

Internet in a nutshell (protocols in practice)

The gory details first

- ❑ The Internet from your computer's view
- ❑ Packet-level traces of what happens when you access a web page

What you need to assume

- ❑ Every host has a network card with a globally unique, 48-bit hardware address typically expressed as 12 hex digits.
 - ❖ `ipconfig /all` OR `ifconfig -a`
 - ❖ This network card = 00-0E-9B-90-1C-50
- ❑ Hop-by-hop link layer communication is done via these hardware addresses.
 - ❖ Payload may have an IP packet
 - ❖ You must know the hardware address of the next hop in order to send a packet there
 - ❖ Special hardware broadcast address for discovery

What you need to assume

- ❑ Every host has a unique 32-bit IP address typically expressed as 4 numbers from 0-255
 - ❖ Portland State = 131.252.x.x
 - ❖ This machine =
 - ❖ Completely decoupled from hardware addresses
 - ❖ Structured like postal addresses.
- ❑ Every network packet has a source and a destination IP address
 - ❖ Routers collaborate to deliver packets based on their destination IP address
- ❑ DNS servers collaborate to map names (i.e. www.google.com) to IP addresses (72.14.213.103)

A day in the life of an Internet host...

□ Booting

- ❖ Dynamically configure network settings
 - DHCP request (Dynamic Host Configuration Protocol)
 - UDP (unreliable datagrams)
 - IP and data-link broadcast

Datalink broadcast header	IP broadcast 255.255.255.255	UDP header	DHCP request Host's datalink (MAC) address 00:50:7e:0d:30:20
---------------------------	---------------------------------	------------	--

- DHCP response from listening server
 - IP address for host to use
 - Netmask (i.e. 255.255.255.0) to determine who is directly connected
 - Default router
 - Local DNS server

Datalink header 00:50:7e:0d:30:20	IP of Host	UDP Header	DHCP reply Host's network settings
--------------------------------------	------------	------------	---------------------------------------

A day in the life of an Internet host...

□ Web request

<http://www.yahoo.com/index.html>

❖ Step #1: Locate DNS server

if (DNS server is directly connected) {

DNS server on local network

ARP for hardware address of IP_{DNS}

} else {

DNS server on remote network

ARP for hardware address of IP_{DefaultRouter}

}

• ARP (Address Resolution Protocol)

– IP address to hardware address mapping

– Request broadcast for all hosts on network to see

– Reply broadcast for all hosts to cache

A day in the life of an Internet host...

□ Step #2: ARP request and reply

Datalink header broadcast	ARP request: Who has MAC address of IP addr “X”? (X=next-hop router, dns server) MAC address of requestor
------------------------------	---

Datalink header MAC of requestor or broadcast addr	ARP reply: MAC address of “X” is a:b:c:d:e:f
--	--

A day in the life of an Internet host...

□ Step #3: DNS request/reply

- ❖ UDP, IP, data-link header

- ❖ DNS request to local DNS server from host

Datalink header (DNS server or next-hop router)	IP of DNS Server	UDP Header	DNS request www.yahoo.com “A” record request
---	---------------------	------------	---

- ❖ DNS reply from local DNS server to host

Datalink header (host)	IP of host	UDP Header	DNS reply www.yahoo.com is 214.11.1.1
---------------------------	------------	------------	--

A day in the life of an Internet host...

- Step #4: TCP connection establishment
 - ❖ TCP 3-way handshake (SYN, SYN-ACK, ACK)
 - ❖ Session establishment to support reliable byte stream

Datalink header (next-hop router)	IP of 216.115.105.2	TCP Header SYN
--------------------------------------	------------------------	-------------------

Datalink header (host)	IP of host	TCP Header SYN-ACK
---------------------------	------------	-----------------------

Datalink header (next-hop router)	IP of 216.115.105.2	TCP Header ACK
--------------------------------------	------------------------	-------------------

A day in the life of an Internet host...

□ Step #5: HTTP request and reply

- HTTP (application data), TCP, IP, data-link header
- HTTP request

Datalink header (next-hop router)	IP of 216.115.105.2	TCP Header	HTTP request GET /index.html HTTP/1.0
--------------------------------------	------------------------	------------	--

- HTTP reply

Datalink header (host)	IP of host	TCP Header	HTTP reply HTTP/1.0 200 OK Date: Mon, 24 Sep 2001 Content-Type: text/html <html> </html>
---------------------------	------------	------------	---

Internet applications

Application protocols

- ❑ Language spoken between a client application (i.e. web browser) and a server application (i.e. a web server)
- ❑ Describes how clients and servers communicate with each other
 - ❖ Defines types of messages exchanged, e.g., request & response messages
 - ❖ Syntax of message types: what fields in messages & how fields are delineated
 - ❖ Semantics of the fields, i.e., meaning of information in fields
 - ❖ Rules for when and how processes send & respond to messages

Must choose which transport layer

TCP service:

- ❑ *connection-oriented*: setup required between client and server processes
- ❑ *reliable transport* between sending and receiving process
- ❑ *flow control*: sender won't overwhelm receiver
- ❑ *congestion control*: throttle sender when network overloaded
- ❑ *does not provide*: timing, minimum bandwidth guarantees

UDP service:

- ❑ unreliable data transfer between sending and receiving process
- ❑ does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee

Internet apps: application, transport protocols

<u>Application</u>	<u>Application layer protocol</u>	<u>Underlying transport protocol</u>
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Vonage, Dialpad)	typically UDP

Web/HTTP

Web and HTTP

First some jargon

- ❑ **Web page** consists of **objects**
- ❑ Object can be HTML file, JPEG image, Java applet, audio file,...
- ❑ Each object is addressable by a **URL**
- ❑ Web page consists of **base HTML-file** which includes several referenced objects
- ❑ Example URL:

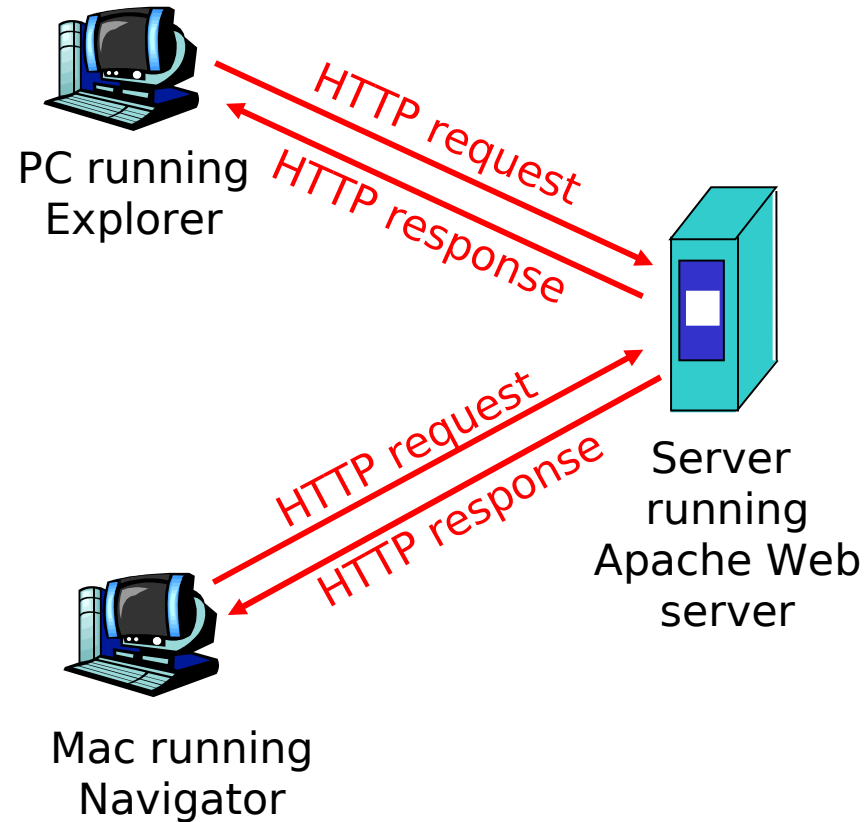
`www.someschool.edu/someDept/pic.gif`

host name

path name

HTTP overview

- ❑ HTTP: hypertext transfer protocol
- ❑ Web's application layer protocol
- ❑ client/server model
- ❑ HTTP 1.0: RFC 1945
 - ❖ <http://www.rfc-editor.org/rfc/rfc1945.tx>
- ❑ HTTP 1.1: RFC 2068
 - ❖ <http://www.rfc-editor.org/rfc/rfc2068.tx>



HTTP overview (continued)

Uses TCP:

- ❑ client initiates bi-directional TCP connection (via socket) to server, port 80
- ❑ server accepts TCP connection from client
- ❑ HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
 - ❖ Messages encoded in text
- ❑ TCP connection closed

HTTP is “stateless”

- ❑ server maintains no information about past client requests

HTTP request message

- two types of HTTP messages: *request*, *response*
- **HTTP request message:**
 - ❖ ASCII (human-readable format)

request line
(GET, POST,
HEAD commands)

header
lines

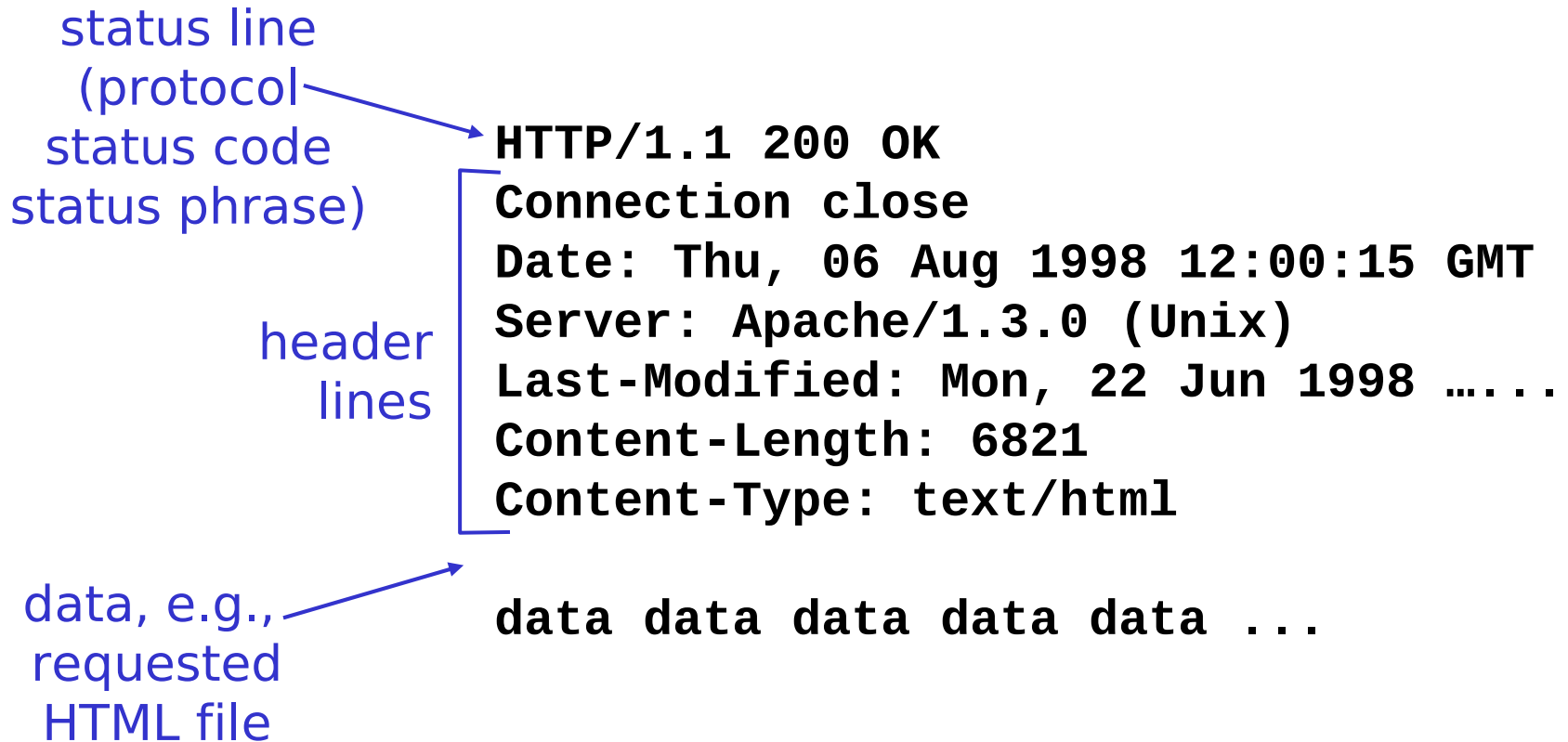
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

Carriage return,
line feed
indicates end
of message

(extra carriage return, line feed)

<http://www.someschool.edu/somedir/page.html>

HTTP response message



User-server state: cookies

HTTP initially “stateless”

- ❖ Didn't remember users or prior requests

Many major Web sites need state

- ❖ Yahoo mail
- ❖ Amazon shopping cart

HTTP state management (cookies): RFC
2109

- ❖ <http://www.rfc-editor.org/rfc/rfc2109.txt>

User-server state: cookies

Four components:

1) cookie header line of HTTP *response* message

Set-cookie:

2) cookie header line in HTTP *request* message

Cookie:

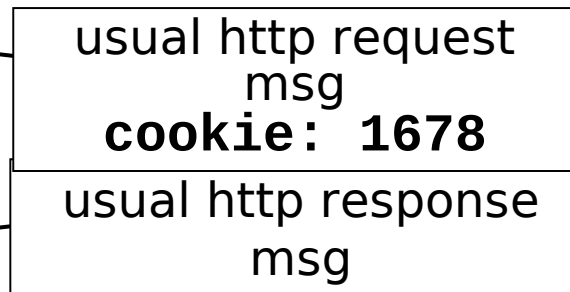
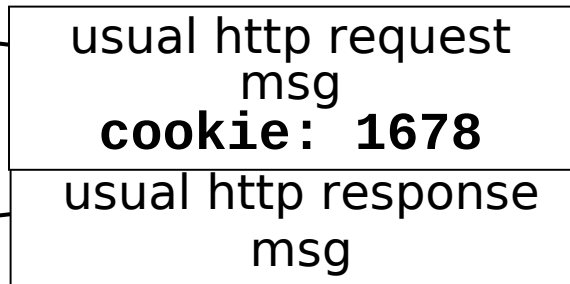
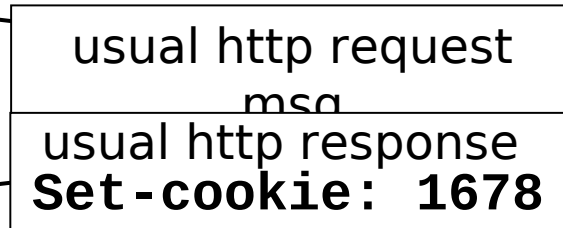
3) cookie file kept on user's host, managed by user's browser

4) back-end database at Web site

Cookies: keeping "state" (cont.)

