Week 13-Lec 1

Data Link layer Intro

Chapter 5: Link layer

our goals:

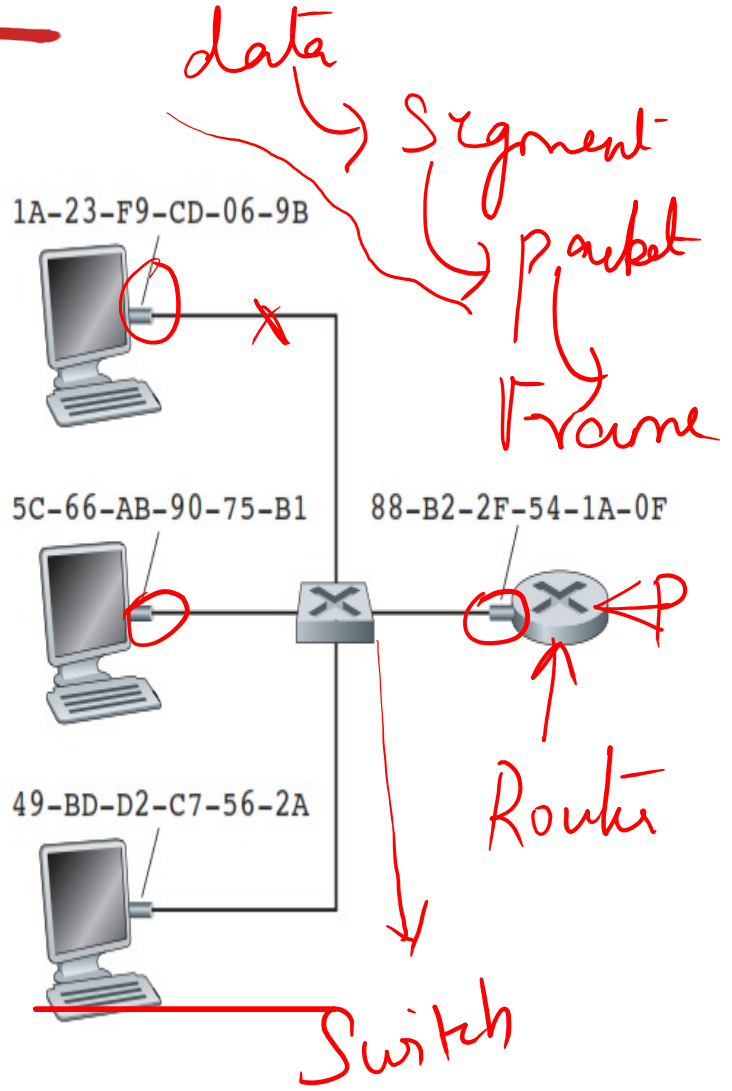
- understand principles behind link layer services:
- What happens when a packet gets inside a network/subnet?
  - Which protocol moves the packet to the final destination machine/host? ETHERNET
  - How is the final destination/host identified?

Link layer: introduction

terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
  - LANs
- layer-2 packet: frame, encapsulates datagram

data-link layer has responsibility of transferring datagram from one node to physically adjacent node over a link



Link layer services

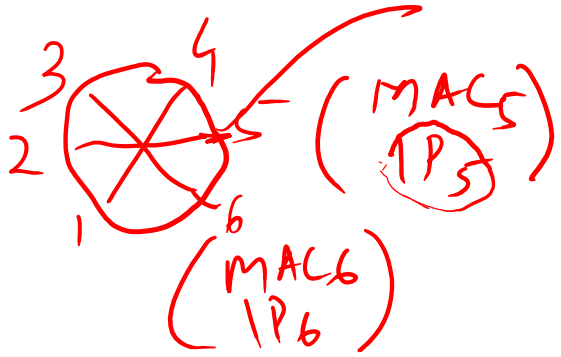
- *reliable delivery between adjacent nodes*

- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
  - channel access if shared medium

MAC addresses

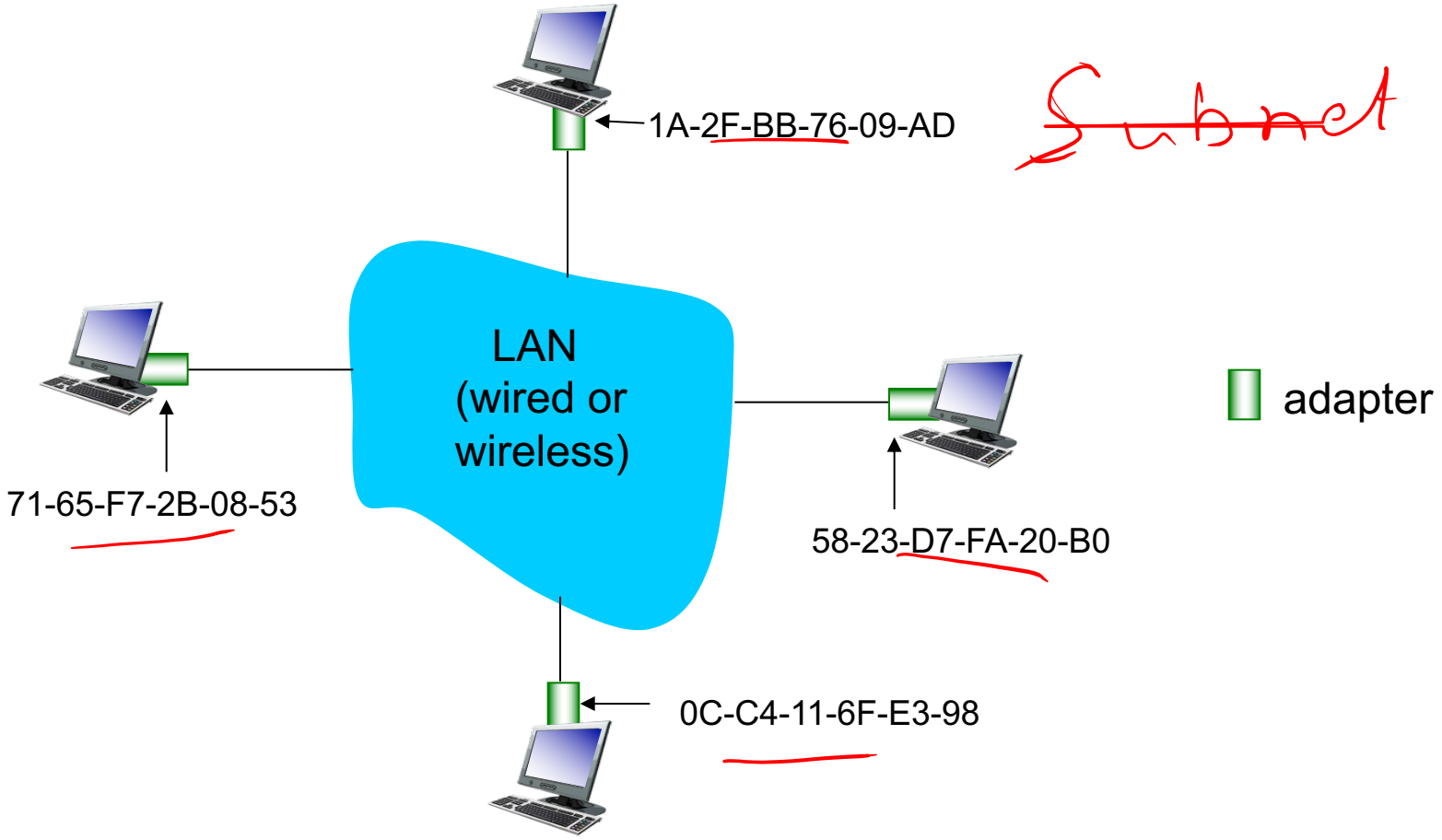
- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
  - function: *used 'locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 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 "number" represents 4 bits)



LAN addresses

each adapter on LAN has unique LAN address



LAN addresses (more)

❖MAC address allocation administered by IEEE

❖manufacturer buys portion of MAC address space (to assure uniqueness)

- each manufacturer of Ethernet devices is allocated a different prefix that must be prepended to the address on every adaptor they build. For example, Advanced Micro Devices has been assigned the 24-bit prefix x080020 (or 8:0:20). A given manufacturer then makes sure the address suffixes it produces are unique.

❖analogy:

- MAC address: like Aadhar number
- IP address: like postal address

❖ MAC flat address ➜ portability

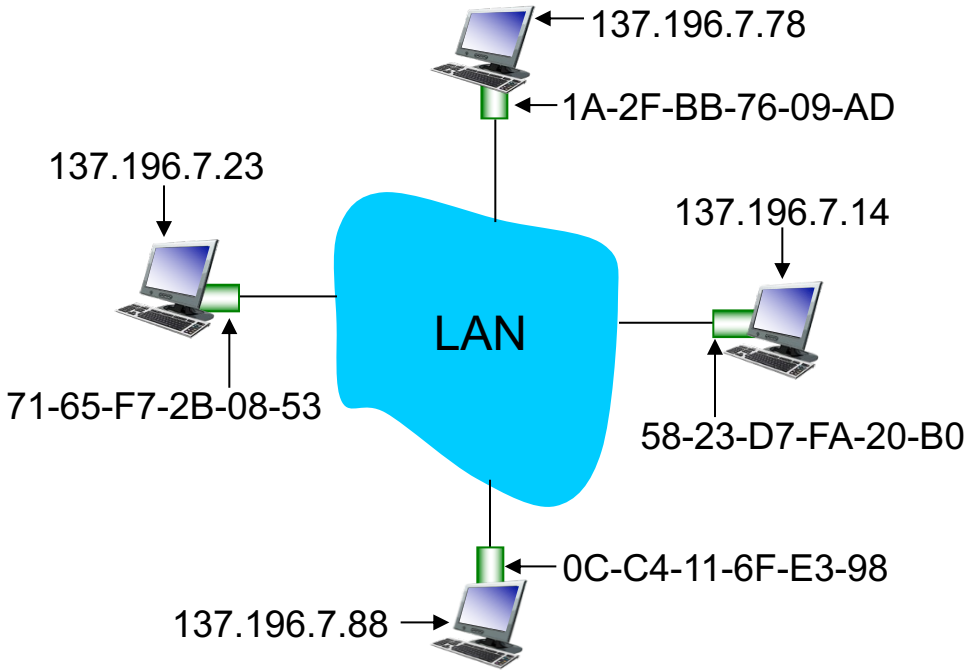
- can move LAN card from one LAN to another

❖IP hierarchical address *not* portable

- address depends on IP subnet to which node is attached

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

ARP: address resolution protocol



Maps IP → MAC

ARP table: each IP node (host, router) on LAN has 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)

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
  - dest 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)

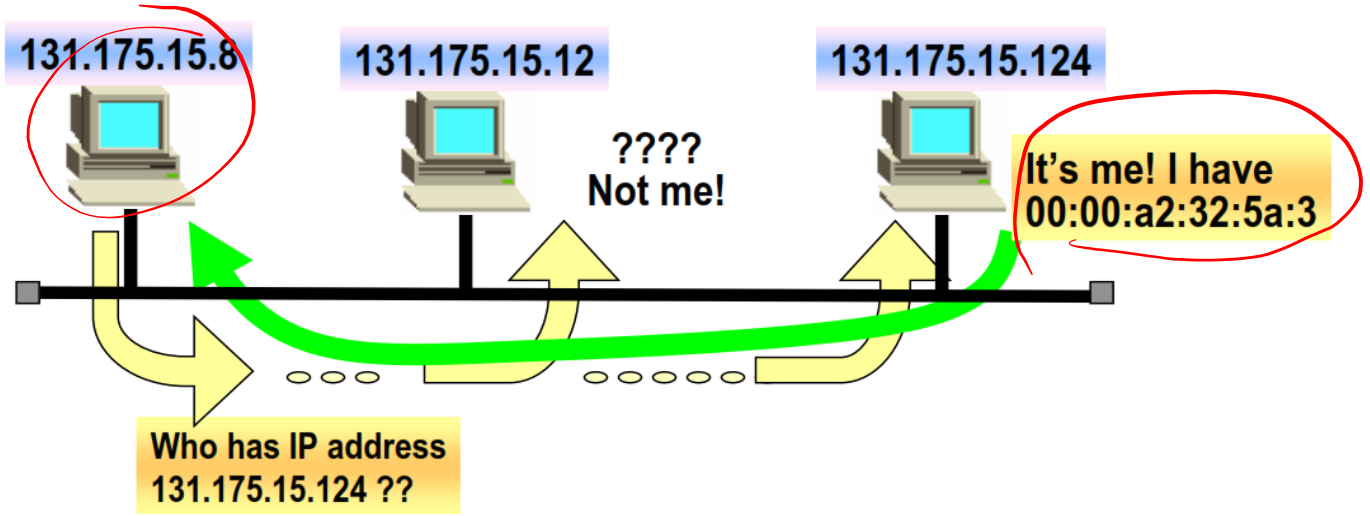
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Example of ARP table

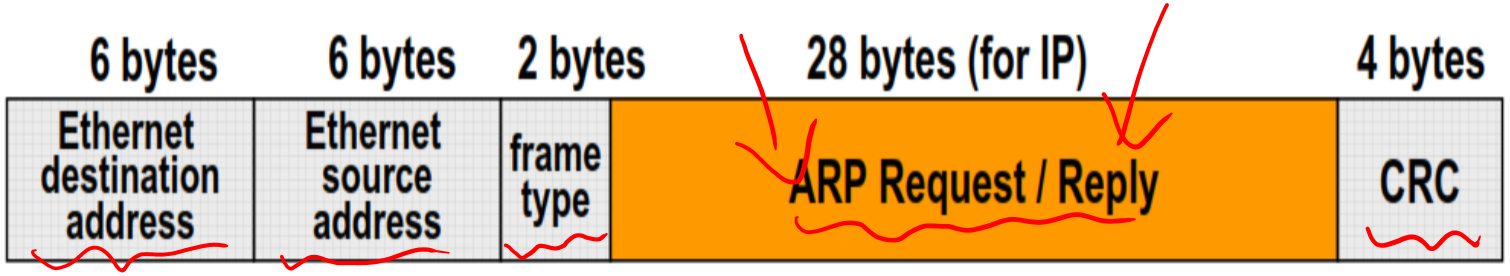
IP address	MAC Address	TTL
137.196.7.78	1A-2F-BB-76-09-AD	13:45:00
137.196.7.14	58-23-D7-FA-20-B0	13:52:00

ARP Idea

ARP provides a dynamic mapping from an IP address to the corresponding hardware address.



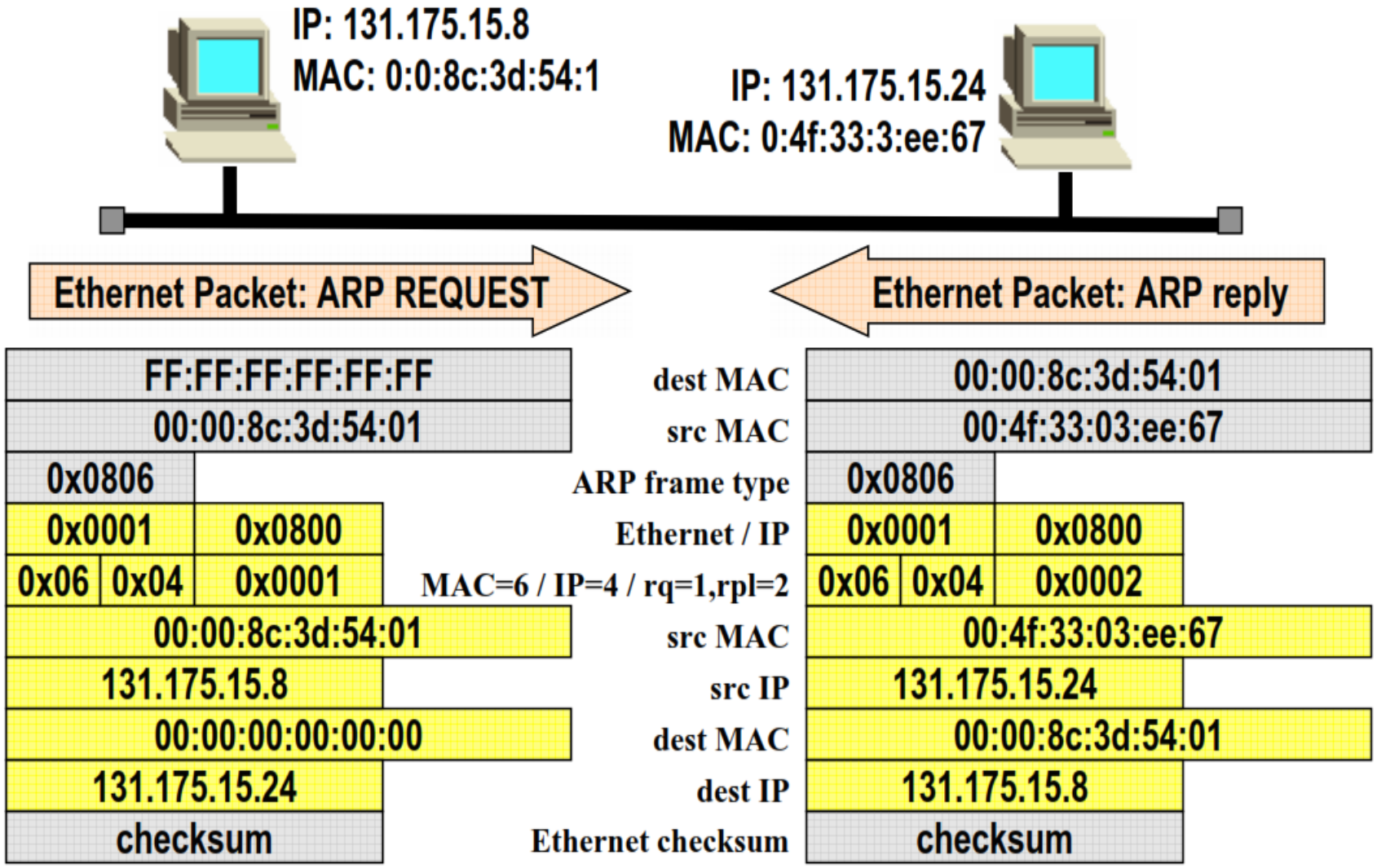
ARP packet encapsulation in an Ethernet Frame



ARP request/reply
Encapsulation in Ethernet Frame

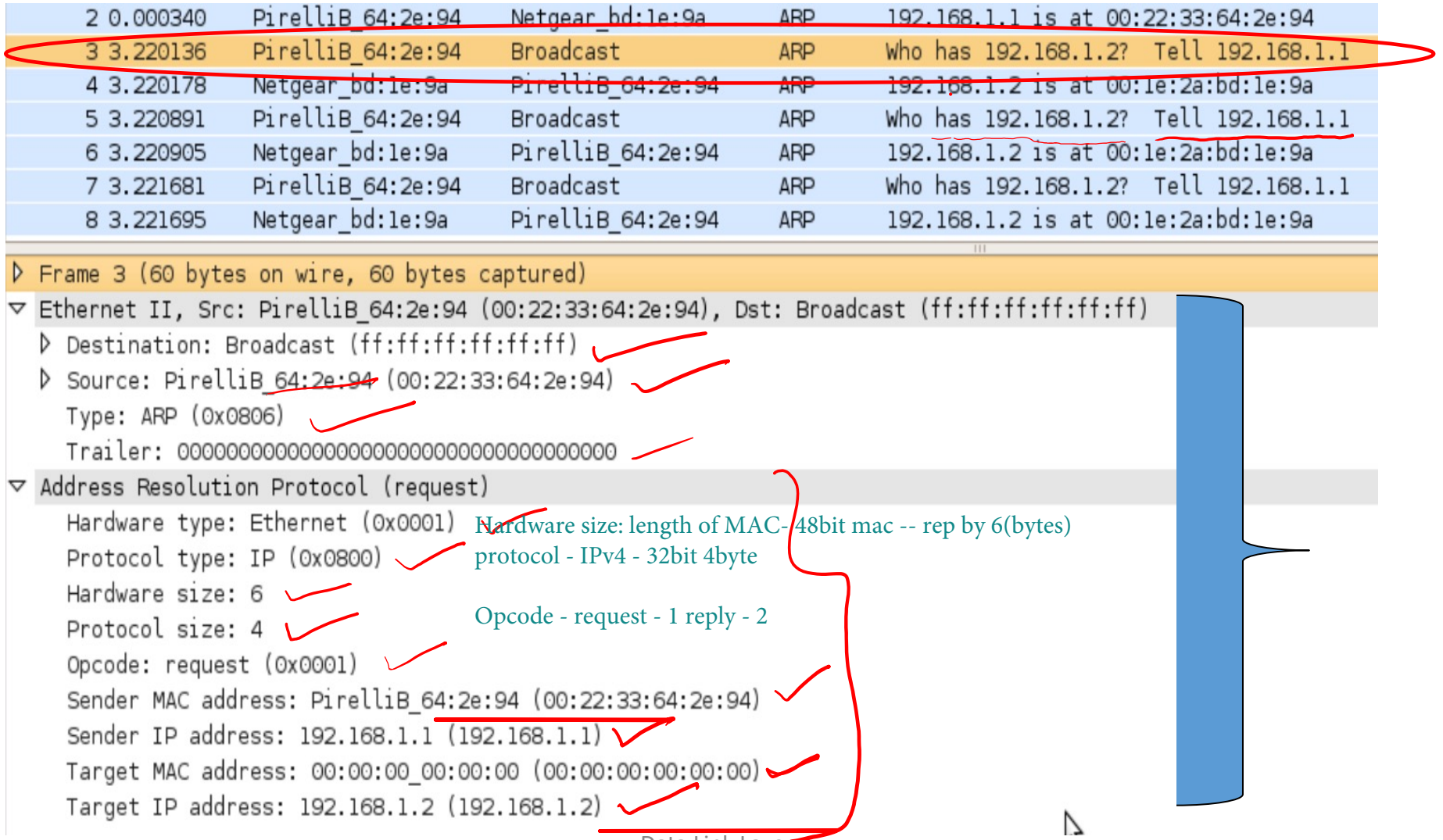
- **Ethernet Destination Address**
  - ff:ff:ff:ff:ff:ff (broadcast) for ARP request
- **Ethernet Source Address**
  - of ARP requester
- **Frame Type**
  - ARP request/reply: 0x0806
  - RARP request/reply: 0x8035
  - IP datagram: 0x0800

Protocol demultiplexing codes!



