer forwarding

# Multiple Access Links and Protocols

- Many link layers use broadcast (shared wire or medium)
  - traditional Ethernet; Cable networks
  - 802.11 wireless LAN; cellular networks
  - satellite




- Problem: if two or more simultaneous transmissions, due to interference, only one node can send successfully at a time (see CDMA later for an exception)

# Multiple Access Protocol

- Protocol that determines how nodes share channel, i.e., determines when nodes can transmit
- Communication about channel sharing must use channel itself !

- Discussion: properties of an ideal multiple access protocol.

# Ideal Mulitple Access Protocol

Broadcast channel of rate R bps

- Efficiency: when only one node wants to transmit, it can send at full rate R
- Rate allocation:
  - simple fairness: when N nodes want to transmit, each can send at average rate R/N
  - we may need more complex rate control
- Decentralized:
  - no special node to coordinate transmissions
  - no synchronization of clocks
- Simple

# MAC Protocols

Goals

- efficient, fair, decentralized, simple

Three broad classes:

- non-partitioning
  - random access
    - allow collisions
  - "taking-turns"
    - a token coordinates shared access to avoid collisions
- channel partitioning
  - divide channel into smaller "pieces" (time slot, frequency, code)

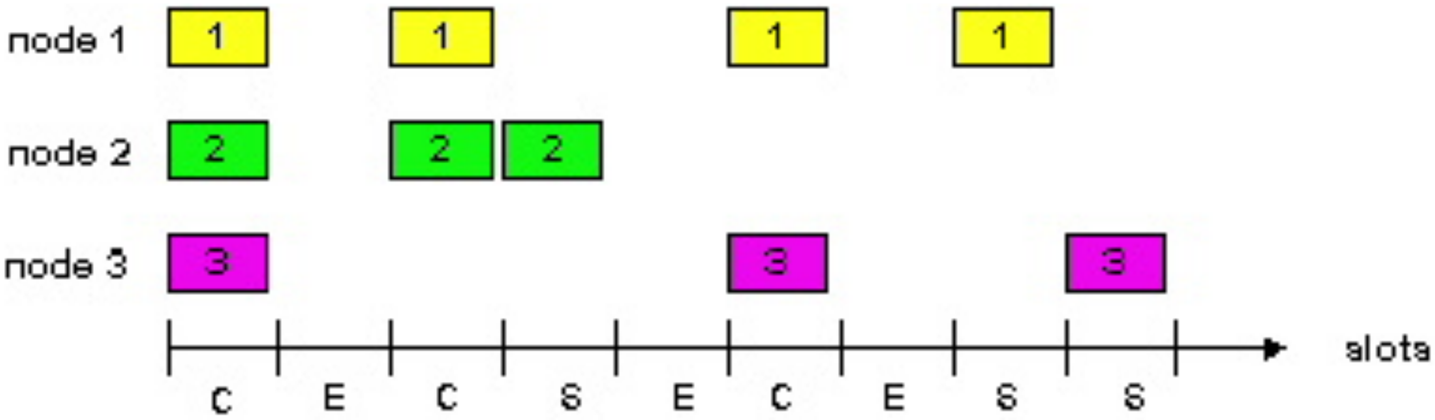
# Focus: Random Access Protocols

- Examples of random access MAC protocols:
  - slotted ALOHA and pure ALOHA
  - CSMA and CSMA/CD, CSMA/CA
  - Ethernet, WiFi 802.11

- Key design points:
  - when to access channel?
  - how to detect collisions?
  - how to recover from collisions?