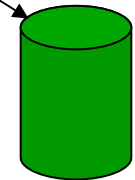
client

server



Amazon server
creates ID
1678 for user

create
entry



backend
database

access

cookie-
specific
action

access

cookie-
specific
action

Cookies (continued)

What cookies can bring:

- ❑ authorization
- ❑ shopping carts
- ❑ Site preferences
- ❑ recommendations
- ❑ user session state (Web e-mail)

Cookies and privacy: aside

- ❑ cookies permit sites to learn a lot about you
- ❑ you may supply name and e-mail to sites
- ❑ search engines use redirection & cookies to learn yet more
- ❑ advertising companies obtain info across sites

DNS

Domain Name System (DNS)

- Internet hosts, routers like to use fixed-length addresses (numbers)
 - ❖ IP address (32 bit) - used for addressing datagrams
- Humans like to use variable-length names
 - ❖ www.cs.pdx.edu
 - ❖ keywords
- DNS, keywords, naming protocols
 - ❖ Map names to numbers (IP addresses)

Original Name to Address Mapping

□ Flat namespace

- ❖ /etc/hosts.txt
- ❖ SRI kept main copy
- ❖ Downloaded regularly

□ Problems

- ❖ Count of hosts was increasing
 - From machine per domain to machine per user
 - Many more downloads of hosts.txt
 - Many more updates of hosts.txt

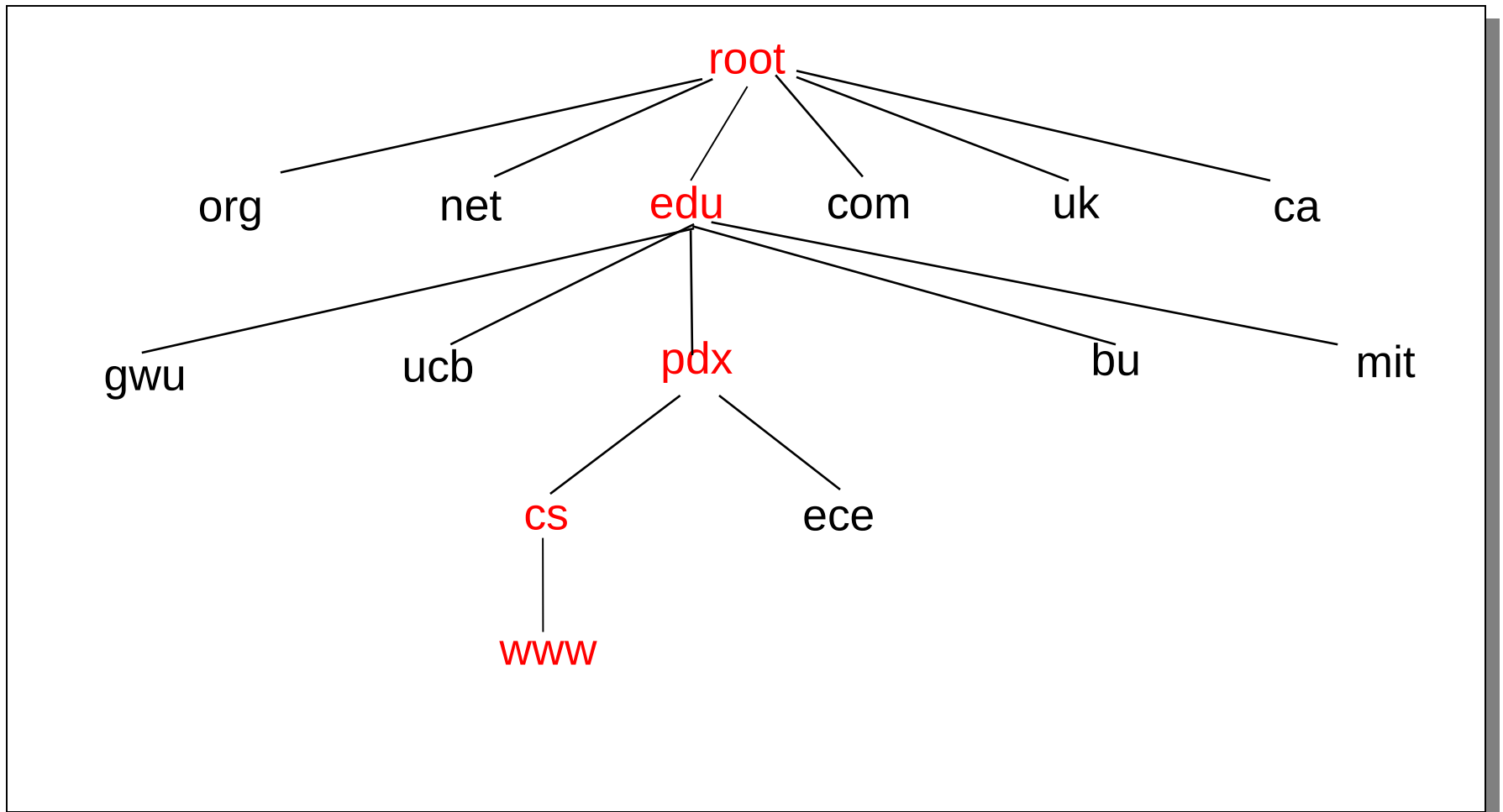
DNS: Domain Name System (1984)

- *Distributed database* implemented as a hierarchy of many *name servers*
 - ❖ Goals
 - Scalability
 - Decentralized maintenance
 - Fault-tolerance
 - Global scope
 - Names mean the same thing everywhere
 - ❖ Why not centralize DNS?
 - Not scalable, hard to maintain, single point of failure
 - ❖ <http://www.rfc-editor.org/rfc/rfc1034.txt>
 - ❖ <http://www.rfc-editor.org/rfc/rfc1035.txt>

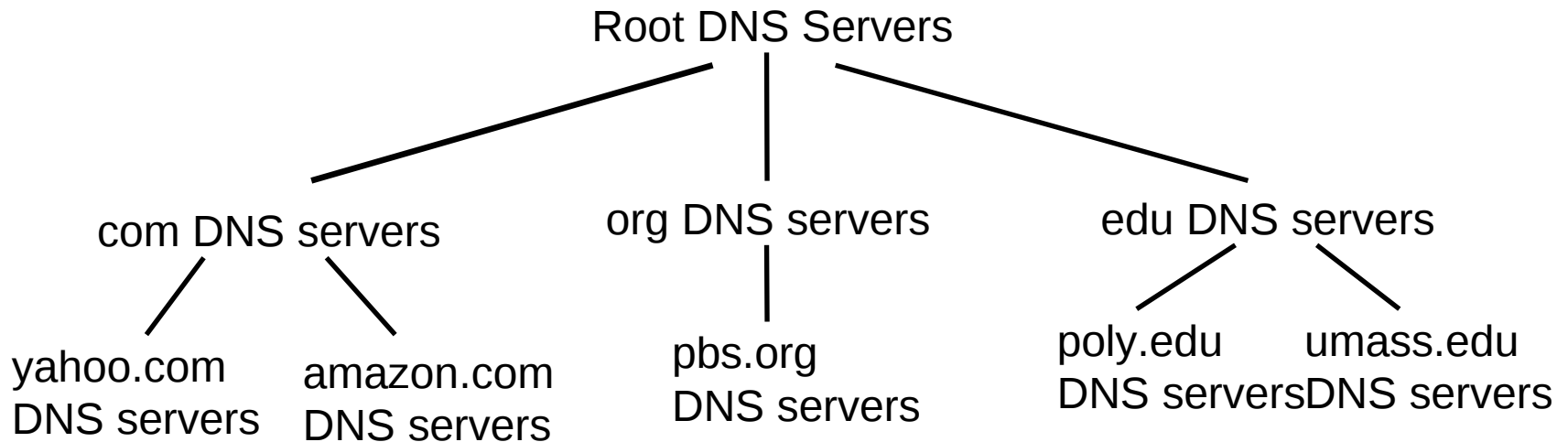
DNS: Domain Name System (1984)

- *Application-layer* protocol used by hosts and name servers
 - ❖ communicate to *resolve* names (address/name translation)
 - ❖ core Internet function, implemented as application-layer protocol
 - complexity at network's "edge"
 - compare to phone network
 - naming (none supported)
 - addressing (complex mechanism within network)

DNS hierarchical canonical name space



Namespace maps closely to name servers



What is stored at these servers?

DNS: distributed db 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**

Main parts of DNS

- ❑ Client resolver
- ❑ Local DNS servers
- ❑ Root servers
- ❑ TLD servers
- ❑ Authoritative servers

Client resolver

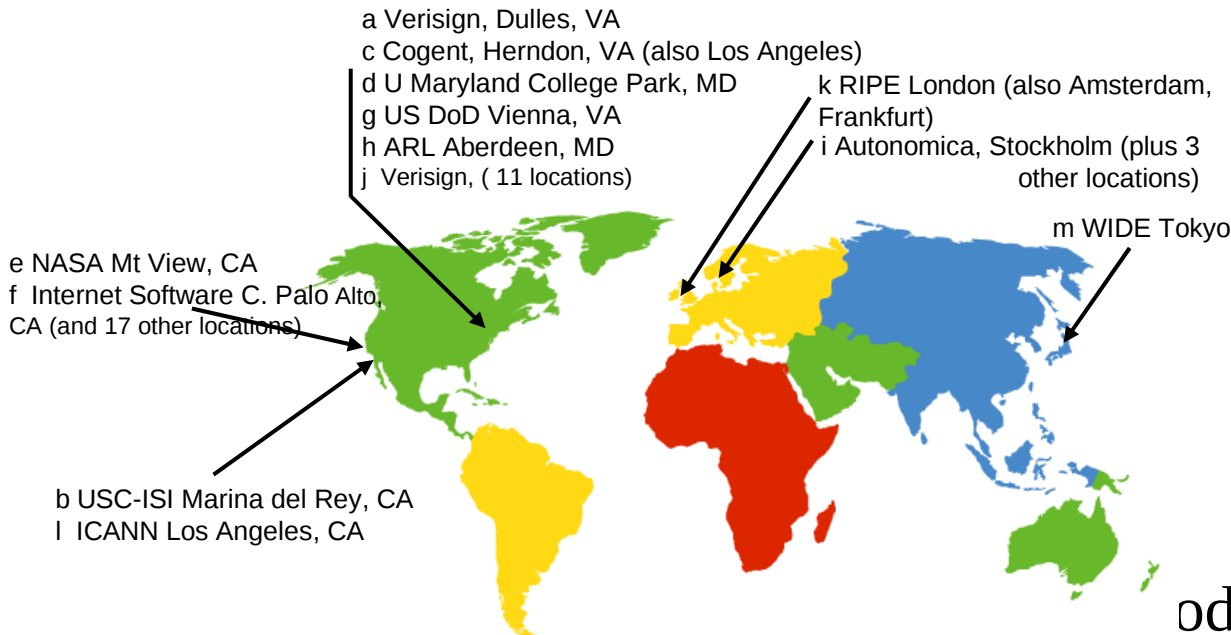
- ❑ Code on client to query DNS hierarchy
 - ❖ `gethostbyname()`
 - ❖ Resolver configuration `/etc/nsswitch.conf`
 - ❖ Local DNS name servers `/etc/resolv.conf`
 - Hand-configured or automatically configured (DHCP)
 - Host queries local name server for unknown names

Local Name Server

- ❑ Does not strictly belong to hierarchy
- ❑ Each ISP (residential ISP, company, university) has one.
 - ❖ Also called “default name server”
 - ❖ Specified in /etc/resolv.conf or given by DHCP
- ❑ When a host makes a DNS query, query is sent to its local DNS server
 - ❖ Acts as a proxy, forwards query into hierarchy.
 - ❖ Typically answer queries about local zone directly
 - ❖ Do a lookup of distant host names for local hosts
- ❑ Each local DNS server has pointers to root servers
 - ❖ Hard-coded IP addresses in all name server distributions
 - ❖ Currently {a-m}.root-servers.net

Root name servers

- ❑ contacted by local name server that can not resolve name
- ❑ root name servers
 - ❖ contacts authoritative name server or intermediate name server if name mapping not known
 - ❖ gets mapping and returns it to local name server
 - ❖ 13 root name servers worldwide for fault-tolerance
 - <http://www.root-servers.org>



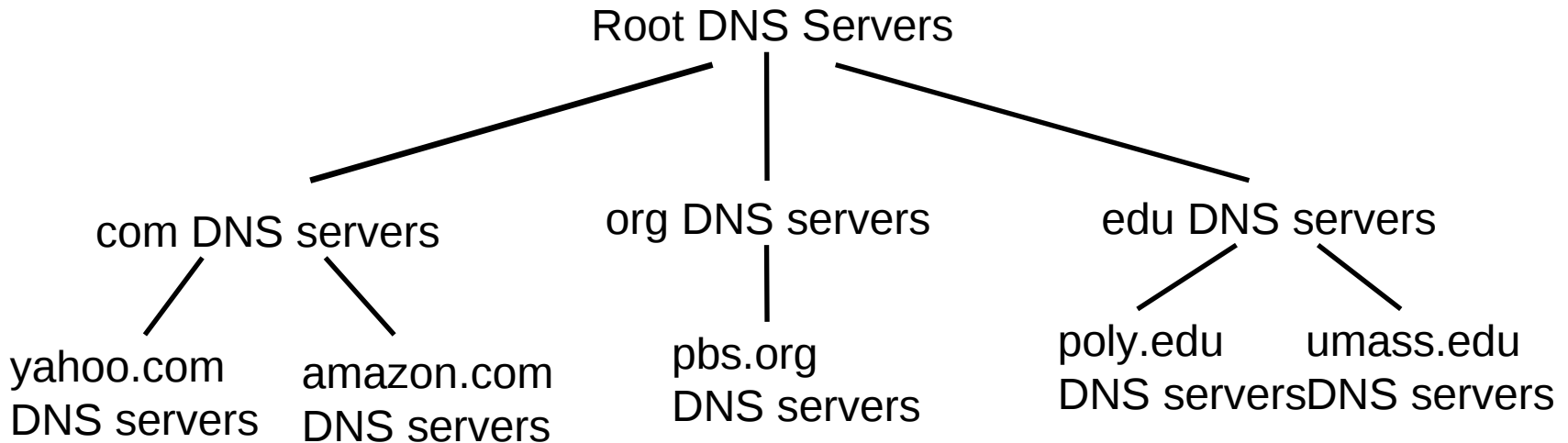
TLD Servers

- **Top-level domain (TLD) servers:**
responsible for com, org, net, edu, etc,
and all top-level country domains uk, fr,
ca, jp.
 - ❖ Network Solutions maintains servers for com
TLD
 - ❖ Educause for edu TLD

Authoritative Servers

- ❑ Provides authoritative hostname to IP mappings
 - ❖ Typically, one per organization
 - ❖ Hand mappings out for organization's servers (Web & mail).
- ❑ Store parts of the database
 - ❖ Each part of a name is assigned to an authoritative server
 - ❖ Server responds to all queries for name it is the authority
 - ❖ Can be maintained by organization or service provider
 - ❖ Example
 - Authority for .edu is a root server
 - Authority for pdx.edu is the ".edu" TLD server
 - Authority for www.pdx.edu is dns0.pdx.edu (131.252.120.128)

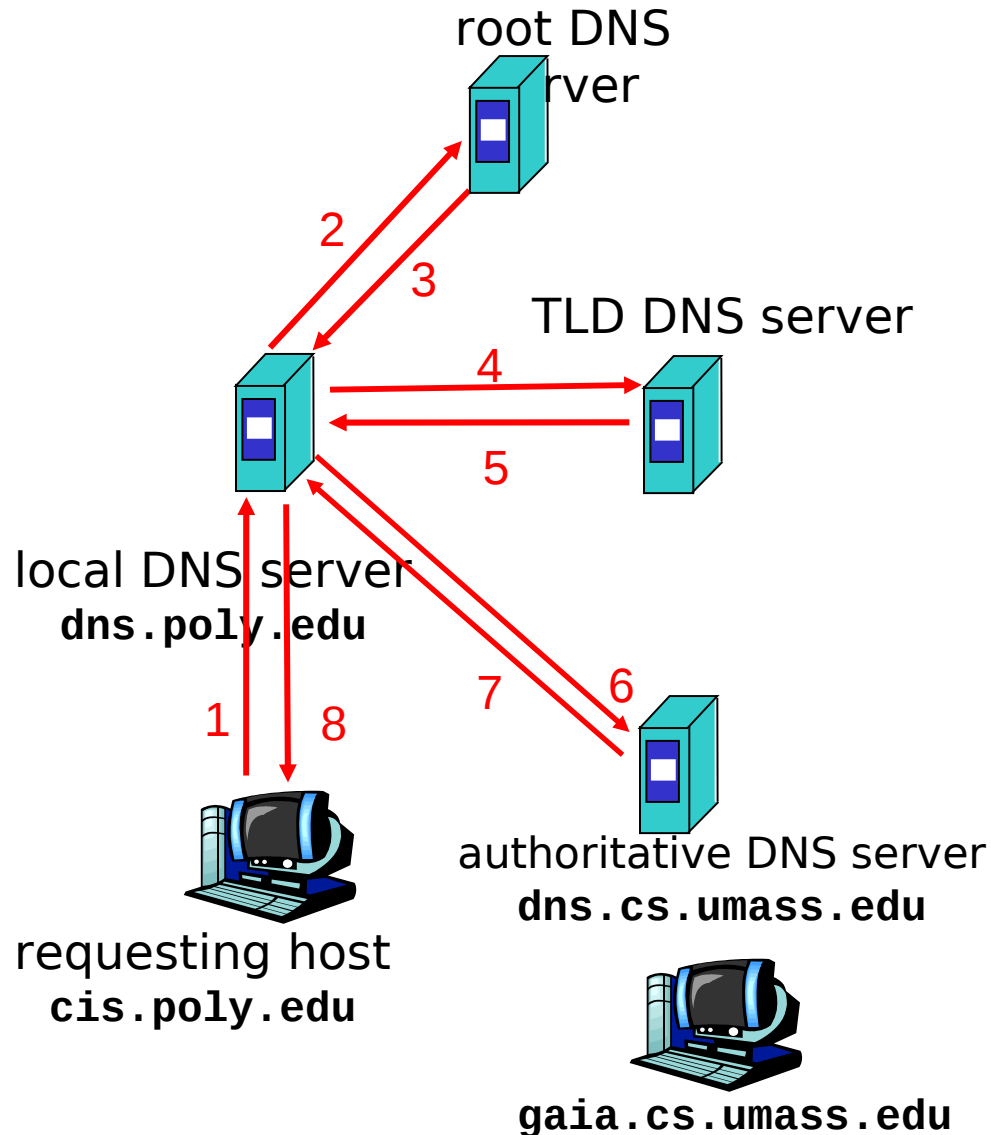
DNS in action



Client wants IP for www.amazon.com

- ❑ client queries local DNS server for www.amazon.com
- ❑ local DNS server queries a root server to find a com DNS server
- ❑ local DNS server queries com DNS server to get amazon.com DNS server
- ❑ local DNS server queries amazon.com DNS server to get IP address for www.amazon.com
- ❑ local DNS server returns IP address to client