ARP request is BROADCAST
ARP Response is UNICAST

Wireshark trace for ARP packet



Revision

- ARP stands for
  - Address resolution protocol
  - Automatic resolution protocol
  - Automatic request protocol

- ARP is used for
  - Mapping IP – MAC

- ARP works
  - Within a subnet
  - Outside a subnet also
  - Both of them

Week 13-Lec 2

Ethernet

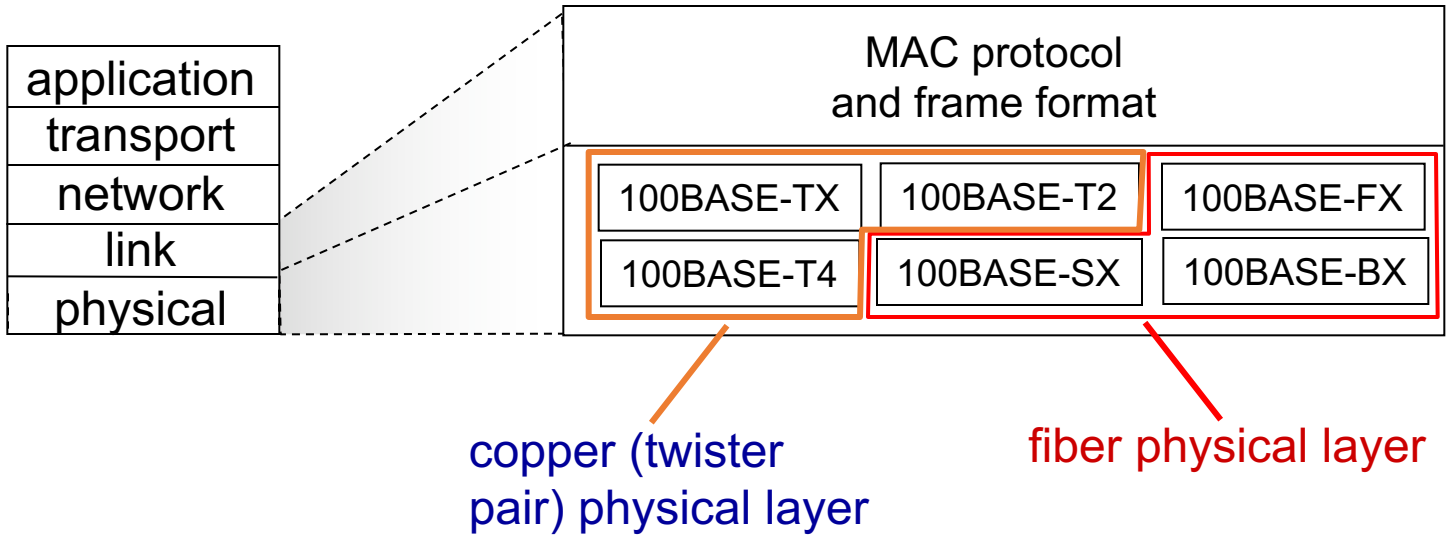
Ethernet

"dominant" wired LAN technology:

- first widely used LAN technology
- kept up with speed race: 10 Mbps – 10 Gbps
- Invented by Metcalfe

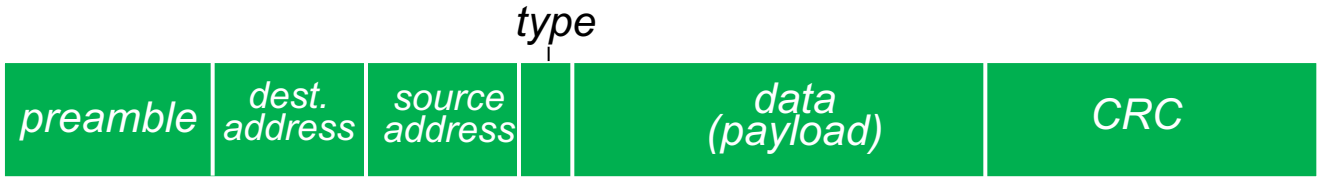
802.3 Ethernet standards: link & physical layers

- *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 frame structure

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

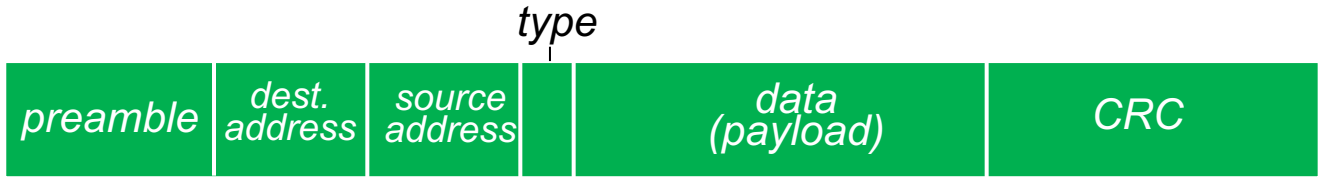


preamble:

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

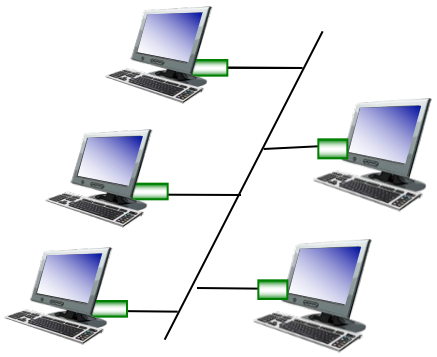
Ethernet frame structure (more)

- *addresses:* 6 byte source, destination MAC addresses
  - if adapter 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)
- *CRC:* cyclic redundancy check at receiver
  - error detected: frame is dropped

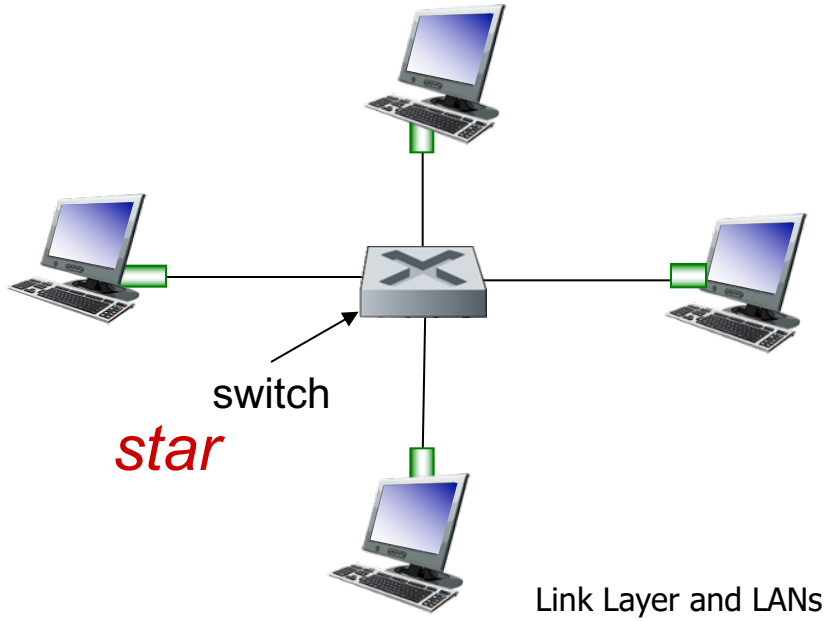


Classic Ethernet: physical topology

- *bus:* popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- *star:* prevails today
  - active *switch* in center
  - each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)



bus: coaxial cable



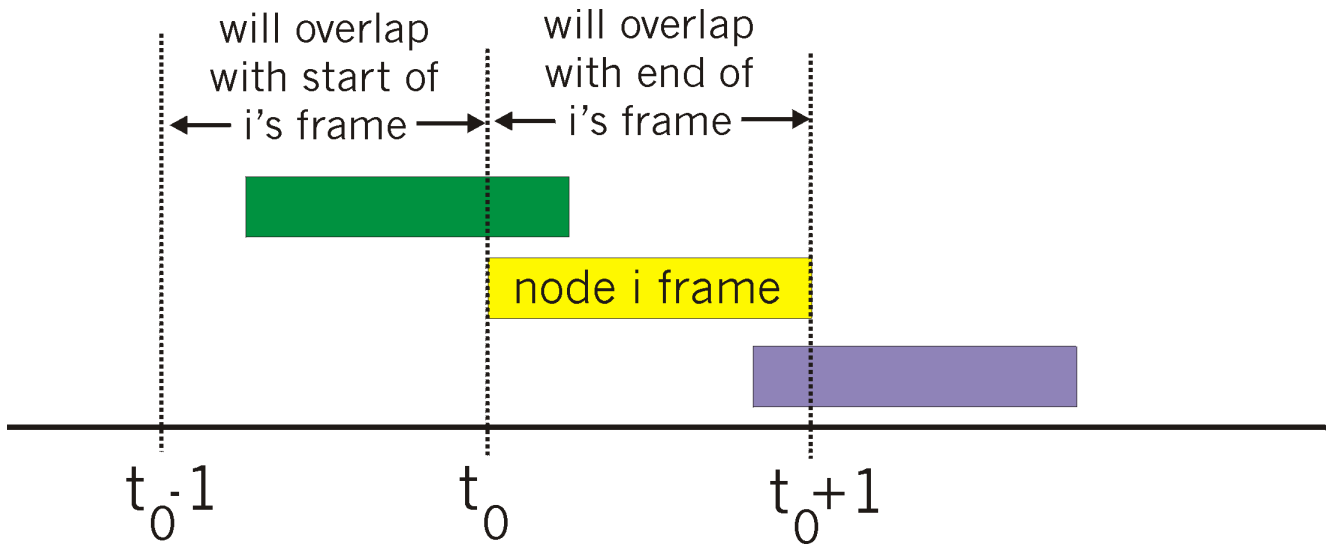
star

Random access protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "collision",
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - ALOHA
  - slotted ALOHA
  - CSMA, CSMA/CD,

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$



Slotted ALOHA

assumptions:

- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

operation:

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with prob. p until success

Pure ALOHA efficiency

P(success by given node) = P(node transmits) ·

P(no other node transmits in $[t_0\text{-}1, t_0]$ ·

P(no other node transmits during $[t_0, t_0, +1]$

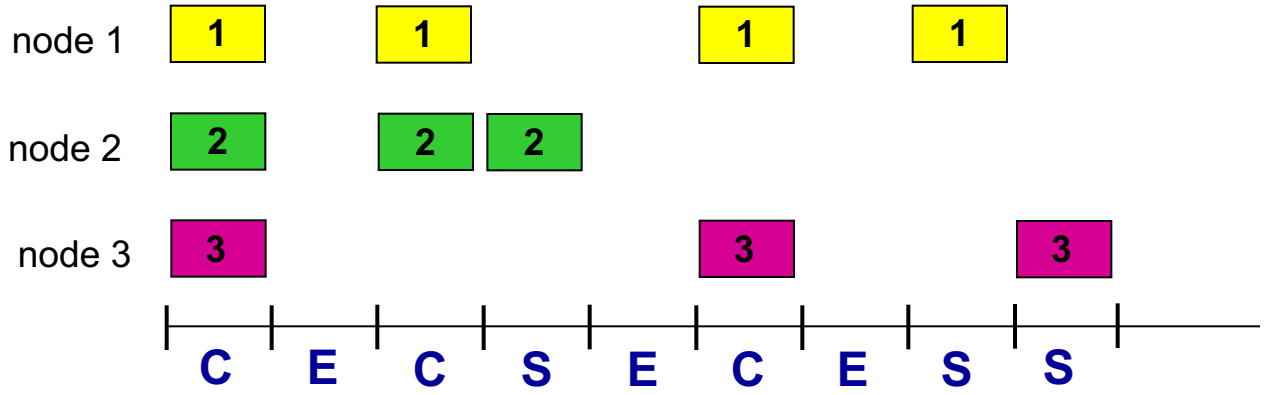
$= p \cdot (1\text{-}p)^{N-1} \cdot (1\text{-}p)^{N-1}$

$= p \cdot (1\text{-}p)^{2(N-1)}$ $\longrightarrow \infty$

… choosing optimum p and then letting *n*

$= 1/(2e) = .18$

Slotted ALOHA



Pros:

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Slotted ALOHA: efficiency

efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

- *suppose:* N nodes with many frames to send, each transmits in slot with probability p
- prob that given node has success in a slot $= p(1-p)^{N-1}$
- prob that *any* node has a success $= Np(1-p)^{N-1}$

- max efficiency: find p^* that maximizes $Np(1-p)^{N-1}$
- for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

max efficiency $= 1/e = .37$

at best: channel used for useful transmissions 37% of time!

!

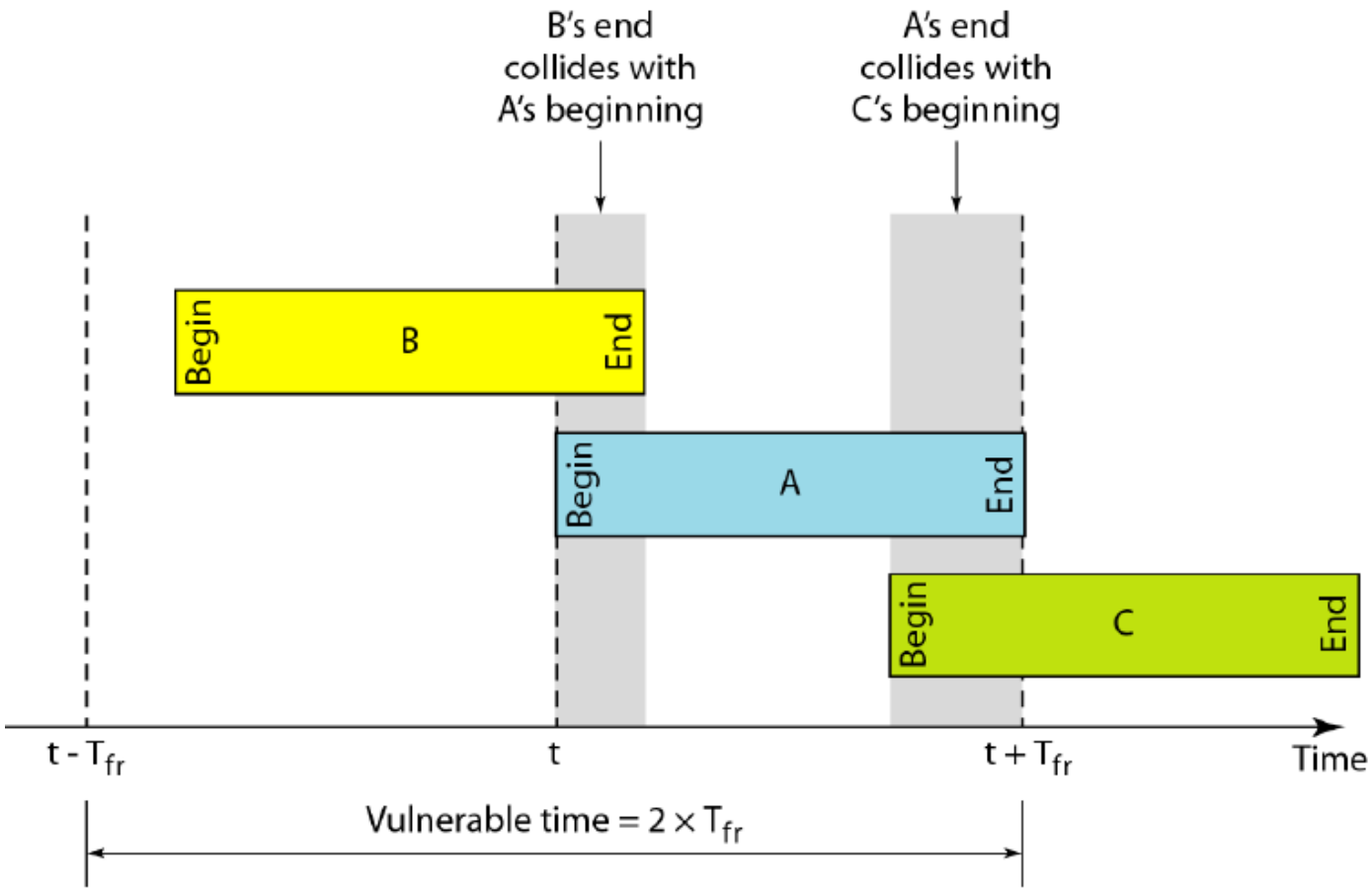
CSMA (carrier sense multiple access)

CSMA: listen before transmit:

if channel sensed idle: transmit entire frame

- if channel sensed busy, defer transmission

- human analogy: don't interrupt others!