# Slotted Aloha [Norm Abramson]

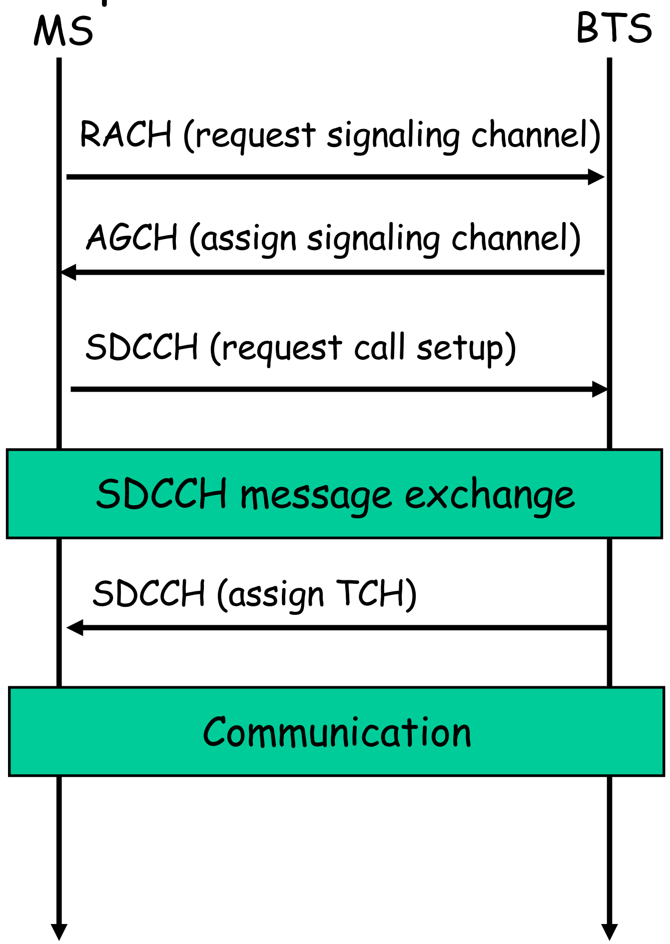
- Time is divided into equal size slots (= pkt trans. time)
- Node with new arriving pkt: transmit at beginning of next slot
- If collision: retransmit pkt in future slots with probability p, until successful.




Success (S), Collision (C), Empty (E) slots

# Slotted Aloha in Real Life

- call setup in GSM




- Notations:
  - Broadcast control channel (BCCH): from base station, announces cell identifier, synchronization
  - Random access channel (RACH): MSs for initial access, **slotted Aloha**
  - access grant channel (AGCH): BTS informs an MS its allocation
  - standalone dedicated control channel (SDCCH): signaling and short message between MS and an MS
  - Traffic channels (TCH)

# Slotted Aloha Efficiency

**Q:** What is the fraction of successful slots?

suppose n stations have packets to send

suppose each transmits in a slot with probability *p*

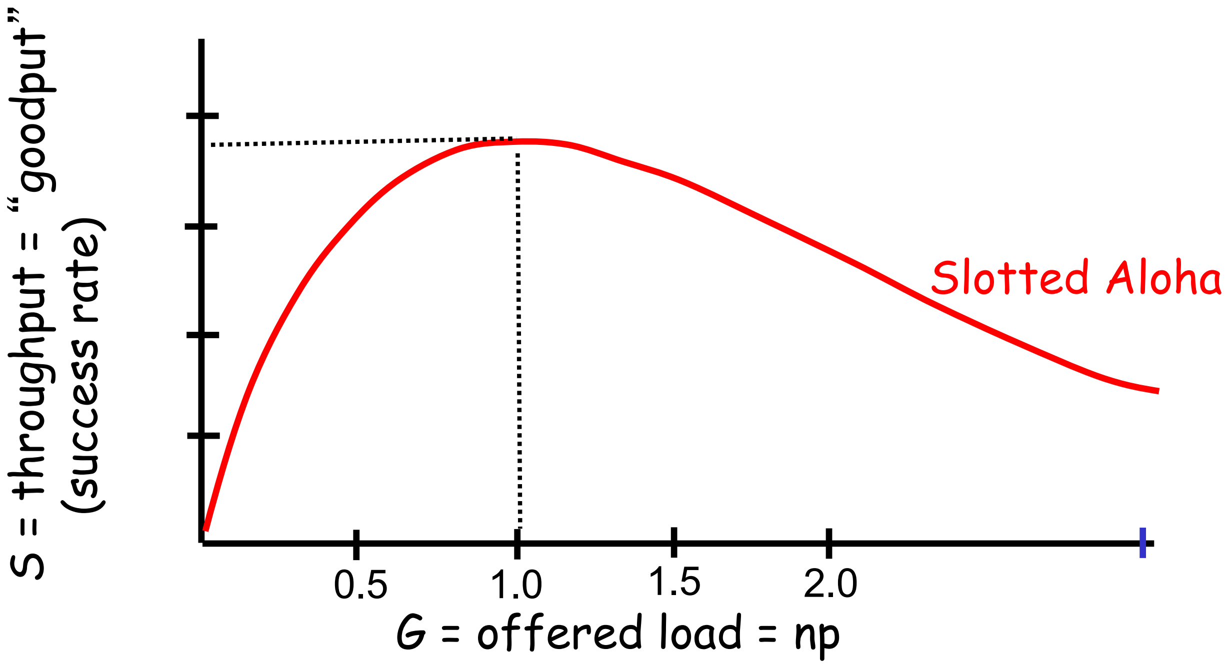
- prob. of succ. by a specific node: $p\,(1-p)^{(n-1)}$