DNS query example



- Host at `cis.poly.edu` wants IP address for `gaia.cs.umass.edu`

Creating your own site

- ❑ Example: just created startup “Network Utopia”
- ❑ Register name networkutopia.com at a registrar (e.g., Network Solutions)
 - ❖ Give registrar names and IP addresses of your authoritative name server
 - ❖ Registrar inserts two RRs into the com TLD server:
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
- ❑ Set up authoritative server (212.212.212.1)
 - ❖ Install DNS server (BIND)
 - ❖ Enter A record for www.networkutopia.com
 - ❖ Enter MX record for networkutopia.com

DNS issues

- UDP used for queries
 - ❖ Need reliability -> Why not TCP?
 - ❖ No rate control
- Centralized caching per site not required
- Vulnerability of 13 static root servers
 - ❖ Attacks on root servers have occurred
 - ❖ Jon Postel and his mobility “experiment”
- Spoofing identity
 - ❖ Adversary on the same network returning a bogus answer

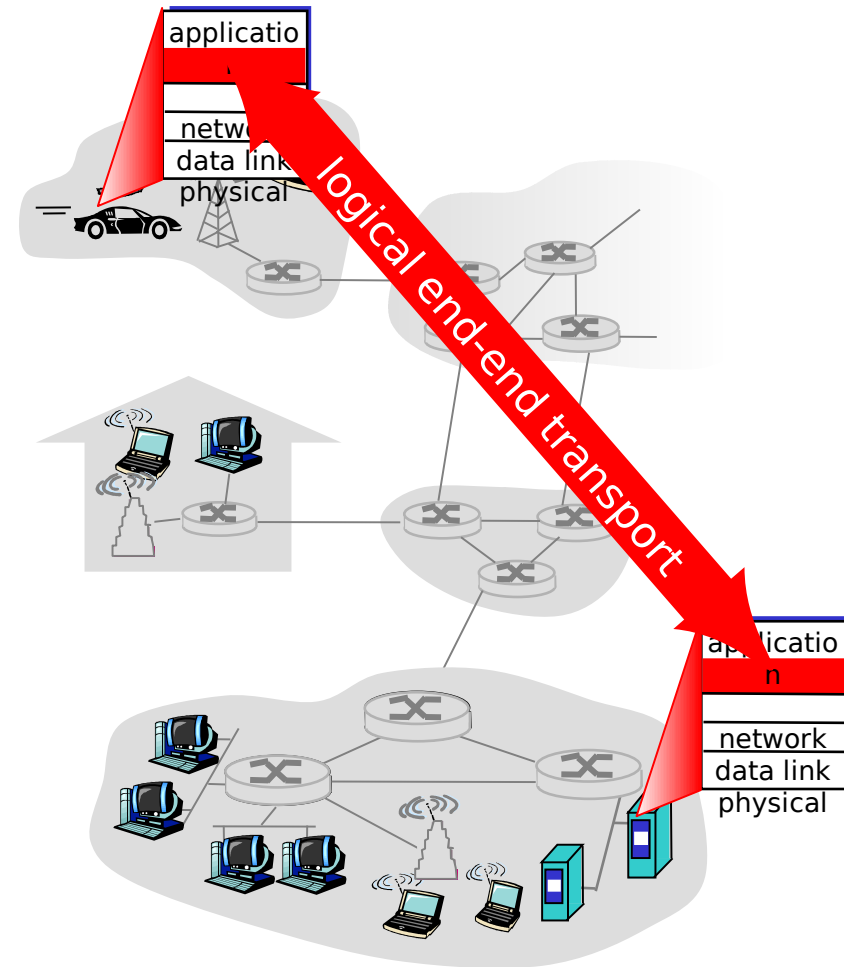
Transport protocols

Transport vs. network layer

- ❑ *network layer*: logical communication between hosts
- ❑ *transport layer*: logical communication between processes on hosts

Transport services and protocols

- Run in end systems
 - ❖ send side: breaks app messages into **segments**, passes to network layer
 - ❖ rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
 - ❖ Internet: TCP and UDP



Transport Layer Functions

- ❑ Demux to upper layer
 - ❖ Delivering data to correct application process
- ❑ Connection setup
 - ❖ Providing a connection abstraction over a connectionless substrate
- ❑ Delivery semantics
 - ❖ Reliable or unreliable
 - ❖ Ordered or unordered
 - ❖ Unicast, multicast, anycast
- ❑ Security
- ❑ Flow control
 - ❖ Prevent overflow of receiver buffers
- ❑ Congestion control
 - ❖ Prevent overflow of network buffers
 - ❖ Avoid packet loss and packet delay

UDP's implementation of transport layer functions

- ❑ Demux to upper layer
 - ❖ UDP port field
- ❑ Connection setup
 - ❖ none
- ❑ Delivery semantics
 - ❖ Unordered, mostly unicast (multicast no longer supported)
 - ❖ Unreliable, but data integrity provided by checksum
- ❑ Security
 - ❖ none
- ❑ Flow control
 - ❖ none
- ❑ Congestion control
 - ❖ none

UDP: User Datagram Protocol [RFC 768]

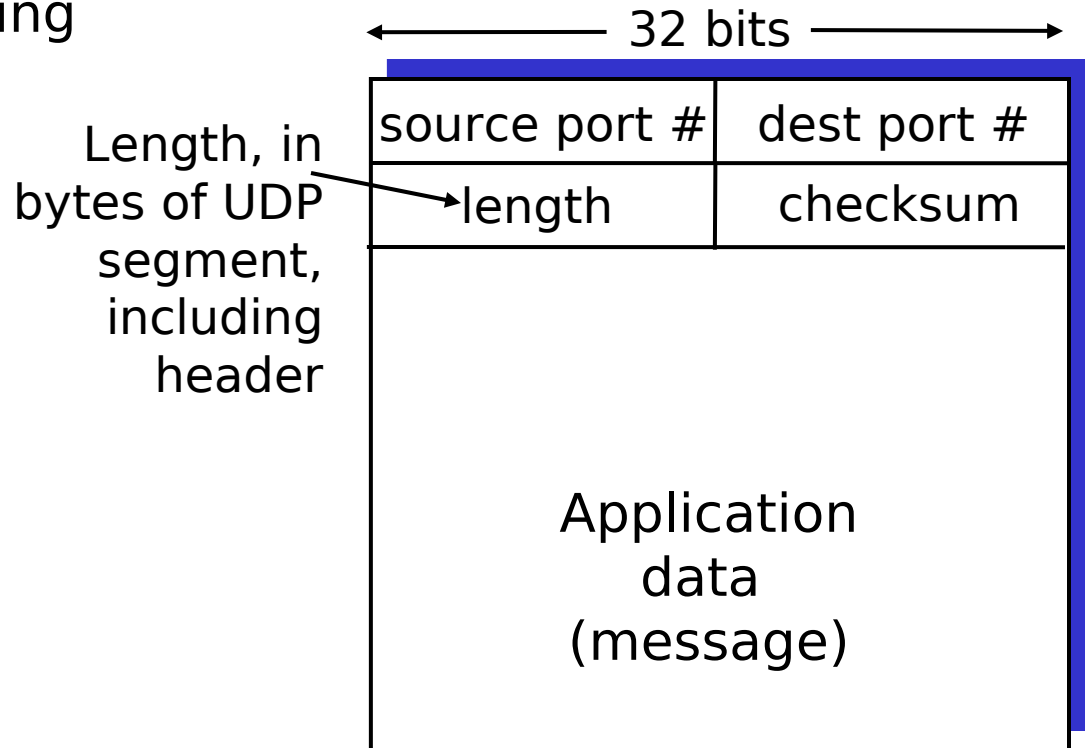
- ❑ “no frills,” “bare bones” Internet transport protocol
- ❑ “best effort” service, UDP segments may be:
 - ❖ lost
 - ❖ delivered out of order to app
- ❑ *connectionless*:
 - ❖ no handshaking between UDP sender, receiver
 - ❖ each UDP segment handled independently of others

Why is there a UDP?

- ❑ no connection establishment (which can add delay)
- ❑ simple: no connection state at sender, receiver
- ❑ small segment header
- ❑ no congestion control: UDP can blast away as fast as desired

UDP: more

- often used for streaming multimedia apps
 - ❖ loss tolerant
 - ❖ rate sensitive
- other UDP uses
 - ❖ DNS
 - ❖ SNMP



UDP segment format

TCP's implementation of transport layer functions

- ❑ Demux to upper layer
 - ❖ TCP port field
- ❑ Connection setup
 - ❖ 3-way handshake
- ❑ Delivery semantics
 - ❖ In-order byte-stream, unicast
 - ❖ Data integrity provided via 32-bit checksum
- ❑ Security
 - ❖ None, added later via SSL and TLS
- ❑ Flow control
 - ❖ Receiver advertised window
- ❑ Congestion control
 - ❖ Window-based

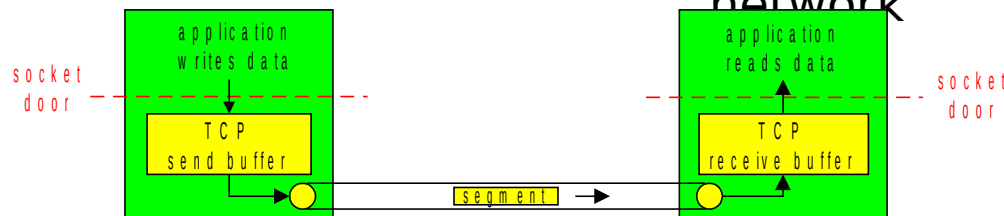
TCP: Overview

2581

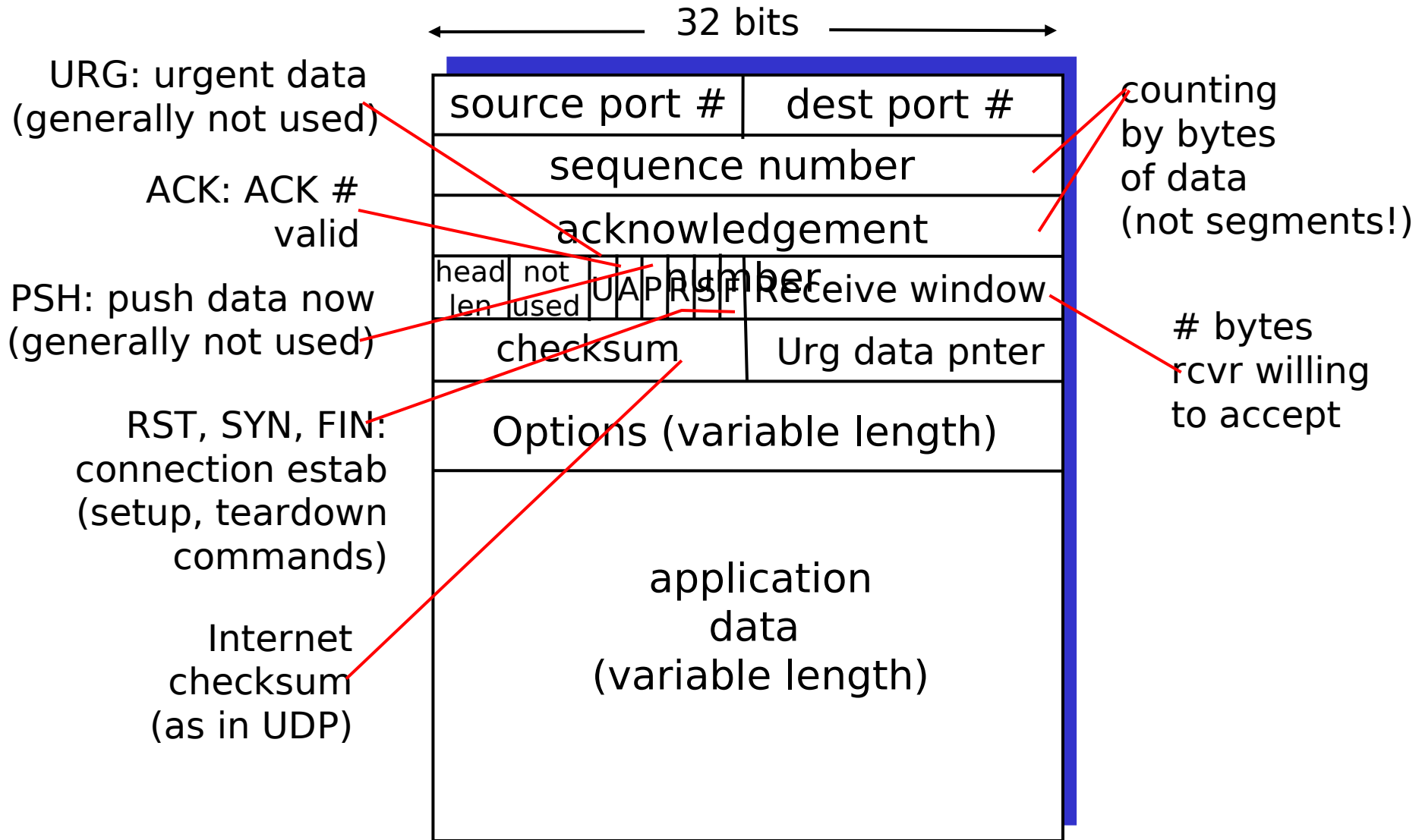
RFCs: 793, 1122, 1323, 2018,

- ❑ **point-to-point:**
 - ❖ one sender, one receiver
- ❑ **connection-oriented:**
 - ❖ handshaking to initialize sender/receiver
 - ❖ connection integrity
- ❑ **reliable, in-order *byte stream*:**
 - ❖ Error detection, correction
 - ❖ Duplicate detection
 - ❖ Retransmission

- ❑ **full duplex:**
 - ❖ bi-directional data flow in same connection
 - ❖ MSS: maximum segment size
- ❑ **pipelined:**
 - ❖ Support high bandwidth
 - ❖ H&H Bagels example
- ❑ **flow and congestion controlled:**
 - ❖ control the size of pipe
 - ❖ sender will not overwhelm receiver or network



TCP segment structure



TCP

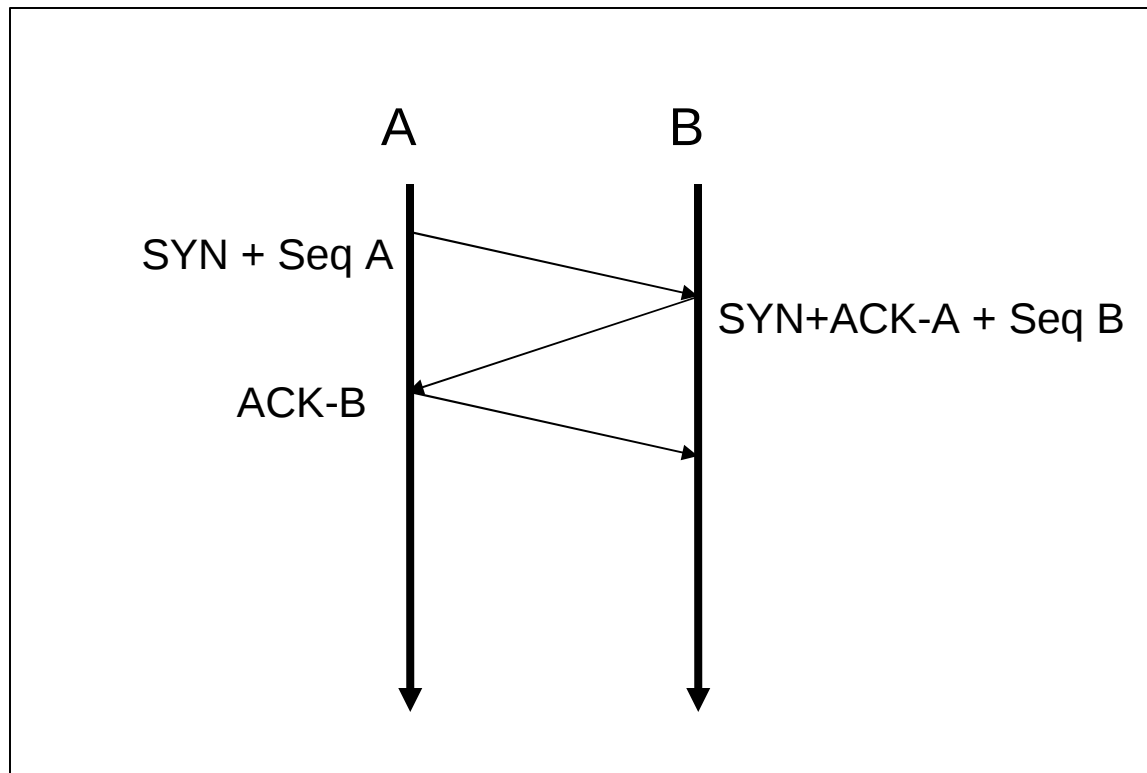
- TCP creates a reliable data transfer service on top of IP's unreliable service via
 - ❖ Checksum
 - ❖ Sequence numbers
 - ❖ Acknowledgments
 - ❖ Retransmissions
 - ❖ Rate limits on sender