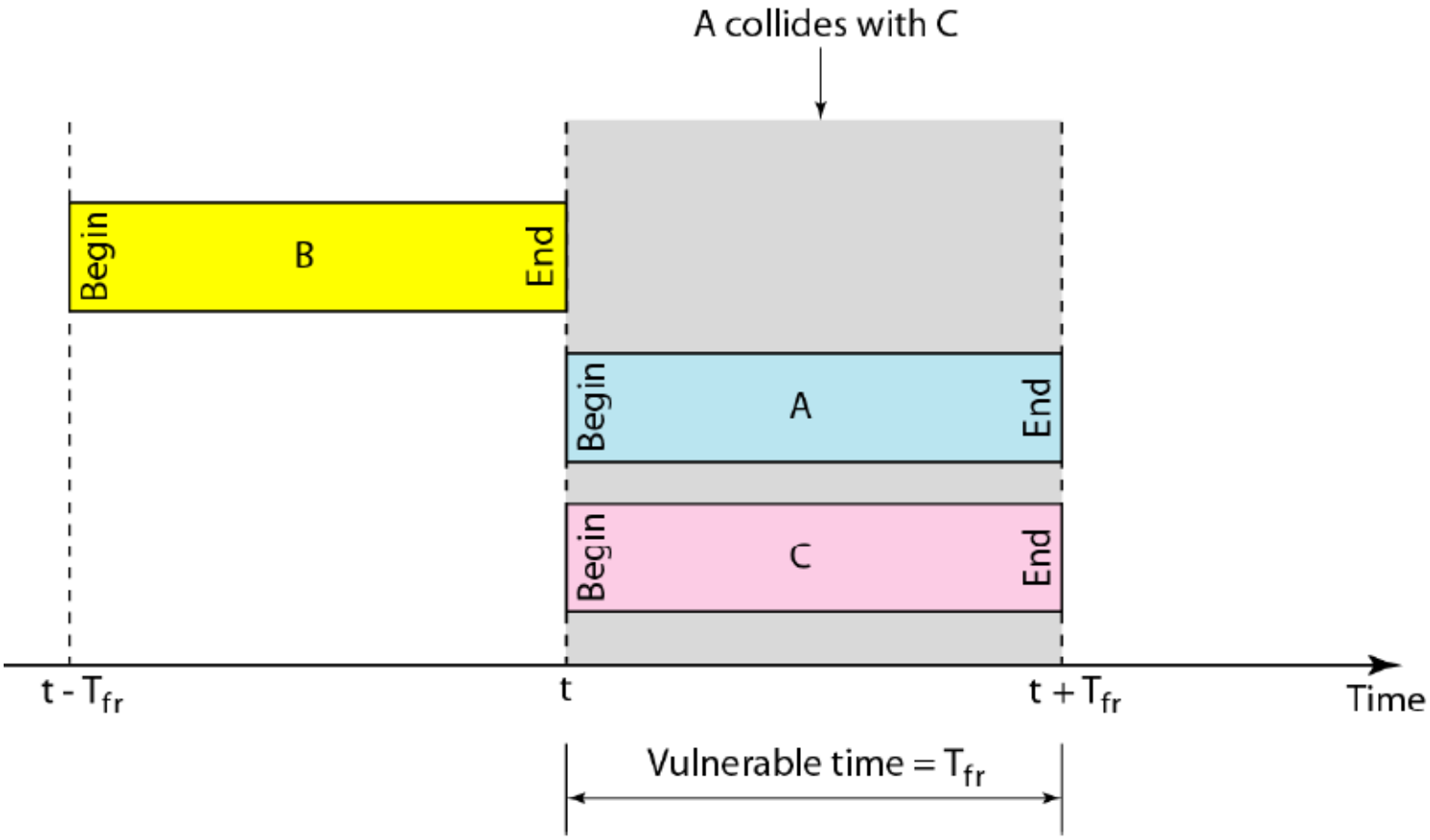

B's end
collides with
A's beginning
A's end
collides with
C's beginning
Begin
B
End
Begin
A
End
Begin
C
End
t - Tfr
t
t + Tfr
Time
Vulnerable time = 2 × Tfr

- A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision-free?
- Solution
- Average frame transmission time

T_{fr} is 200 bits/200 kbps = 1 ms.

The vulnerable time is 2 × 1 ms = 2 ms. This means

- no station should send later than 1 ms before this station starts transmission
- no station should start sending during the one 1ms period that this station is sending.


A collides with C
Begin
B
End
Begin
A
End
Begin
C
End
t - Tfr
t
t + Tfr
Time
Vulnerable time = Tfr

CSMA (carrier sense multiple access)

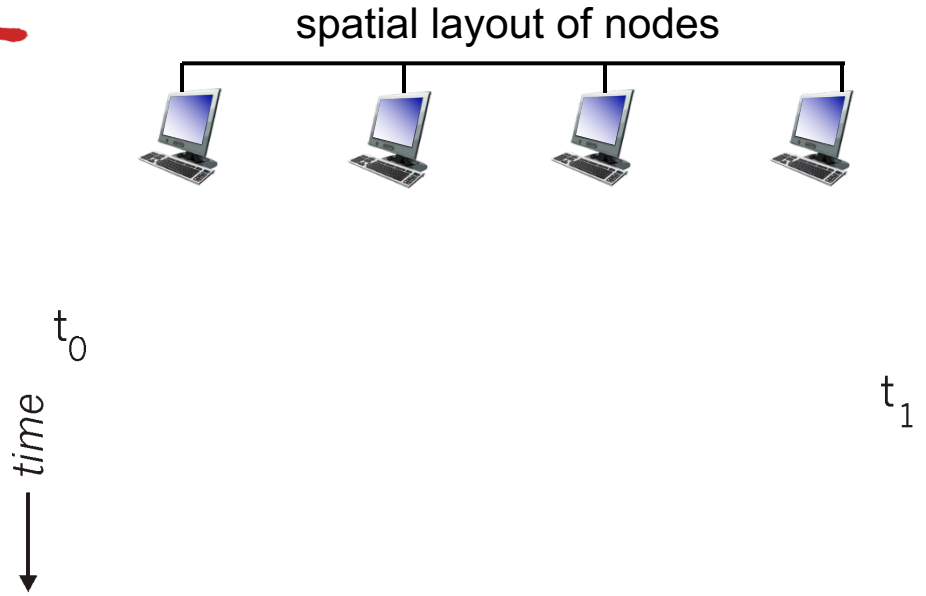
CSMA: listen before transmit:

if channel sensed idle: transmit entire frame

- if channel sensed busy, defer transmission

CSMA collisions

- collisions *can* still occur: propagation delay means two nodes may not hear each other's transmission
- collision: entire packet transmission time wasted



- *Collision chance Depends on the number of frames/* ***bandwidth-delay product***
- ***If #****frames in channel less, the chance of a collision happening is small.*
- *The larger the bandwidth-delay product, more collisions, performance of protocol worse*

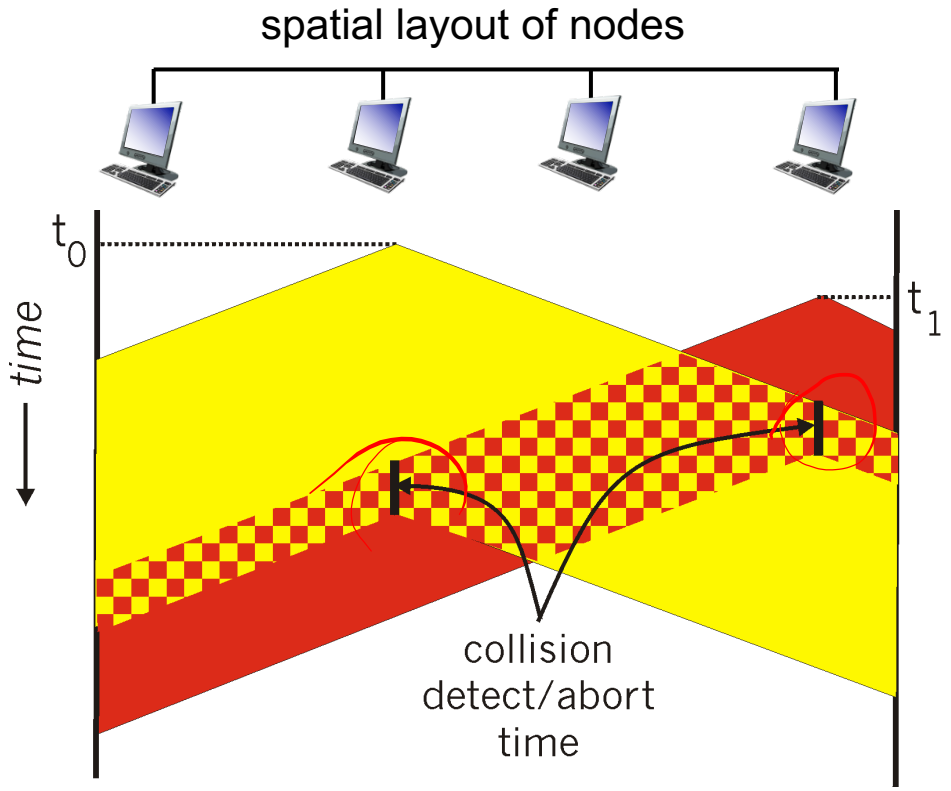
CSMA/CD (collision detection)

CSMA/CD: carrier sensing,

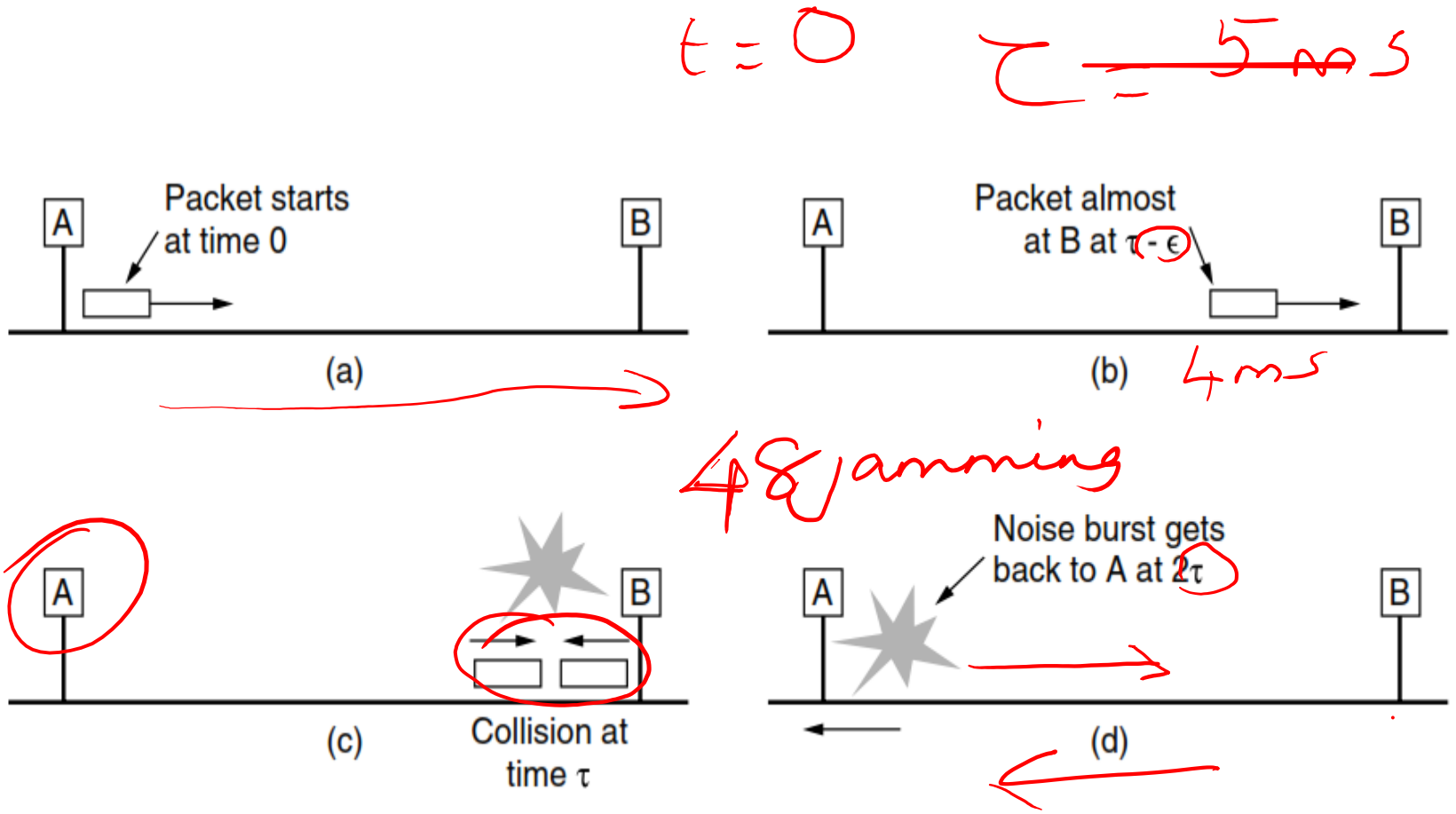
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals

CSMA/CD (collision detection)



Time for collision detection



Successful transmission if Tx done for atleast 2 τ time units

How is wait/back off time chosen?

Need an interval that is short when the number of colliding nodes is small,

Long when the number of colliding nodes is large.

How is wait/back off time chosen?

- Suppose that a node attempts to transmit a frame for the first time
- Detects a collision. i.e., num of collision is 1. so $2^0 = 1$. therefore possible values of K ={0,1}
  - The node then chooses *K=0 or K=1 with probability 0.5.*
  - *If K=0, then it immediately begins sensing the channel.*
  - *If the node chooses K 1, it waits K.* 512 bit times i.e,.(1)* 512 (e.g., 0.01 microseconds for a 100 Mbps Ethernet) before beginning the sense-and-transmit-when-idle cycle.
- Similarly, after a second collision, *K is chosen with* 2^2 equal probability from {0,1,2,3}.

- After three collisions, K is chosen with equal probability 2^3 from {0,1,2,3,4,5,6,7}.
- After 10 or more collisions, K is chosen with equal probability from {0,1,2, . . . , 1023}.
- Thus, the size of the sets from which K is chosen grows exponentially with the number of collisions;
- Hence Binary exponential backoff

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If channel idle → starts Tx

   If channel busy → Wait
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC needs to wait for a random time

   enters *binary (exponential) backoff:*

   - after n collisions, NIC chooses *K* at random from $\{0,1,2, \ldots, 2^n-1\}$.
   - NIC waits K·512 bit times, returns to Step 2
   - longer backoff interval with more collisions

Why 512 bits?

- 10Mbps Classic Ethernet
- Min frame size cannot be zero. THINK!
- Worst case scenario, If distance long, i.e., 2500 m, 10 Mbps,
- Then 64 bytes (512 bits) of data to be sent
  - 64 bytes = 14 bytes (header) + 46 bytes(payload, could be all zeros) + 4 bytes(CRC)
- Wait time can also be calculated in terms of time (micro sescs) = 512 bits on 10 Mbps = 51.2microsecs
- 0, 51.2, 102.4, or 153.6 μs (k=0, 1, 2, 3)…and so on

Recall that with the CSMA/CD protocol, the adapter waits K·512 bit times after a collision, where K is drawn randomly. For K=100, how long does the adapter wait until returning to Step 2 for a 1Mbps Ethernet? For a 10 Mbps Ethernet?

Solution: For k = 100, the adapter waits for 51200 bit times.
For 1 Mbps, this wait is 51200 $bits / 10^6 bps = 51.2 msec$
For 10 Mbps, the adapter waits for 5.12m sec

Summary of MAC protocols

- *channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
- *random access* (dynamic),
  - CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing
  - bluetooth, FDDI, token ring

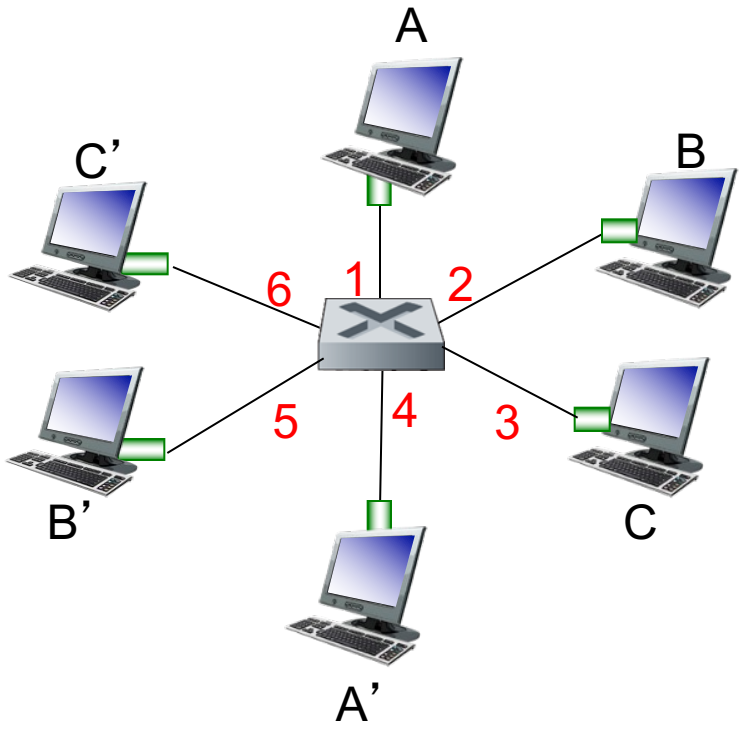
Week14-lec2

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)

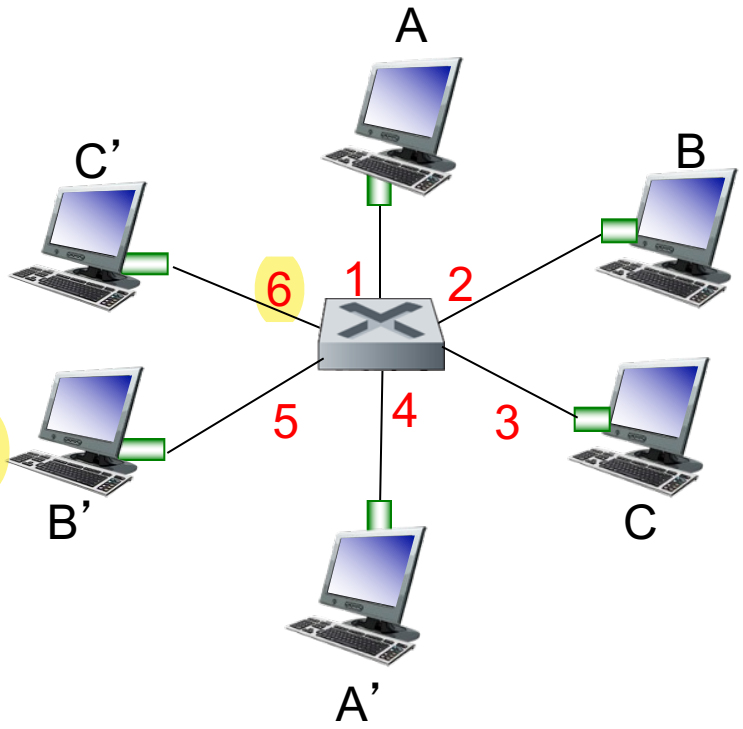
Switch forwarding table

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?

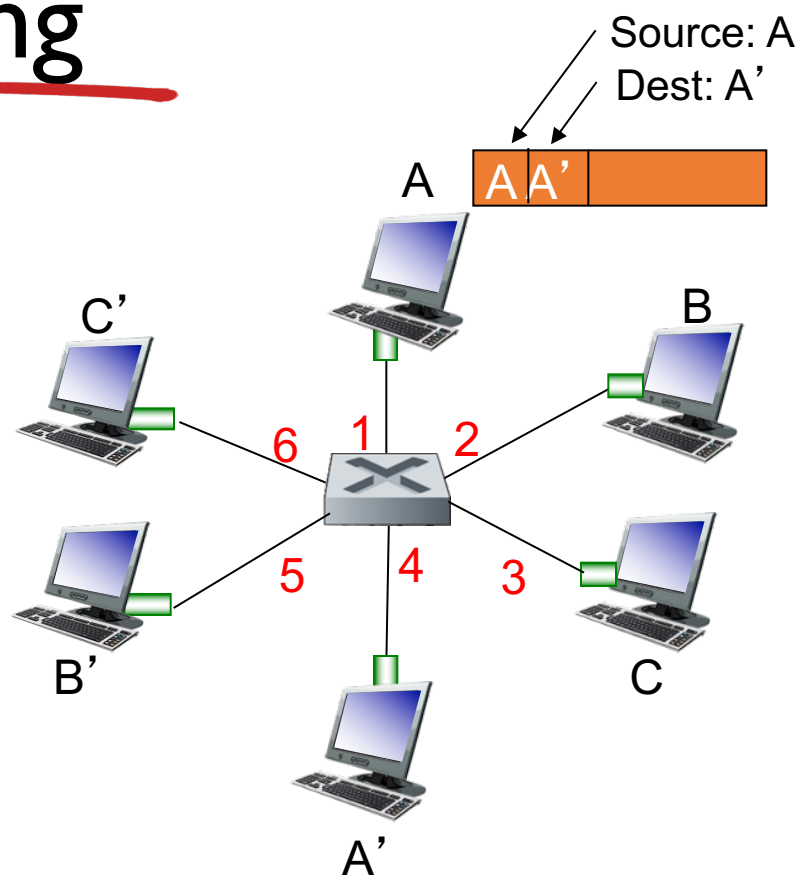
- something like a routing protocol?