- prob. of succ. by any one of the N nodes

  S(p) = n * Prob (only one transmits)

  $= n\,p\,(1-p)^{(n-1)}$

# Goodput vs. Offered Load for Slotted Aloha

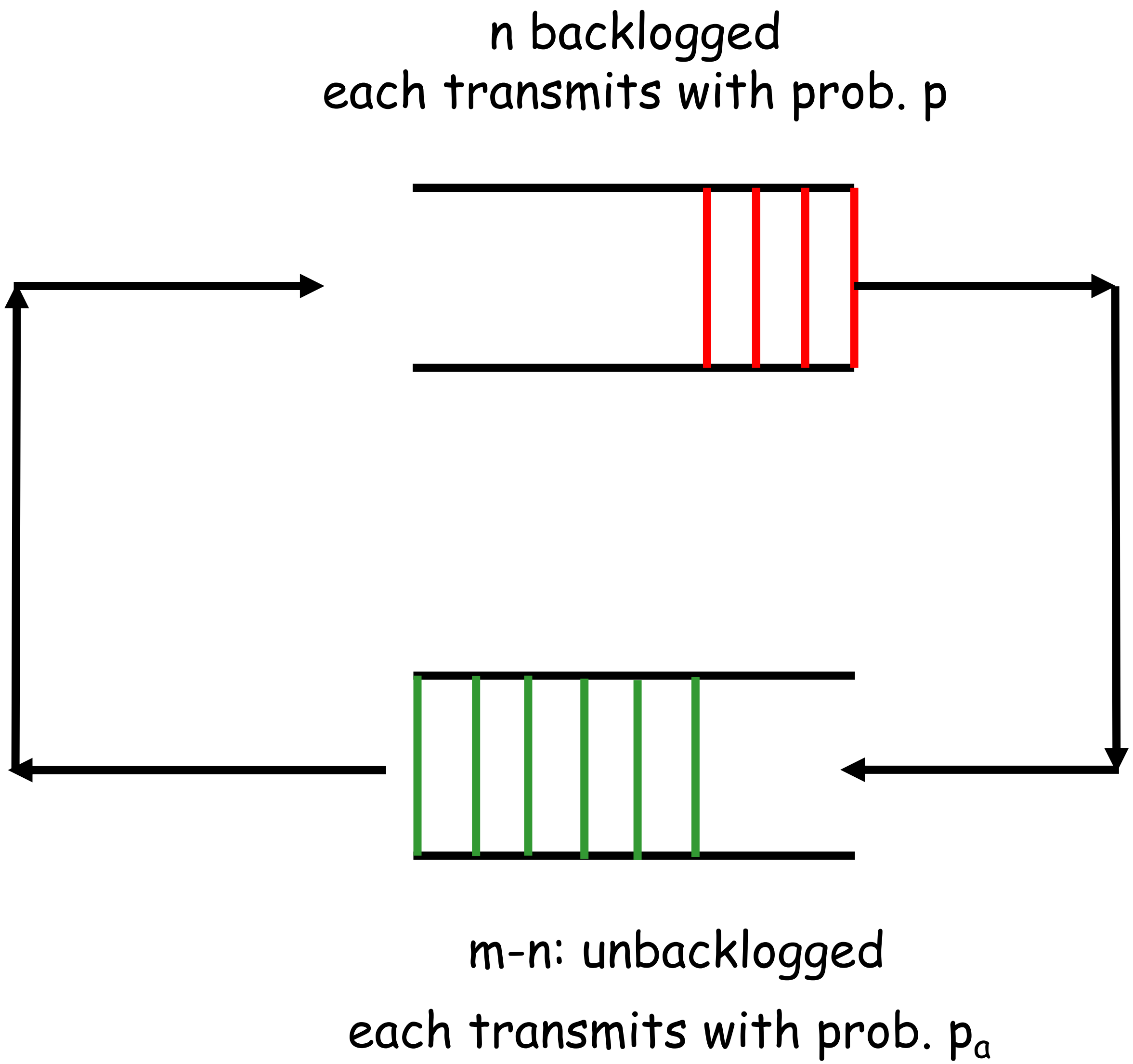



- when p n < 1, as p (or n) increases
  - probability of empty slots reduces
  - probability of collision is still low, thus goodput increases
- when p n > 1, as p (or n) increases,
  - probability of empty slots does not reduce much, but
  - probability of collision increases, thus goodput decreases
- goodput is optimal when p n = 1, n -> infinite, S -> 1/e (~37%)