Sequence numbers

- Data packet in each packet is labeled with a unique* number
 - ❖ Establishes ordering amongst packets
 - ❖ Allows receiver to identify which packets have been received and which have not
 - ❖ Prevents adversary from injecting bogus data into the connection
 - If initial sequence number is random
 - ❖ Initialized during connection setup (i.e. 3-way handshake)

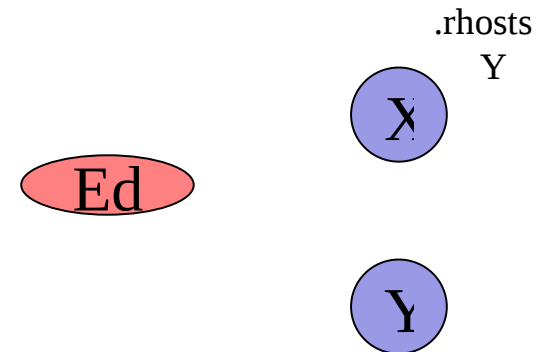
Sequence numbers

- 3-way handshake with initial sequence number selection



Sequence Numbers

- Why is selecting a random initial sequence number important?
 - ❖ Predictable ISNs allow adversary to blindly “spoof” connections from “trusted” hosts
 - Hijack TCP connections
 - Reset existing TCP connections
 - Create new connections as someone else
 - Attack popularized by K. Mitnick
 - X trusts Y
 - Logins from Y are accepted without credential check
 - Predictable ISN of X allows Evil Ed to impersonate Y and access X without credential check



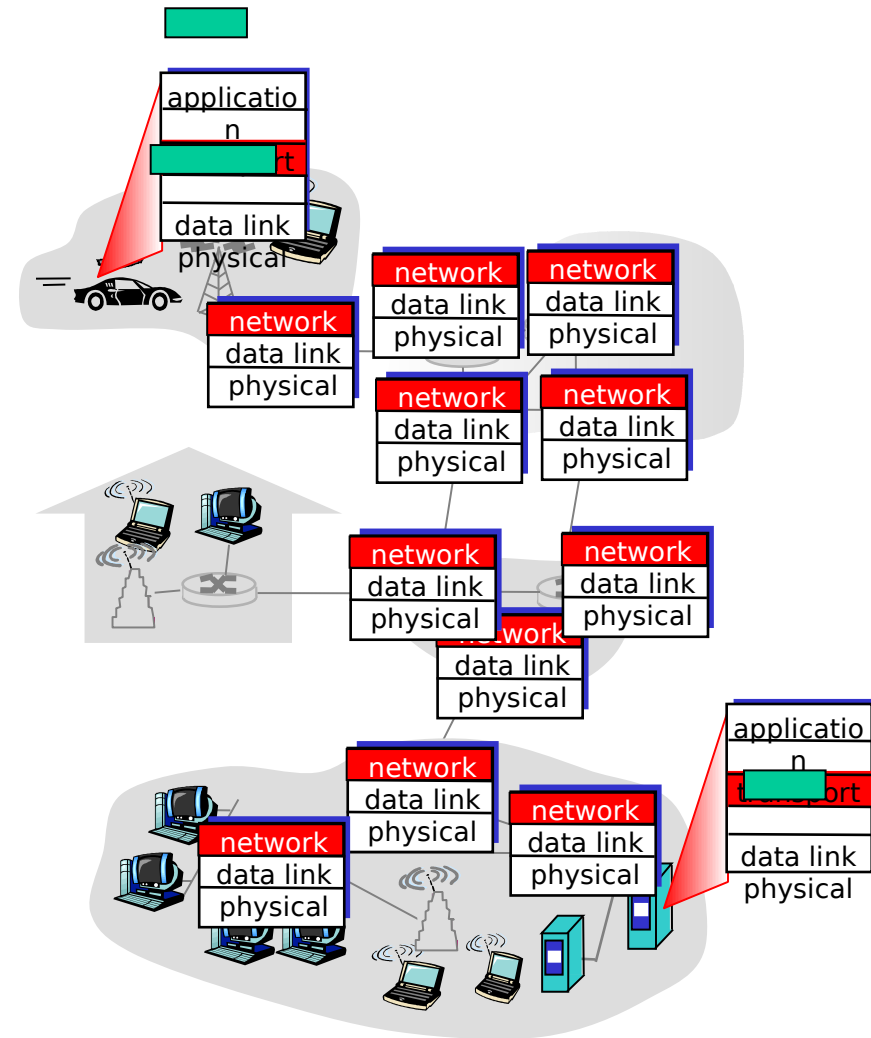
TCP Congestion Control

- Mainly end-host, window-based congestion control
 - ❖ Only place to really prevent collapse is at end-host
 - ❖ Reduce sender window when congestion is perceived
 - ❖ Increase sender window otherwise (probe for bandwidth)

Network layer

Network layer

- ❑ transport segment from sending to receiving host
- ❑ on sending side puts segments into datagrams
- ❑ on rcving side, delivers segments to transport layer
- ❑ network layer protocols in every host, router

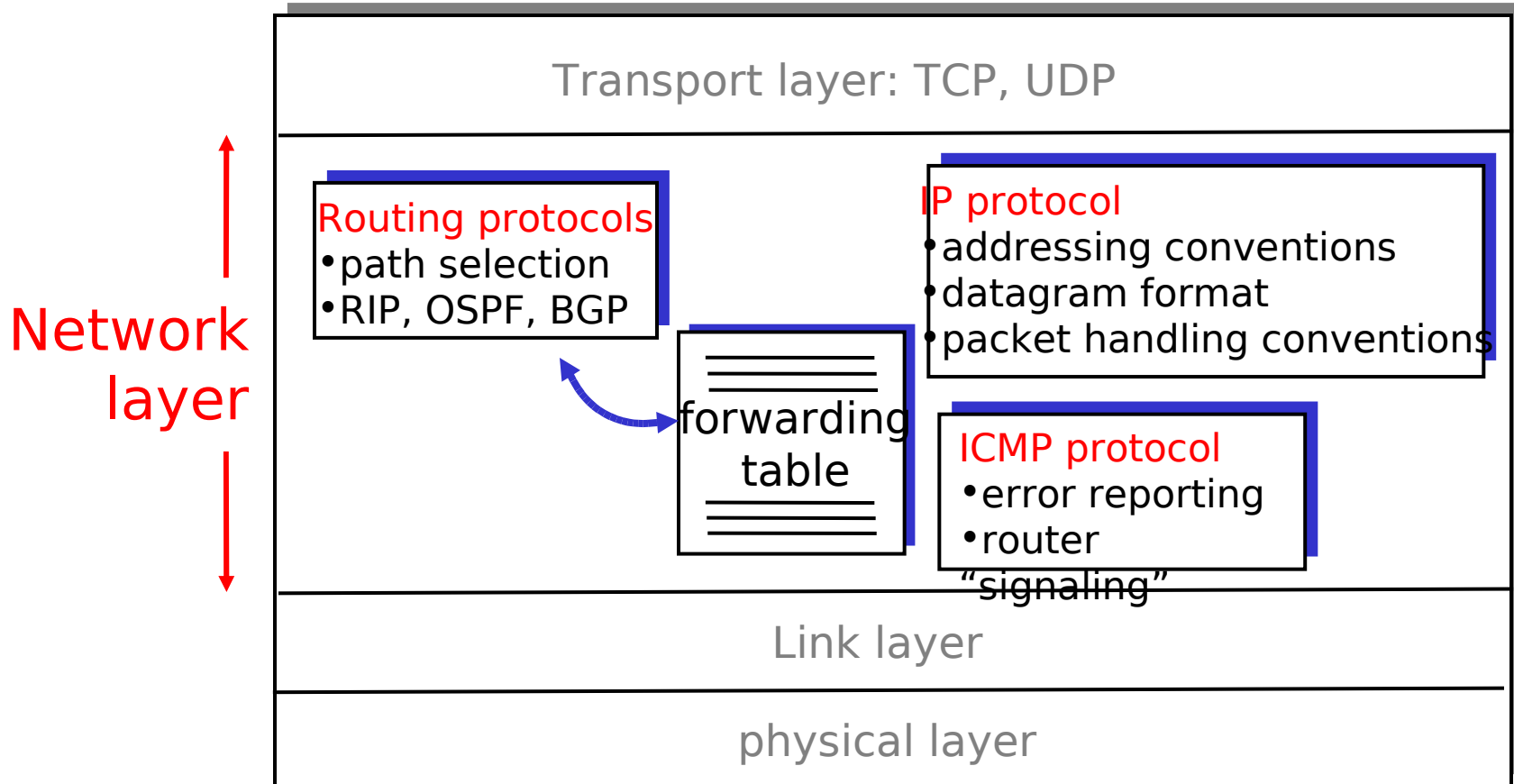


Network layer functions

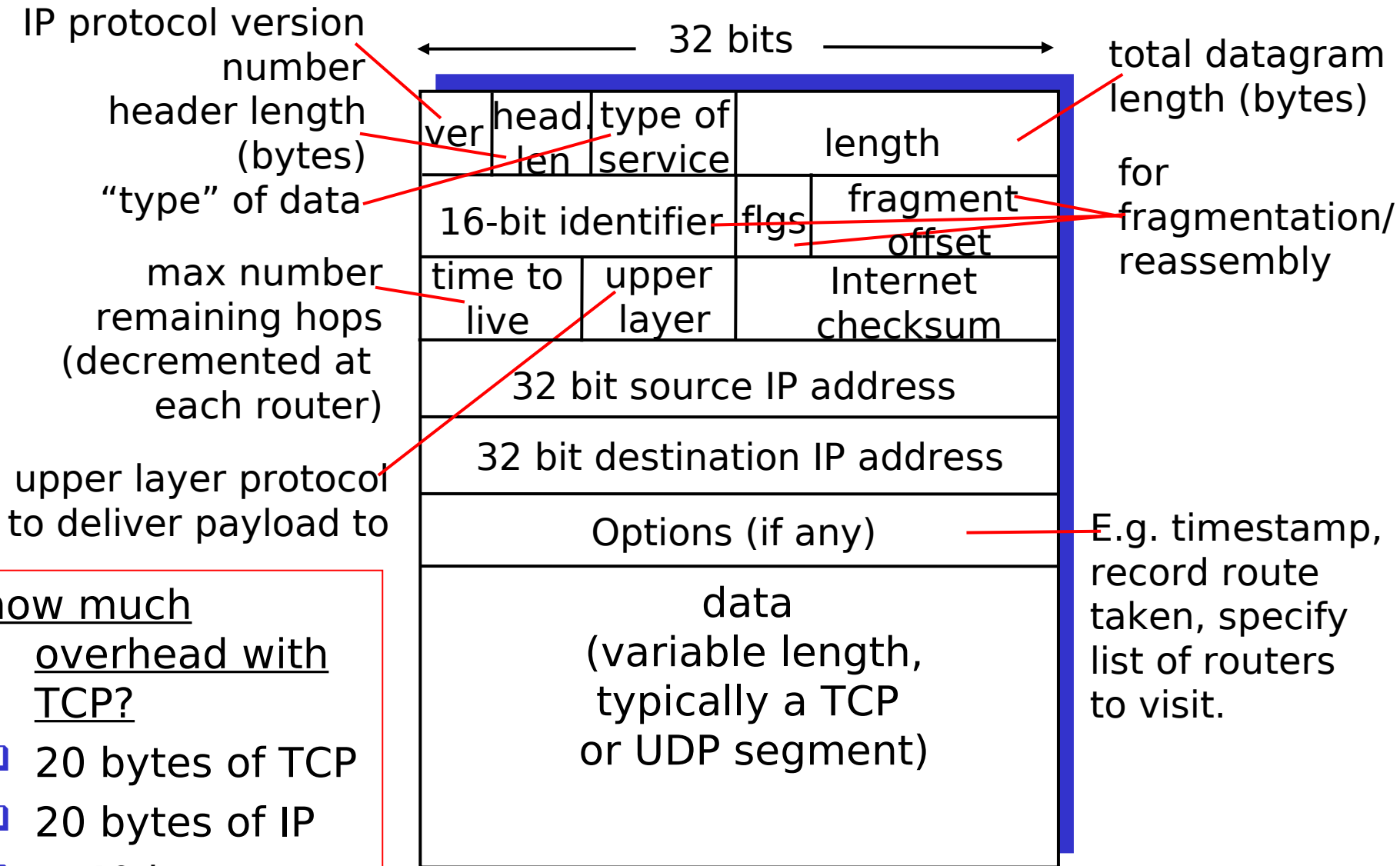
- ❑ Connection setup
 - ❖ datagram
 - ❖ connection-oriented, host-to-host connection
- ❑ Delivery semantics:
 - ❖ Unicast, broadcast, multicast, anycast
 - ❖ In-order, any-order
- ❑ Security
 - ❖ secrecy, integrity, authenticity
- ❑ Demux to upper layer
 - ❖ next protocol
 - ❖ Can be either transport or network (tunneling)
- ❑ Quality-of-service
 - ❖ provide predictable performance
- ❑ Fragmentation
 - ❖ break-up packets based on data-link layer properties
- ❑ Routing
 - ❖ path selection and packet forwarding
- ❑ Addressing
 - ❖ flat vs. hierarchical
 - ❖ global vs. local
 - ❖ variable vs. fixed length

The Internet Network layer

Host, router network layer functions:



IP datagram format



how much

overhead with TCP?

- ❑ 20 bytes of TCP
- ❑ 20 bytes of IP
- ❑ = 40 bytes + app layer

Recall network layer functions

- How does IPv4 support..
 - ❖ Connection setup
 - ❖ Delivery semantics
 - ❖ Security
 - ❖ Demux to upper layer
 - ❖ Quality-of-service
 - ❖ Fragmentation
 - ❖ Addressing
 - ❖ Routing

IP connection setup

- Hourglass design
- No support for network layer connections
 - ❖ Unreliable datagram service
 - ❖ Out-of-order delivery possible
 - ❖ Connection semantics only at higher layer
 - ❖ Compare to ATM and phone network...