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)

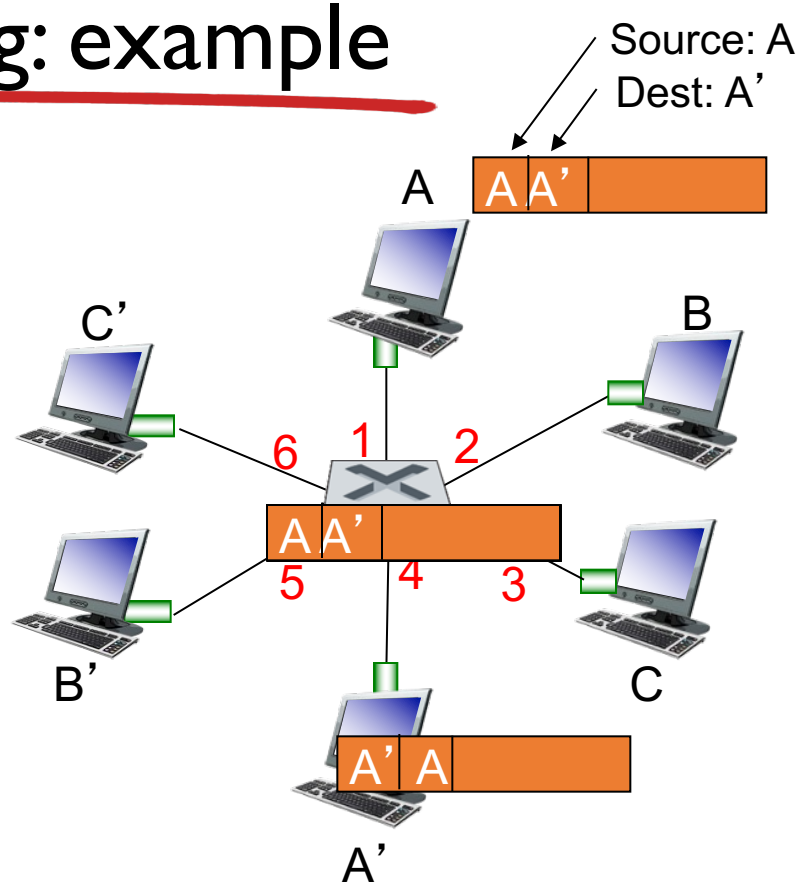
Switch: frame filtering/forwarding

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

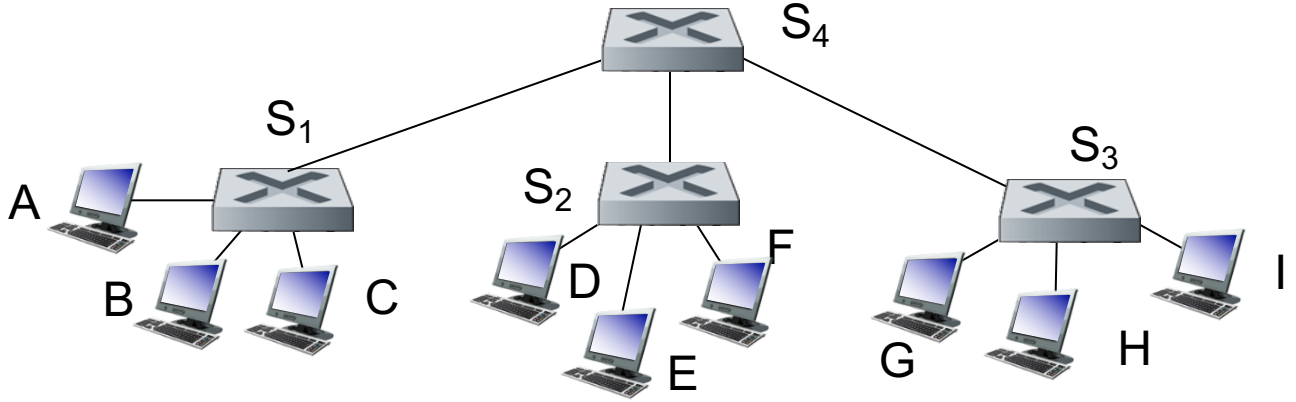


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Interconnecting switches

self-learning switches can be connected together:

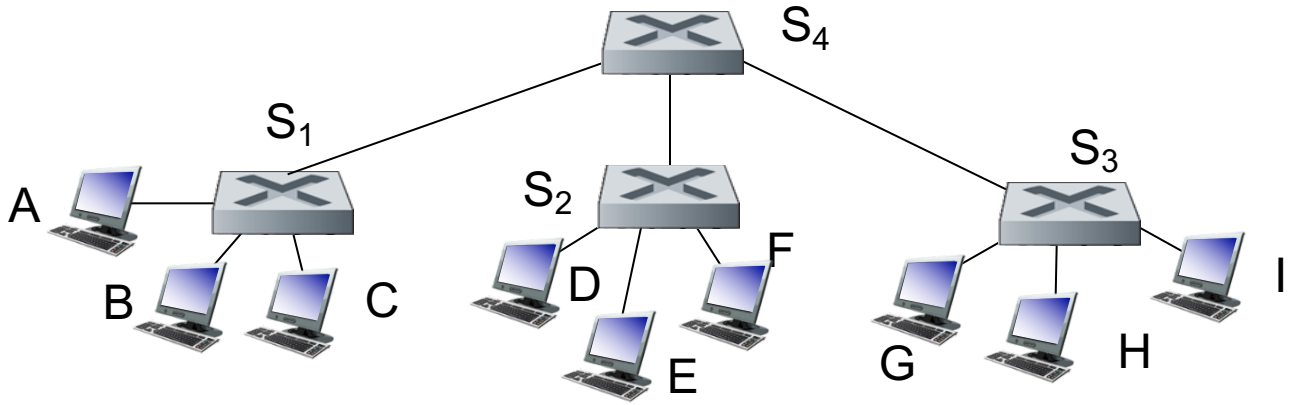


Q: sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?

- *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



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

Week 16-Lec 1

DNS and NAT

DNS: domain name system

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., www.yahoo.com - used by humans

Q: how to map between IP address and name, and vice versa ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS: services, structure

DNS services

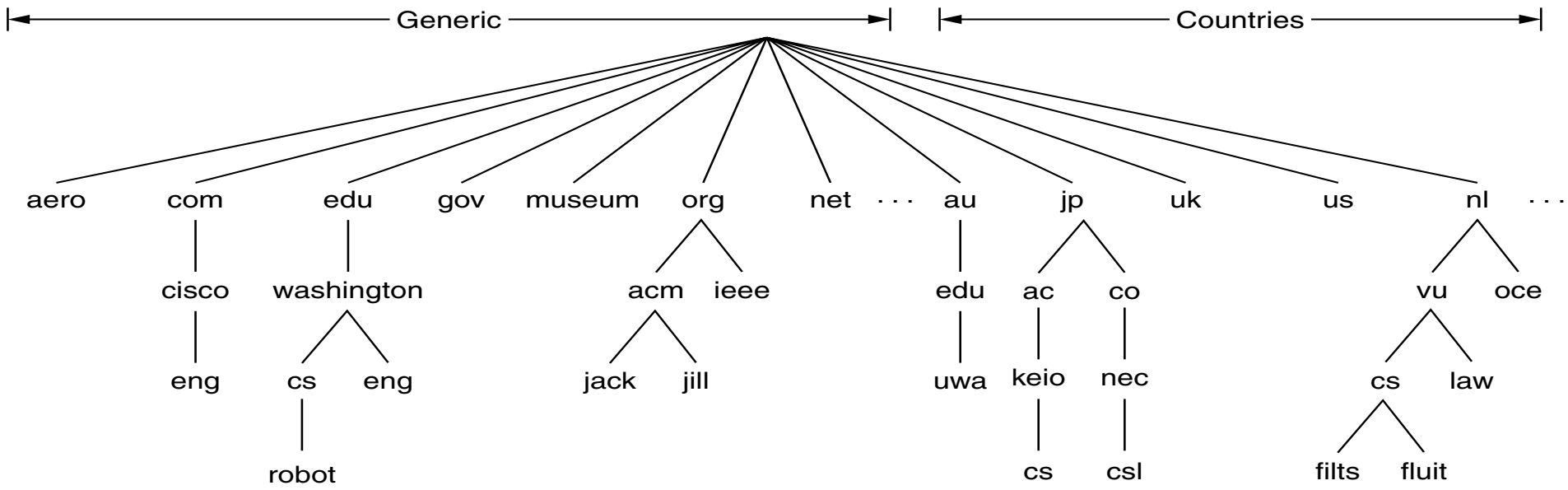
- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

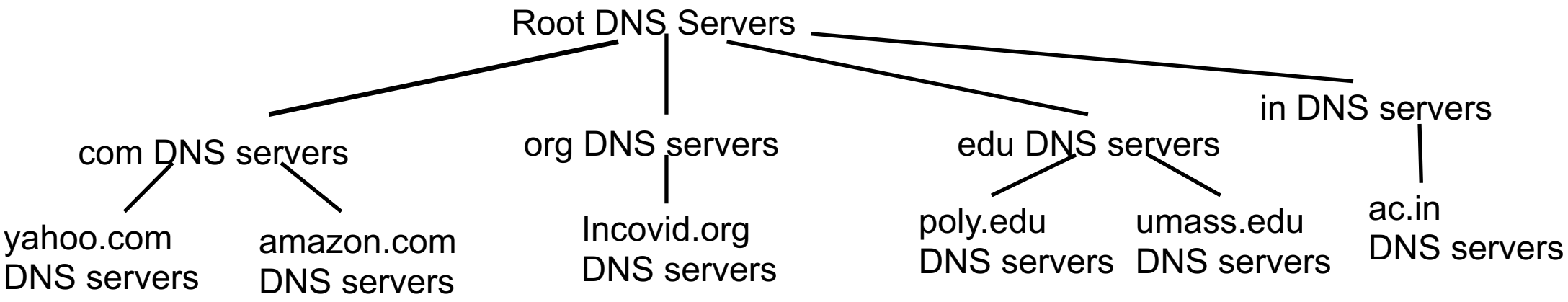
why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: doesn't scale!

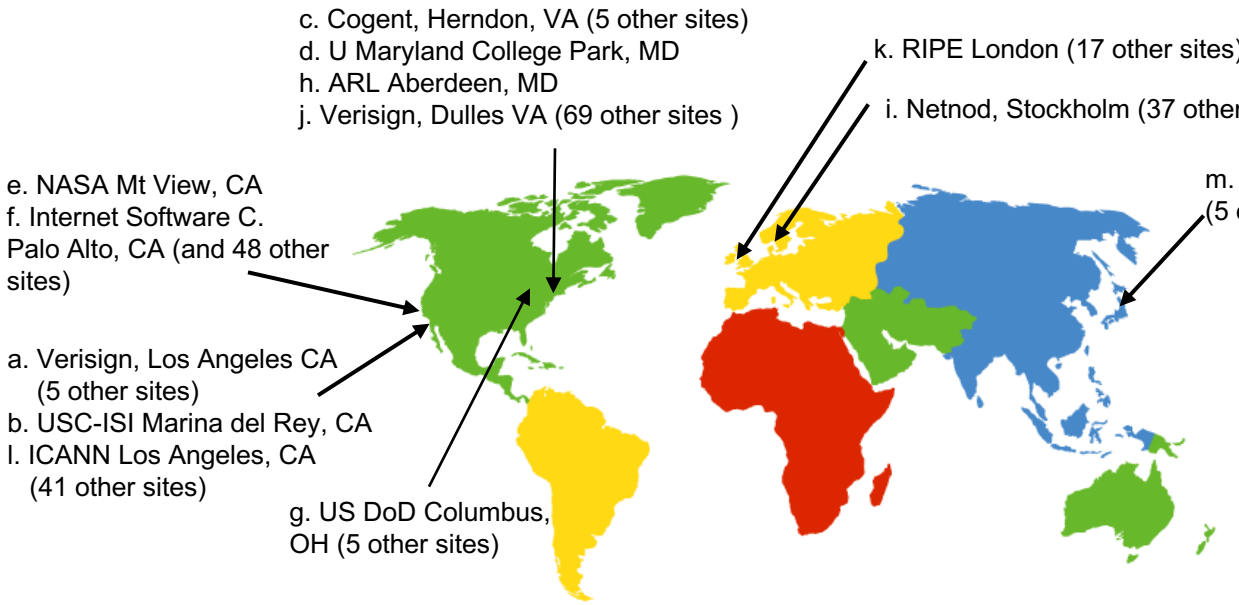
DNS: a distributed, hierarchical database





DNS: root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server



13 logical root name "servers" worldwide

- *each "server" replicated many times*

TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

authoritative DNS servers:

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

Local DNS name server

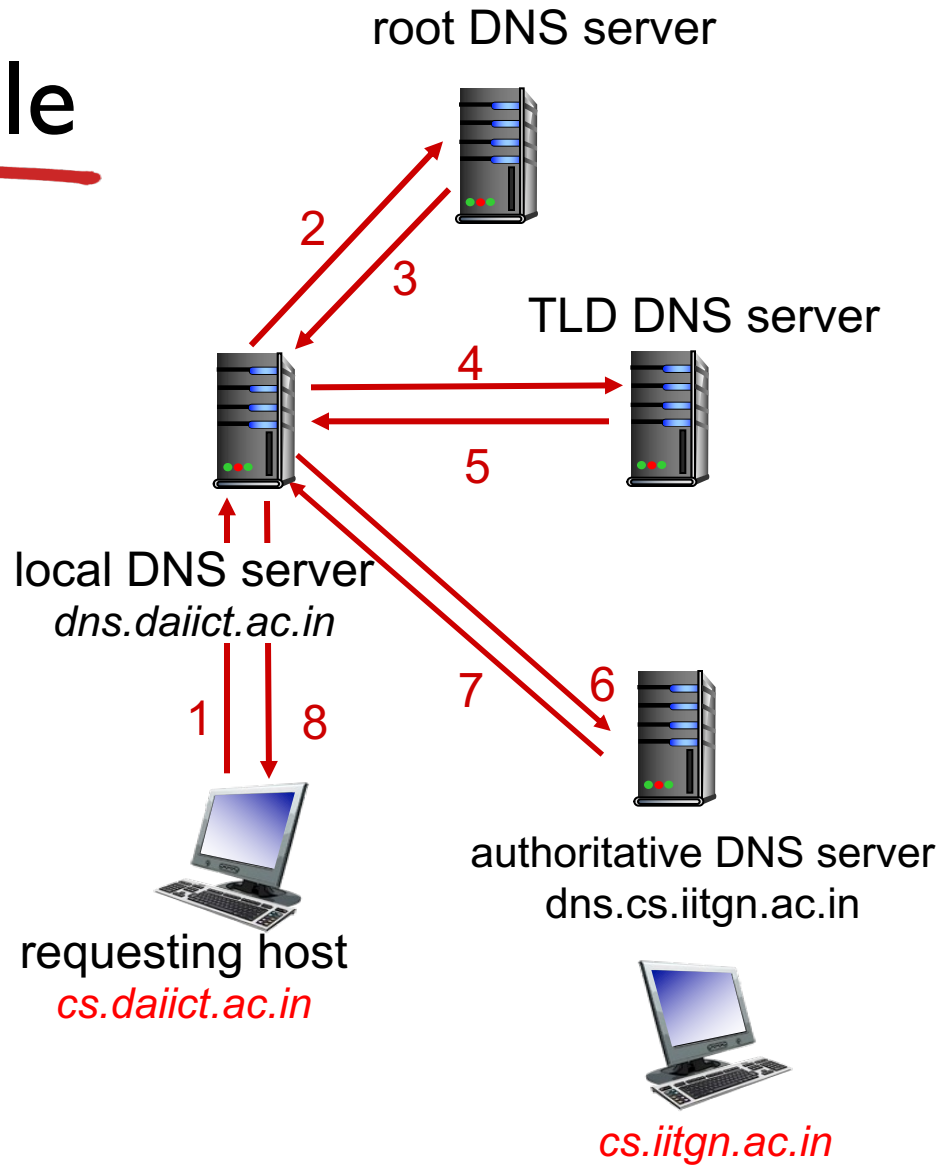
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

- host at *cs.daiict.ac.in* wants IP address for *cs.iitgn.ac.in*

iterated query:

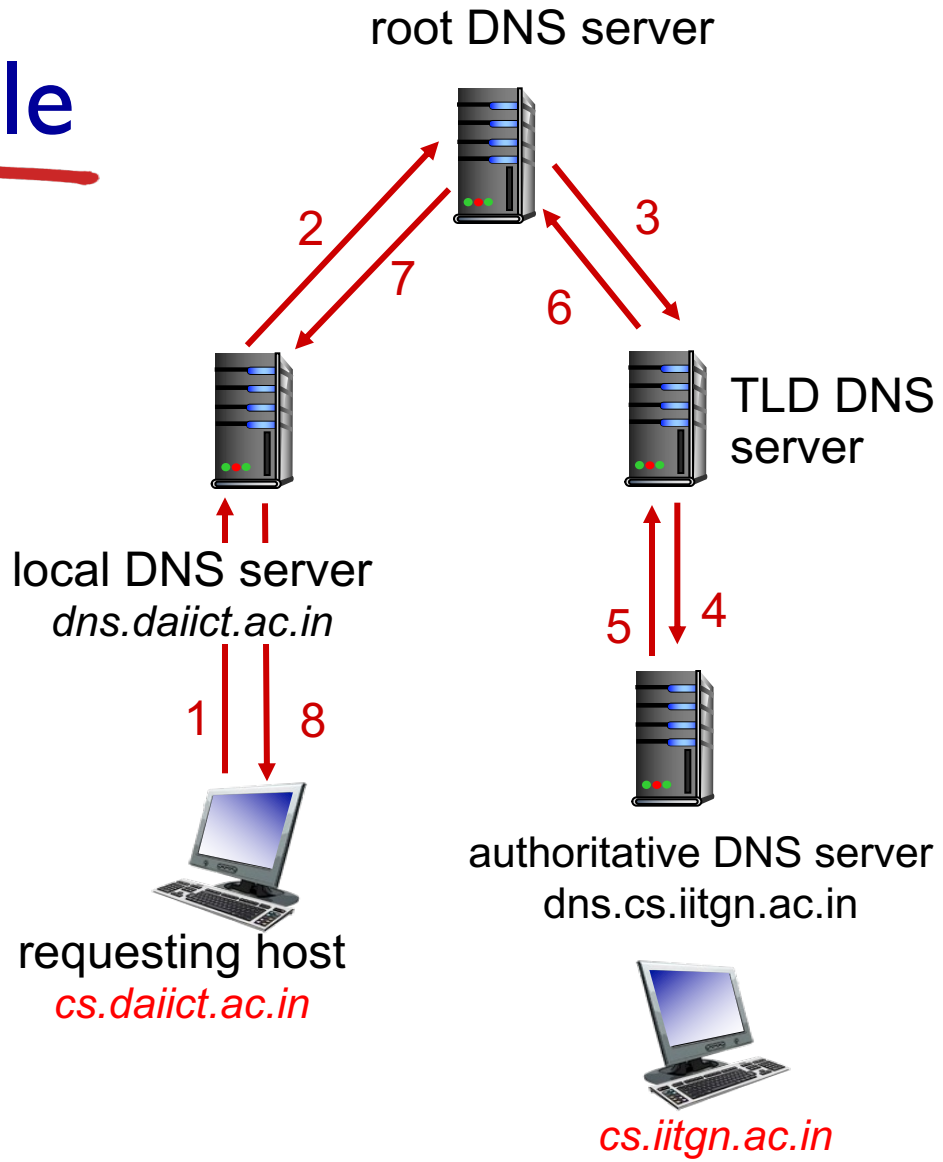
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS: caching, updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited
- cached entries may be *out-of-date* (best effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire
- update/notify mechanisms proposed IETF standard
  - RFC 2136

DNS records

DNS: distributed database storing resource records (RR)

RR format: `(name, value, type, ttl)`

type=A

- **`name`** is hostname
- **`value`** is IP address

type=NS

- **`name`** is domain (e.g., foo.com)
- **`value`** is hostname of authoritative name server for this domain