# Dynamics of (Slotted) Aloha

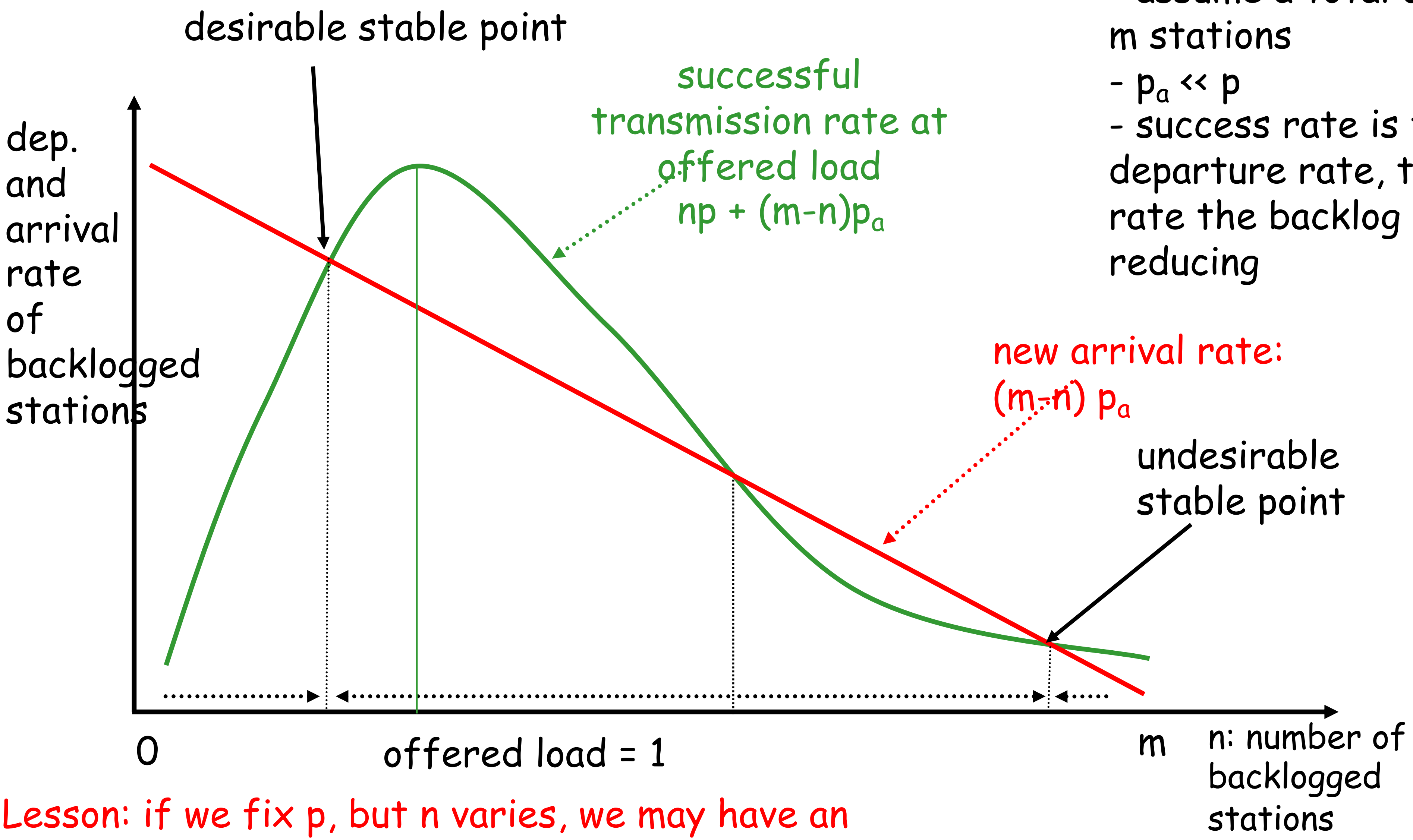
- Slotted Aloha has maximum throughput when np = 1
  - Implies we need to adjust p as the number of backlog stations varies.
- Early design question: what is the effect if we do not change p--use a fixed p
  - Assume we have a total of m stations (the machines on a LAN):
    - n of them are currently backlogged, each tries with a (fixed) probability p
    - the remaining m-n stations are not backlogged. They may start to generate packets with a probability $p_a$, where $p_a$ is much smaller than p

# Model



n backlogged
each transmits with prob. p
m-n: unbacklogged
each transmits with prob. $p_a$

