COMP 4622

Computer Communication Networks II

Fall 2013 HKUST

Qian Zhang
Department of Computer Science and Engineering

Who's Who

- Instructor:

  Prof. Qian Zhang, qianzh AT ust domain
  - Rm. 3533, Tel: 2358-7688
  - Office hours: by appointment
- TAs:

  Wei Wang, gswwang AT ust domain
- Course web site:
  - http://course.cs.ust.hk/comp4622/index.html
- No lab for comp4622

Conduct in the Classroom

1. Be on-time to class.

When you come in late, you disrupt your class. Respect your fellow students, and be on-time to class!

Conduct in the Classroom

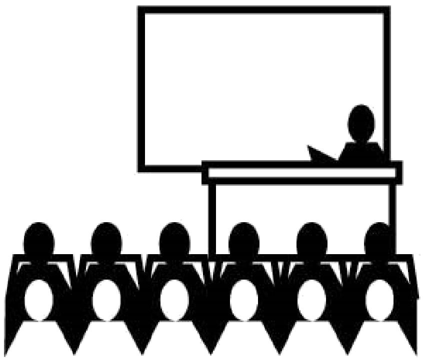
2. No talking in class except to raise or answer questions.

During class, you should not do anything that disrupts the class or distracts your classmates. If you have a pager or cellular phone, turn it off when you are in class.

And please pay attention to the signs that tell you not to eat or drink in the classrooms.

Conduct in the Classroom

3. Anti-cheating



Unless otherwise stated, all work submitted by you should be your own. If there is any doubt about the appropriateness of your actions, please contact the instructor for explicit clarification. Cheating is a serious offense and will result in appropriate disciplinary actions against those involved.

Grading

Project	5%
Homeworks (2)	20%
Midterm	35%
Final	40%

The detailed instruction for course project will be given 2-3 weeks later

- Introduction: course description & calendar
- Prerequisites
- What we have learnt, and what we will learn

Introduction

- What you have learnt
  - COMP 361/4621 or ELEC 315/4120 has already laid the foundation of computer networks
  - We will review the concepts and techniques discussed in these prerequisite courses

- In this course we concentrate on advanced topics in computer networks beyond what you learnt

What you Learnt!: Overview

Goal:

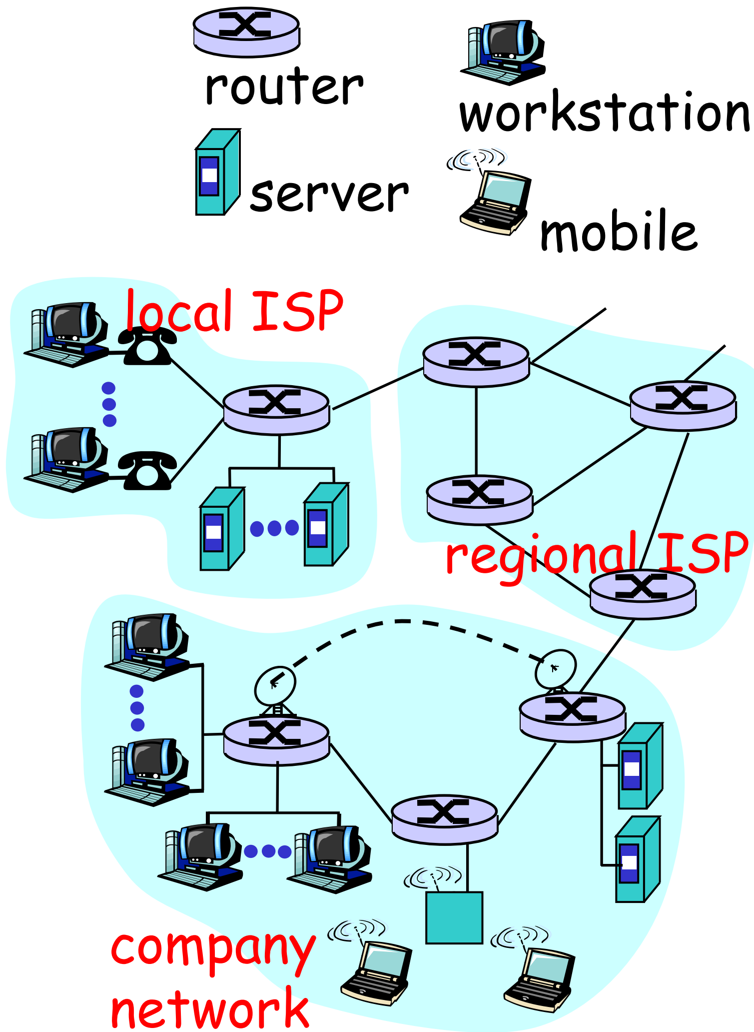
- broader coverage of networking
- approach:
  - descriptive
  - use Internet as example