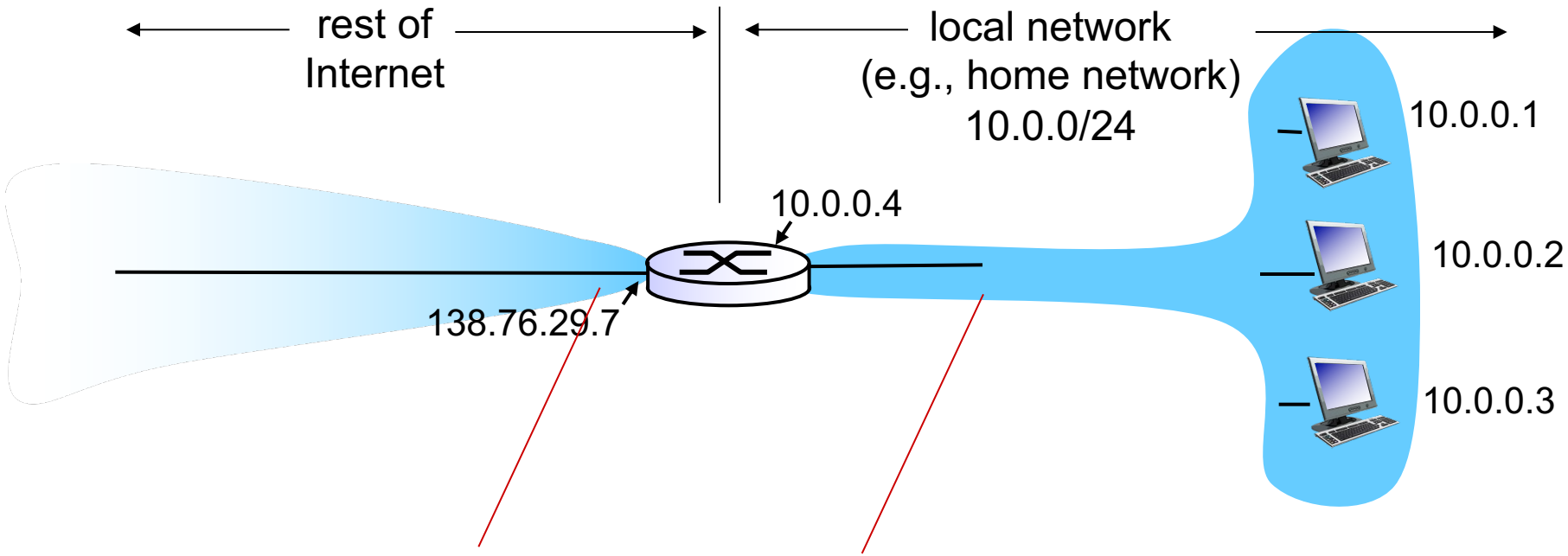
type=CNAME

- **`name`** is alias name for some "canonical" (the real) name
- **`www.ibm.com`** is really **`servereast.backup2.ibm.com`**
- **`value`** is canonical name

type=MX

- **`value`** is name of mailserver associated with **`name`**

NAT: network address translation



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

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

NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

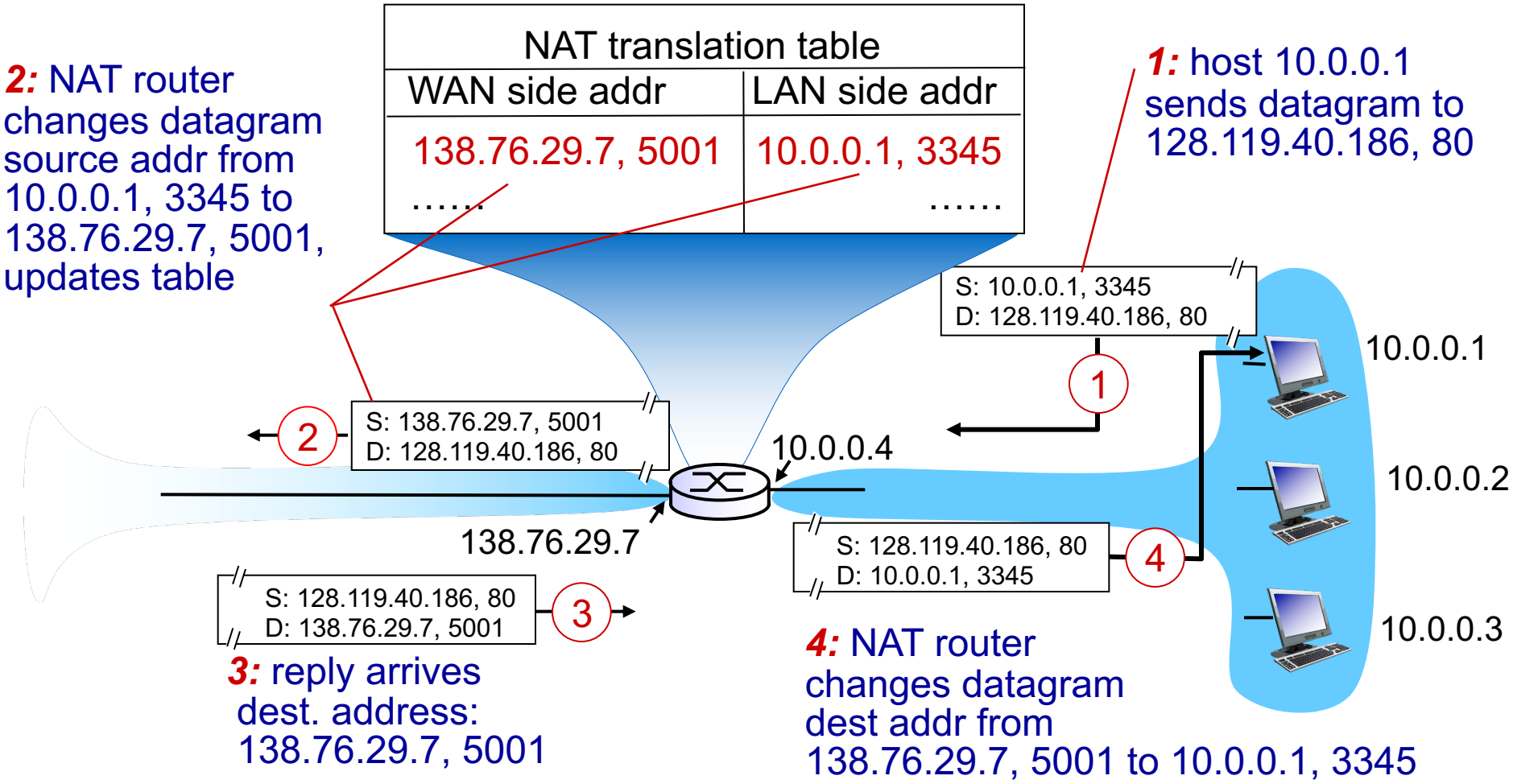
- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation



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

NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - address shortage should be solved by IPv6
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - NAT traversal: what if client wants to connect to server behind NAT?

Week 15-Lec 2

CDN, P2P

Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users
- challenge: scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- *solution:* distributed, application-level infrastructure

Multimedia networking: 3 application types

- *streaming, stored* audio, video
  - *streaming:* can begin playout before downloading entire file
  - *stored (at server):* can transmit faster than audio/video will be rendered (implies storing/buffering at client)
  - e.g., YouTube, Netflix, Hulu
- *conversational* voice/video over IP
  - interactive nature of human-to-human conversation limits delay tolerance
  - e.g., Skype
- *streaming live* audio, video
  - e.g., live sporting event

Multimedia: video

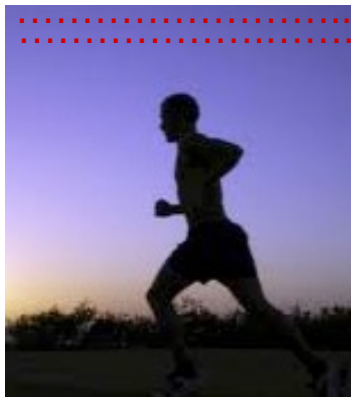
- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

	Bit rate	Bytes transferred in 67 min
Facebook Frank	160 kbps	80 Mbytes
Martha Music	128 kbps	64 Mbytes
Victor Video	2 Mbps	1 Gbyte

Multimedia: video

- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) and *number of repeated values* (N)



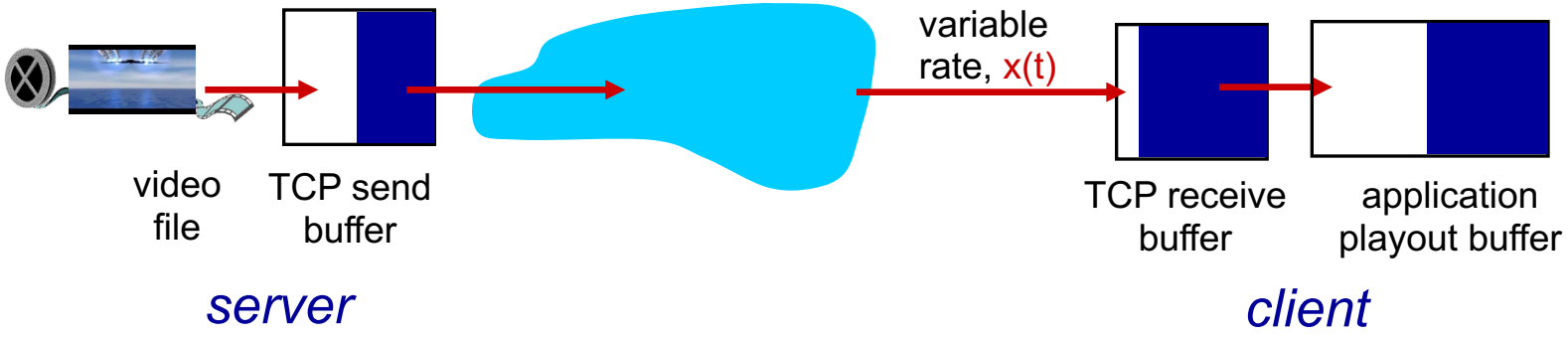
frame *i*

frame *i+1*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Streaming multimedia: HTTP

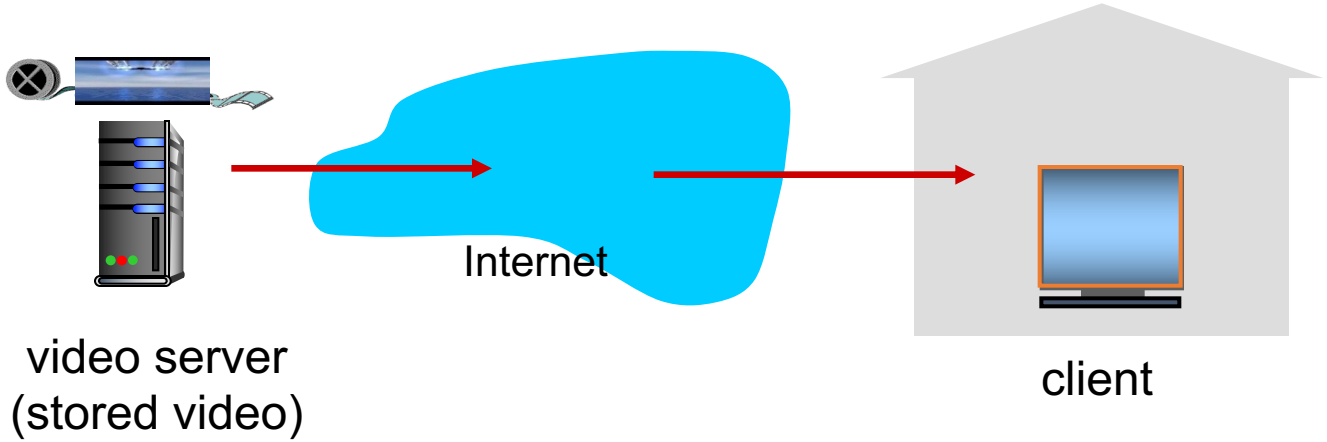
- multimedia file retrieved via HTTP GET
- send at maximum possible rate under TCP



- fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls

Streaming stored video:

simple scenario:



Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- *option 1:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

….quite simply: this solution *doesn't scale*

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

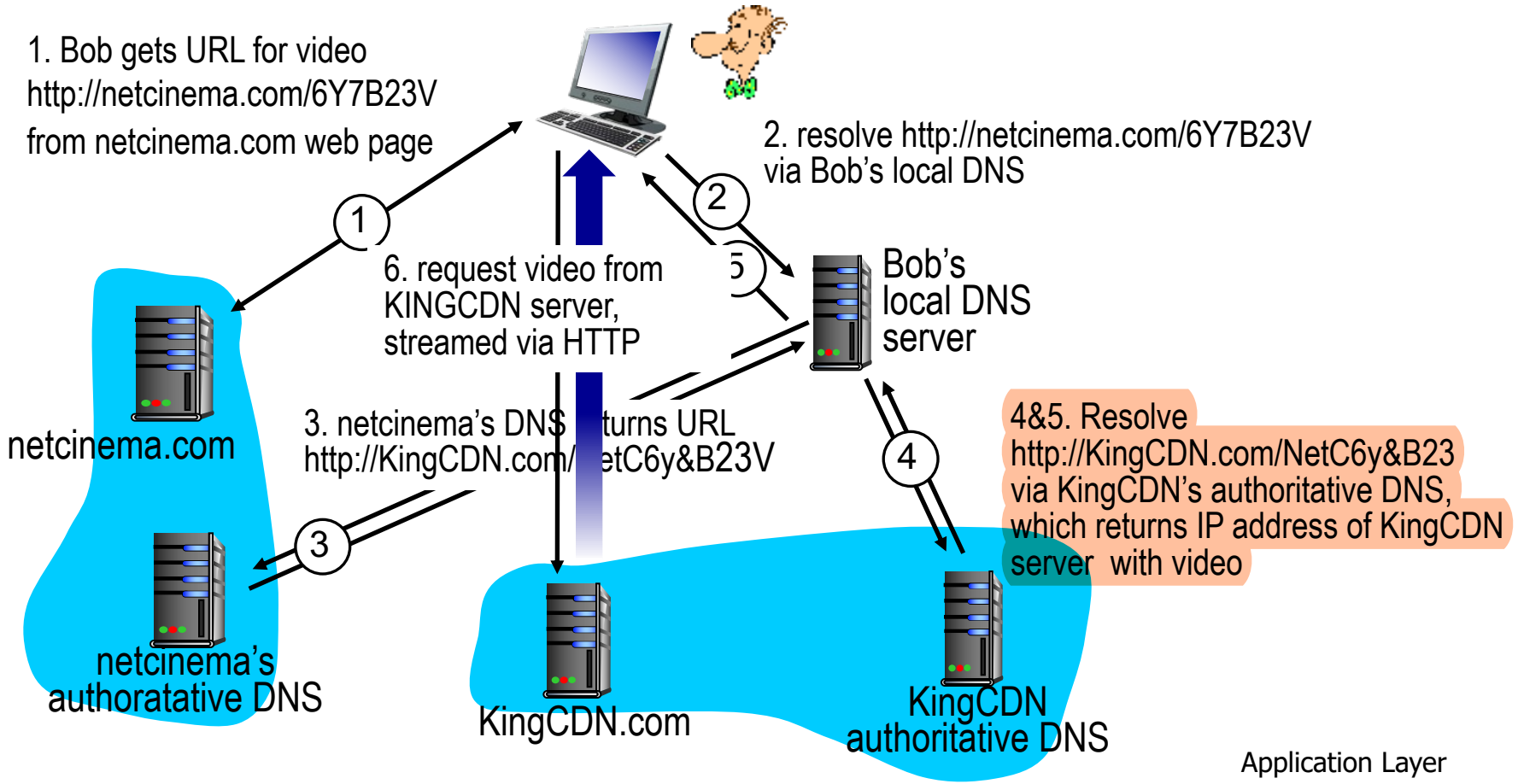
- *option 2:* store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - used by Akamai, 1700 locations

A content delivery network (CDN) is a group of geographically distributed servers that speed up the delivery of web content by bringing it closer to where users

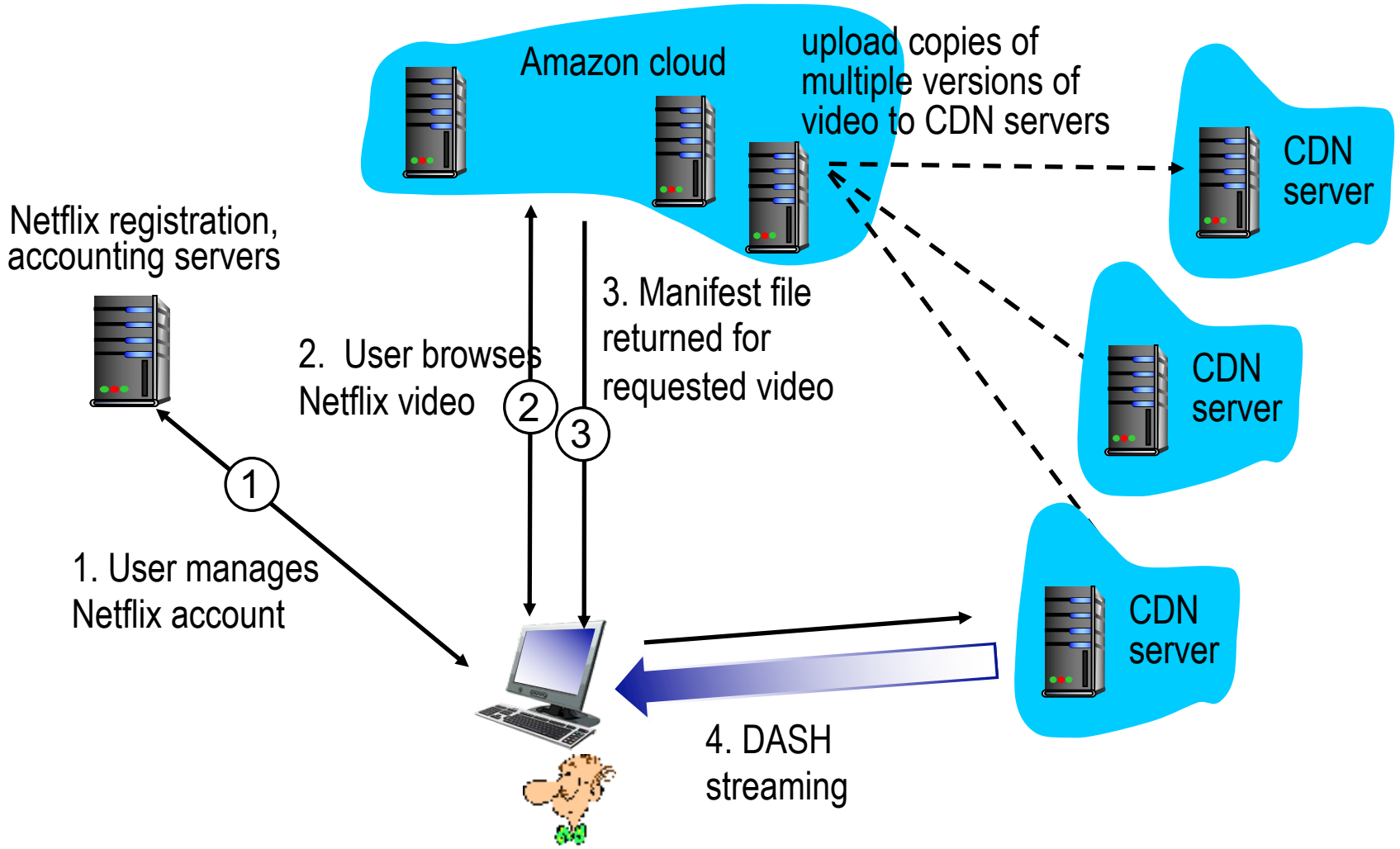
CDN content access: a closer look

Bob (client) requests video http://netcinema.com/6Y7B23V

- video stored in CDN at http://KingCDN.com/NetC6y&B23V



Case study: Netflix



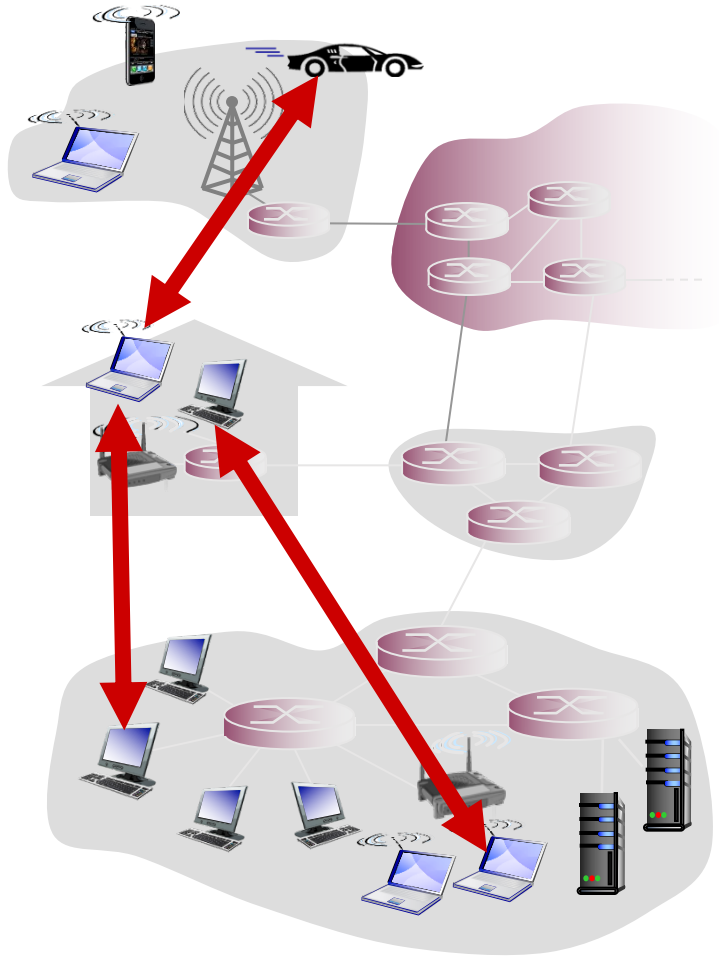
- Content Ingestion
- Content Processing
- Uploading versions to CDN

Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

examples:

- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)

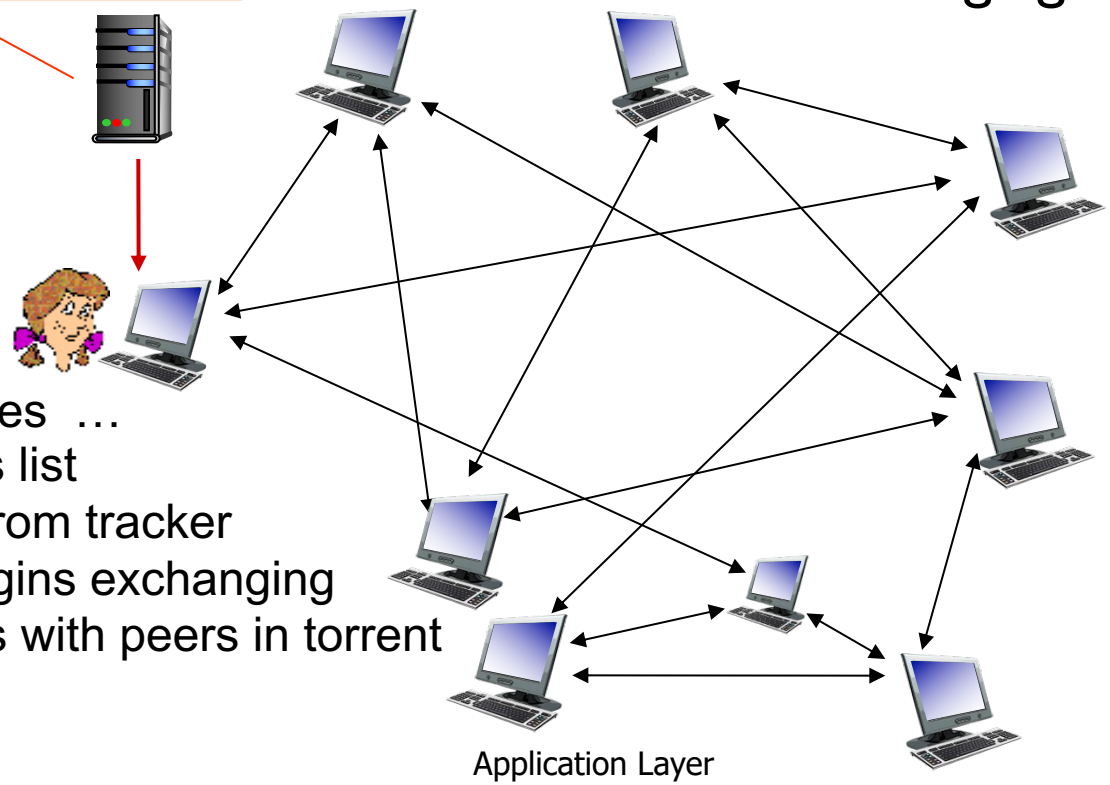


P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

tracker: tracks peers participating in torrent

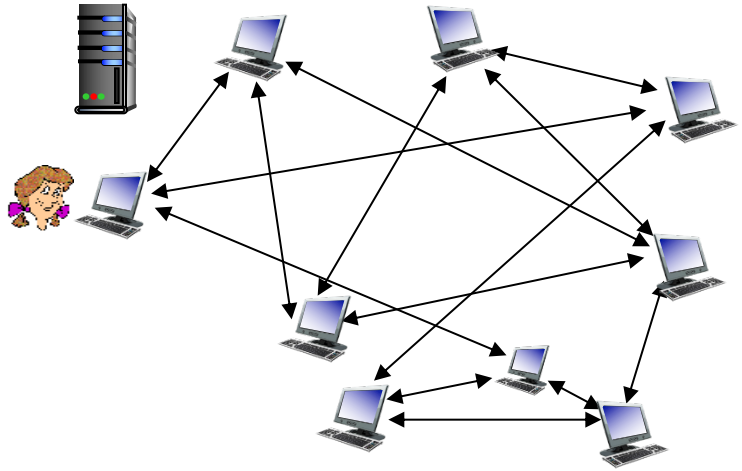
torrent: group of peers exchanging chunks of a file



Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

P2P file distribution: BitTorrent

- peer joining torrent:
  - has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")



- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- *churn:* peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

BitTorrent: requesting, sending file chunks

requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

sending chunks: tit-for-tat

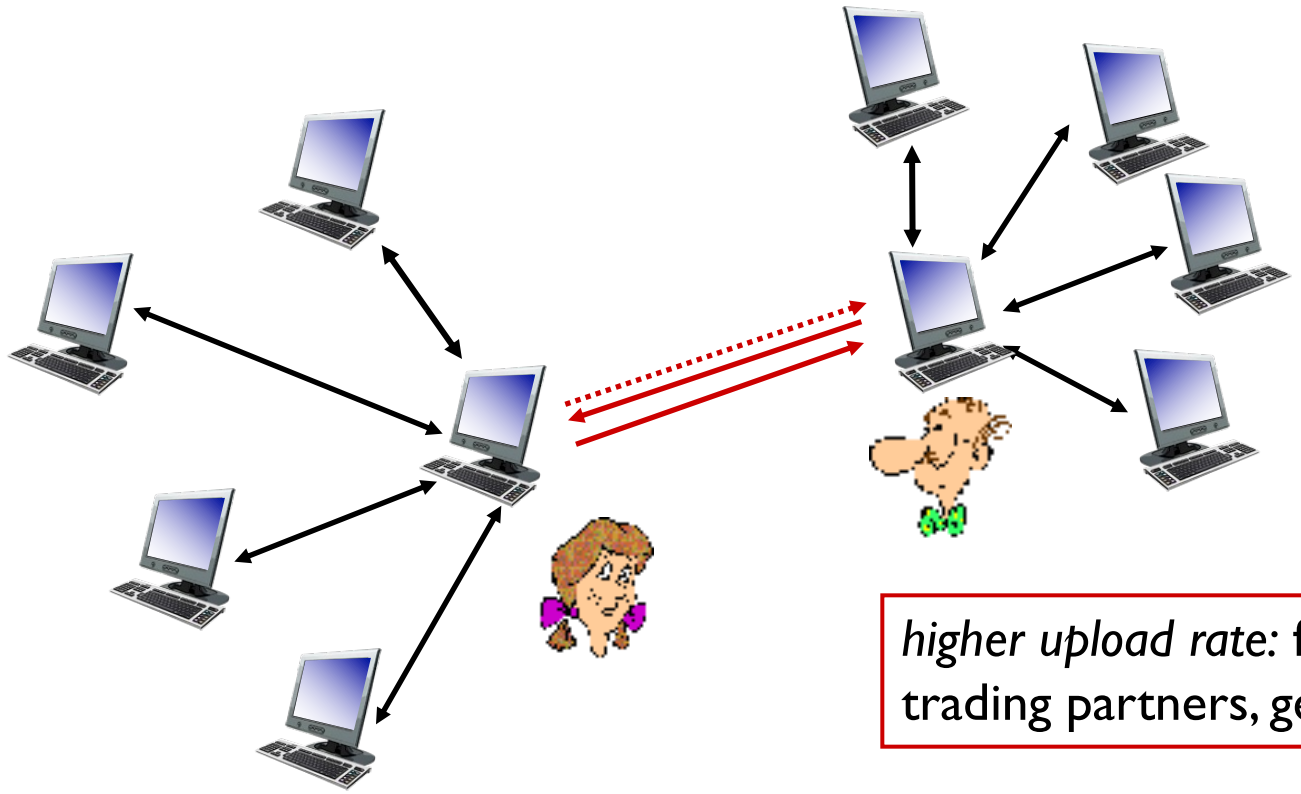
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster !

Data center networks

Inside a 40-ft Microsoft container, Chicago data center

- 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
  - e-business (e.g. Amazon)
  - content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
  - search engines, data mining (e.g., Google)

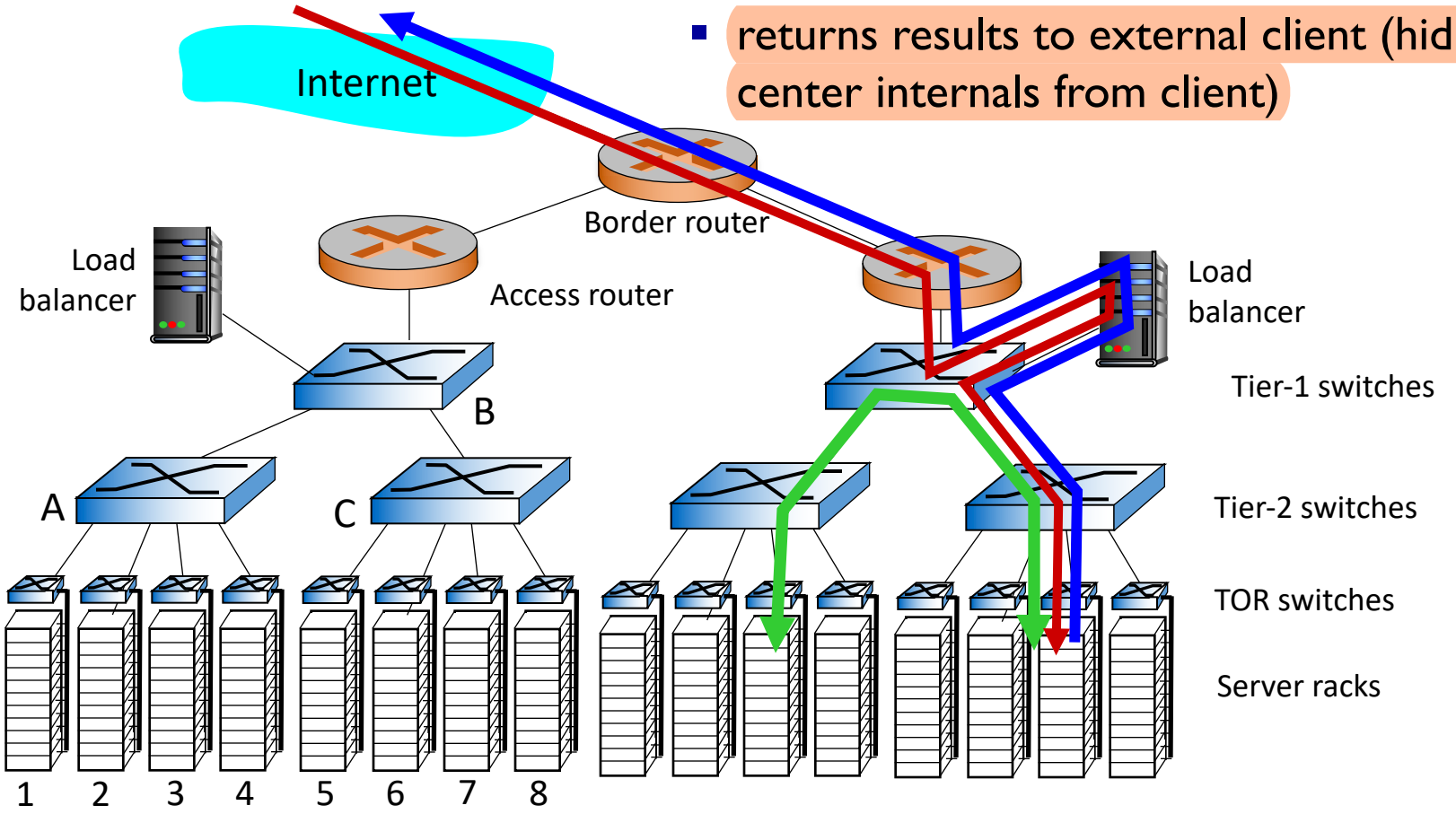
- challenges:
  - multiple applications, each serving massive numbers of clients
  - managing/balancing load, avoiding processing, networking, data bottlenecks

Inside Facebook's datacenter

Data center networks

load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)



Connectivity

- The hosts (**blades)**
  - Include CPU, memory, and disk storage.
  - Stacked in racks, with each rack typically having 20 to 40 blades.
  - At the top of each rack there is a switch, **Top of Rack (TOR) switch**
    - **interconnects** the hosts in the rack with each other and with other switches in the data center
    - Each host has a NIC card that connects to its TOR switch,
    - Each TOR switch has additional ports that can be connected to other switches.
- hosts below each access router form a single subnet.
- In order to localize ARP broadcast traffic,
  - each of these subnets is further partitioned into smaller VLAN subnets
    - each comprising a few hundred hosts

Traffic types

- traffic flowing between external clients and internal hosts
  - One of more Border routers – Data center to public Internet
- Traffic flowing between internal hosts
- External requests directed to load balancer
  - Distribute jobs/requests to servers
  - Destination port num and IP address

How are datacenter networks different from networks we've seen before?

- **Scale**: very few local networks have so many machines in one place: 10's of thousands of servers — and they all work together like one computer!

- **Control**: entirely administered by one organization — unlike the Internet, datacenter owners control every switch in the network **and** the software on every host

- **Performance:** datacenter latencies are 10s of us, with 10, 40, even 100Gbit links.

How do these factors change how we *design* datacenter networks?

Desirable Properties

- **Low Latency:** Very few "hops" between destinations
- **Resilience:** Able to recover from link failures
- **Good Throughput:** Lots of endpoints can communicate, all at the same time.
- **Cost-Effective:** Does not rely too much on expensive equipment like very high bandwidth, high port-count switches.
- **Easy to Manage:** Won't confuse network administrators who have to wire so many cables together!

Border Gateway routing protocol

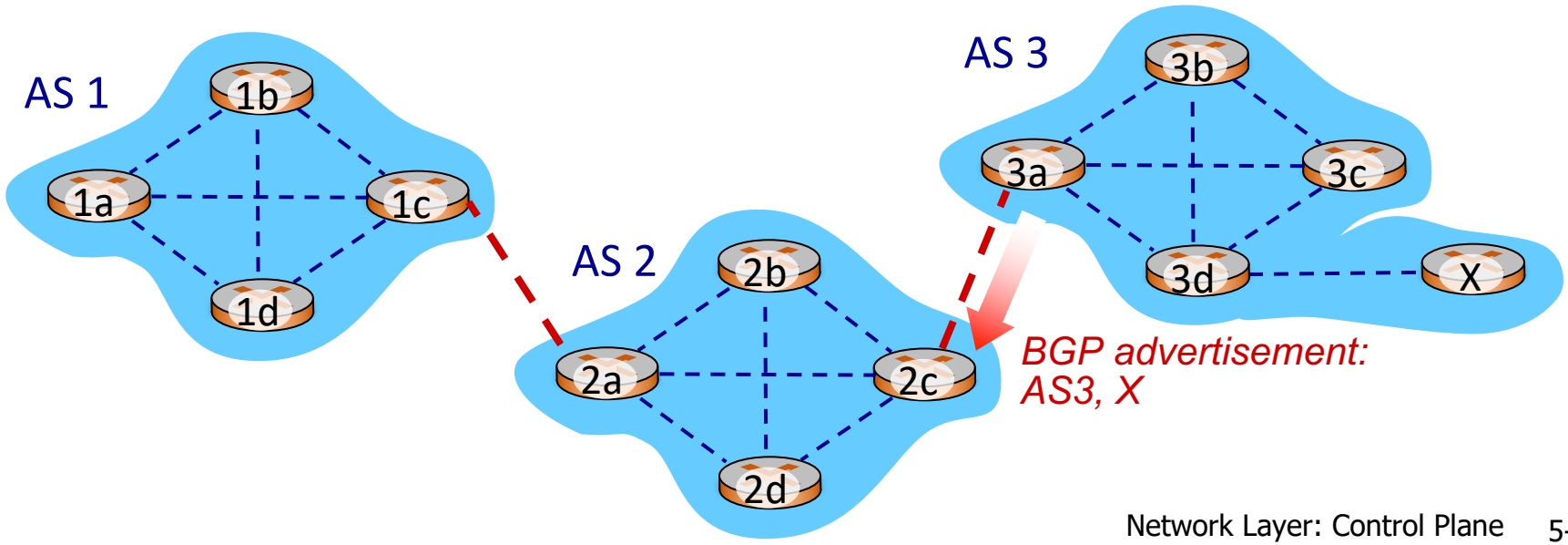
Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASes
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and *policy*
- allows subnet to advertise its existence to rest of Internet: *"I am here"*

Autonomous System

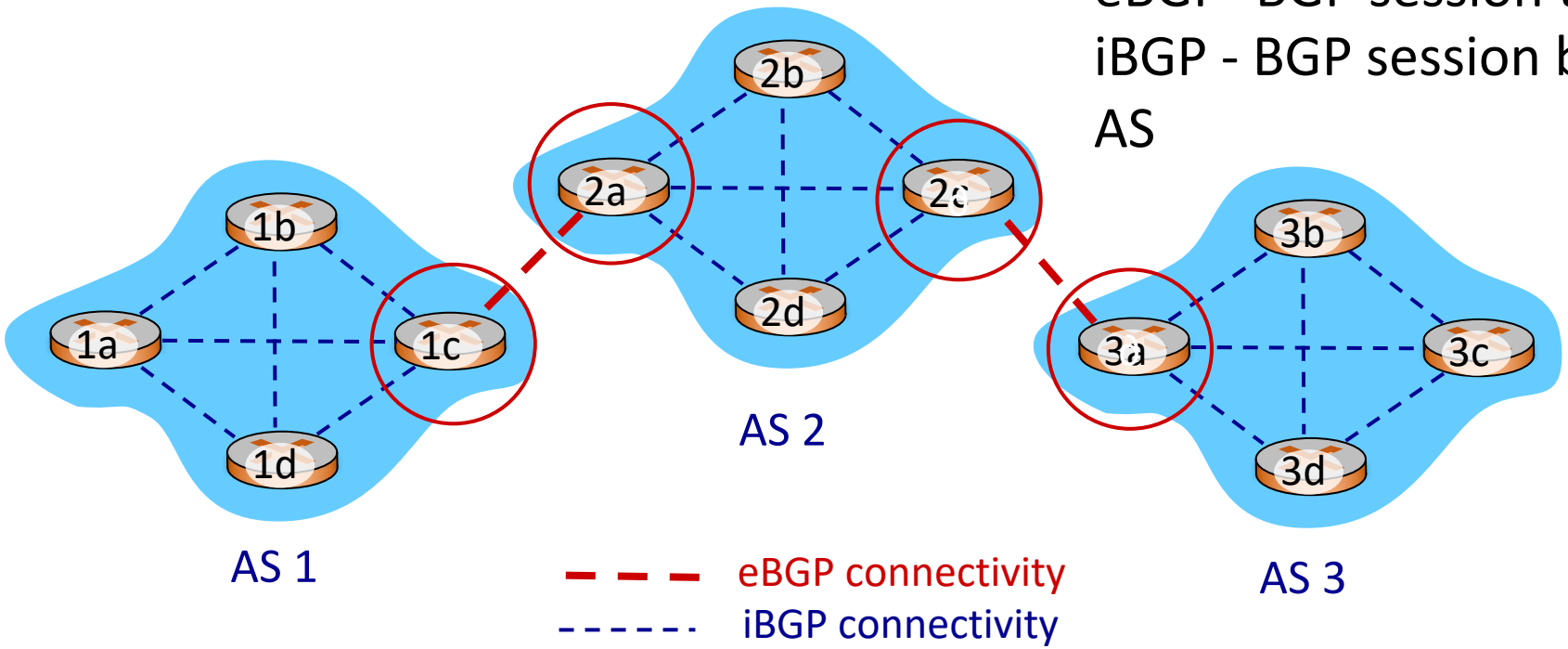
BGP basics

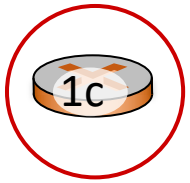
- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection, port 179:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- when AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards X



eBGP, iBGP connections

eBGP- BGP session that spans two Ass
iBGP - BGP session between routers within an AS