# Dynamics of Aloha: Effects of Fixed Probability




- assume a total of m stations
- $p_a << p$
- success rate is the departure rate, the rate the backlog is reducing

Lesson: if we fix p, but n varies, we may have an undesirable stable point

# Summary of Problems of Aloha Protocols

- Problems
  - slotted Aloha has better efficiency than pure Aloha but clock synchronization is hard to achieve
  - Aloha protocols have low efficiency due to collision or empty slots
    - when offered load is optimal (p = 1/N), the goodput is only about 37%
    - when the offered load is not optimal, the goodput is even lower
  - undesirable steady state at a fixed transmission rate, when the number of backlogged stations varies

- Ethernet design: address the problems:
  - optimal transmission rate

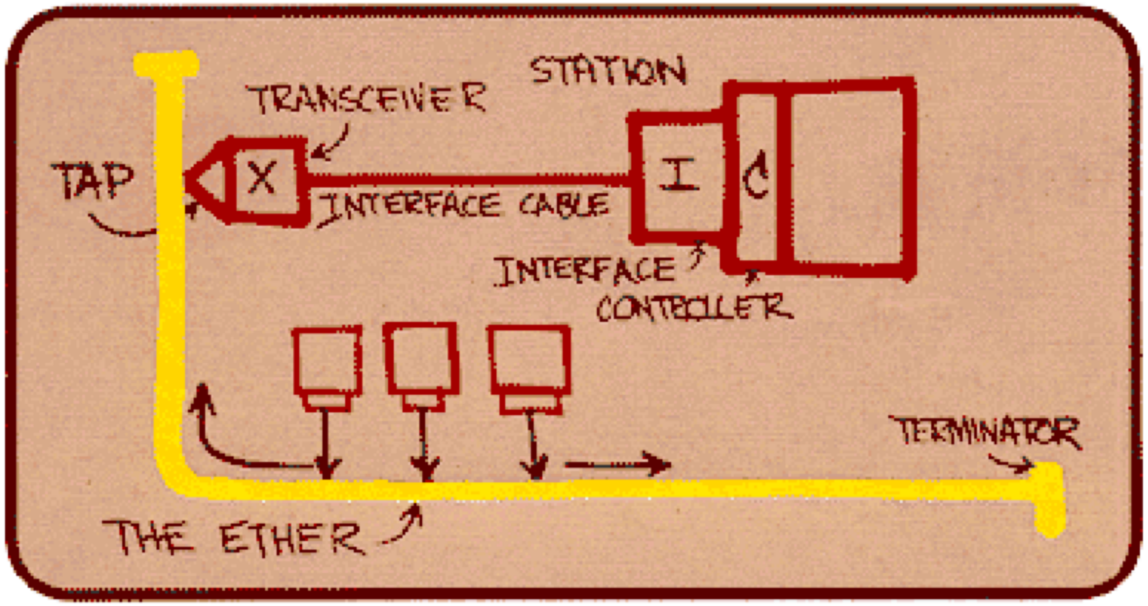
# The Basic MAC Mechanisms of Ethernet

```
get a packet from upper layer;
K := 0; n := 0; // K: control wait time; n: no. of collisions
repeat:
  wait for K * 512 bit-time;
  while (network busy) wait;
  wait for 96 bit-time after detecting no signal;
  transmit and detect collision;
  if detect collision
     stop and transmit a 48-bit jam signal;
     n ++;
     m:= min(n, 10), where n is the number of collisions
     choose K randomly from {0, 1, 2, …, 2^m-1}.
     if n < 16 goto repeat
     else give up
```

# Ethernet


"Dominant" LAN technology:

- First widely used LAN technology
- Kept up with speed race: 10 Mbps, 100 Mbps, 1 Gbps, 10 Gbps