Overview:

- what's the Internet
- what's a protocol?
- network edge
- network core
- access net, physical media
- Internet/ISP structure
- performance: loss, delay
- protocol layers, service models
- history

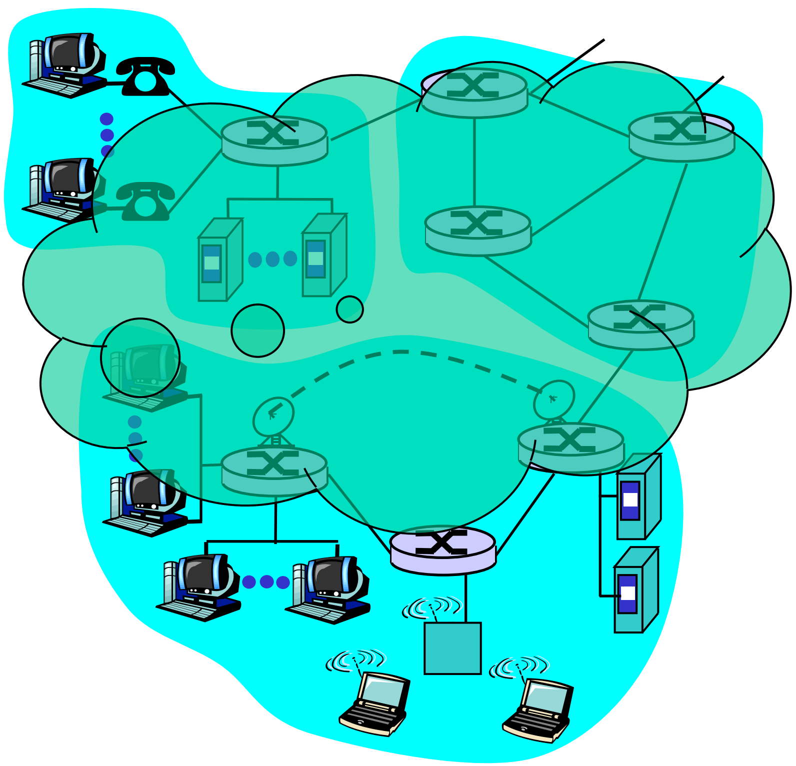
What's the Internet: "Nuts and Bolts" View

- tens of millions of connected computing devices: *hosts = end systems*
  - New members – pervasive wireless devices
- running *network apps*
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate = ***bandwidth***
- *routers:* forward packets (chunks of data)



What's the Internet: a Service View

- communication *infrastructure* enables distributed applications:
  - Popular applications: Web, email, games, e-commerce, file sharing, blogging, twitter, …
- communication services provided to apps:
  - Connectionless unreliable
  - connection-oriented reliable



What's a Protocol?

human protocols:

- "what's the time?"
- "I have a question"
- introductions

… specific msgs sent

… specific actions taken when msgs received, or other events

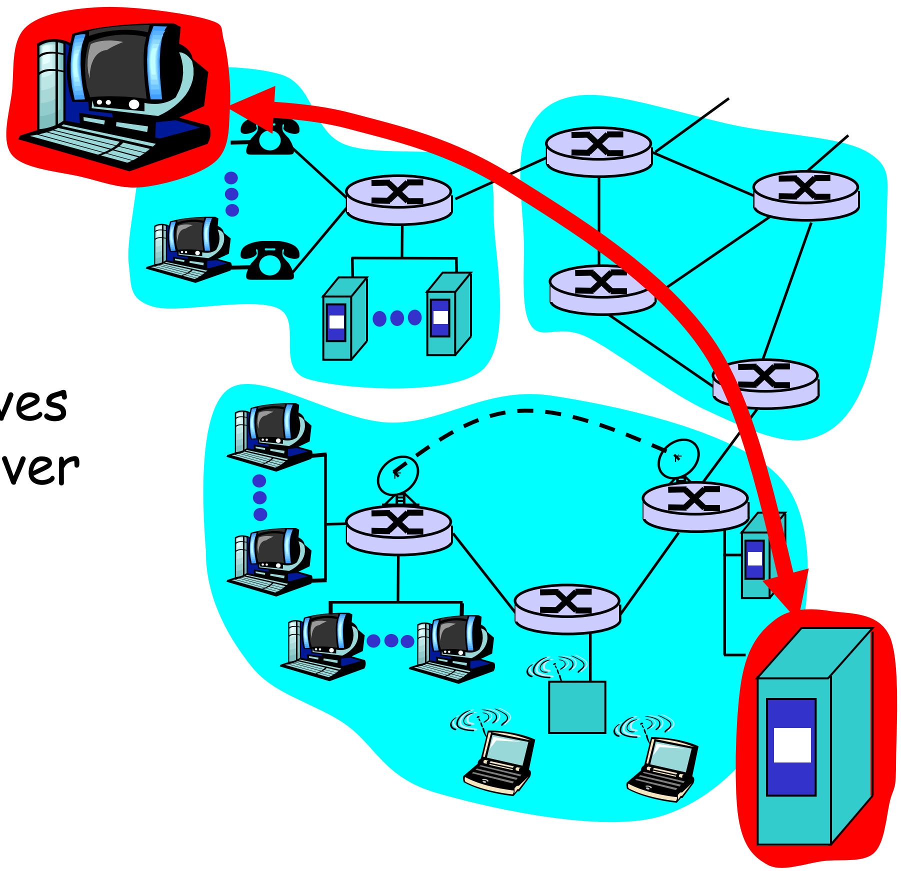
network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

protocols define the format, and the order of msgs sent and received among network entities, and actions taken on msg transmission and receipt

The Network Edge:

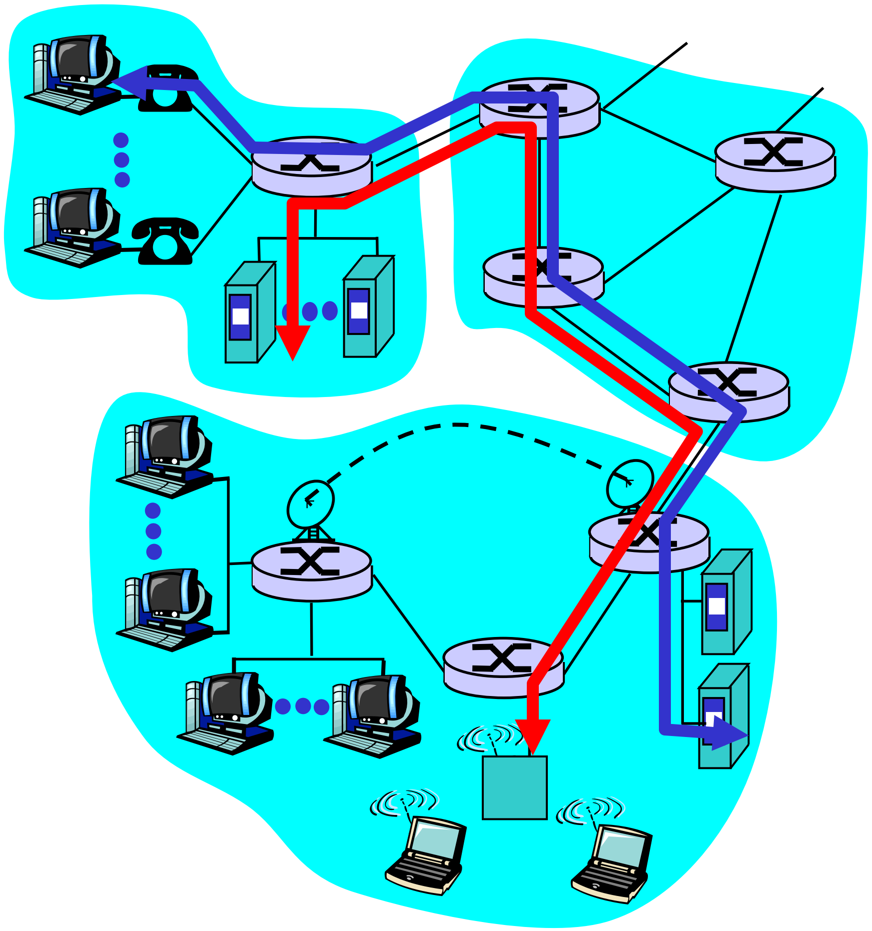
- end systems (hosts):
  - run application programs
  - e.g. Web, email
  - at "edge of network"
- client/server model
  - client host requests, receives service from always-on server
  - e.g. Web browser/server; email client/server
- peer-peer model:
  - minimal (or no) use of dedicated servers
  - e.g. Gnutella, KaZaA