IP delivery semantics

- ❑ No reliability guarantees
 - ❖ Loss
- ❑ No ordering guarantees
 - ❖ Out-of-order delivery possible
- ❑ Unicast mostly
 - ❖ IP broadcast (255.255.255.255) not forwarded
 - ❖ IP multicast supported, but not widely used
 - 224.0.0.0 to 239.255.255.255

IP security

❑ Weak support for integrity

❖ IP checksum

- IP has a header checksum, leaves data integrity to TCP/UDP

- <http://www.rfc-editor.org/rfc/rfc1141.txt>

❖ No support for secrecy, authenticity

❑ IPsec

❖ Retrofit IP network layer with encryption and authentication

- ❖ <http://www.rfc-editor.org/rfc/rfc2411.txt>

IP demux to upper layer

□ <http://www.rfc-editor.org/rfc/rfc1700.txt>

❖ Protocol type field

- 1 = ICMP
- 4 = IP in IP
- 6 = TCP
- 8 = EGP
- 9 = IGP
- 17 = UDP

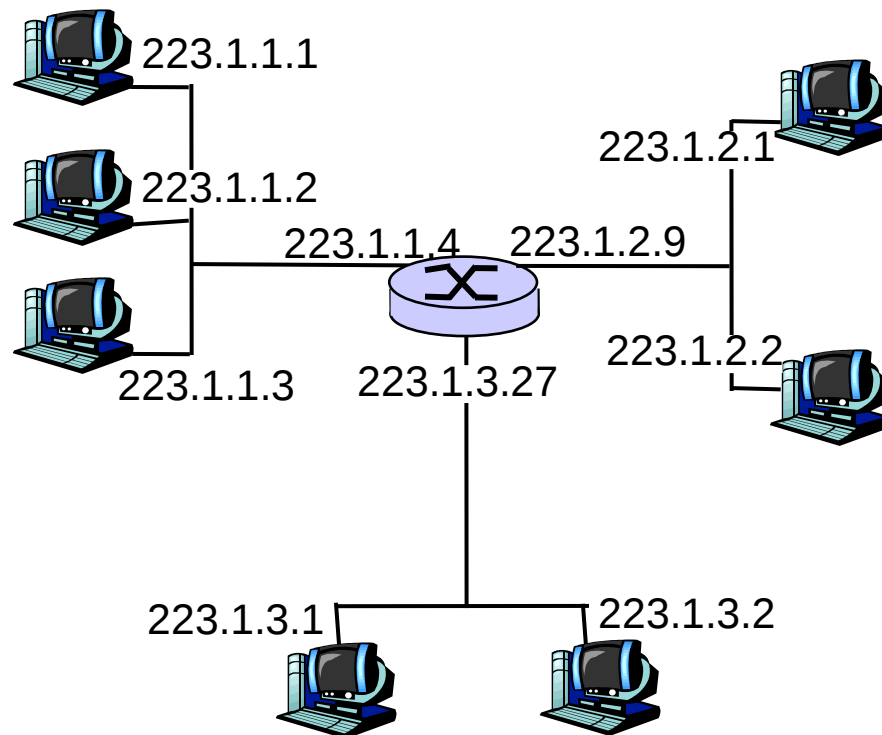
IP quality of service

- ❑ IP originally had “type-of-service” (TOS) field to eventually support quality
 - ❖ Not used, ignored by most routers
 - ❖ Redefined into DiffServ fields (Drop precedence, Low delay)
 - ❖ Used in network management

IP Addressing

□ IP address:

- ❖ 32-bit identifier for host/router *interface*
- ❖ routers typically have multiple interfaces
- ❖ Addresses hierarchical (like post office)



223.1.1.1 = $\underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$

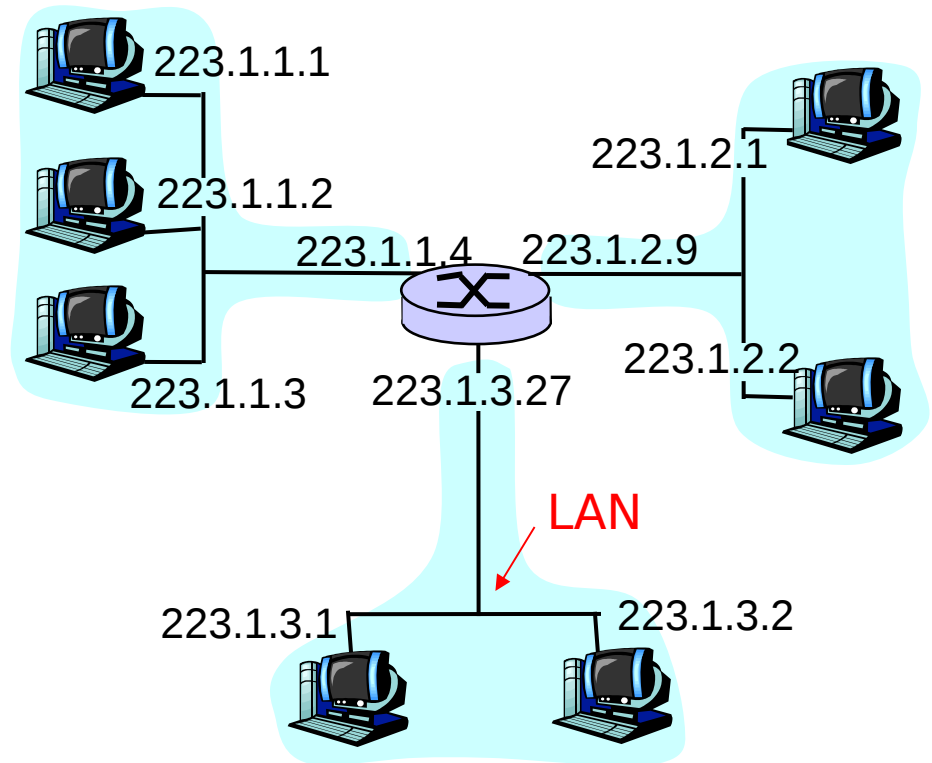
IP Addressing

□ IP address:

- ❖ network part (high order bits)
- ❖ host part (low order bits)

□ *What's a network ?*

- ❖ all interfaces that can physically reach each other without intervening router
- ❖ each interface shares the same network part of IP address

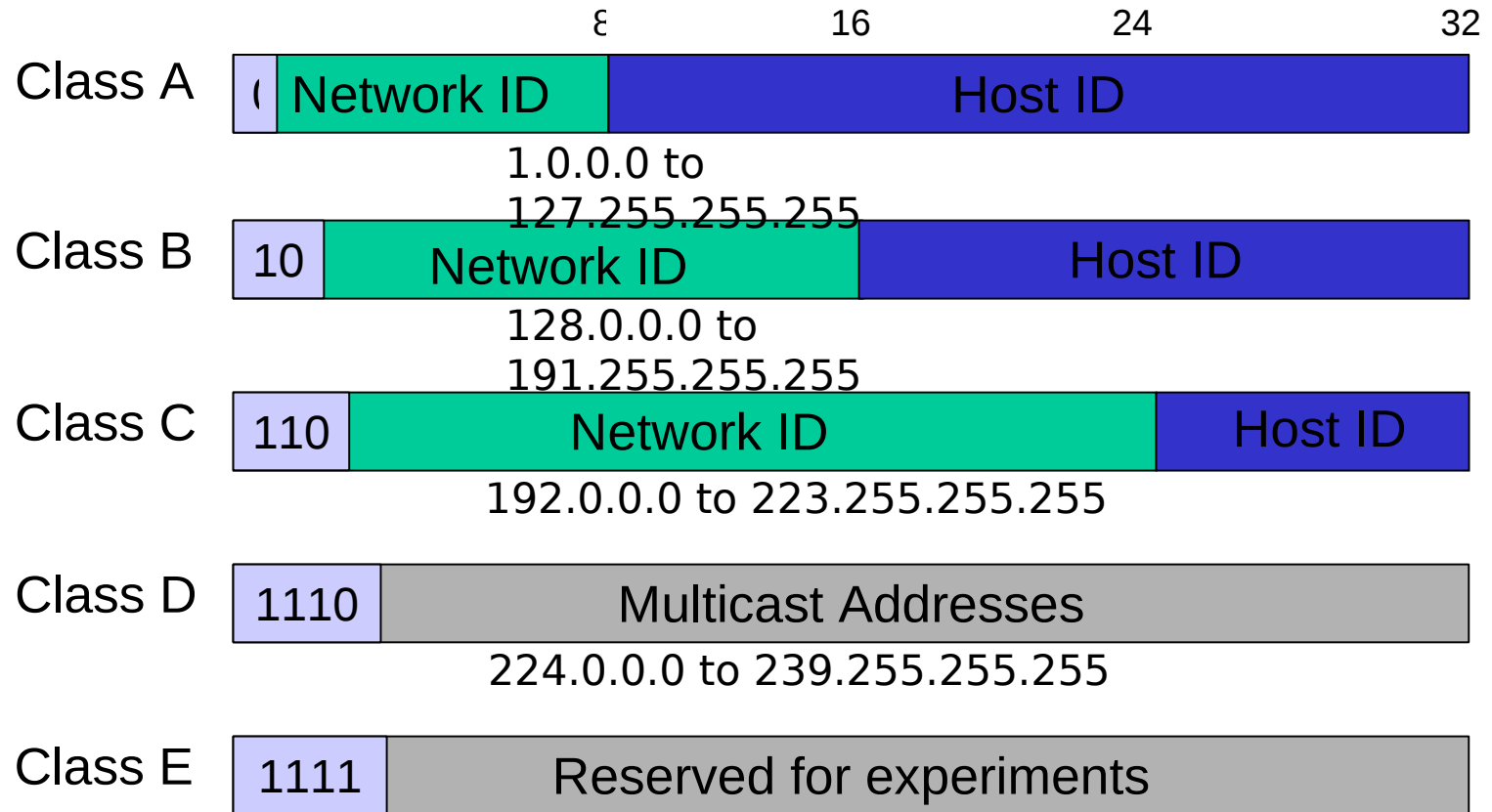


network consisting of 3 IP networks
(for IP addresses starting with 223,
first 24 bits are network address)

How did networks get IP addresses?

- ❑ Total IP address size: 4 billion
- ❑ Initially one large class (8-bit network, 24-bit host)
 - ❖ ISP given an 8-bit network number to manage
 - ❖ Each router keeps track of each network (28=256 routes)
 - ❖ Each network has 16 million hosts
 - ❖ Problem: one size does not fit all
- ❑ Classful addressing
 - ❖ Accommodate smaller networks (LANs)
 - ❖ Class A: 128 networks, 16M hosts
 - ❖ Class B: 16K networks, 64K hosts
 - ❖ Class C: 2M networks, 256 hosts
 - ❖ Total routes potentially > 2,113,664 networks and network routes !

IP address classes



Special IP Addresses

❑ Private addresses

- <http://www.rfc-editor.org/rfc/rfc1918.txt>
- Class A: 10.0.0.0 - 10.255.255.255 (10.0.0.0/8 prefix)
- Class B: 172.16.0.0 - 172.31.255.255 (172.16.0.0/12 prefix)
- Class C: 192.168.0.0 - 192.168.255.255 (192.168.0.0/16 prefix)

❑ 127.0.0.1: local host (a.k.a. the loopback address)

IP Addressing problems

- ❑ Inefficient use of address space
 - ❖ Class A (rarely given out, sparse usage)
 - ❖ Class B = 64k hosts (sparse usage)
 - Very few LANs have close to 64K hosts
- ❑ Address space depletion
 - ❖ Classes A and B take huge chunks of space but not used much
 - ❖ Not many class C addresses left to give out
- ❑ Explosion of routes
 - ❖ Increasing use of class C explodes # of routes

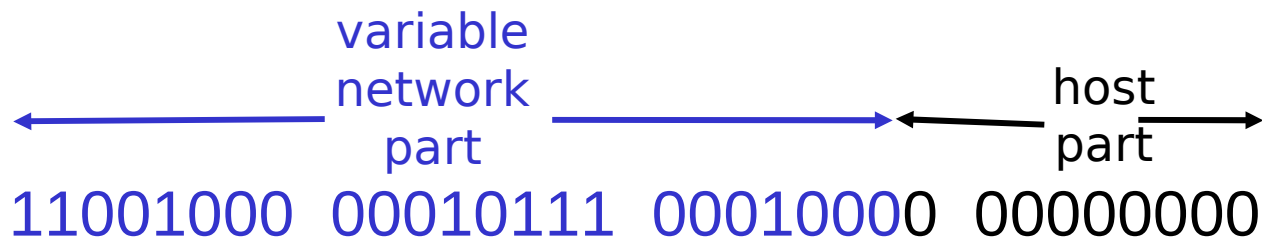
IP addressing: CIDR

□ Original classful addressing

- ❖ Use class structure (A, B, C) to determine network ID for route lookup

□ CIDR: Classless InterDomain Routing

- ❖ Arbitrarily aggregate and split up adjacent network addresses



200.23.16.0/23

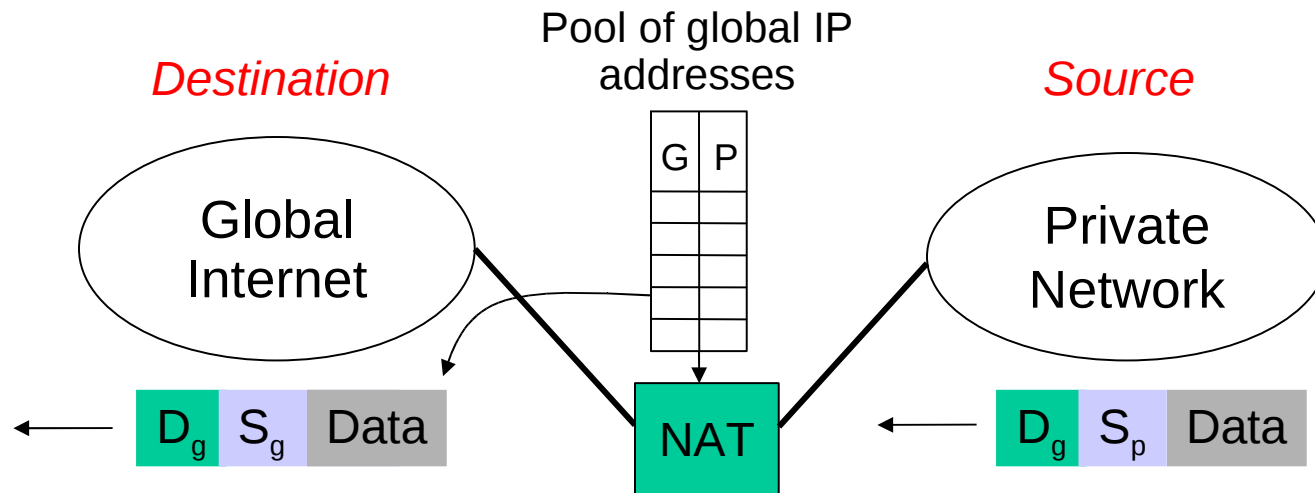
CIDR

- Assign any range of addresses to network
 - ❖ Allows one to split large network blocks into multiple smaller ones (increase usage of Class A & B)
 - ❖ Allows one to combine small network blocks into a single large one (reduce routes from Class C usage)

IP Address depletion

- ❑ Even with CIDR, address space running out
 - ❖ IPv6 still being developed, a long way from being deployed
- ❑ Network Address Translation (NAT)
 - ❖ Alternate solution to address space depletion problem
 - Kludge (but useful)
 - ❖ Sits between your network and the Internet
 - ❖ Dynamically assign source address from a pool of available addresses
 - “Statistically multiplex” address usage
 - Each machine gets unique, external IP address out of pool
 - Replaces local, private, network layer source IP addresses to global IP addresses
 - ❖ Has a pool of global IP addresses (less than number of hosts on your network)

NAT Illustration



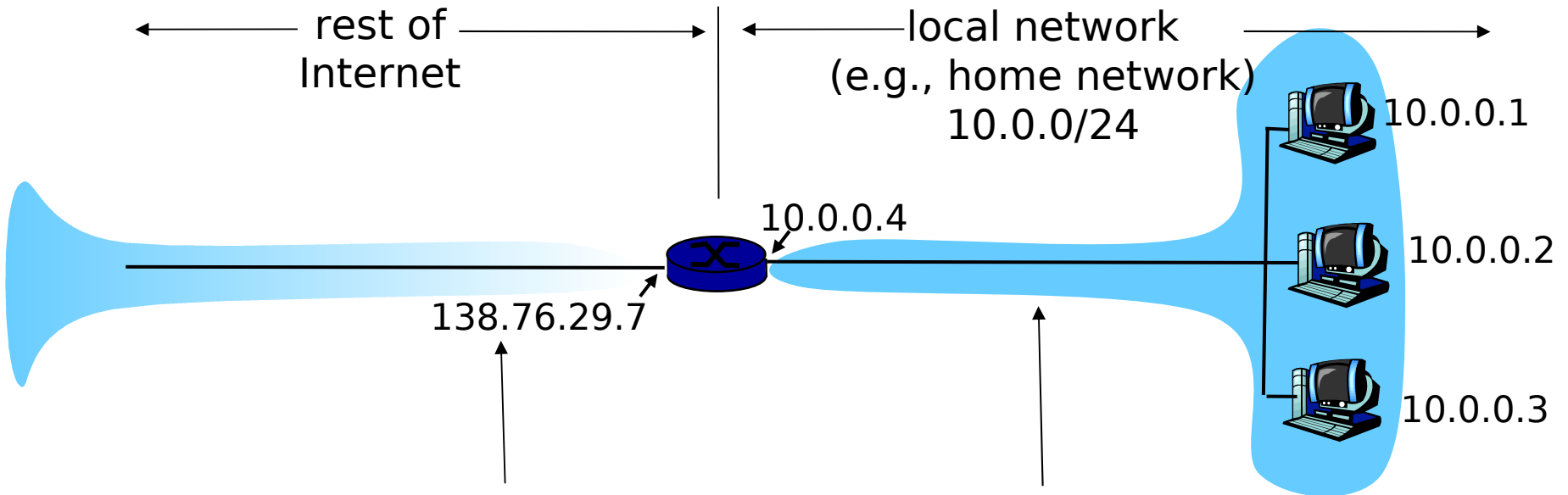
• **Operation: Source (S) wants to talk to Destination (D):**

- Create S_g - S_p mapping
- Replace S_p with S_g for outgoing packets
- Replace S_g with S_p for incoming packets

IP addressing and NAT

- ❑ What if we only have one IP address?
 - ❖ Add port translation to NAT
 - Sometimes referred to as NAPT (Network Address Port Translator)
 - ❖ Both addresses and ports are translated
 - Translates Paddr + flow info to Gaddr + new flow info
 - Uses TCP/UDP port numbers
 - ❖ Potentially thousands of simultaneous connections with one global IP address
 - 16-bit port-number field:
 - 60,000 simultaneous connections with a single LAN-side address!