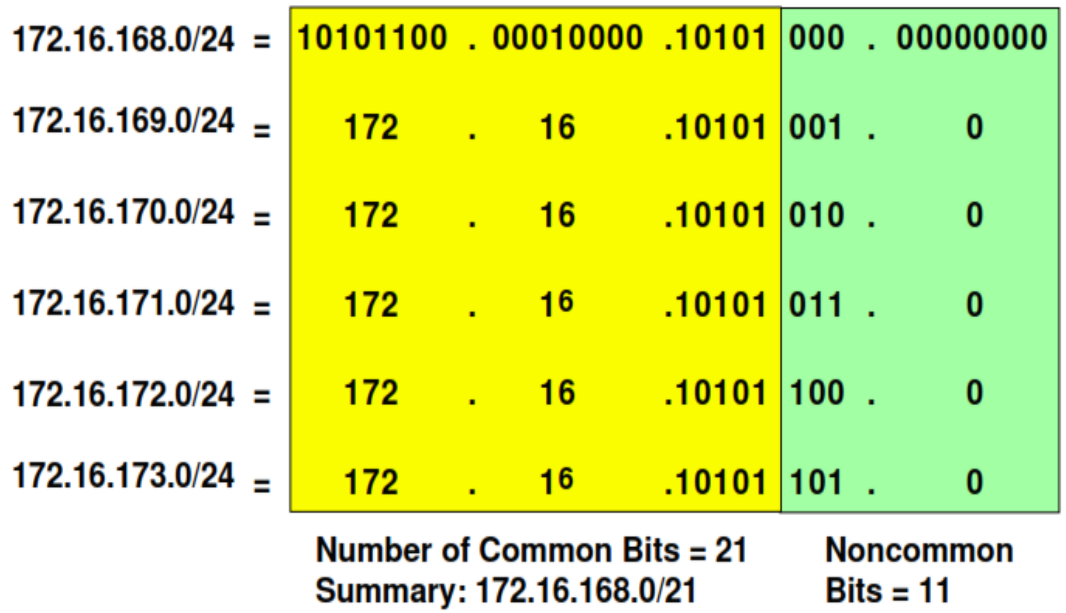




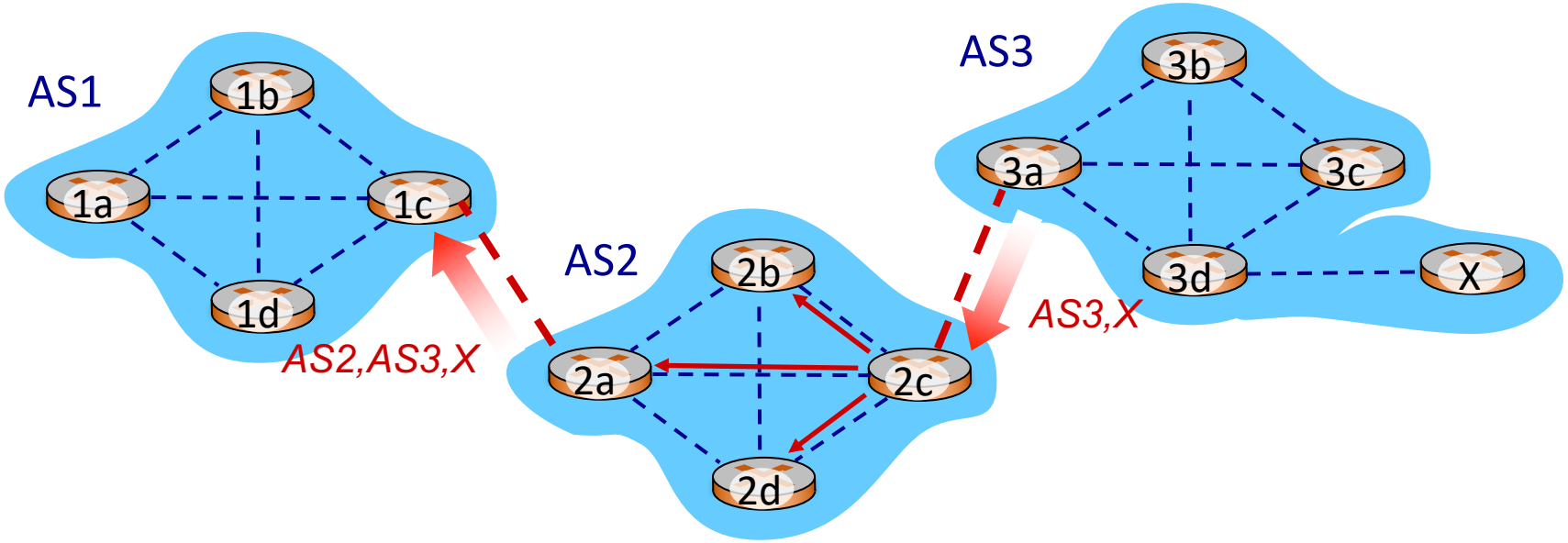
gateway routers run both eBGP and iBGP protools

each prefix representing a subnet or a collection of subnets. Thus, for example, suppose there are four subnets attached to AS2: **138.16.64/24**, **138.16.65/24,** **138.16.66/24**, and **138.16.67/24**. Then AS2 could aggregate the prefixes for these four subnets and use BGP to advertise the single prefix to **138.16.64/22** to AS1.

Route summarization



BGP path advertisement

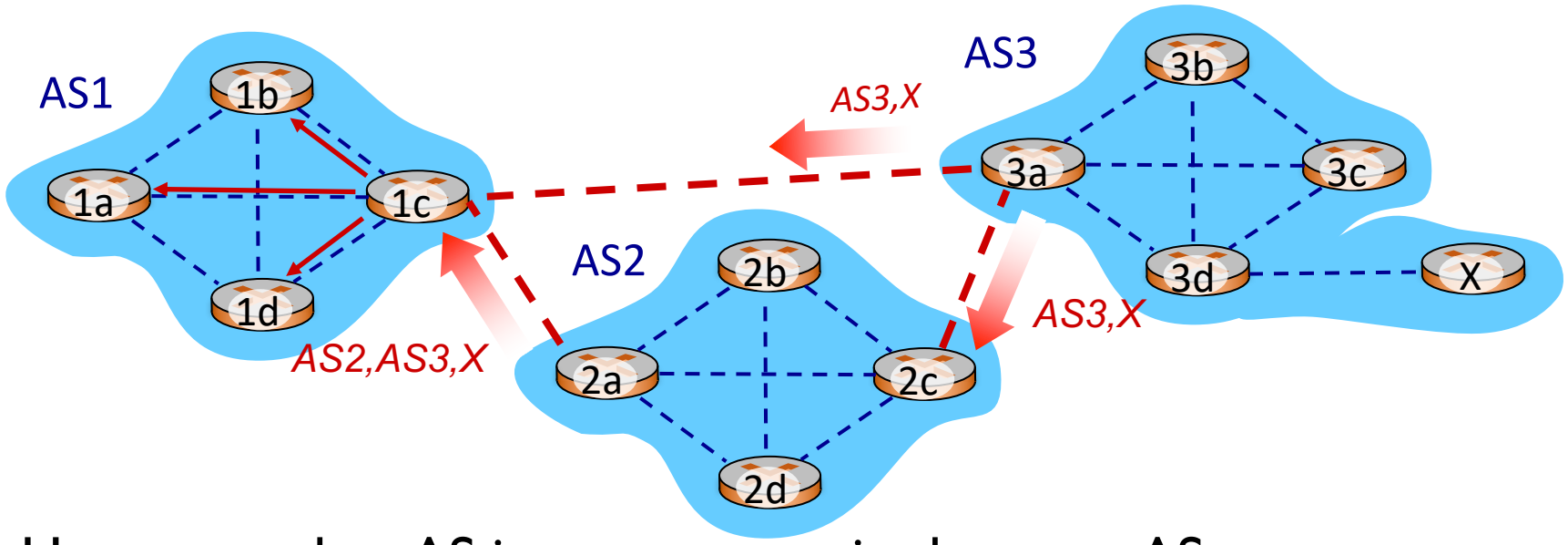


- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

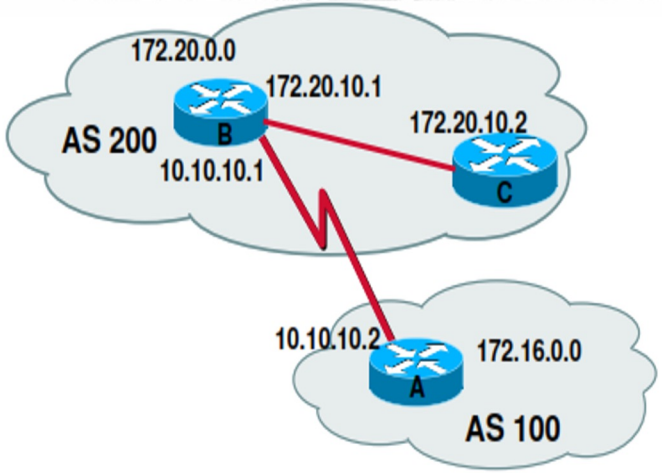
Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: list of ASes through which prefix advertisement has passed
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS
- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other other neighboring ASes

BGP Next Hop



- Next Hop to another AS is a router not in the same AS
- However, the subnet that contains this IP address directly attaches to AS1:
- **1c and 2a (Next HOP) are connected., they share a separate subnet address**
- **Similarly, 1c and 3a(Next HOP) are connected**



Next-Hop to reach a network

- Router A will advertise network 172.16.0.0 to Router B in EBGP, with a next hop of 10.10.10.2
- Router B advertises 172.16.0.0 in IBGP to Router C keeping 10.10.10.2 as the next hop address

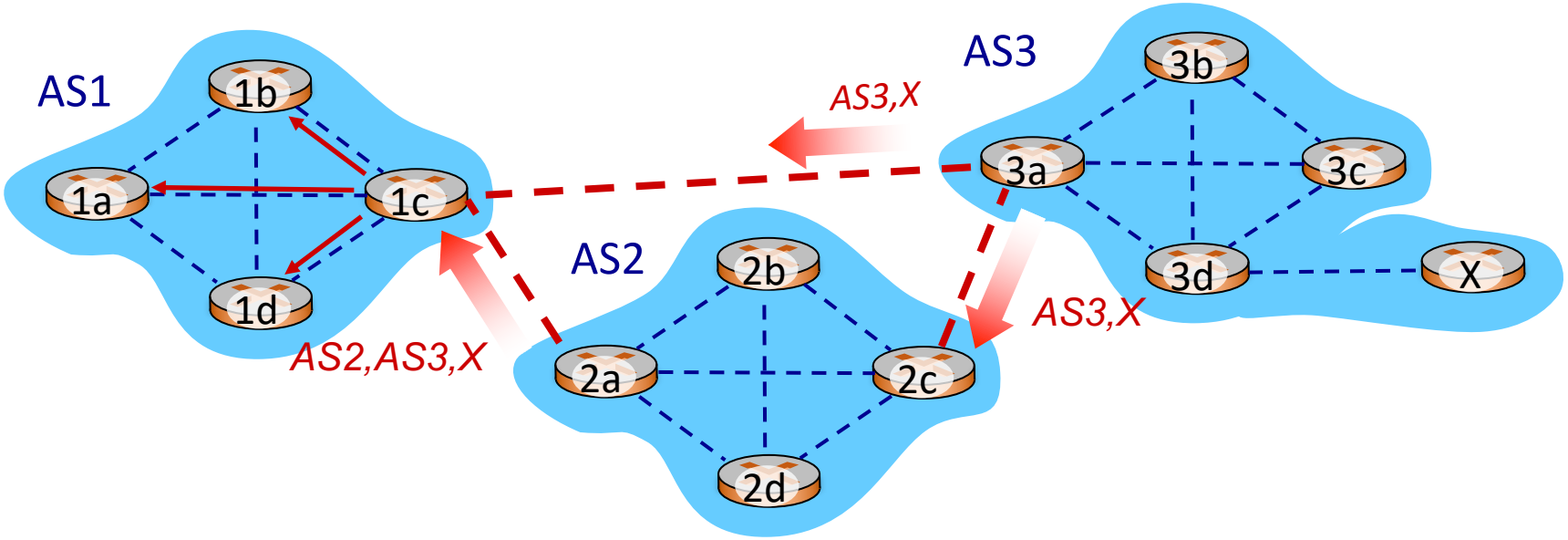
For eBGP, the next hop is the IP address of the neighbor specified who sent the update.
Router A will advertise 172.16.0.0 to Router B, with a next hop of 10.10.10.2

For iBGP, the protocol states that *the next hop advertised by EBGP should be carried into iBGP. Because of that rule, Router B will advertise 172.16.0.0 to its i*BGP peer Router C, with a next hop of 10.10.10.2 (Router A's address).
Therefore Router C knows the next hop to reach 172.16.0.0 is **10.10.10.2**, not **172.20.10.1**
But how does Router C find out how to reach 10.10.10.2?
 OSPF or RIP

BGP route selection

- router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

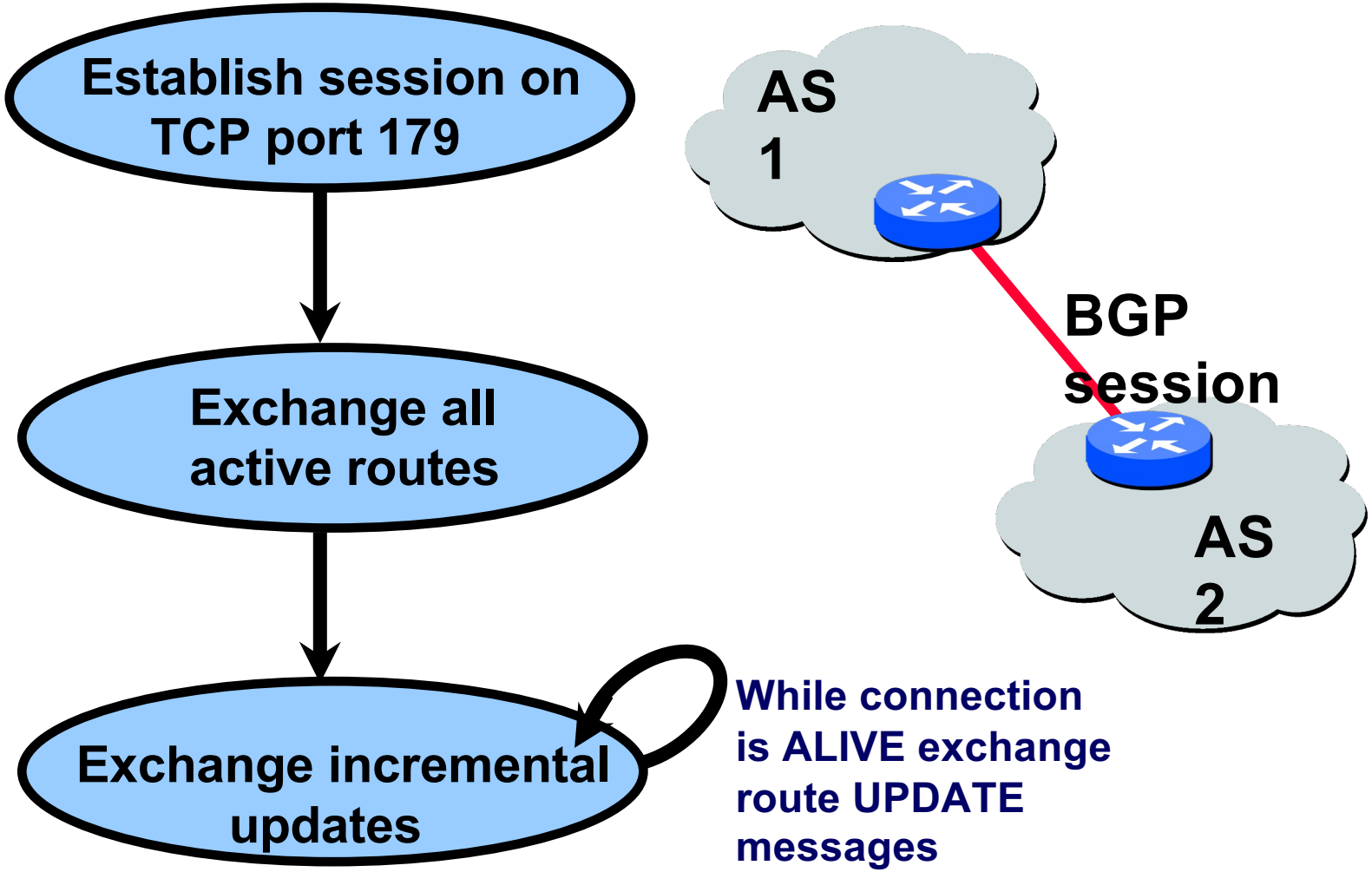
BGP path advertisement



gateway router may learn about multiple paths to destination:

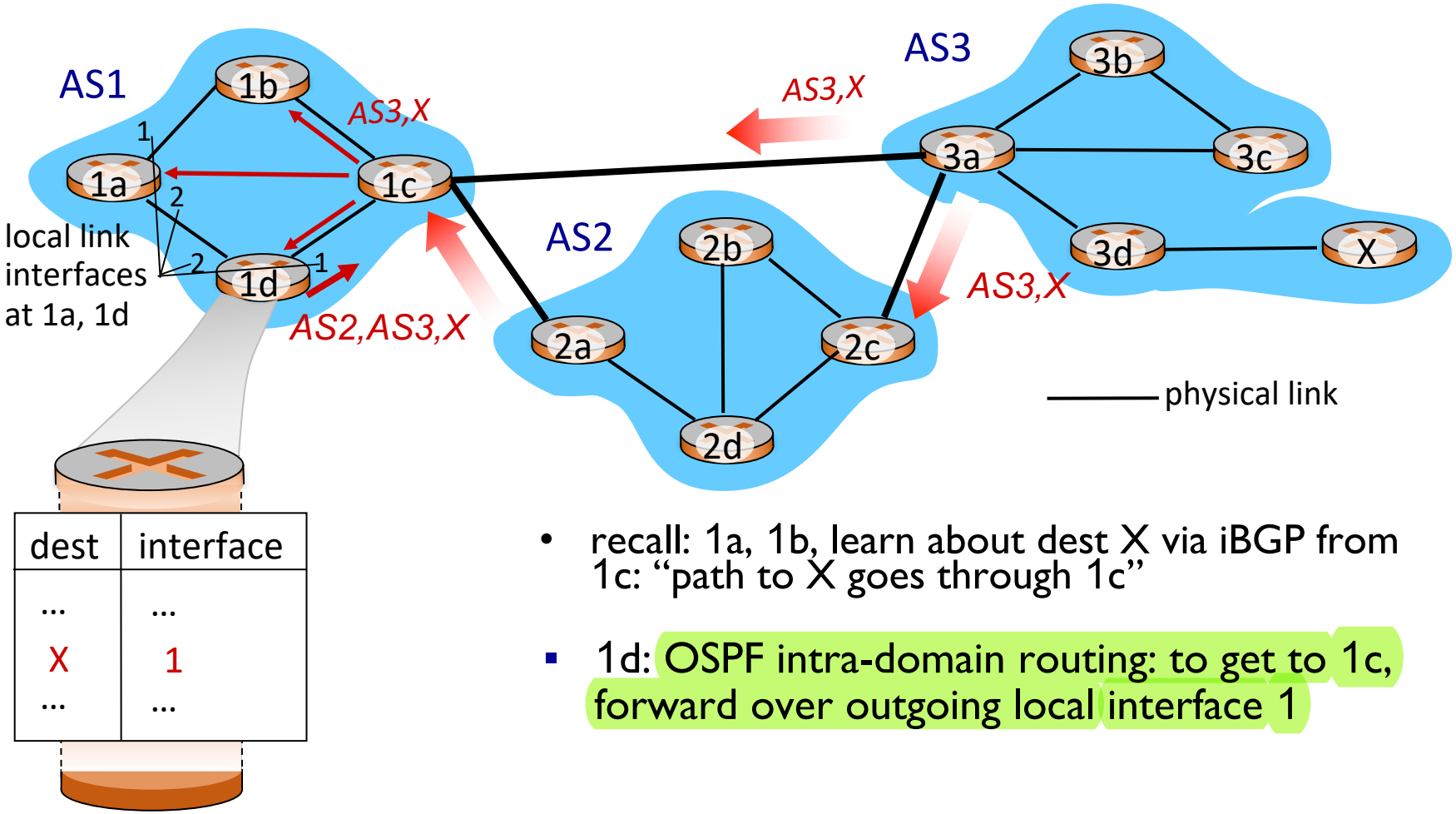
- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- Based on policy, AS1 gateway router 1c chooses path *AS3,X, and advertises path within AS1 via iBGP*