Network Core: Circuit Switching

End-end resources reserved for “call”

- link bandwidth, switch capacity
- dedicated resources: no sharing
- circuit-like (guaranteed) performance
- call setup required



Network Core: Packet Switching

each end-end data stream divided into *packets*

- packets *share* network resources
- each packet usually uses full link bandwidth
- resources used *as needed*

Bandwidth division into "pieces"
Dedicated allocation
Resource reservation

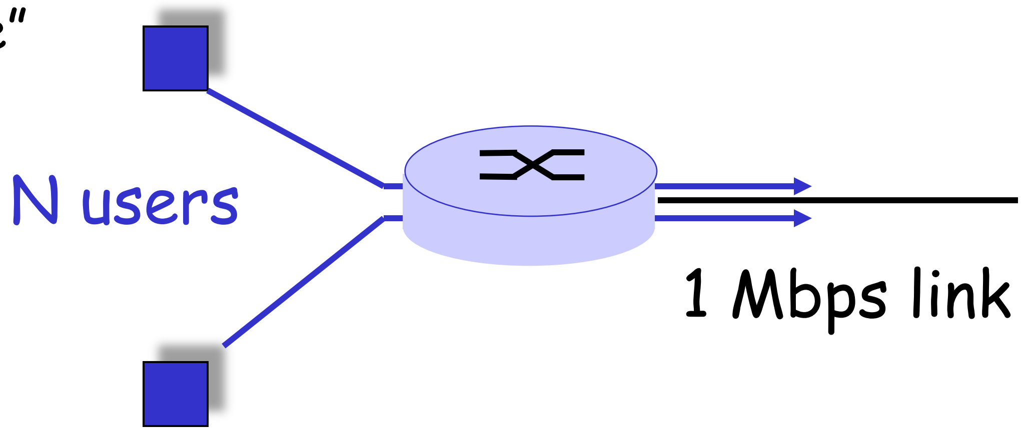
resource contention:

- aggregate resource demand can exceed amount available
- congestion: packets queue, wait for link use
- store and forward: packets move one hop at a time
  - Node receives complete packet before forwarding

Packet Switching vs. Circuit Switching

Packet switching allows more users to use network!

- 1 Mbps link
- each user:
  - 100 kbps when "active"
  - active 10% of time
- circuit-switching:
  - 10 users
- packet switching:
  - with 35 users, probability > 10 simultaneous active uses less than .0004