Metcalfe's Ethernet sketch

# Outline

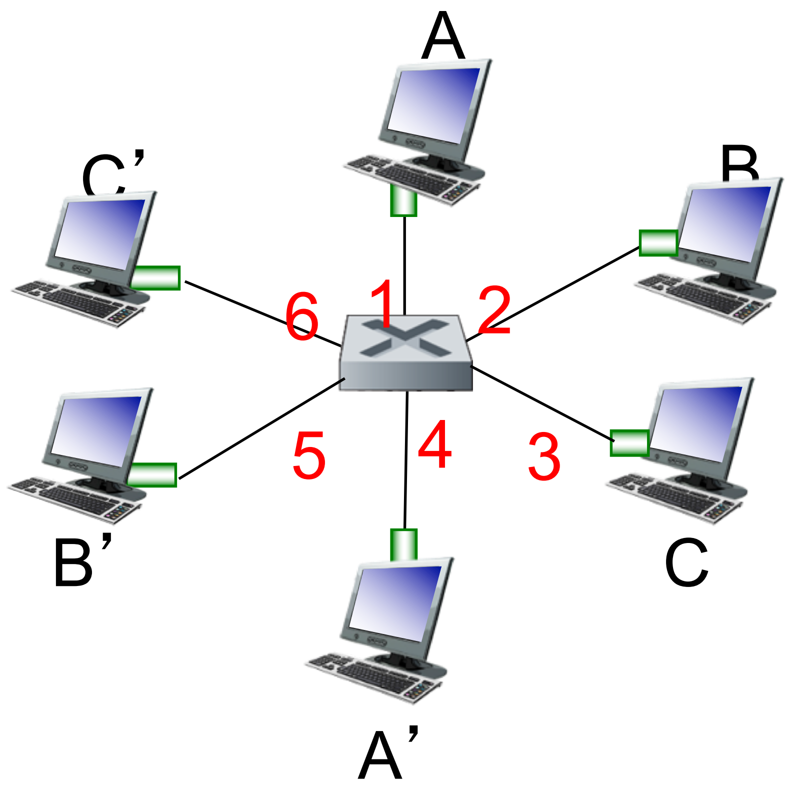
- Admin and recap
- Link layer
  - Ethernet switch