BGP Operations



BGP, OSPF, forwarding table entries

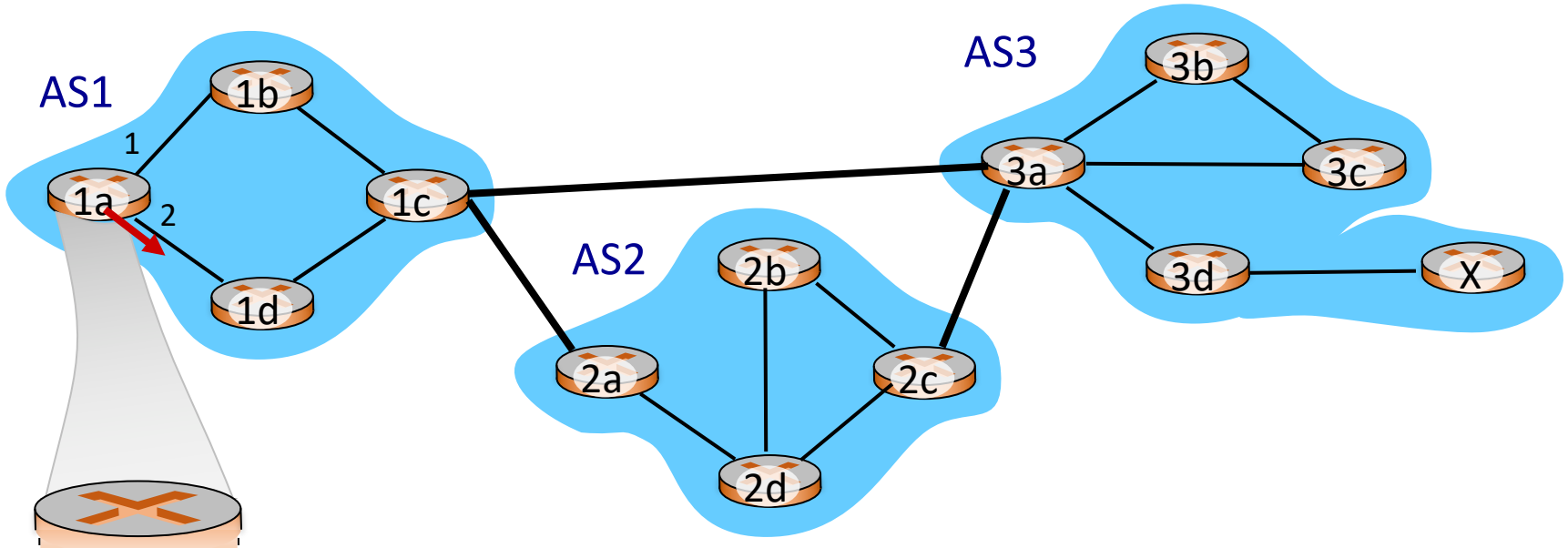
Q: how does router set forwarding table entry to distant prefix?



- recall: 1a, 1b, learn about dest X via iBGP from 1c: "path to X goes through 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

BGP, OSPF, forwarding table entries

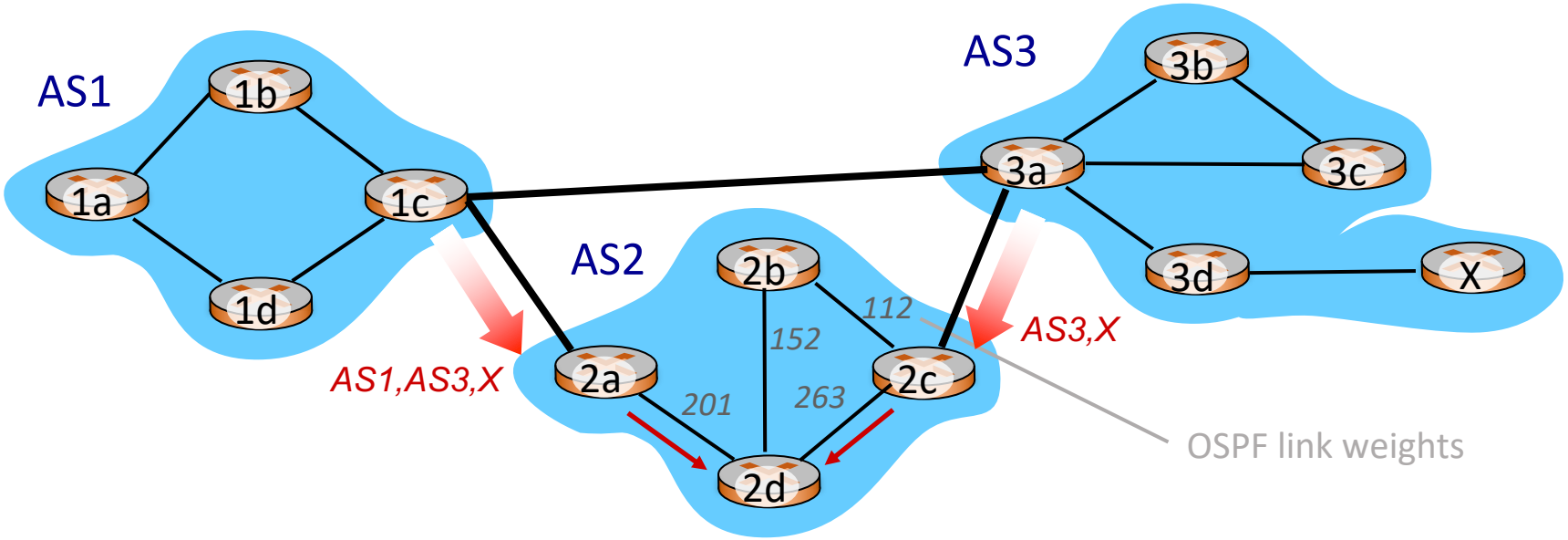
Q: how does router set forwarding table entry to distant prefix?



dest	interface
...	...
X	2
...	...

- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1
- 1a: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 2

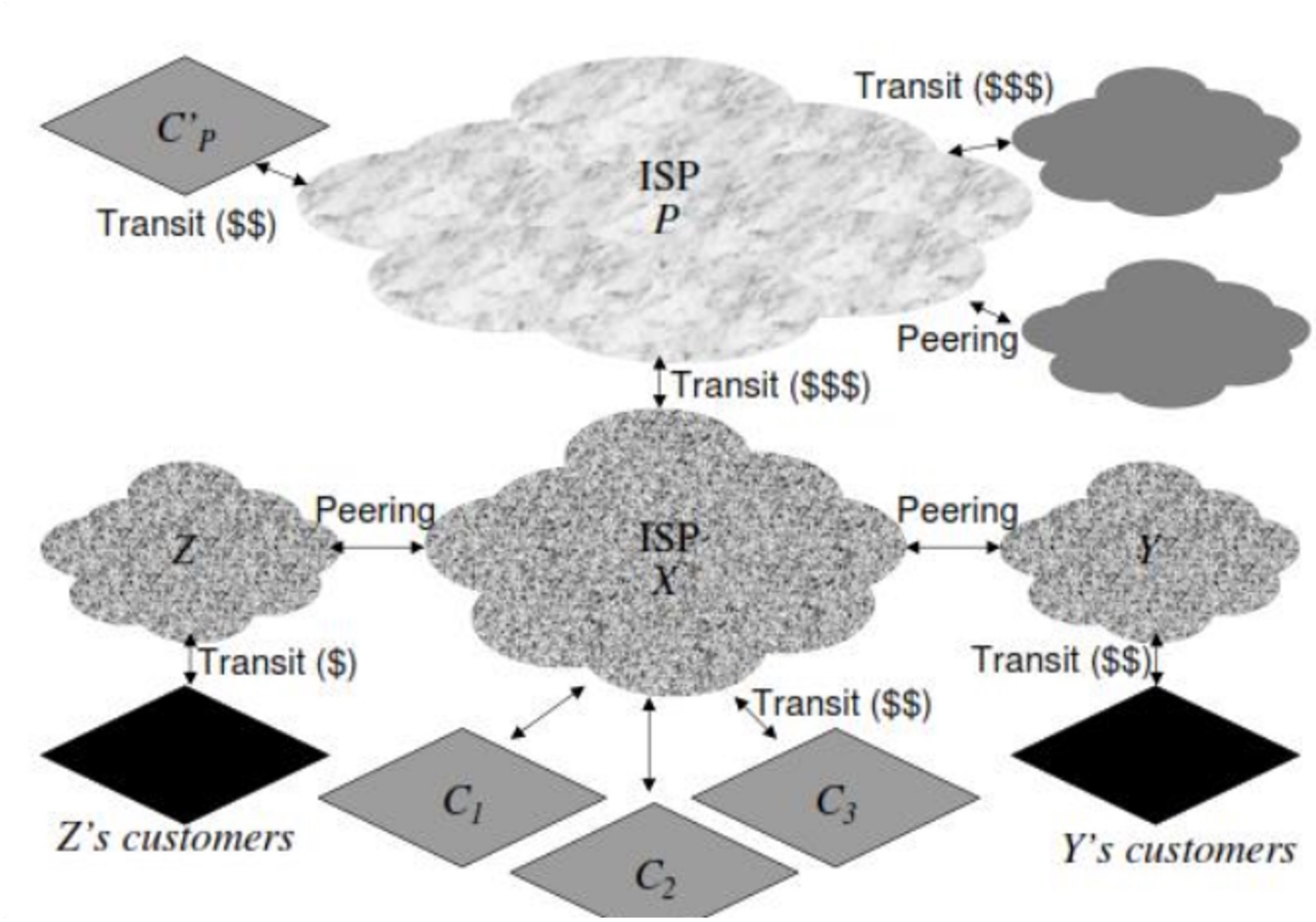
Hot Potato Routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- *hot potato routing:* choose local gateway that has least intra-domain cost (e.g., 2d chooses 2a, even though more AS hops to *X*): don't worry about inter-domain cost!

BGP relationships & policy

- Transit
- Peering



Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

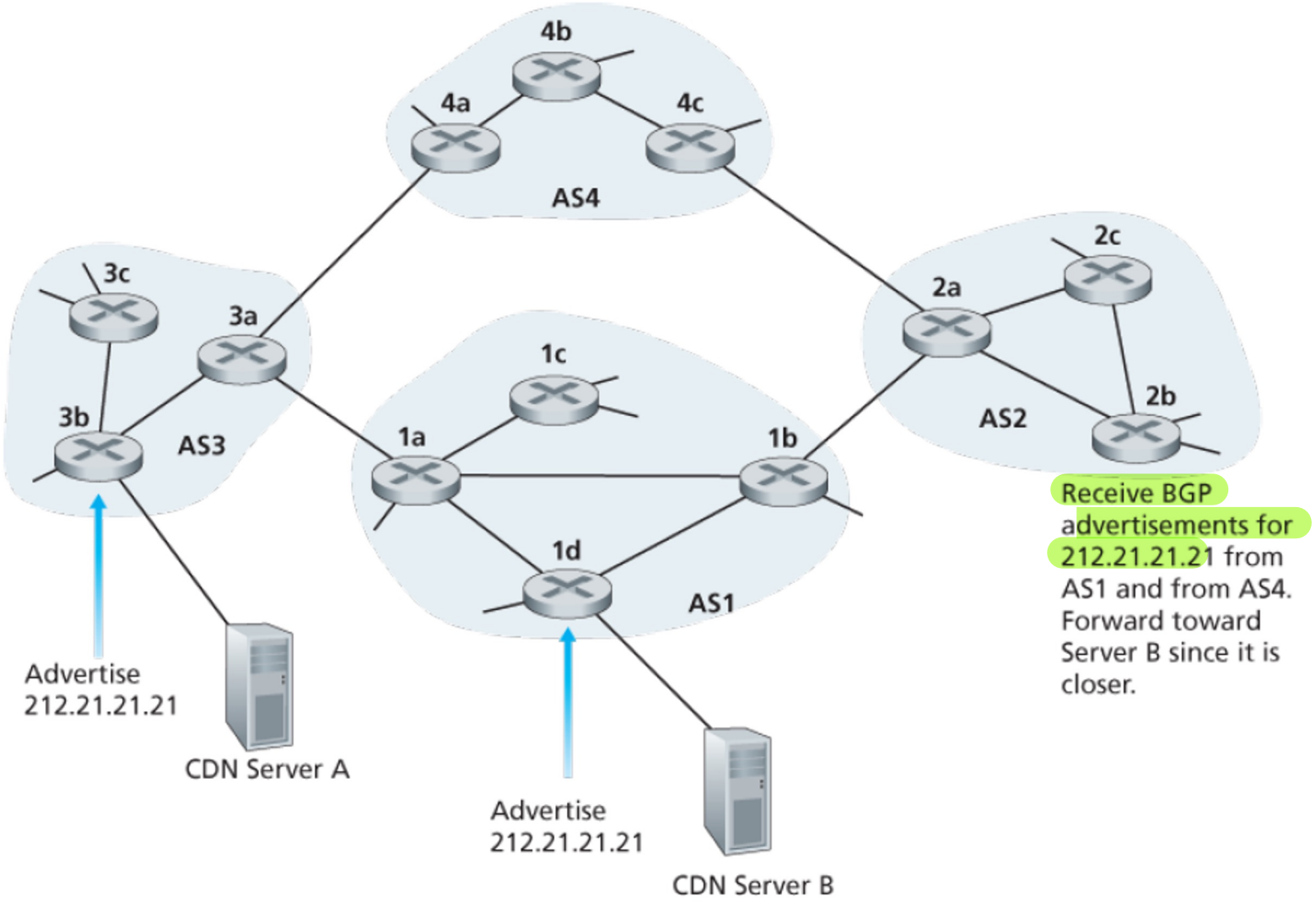
performance:

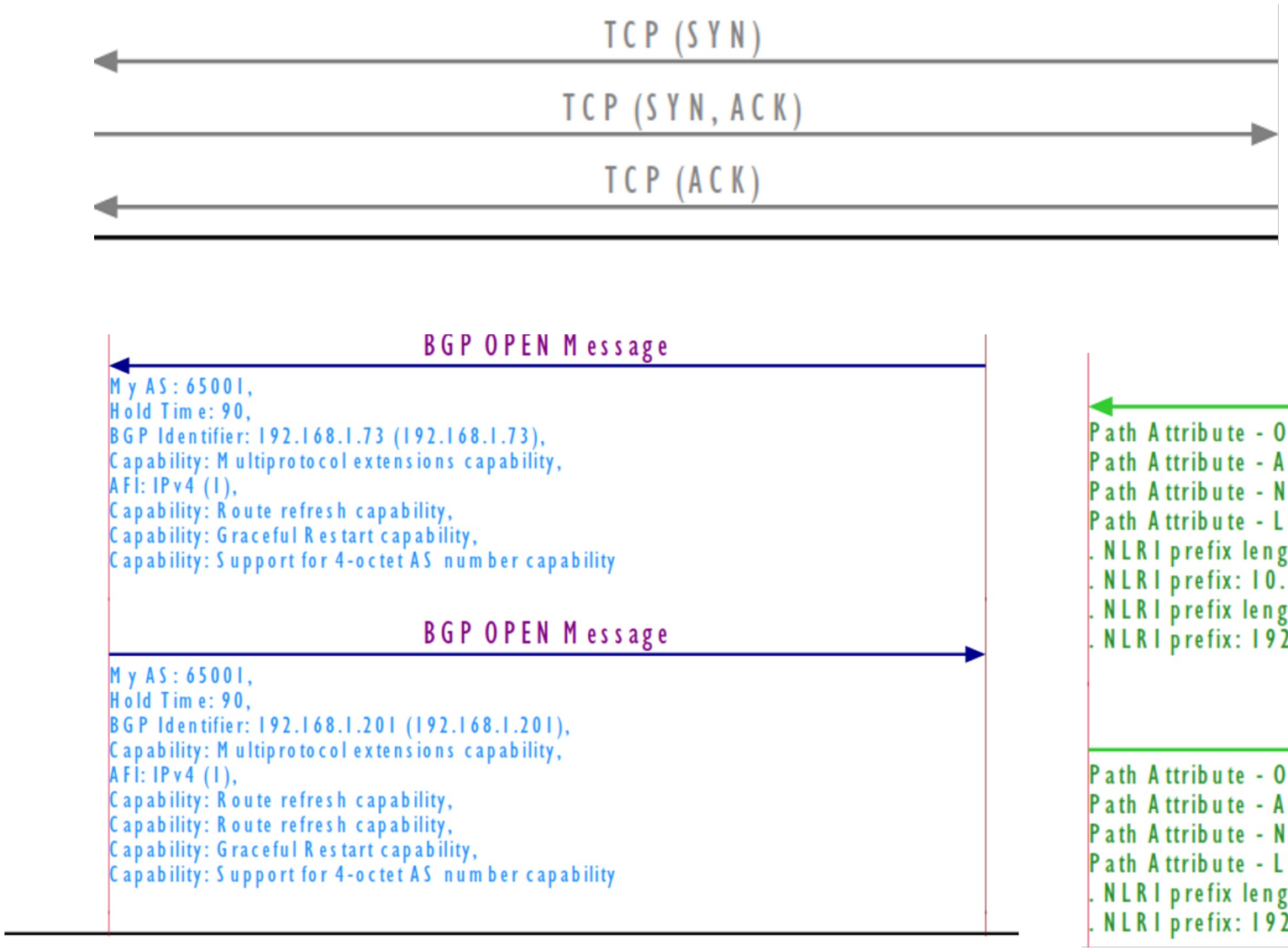
- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

BGP messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

IP AnyCast


4b
4a
4c
AS4
3c
3a
3b
AS3
1c
1a
1b
1d
AS1
2c
2a
2b
AS2
Advertise
212.21.21.21
CDN Server A
Advertise
212.21.21.21
CDN Server B
Receive BGP advertisements for 212.21.21.21 from AS1 and from AS4. Forward toward Server B since it is closer.


TCP (SYN)
TCP (SYN, ACK)
TCP (ACK)
BGP OPEN Message
My AS: 65001,
Hold Time: 90,
BGP Identifier: 192.168.1.73 (192.168.1.73),
Capability: Multiprotocol extensions capability,
AFI: IPv4 (1),
Capability: Route refresh capability,
Capability: Graceful Restart capability,
Capability: Support for 4-octet AS number capability
BGP OPEN Message
My AS: 65001,
Hold Time: 90,
BGP Identifier: 192.168.1.201 (192.168.1.201),
Capability: Multiprotocol extensions capability,
AFI: IPv4 (1),
Capability: Route refresh capability,
Capability: Route refresh capability,
Capability: Graceful Restart capability,
Capability: Support for 4-octet AS number capability
BGP UPDATE Message
Path Attribute - ORIGIN: IGP,
Path Attribute - AS_PATH: empty,
Path Attribute - NEXT_HOP: 192.168.1.73,
Path Attribute - LOCAL_PREF: 100,
. NLRI prefix length: 24.,
. NLRI prefix: 10.100.0.0,
. NLRI prefix length: 24.,
. NLRI prefix: 192.168.1.0
BGP KEEPALIVE + UPDATE Message
Path Attribute - ORIGIN: IGP,
Path Attribute - AS_PATH: empty,
Path Attribute - NEXT_HOP: 192.168.1.201,
Path Attribute - LOCAL_PREF: 100,
. NLRI prefix length: 24.,
. NLRI prefix: 192.168.1.0
TCP (ACK)