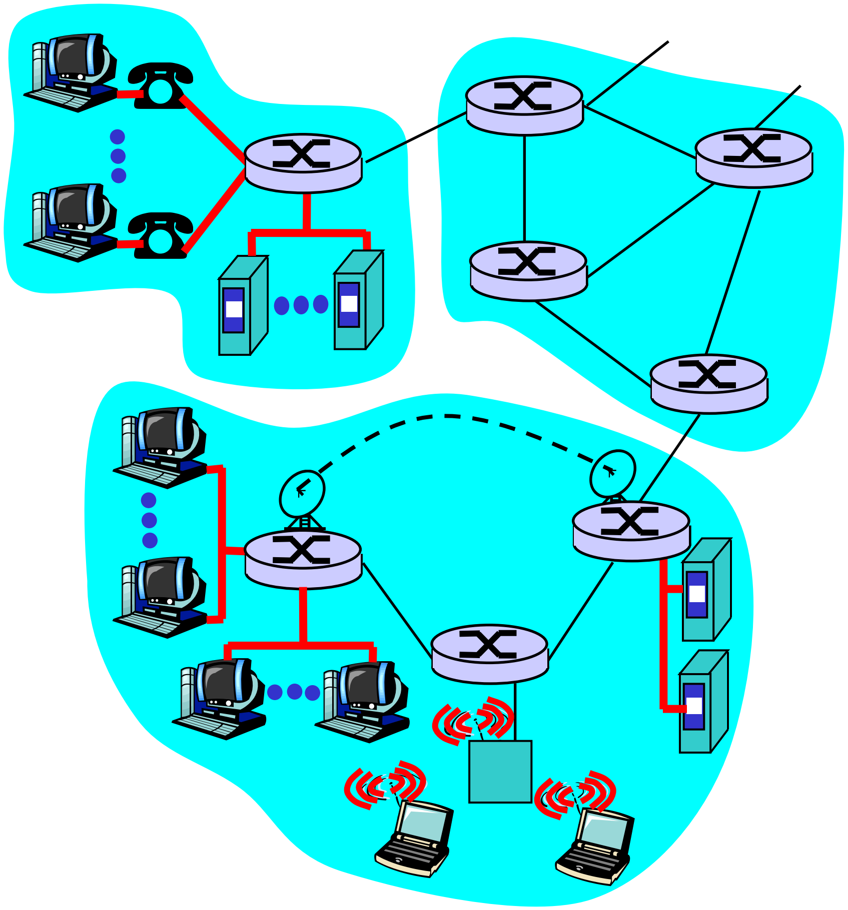
Access Networks and Physical Media

Q: How to connect end systems to edge router?

- residential access nets
- institutional access networks (school, company)
- mobile access networks

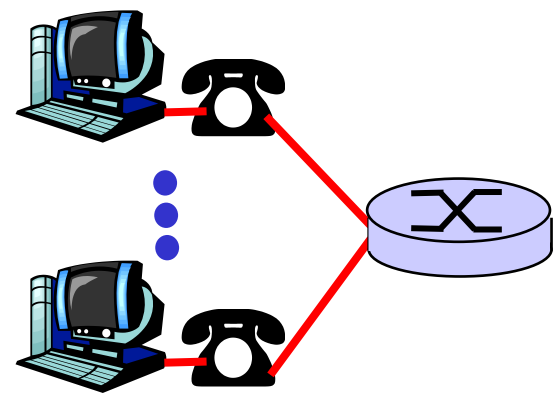
Keep in mind:

- bandwidth (bits per second) of access network?
- shared or dedicated?



Residential Access: Point to Point Access

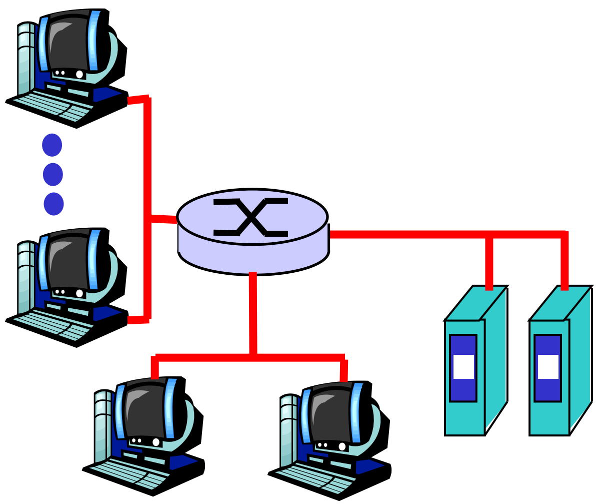
- Dialup via modem
  - up to 56Kbps direct access to router (often less)
  - Can't surf and phone at same time: can't be "always on"



- ADSL: asymmetric digital subscriber line
  - up to 1 Mbps upstream (today typically < 256 kbps)
  - up to 8 Mbps downstream (today typically < 1 Mbps)
  - FDM: 50 kHz - 1 MHz for downstream
    4 kHz - 50 kHz for upstream
    0 kHz - 4 kHz for ordinary telephone