# 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



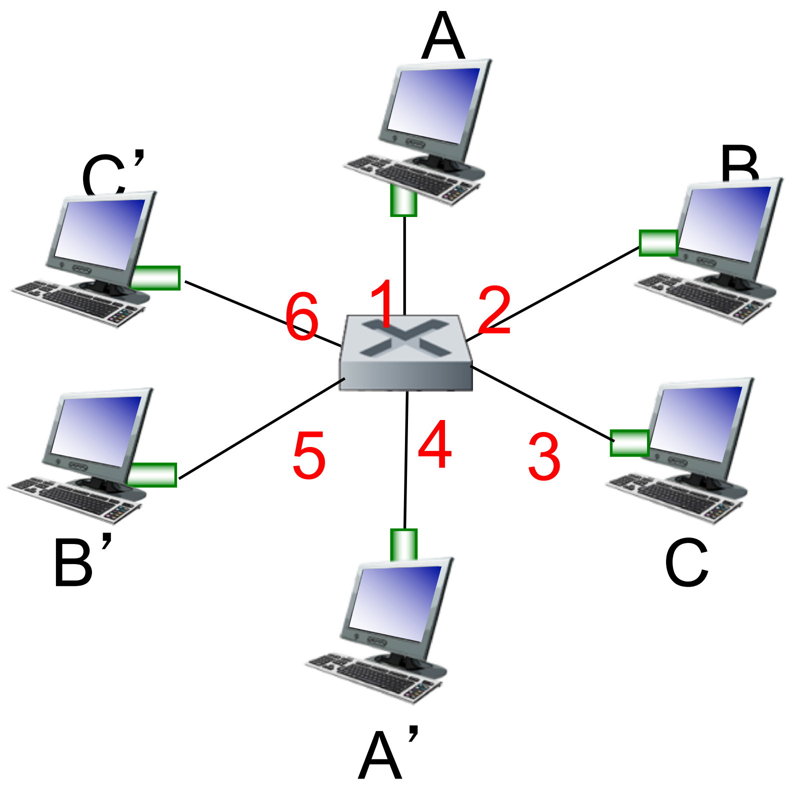

*switch with six interfaces (1,2,3,4,5,6)*

# 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!

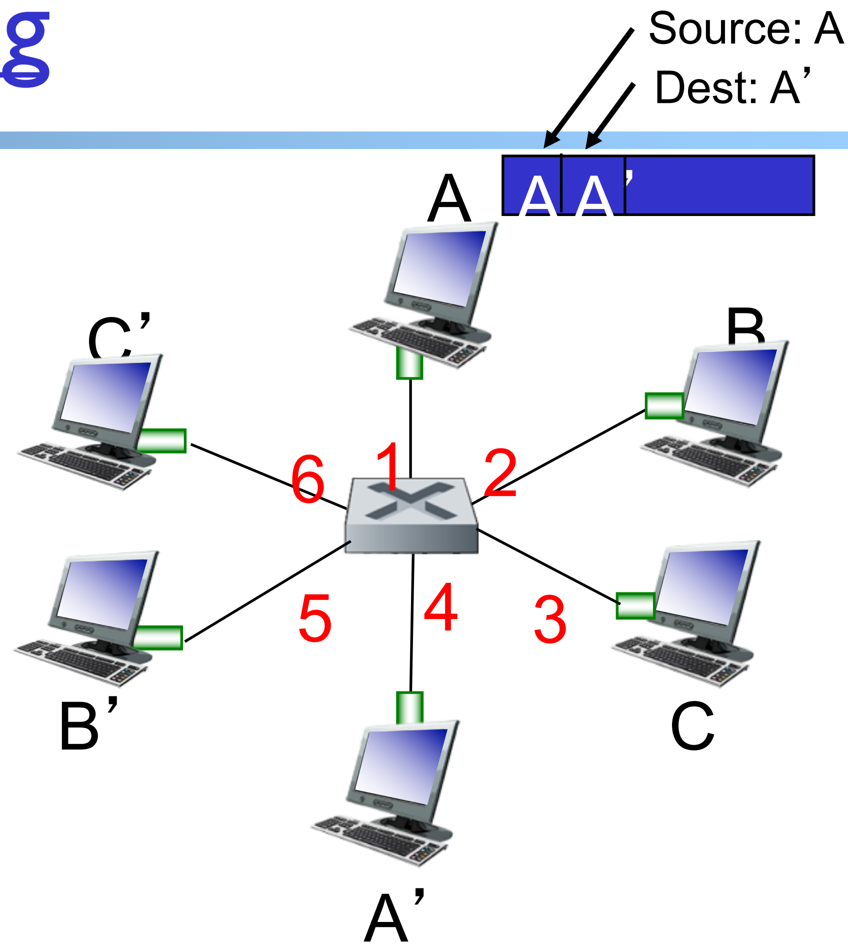
*Q:* how are entries created, maintained in switch table?
  - something like a routing protocol?