NAT with port 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

❑ Advantages

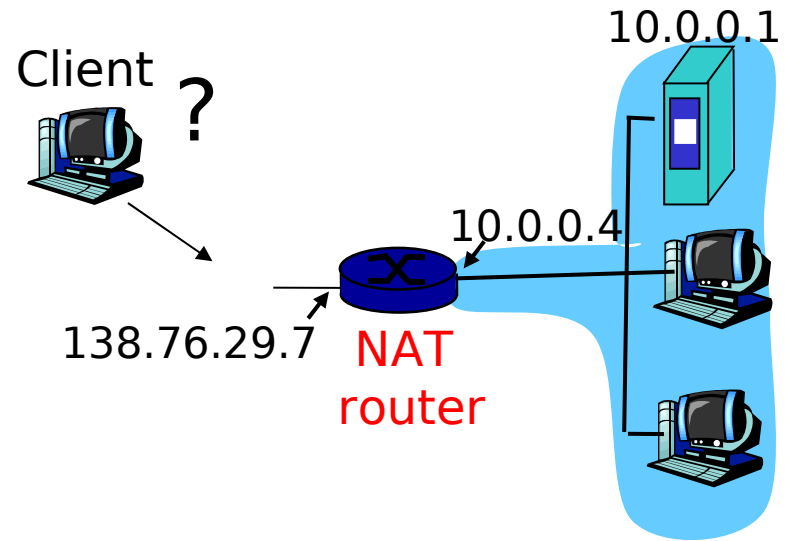
- ❖ can change addresses of devices in local network without notifying outside world
- ❖ can change ISP without changing addresses of devices in local network
- ❖ reduce IP address usage
- ❖ devices inside local net not explicitly addressable, visible by outside world (a security plus).

NAT is controversial

- ❑ Routers should only process up to layer 3
 - ❖ violates network transparency
 - key feature that allows one to deploy any application without coordinating with network infrastructure
 - ❖ implicit assumption that network header is unchanged in network
 - ❖ address shortage should instead be solved by IPv6
- ❑ Other problems
 - ❖ No inbound connections
 - Must be taken into account by app designers, eg, P2P applications
 - ❖ Some protocols carry addresses
 - e.g., FTP carries addresses in text
 - What is the problem?
 - ❖ Encryption

NAT problem #1: traversal

- Incoming connections
 - ❖ client want to connect to server with address 10.0.0.1
 - ❖ server address 10.0.0.1 local to LAN (client can't use it as destination addr)
 - ❖ only one externally visible NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
 - ❖ e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000
 - ❖ Or use DMZ host

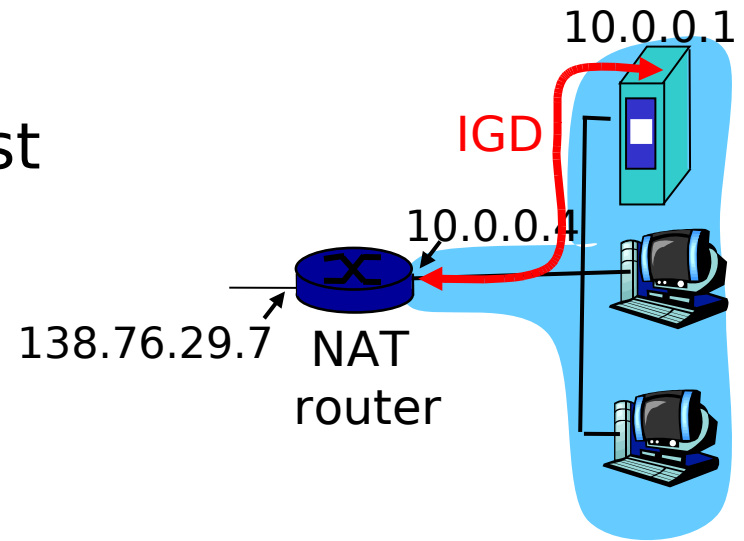


NAT problem #1: traversal

- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:

- ❖ learn public IP address (138.76.29.7)
- ❖ enumerate existing port mappings
- ❖ add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration



NAT problem #1: traversal

- solution 3: relaying (used in Skype)
 - ❖ NATed server establishes connection to relay
 - ❖ External client connects to relay
 - ❖ relay bridges packets between to

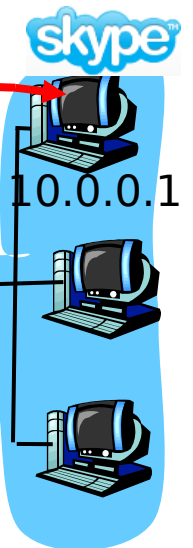
2. connection

to relay initiated by client

3. relaying established

1. connection to relay initiated by NATted host

138.76.29.7 NAT router



NAT problem #2: loss of transparency

- ❑ Breaks applications that assume network does not modify packets
- ❑ Prevents new applications that make the same assumption
- ❑ Example
 - ❖ ftp, NAT, and PORT command

Routing

Malware typically does not attack core routing (but does impact its behavior)

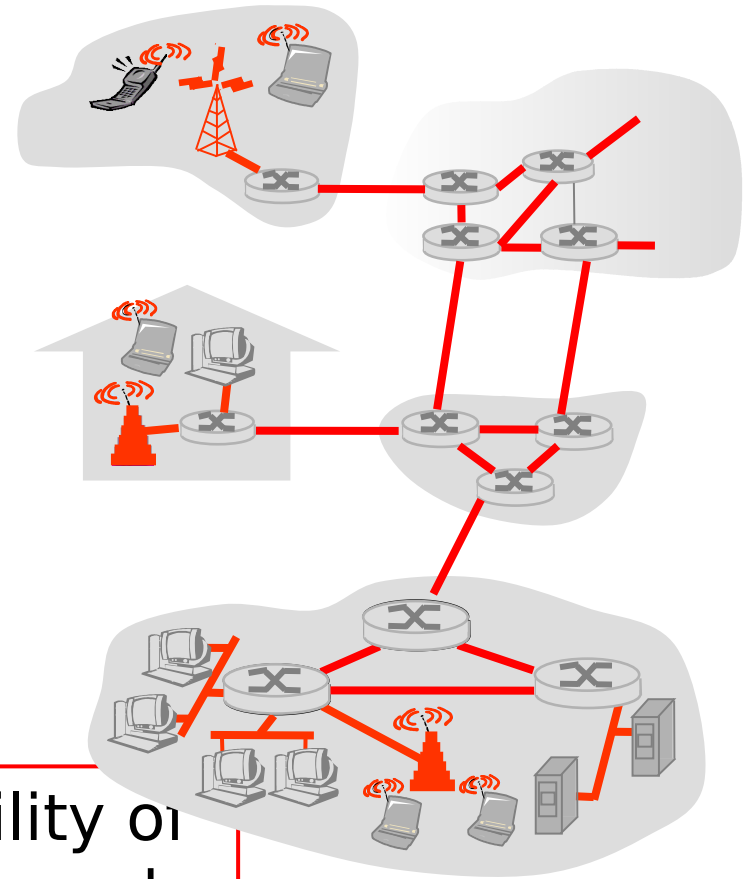
Will not cover...

Data link layer

Link Layer: Introduction

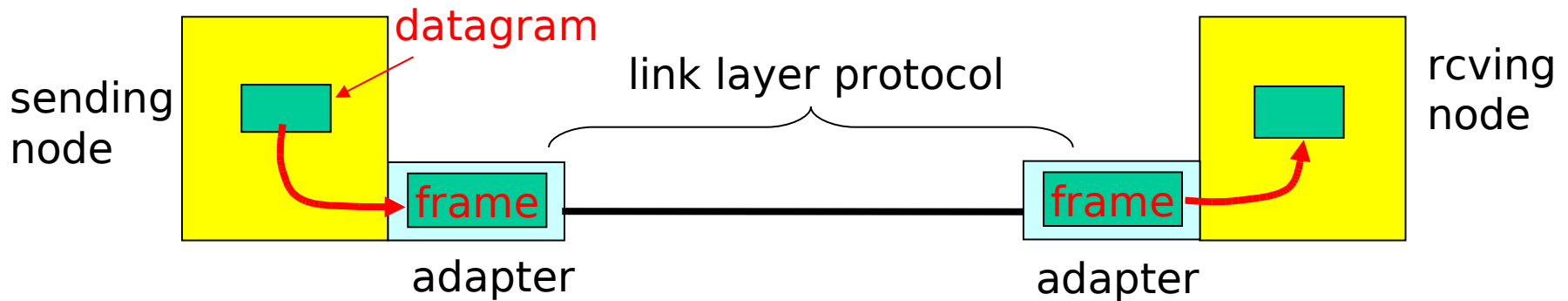
Some terminology:

- ❑ hosts and routers are **nodes**
- ❑ communication channels that connect adjacent nodes along communication path are **links**
 - ❖ wired links
 - ❖ wireless links
 - ❖ LANs
- ❑ layer-2 packet is a **frame**, encapsulates datagram



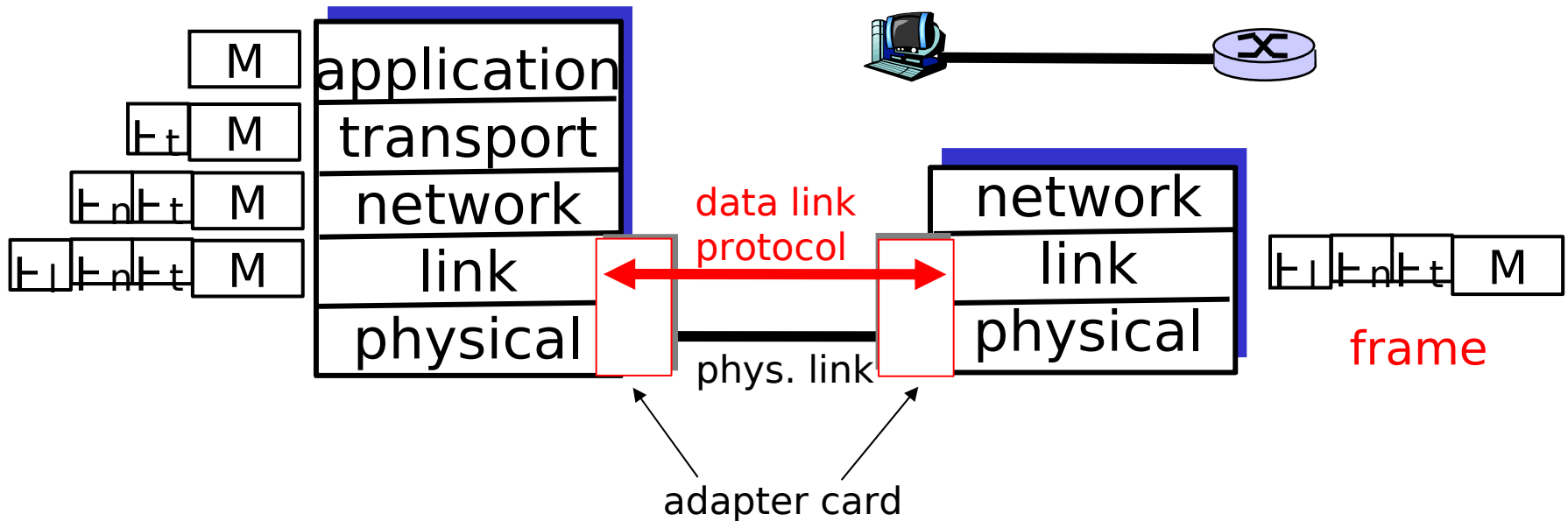
data-link layer has responsibility of transferring datagram from one node to adjacent node over a link

Adaptors Communicating

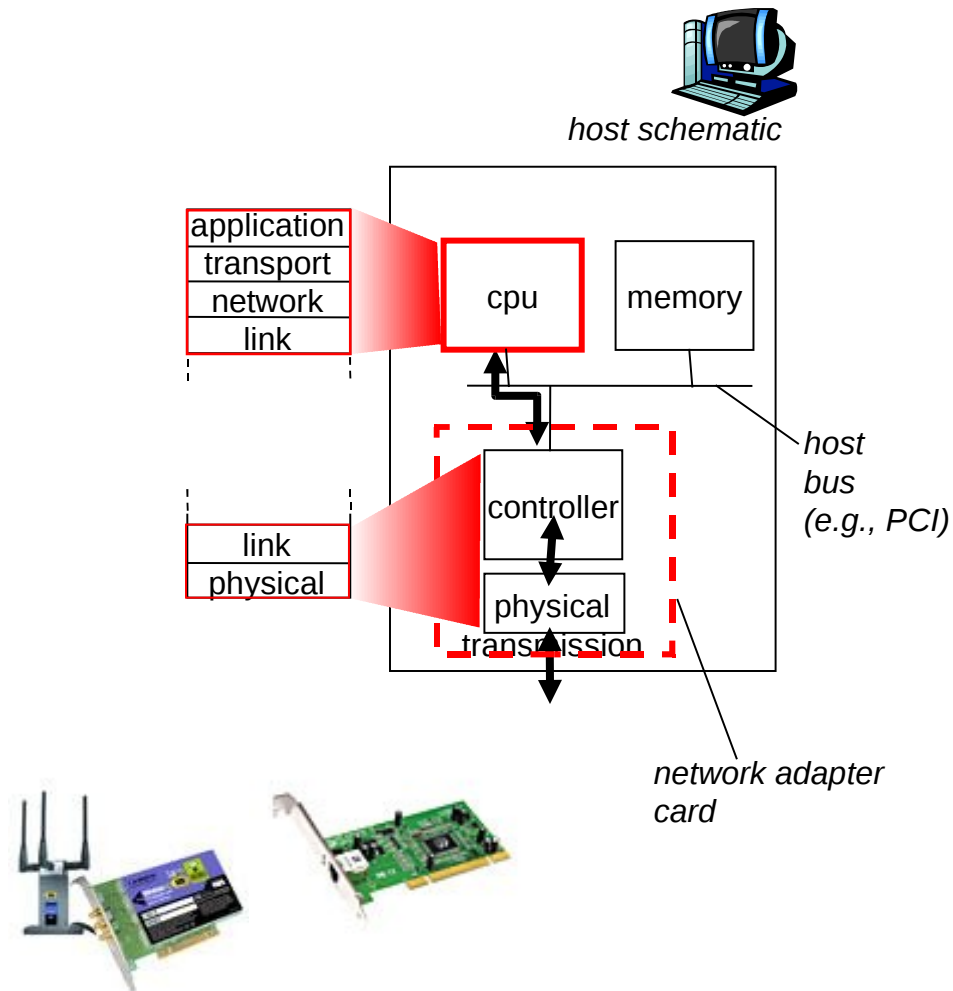


- ❑ link and physical layers implemented in adaptor/NIC (Network Interface Card)
 - ❖ RAM, DSP chips, host bus interface, and link interface
 - ❖ Ethernet card, PCMCIA card, 802.11 card
- ❑ sending side:
 - ❖ encapsulates datagram in a frame
 - ❖ adds error checking bits, rdt, flow control, etc.
- ❑ receiving side
 - ❖ looks for errors, rdt, flow control, etc
 - ❖ extracts datagram, passes to upper layer at receiving side
- ❑ datagram transferred by different link protocols over different links:
 - ❖ e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link

Protocol stack picture



Host adaptor



Link Layer Functions

❑ Flow Control

- ❖ pacing between adjacent sending and receiving nodes

❑ Reliable delivery between adjacent nodes

- ❖ we learned how to do this already (chapter 3)!
- ❖ seldom used on low bit error link (i.e. fiber, twisted pair)
- ❖ wireless links: high error rates
 - Eschew end-to-end principle for performance

Link Layer Functions

❑ Security

- Mainly for broadcast data-link layers
- Encrypt payload of higher layers
- Hide IP source/destination from eavesdroppers
- Important for wireless LANs especially
 - Parking lot attacks with 802.11b
 - WEP, WPA

❑ Demux to upper protocol

❑ Framing

- ❖ encapsulate datagram into frame, adding header, trailer

Link Layer Functions (more)

❑ Error Detection

- ❖ errors caused by signal attenuation, noise.
- ❖ receiver detects presence of errors:
 - signals sender for retransmission or drops frame

❑ Error Correction

- ❖ receiver identifies and *corrects* bit error(s) without resorting to retransmission

❑ Medium access and quality of service

- ❖ channel access if shared medium

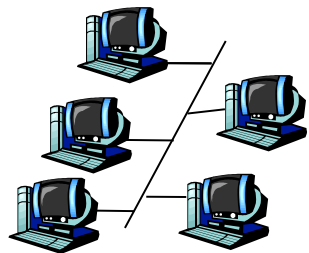
❑ Addressing

- ❖ “MAC” addresses used in frame headers to identify source, dest (different from IP address)

Multiple Access Links and Protocols

Two types of “links”:

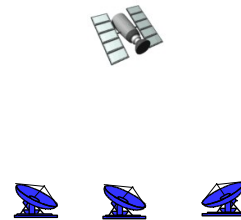
- point-to-point
 - ❖ PPP for dial-up access
 - ❖ point-to-point link between Ethernet switch and host
- **broadcast** (shared wire or medium)
 - ❖ old-fashioned Ethernet
 - ❖ upstream HFC (cable)
 - ❖ 802.11 wireless LAN



shared wire (e.g.,
cabled Ethernet)



shared RF
(e.g., 802.11 WiFi)



shared RF
(satellite)



humans at a
cocktail party
(shared air, acoustical)

Multiple access problem

- ❑ Point-to-point link and switched media no problem
- ❑ Broadcast links require network arbitration
 - Fixed time/freq slot?
 - Centralized arbiter
 - Distributed arbitration

Multiple access protocols

- ❑ single shared communication channel
- ❑ two or more simultaneous transmissions by nodes:
interference
 - only one node can send **successfully** at a time
- ❑ *multiple access protocol:*
 - distributed algorithm that determines how stations share channel, i.e., determine when station can transmit
 - communication about channel sharing uses channel itself!