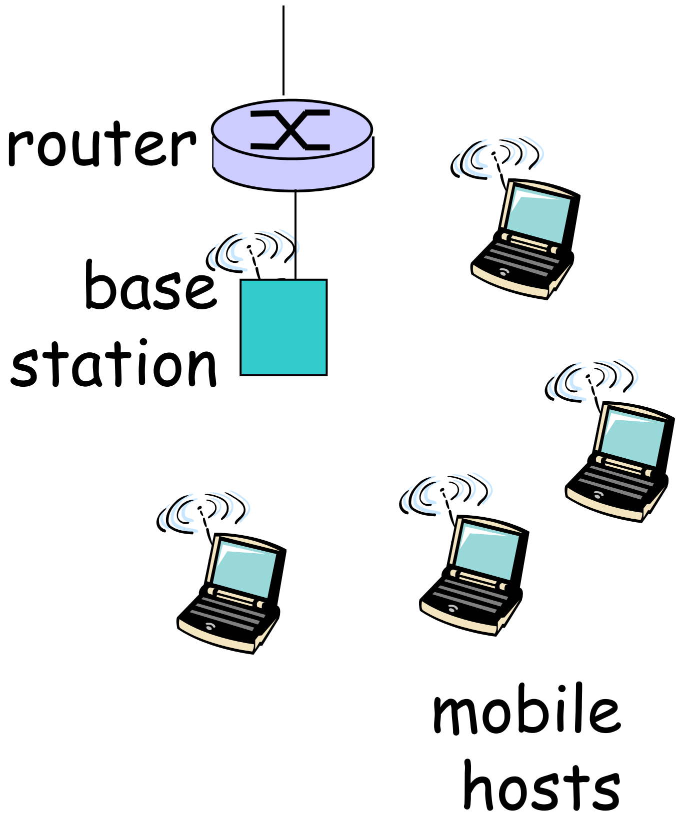
Company Access: Local Area Networks

- company/univ local area network (LAN) connects end system to edge router
- Ethernet:
  - shared or dedicated link connects end system and router
  - 10 Mbps, 100Mbps, Gigabit Ethernet