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

# Switch: 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)*

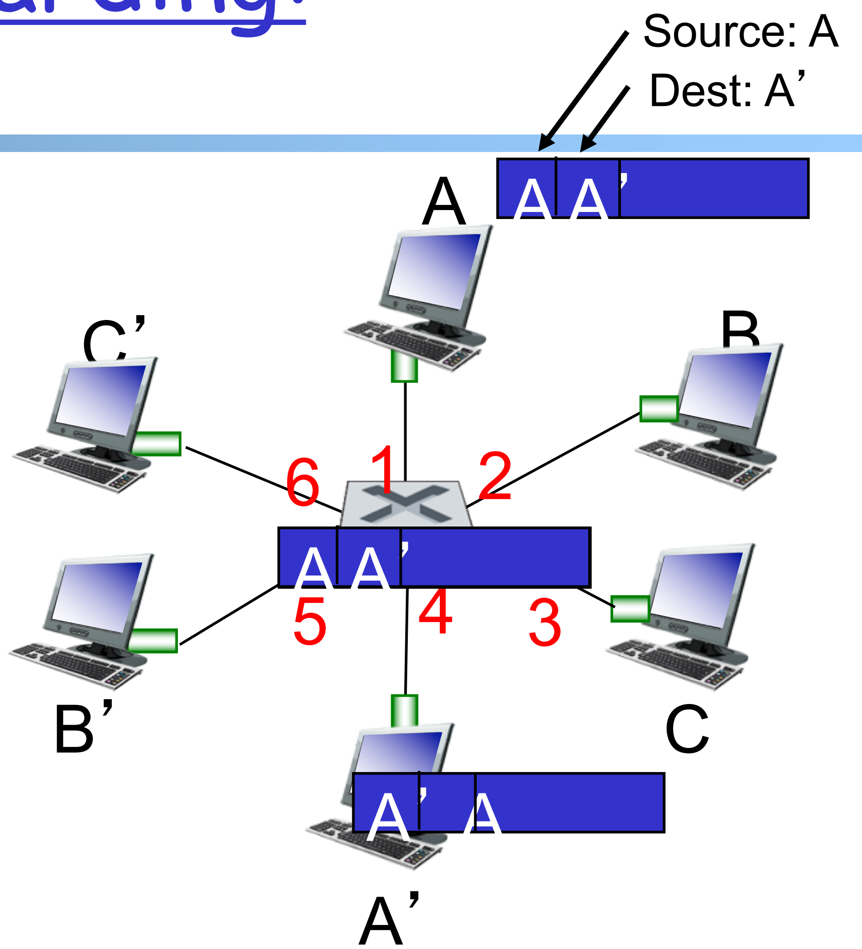
# Switch: Frame Filtering /Forwarding

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

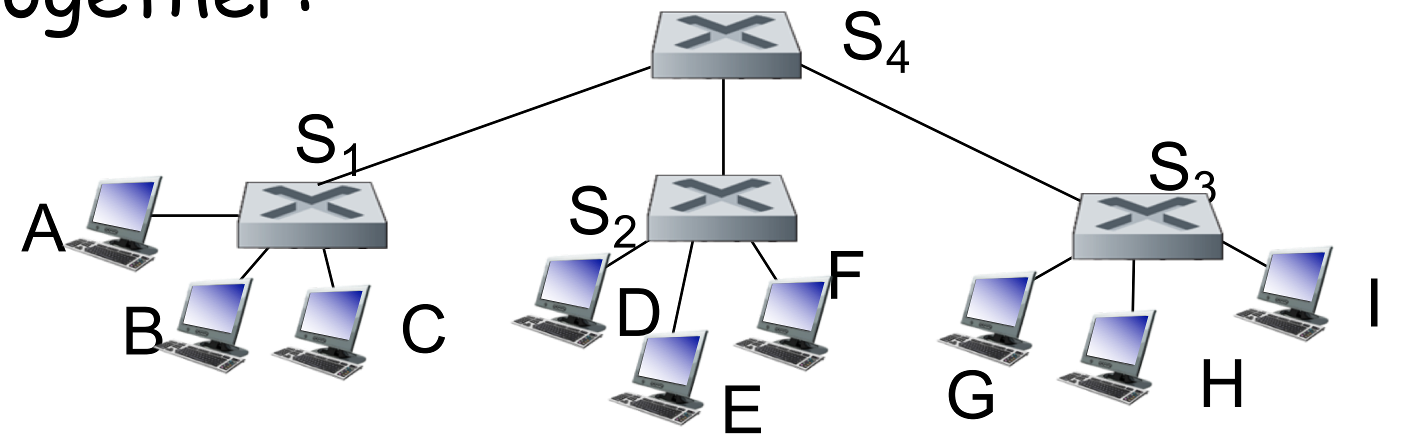



| MAC addr | interface | TTL |
|---|---|---|
| *A* | *1* | *60* |
| *A'* | *4* | *60* |

*switch table (initially empty)*

# Interconnecting Switches

self-learning switches can be connected together:

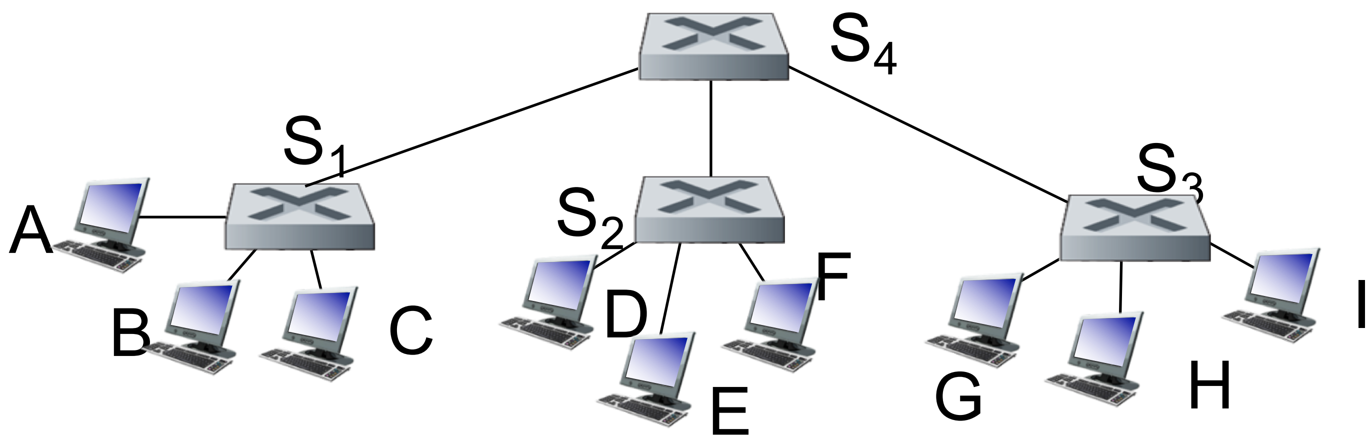



*Q:* sending from A to G - how does $S_1$ know to forward frame destined to G via $S_4$ and $S_3$?

- *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




- Offline Exercise: show switch tables and packet forwarding in $S_1$, $S_2$, $S_3$, $S_4$