MAC Protocols: a taxonomy

Three broad classes:

❑ Channel Partitioning

- ❖ divide channel into smaller “pieces” (time slots, frequency, code)
- ❖ allocate piece to node for exclusive use

❑ Random Access

- ❖ channel not divided, allow collisions
- ❖ “recover” from collisions

❑ “Taking turns”

- ❖ tightly coordinate shared access to avoid collisions
- ❖ Nodes take turns, but nodes with more to send can take longer turns

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”,
- ❑ To avoid deterministic collisions: randomize
 - ❖ **random access MAC protocol** specifies:
 - how to detect collisions
 - how to recover from collisions (e.g., via delayed retransmissions)
 - “Asynchronous” TDMA
- ❑ Examples of random access MAC protocols:
 - ❖ slotted ALOHA
 - ❖ ALOHA
 - ❖ CSMA, CSMA/CD, CSMA/CA

MAC Addresses

- MAC/LAN/physical/Ethernet address:
 - ❖ *used to get frame from one interface to another physically-connected interface (same network)*
 - ❖ Globally unique 48 bit address (for most LANs)
burned in the adapter ROM
 - `ifconfig -a`
 - ❖ Administered by IEEE
 - manufacturer buys portion of MAC address space to assure uniqueness

MAC vs IP addressing

- ❑ MAC address
 - ❖ Flat (not hierarchical)
 - Like Social Security Numbers
 - Does not change when machine is moved (portable)
- ❑ IP addresses
 - ❖ Hierarchically organized
 - Like postal address
 - Depends on IP subnet that node is attached to
 - Must change when machine is moved (not portable)
- ❑ Why have separate IP and hardware addresses?
 - ❖ Assign adapters an IP address
 - Hardware only works for IP (no IPX, DECNET)
 - ❖ Use hardware address as network address
 - No route aggregation

ARP: Address Resolution Protocol

Question: how to determine MAC address of B given B's IP address?

□ ARP

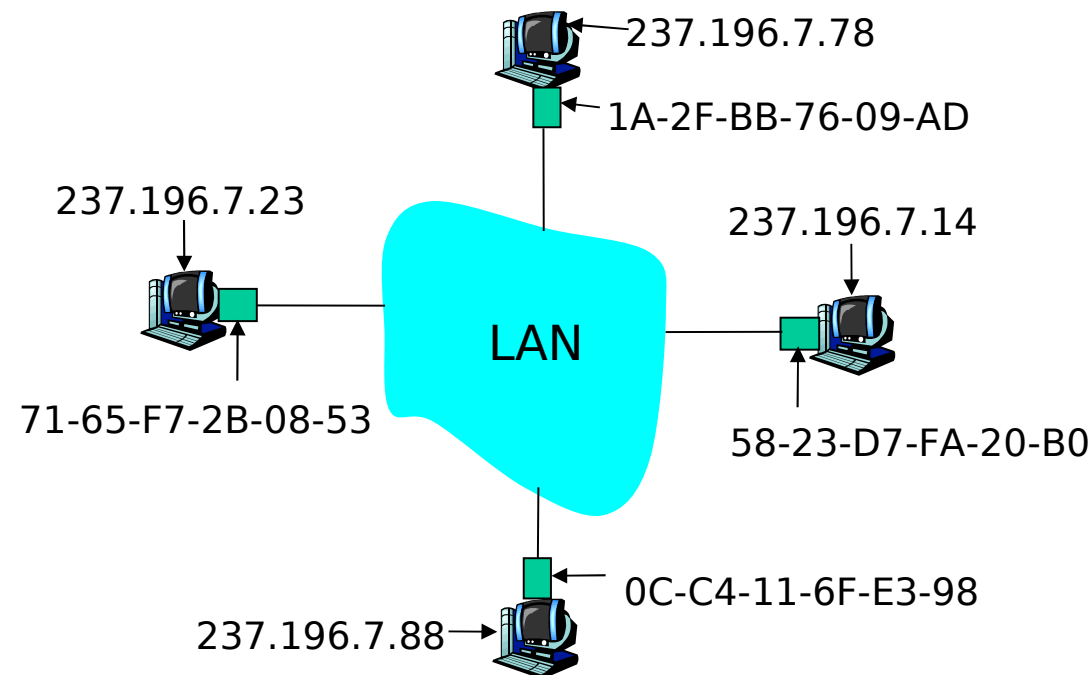
- ❖ Broadcast interest in B's MAC address
- ❖ B responds with its MAC address
- ❖ Keep track of mappings in **ARP** table
 - IP/MAC address mappings for LAN nodes

< IP address; MAC address; TTL >

- TTL (Time To Live)

• Soft state

1-

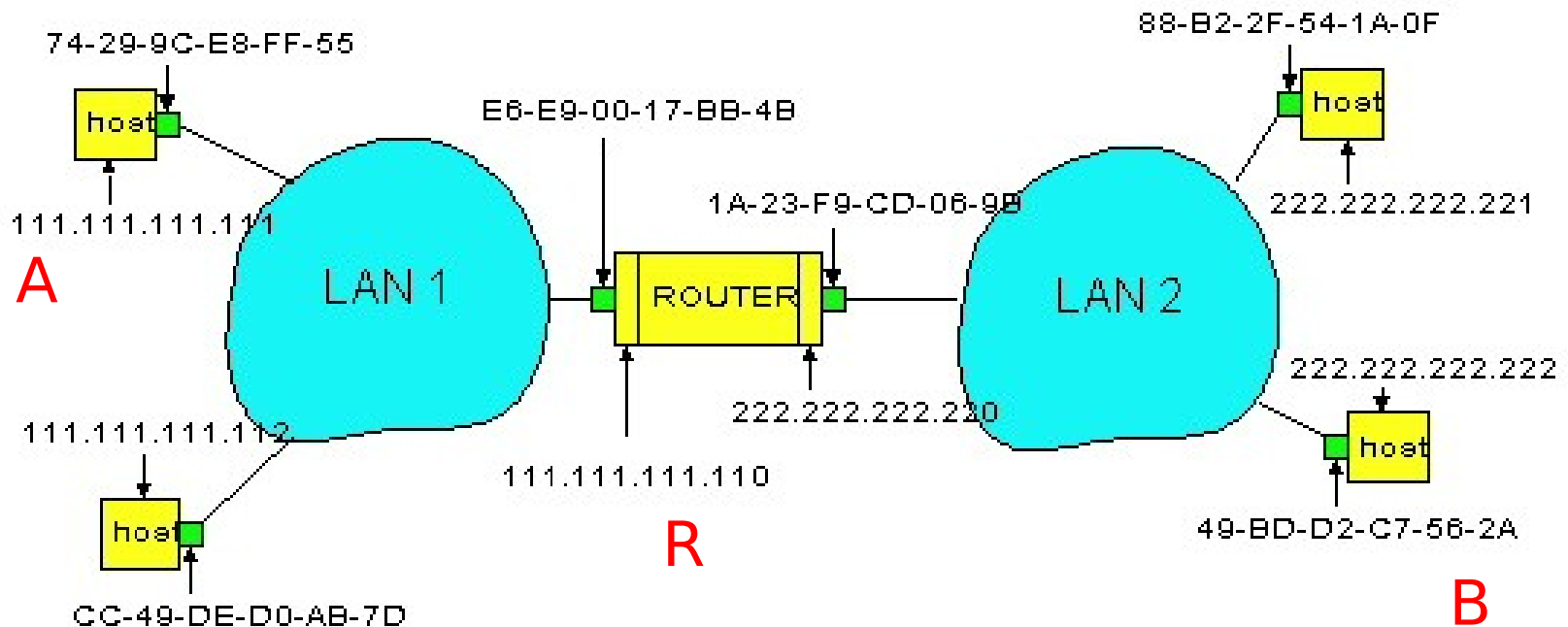


ARP protocol: Same LAN (network)

- ❑ A knows B's IP address and wants to send datagram to B, and 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 machines 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
 - `arp -a`
 - `/proc/net/arp`

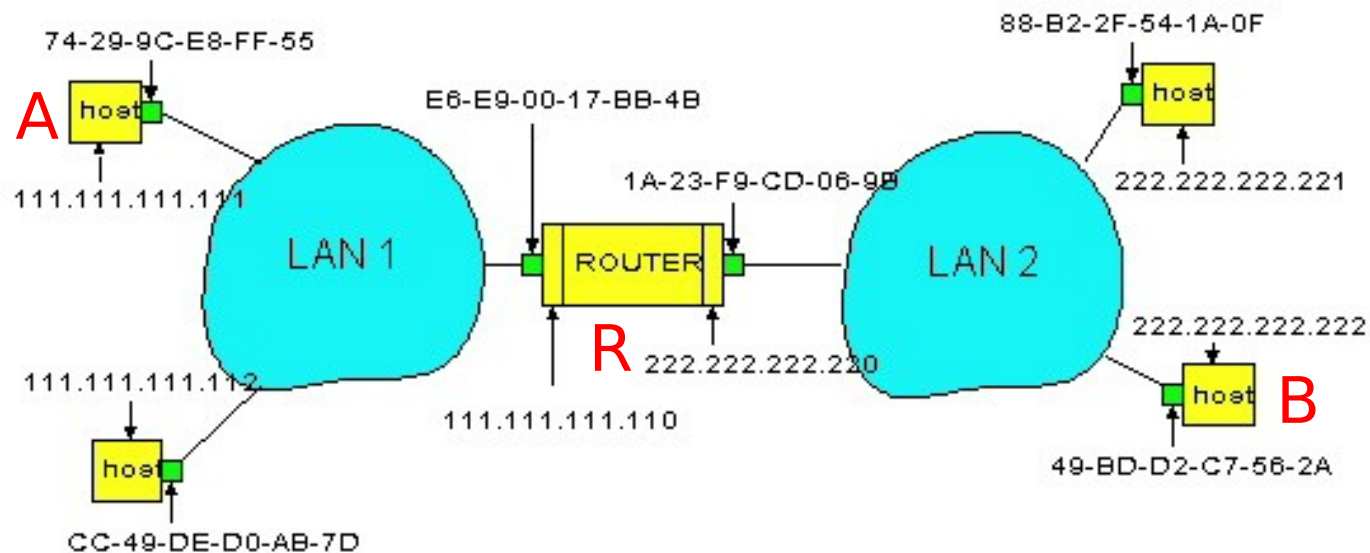
Routing to another LAN

walkthrough: **send datagram from A to B via R**
assume A knows B's IP address



- ❑ Two ARP tables in router R, one for each IP network (LAN)
- ❑ In routing table at source Host, default route 111.111.111.110
- ❑ A creates datagram with source A, destination B

- ❑ A checks route table to find B is not on its network
- ❑ A uses ARP to get R's MAC address (ARP for 111.111.111.110)
- ❑ A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram
- ❑ A's adapter sends frame
- ❑ R's adapter receives frame
- ❑ R removes IP datagram from Ethernet frame, sees its destined to B
- ❑ R uses ARP to get B's MAC address
- ❑ R creates frame containing A-to-B IP datagram sends to B



DHCP

Q: How does *host* get an IP address on subnet?

- hard-coded by system admin in a file
 - ❖ Wintel: control-panel->network->configuration->tcp/ip->properties
 - ❖ UNIX: /etc/rc.config
- **DHCP: Dynamic Host Configuration Protocol:**
dynamically get address from server
 - ❖ “plug-and-play”
 - ❖ Given a hardware address, give me the IP address
 - Predecessors: RARP, BOOTP
 - Opposite of ARP (given IP address, give me MAC address)

Link-layer devices

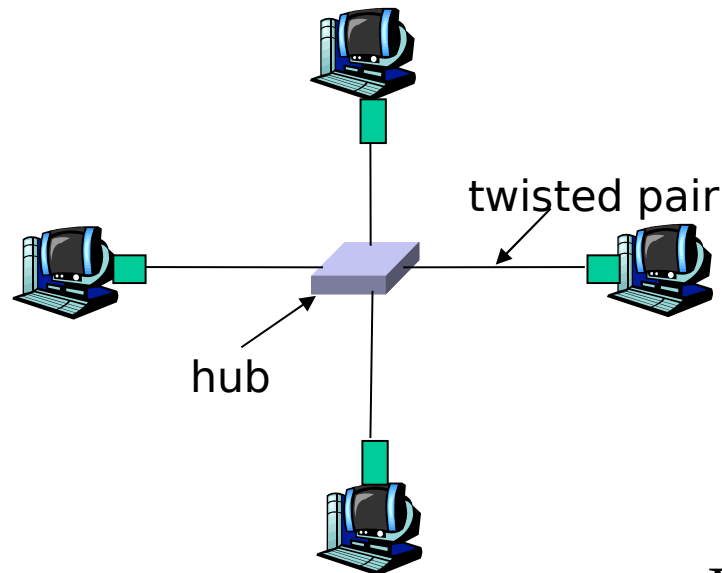
Q: Why not just one big LAN?