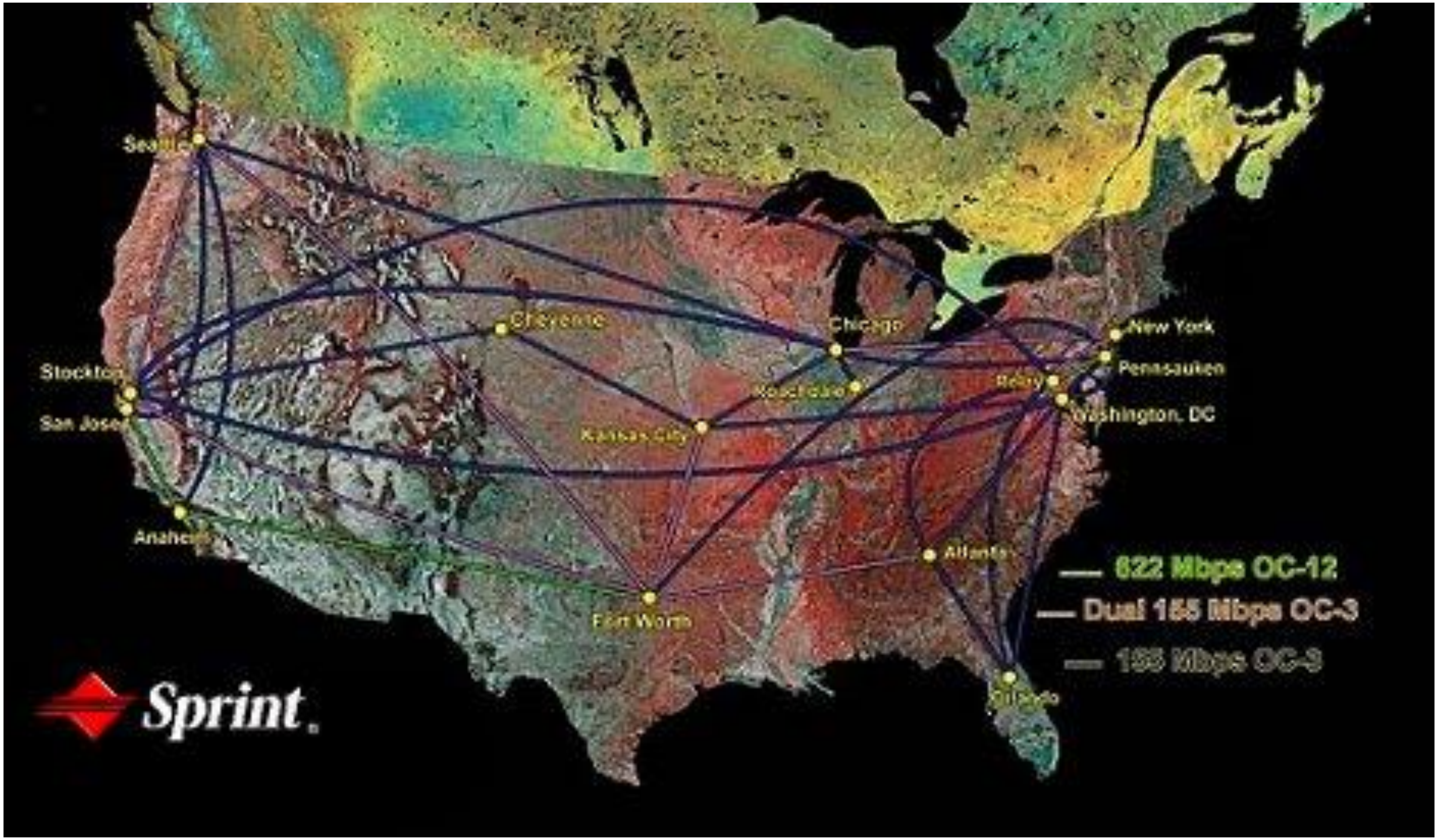
Wireless Access Networks

- shared *wireless* access network connects end system to router
  - via base station a.k.a. "access point"
- wireless LANs:
  - 802.11b (WiFi): 11 Mbps
- wider-area wireless access
  - provided by telecom operators
  - 2G, 3G, 4G, …
  - WAP/GPRS