- ❑ limited aggregate bandwidth
- ❑ limited length: electrical limitations
- ❑ large “collision domain” (can collide with many stations)
- ❑ access delay (eg 802.5 token passing delay)

Hubs

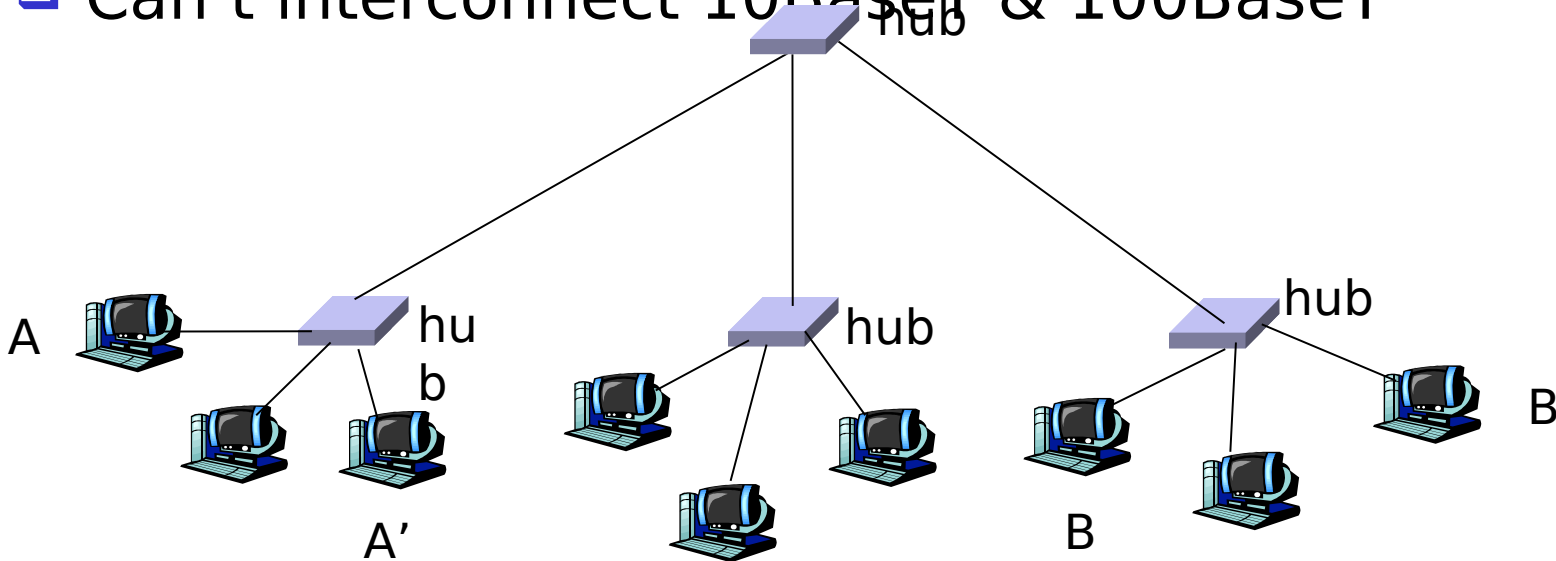
Hubs are essentially physical-layer, multi-port repeaters:

- ❖ bits coming from one link go out *all* other links at the same rate
- ❖ addresses electrical limitations
- ❖ no frame buffering
- ❖ no CSMA/CD at hub: adapters detect collisions
 - all nodes connected to hub can collide with one another
 - Does not isolate collision domains



Interconnecting with hubs

- ❑ Backbone hub interconnects LAN segments
- ❑ But individual segment collision domains become one large collision domain
 - ❖ Single collision domain results in no increase in max throughput
 - ❖ Simultaneous transfers between A to A' and B to B' collide
 - ❖ Multi-tier throughput same as single segment throughput
- ❑ Can't interconnect 10BaseT & 100BaseT

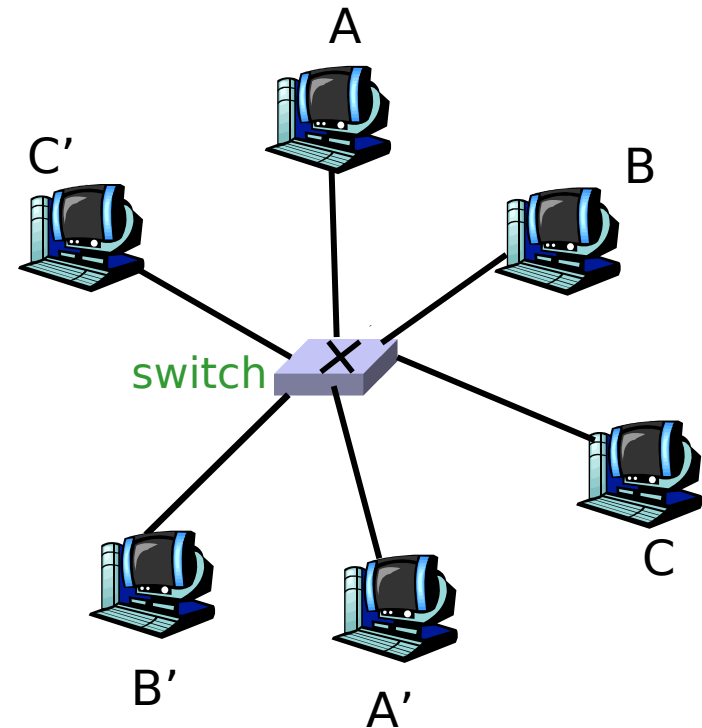


Switches

- ❑ **Link Layer device**
 - ❖ Smarter than hubs
 - ❖ Actively stores and forwards Ethernet frames
 - ❖ Examines frame header and **selectively** forwards frame based on destination MAC address
 - ❖ Two-port switch known as a “bridge”
 - ❖ Switches known as “multi-port” bridges
- ❑ A switch **isolates collision** domains since it buffers frames
- ❑ Uses CSMA/CD to access individual network segments to transmit frames
 - ❖ Transparent to hosts
 - ❖ Plug-and-play, self-learning (do not need to be configured)

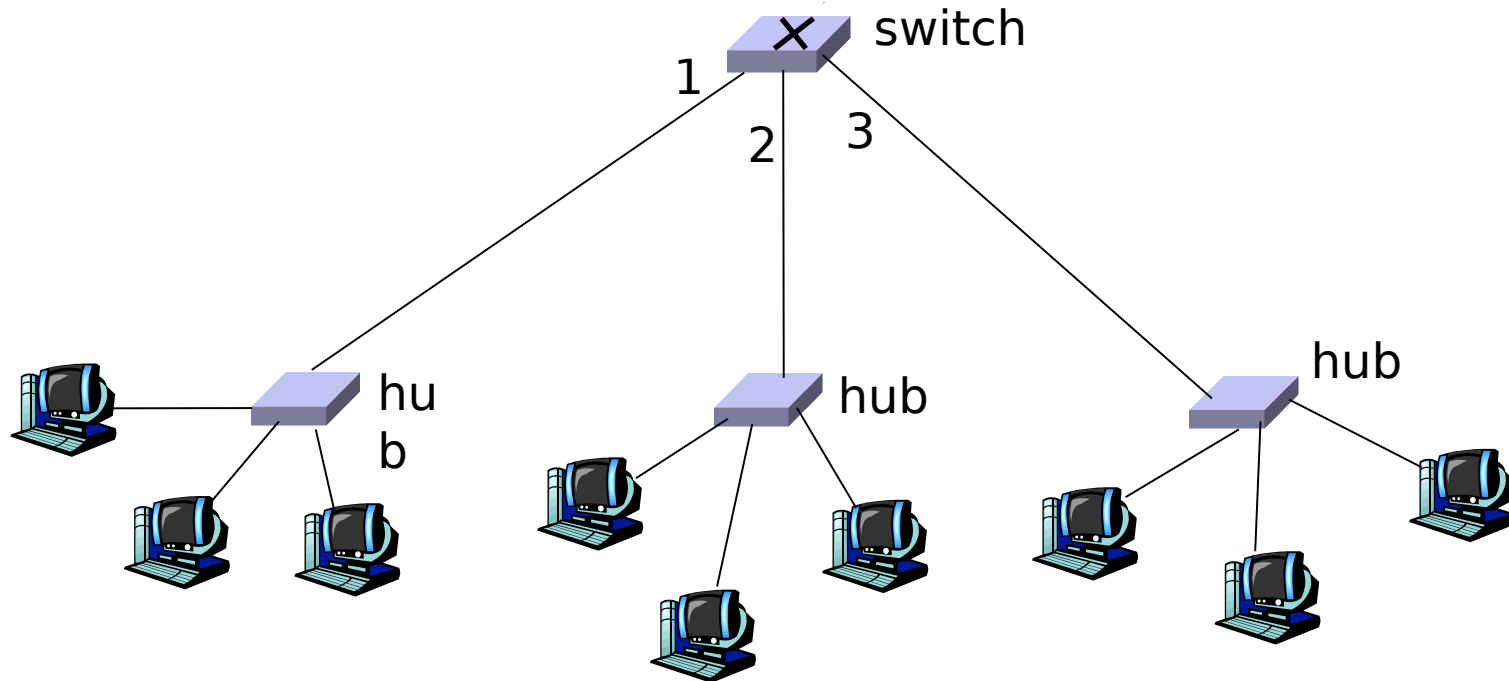
Switches: multiple simultaneous transmissions

- ❑ Hosts have dedicated direct connection to switch
- ❑ Ethernet protocol and frame used, but...
 - ❖ No collisions
 - Each link is its own collision domain
 - ❖ Full duplex operation
- ❑ Switch buffers frames
- ❑ Much greater aggregate bandwidth
 - ❖ Data backplane of switches typically large to support simultaneous transfers amongst ports



Switching: A-to-A' and B-to-B' simultaneously, no collisions

Switch operation



- How do determine onto which LAN segment to forward frame?
 - ❖ Looks like a routing problem...


Self learning

□ Approach

- ❖ Monitor traffic to build a cache (**switch table**) of which nodes are downstream of which ports
 - (MAC Address, Interface, Time Stamp)
 - *learns* which hosts can be reached through which interfaces
- ❖ Selectively forward frames based on cache entries
- ❖ Flood network for frames with unknown (MAC) destinations

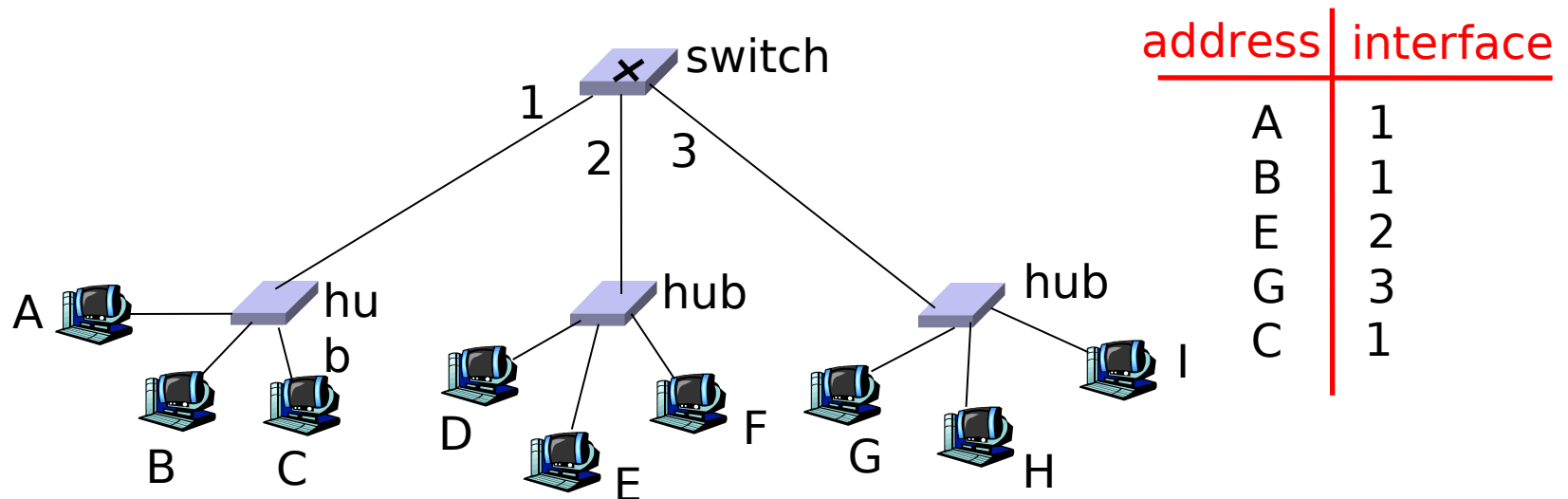
Switch algorithm

When frame received:

1. record link associated with sending host
2. index switch table using MAC dest address
3. **if** entry found for destination
 then {
 if dest on segment from which frame arrived
 then drop the frame
 else forward the frame on interface
indicated
 }
 else flood 

Switch example

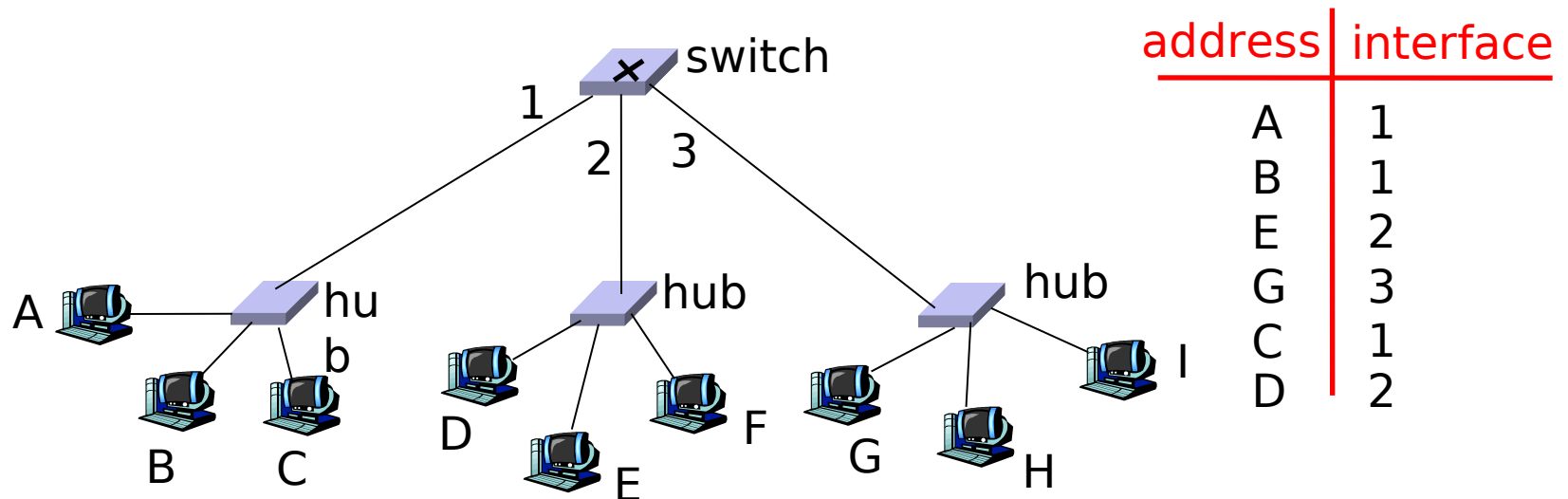
Suppose C sends frame to D



- ❑ Switch receives frame from from C
 - notes in bridge table that C is on interface 1
 - because D is not in table, switch forwards frame into interfaces 2 and 3
- ❑ frame received by D

Switch example

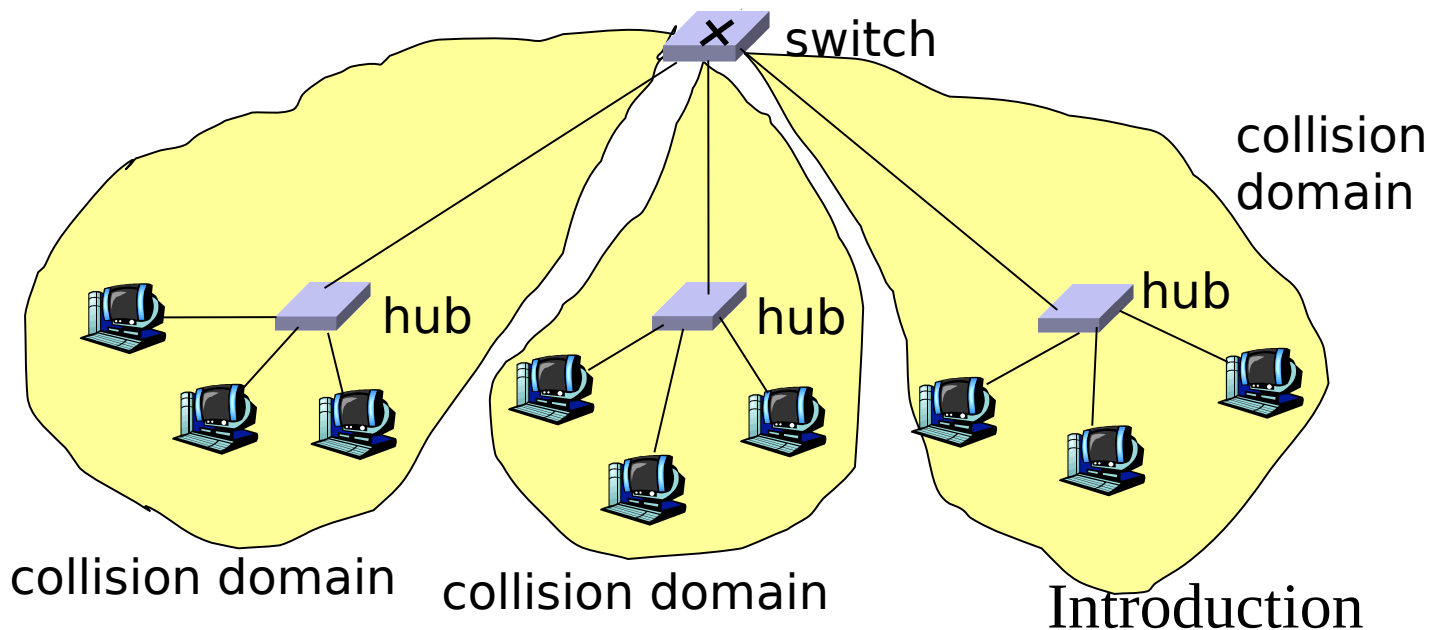
Suppose D replies back with frame to C.



- ❑ Switch receives frame from from D
 - notes in bridge table that D is on interface 2
 - because C is in table, switch forwards frame only to interface 1
- ❑ frame received by C

Switch: traffic isolation

- switch installation breaks subnet into LAN segments
- switch **filters** packets:
 - ❖ same-LAN-segment frames not usually forwarded onto other LAN segments
 - ❖ segments become separate **collision domains**



Internet overview complete

- Technical background for the rest of the course