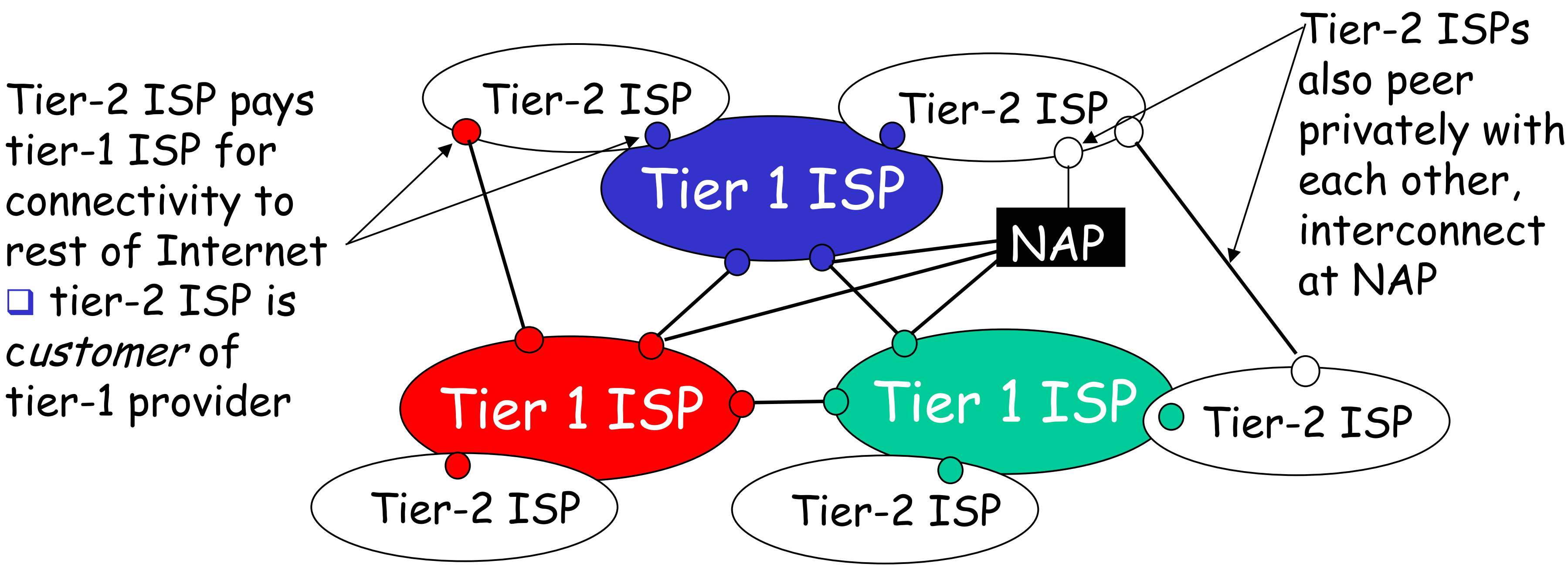
Tier-1 ISP: e.g., Sprint

Sprint US backbone network



Internet Structure: Network of Networks

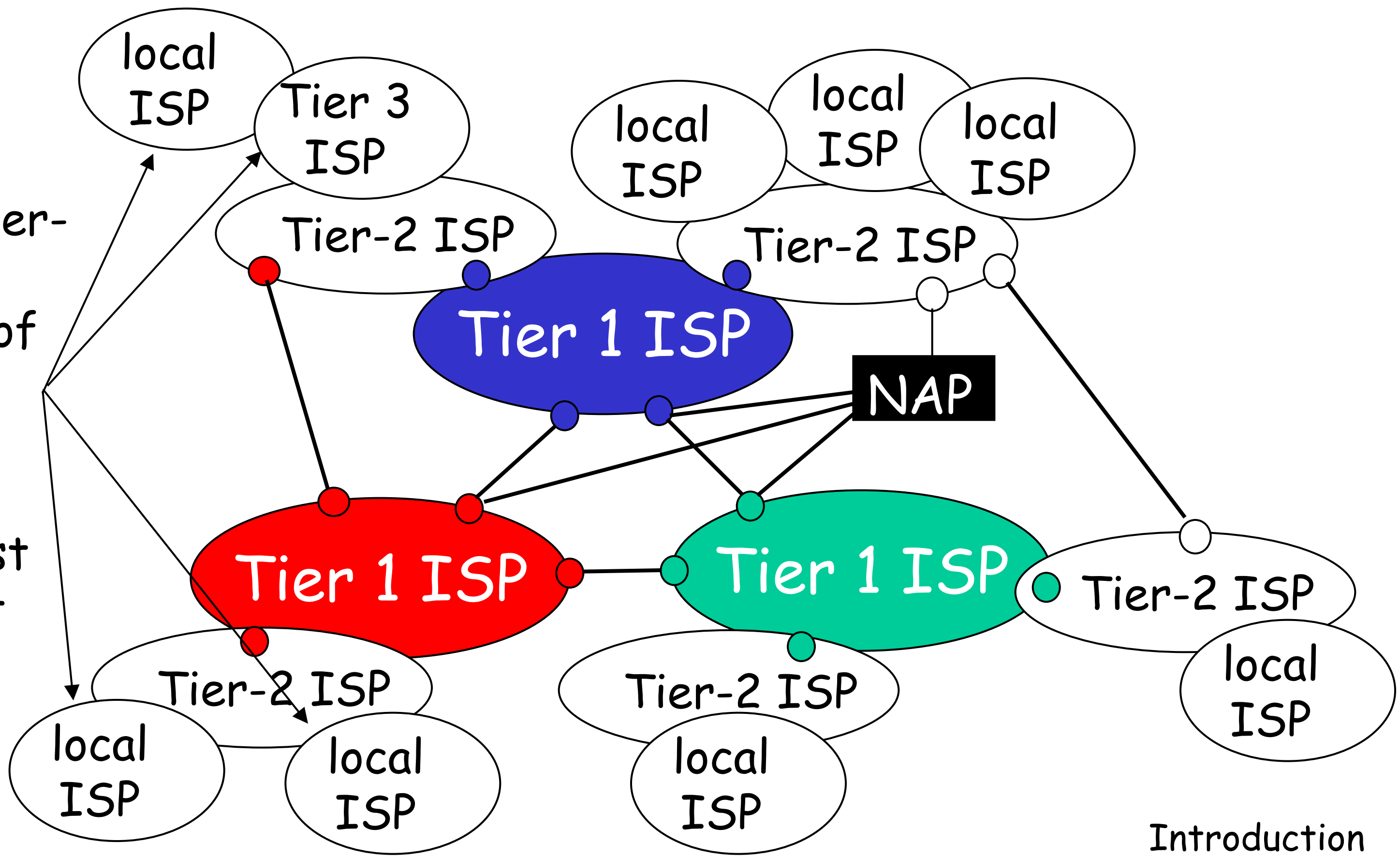
- "Tier-2" ISPs: smaller (often regional) ISPs
  - Connect to one or more tier-1 ISPs, possibly other tier-2 ISPs
  - NAP: Network Access Point



Internet Structure: Network of Networks

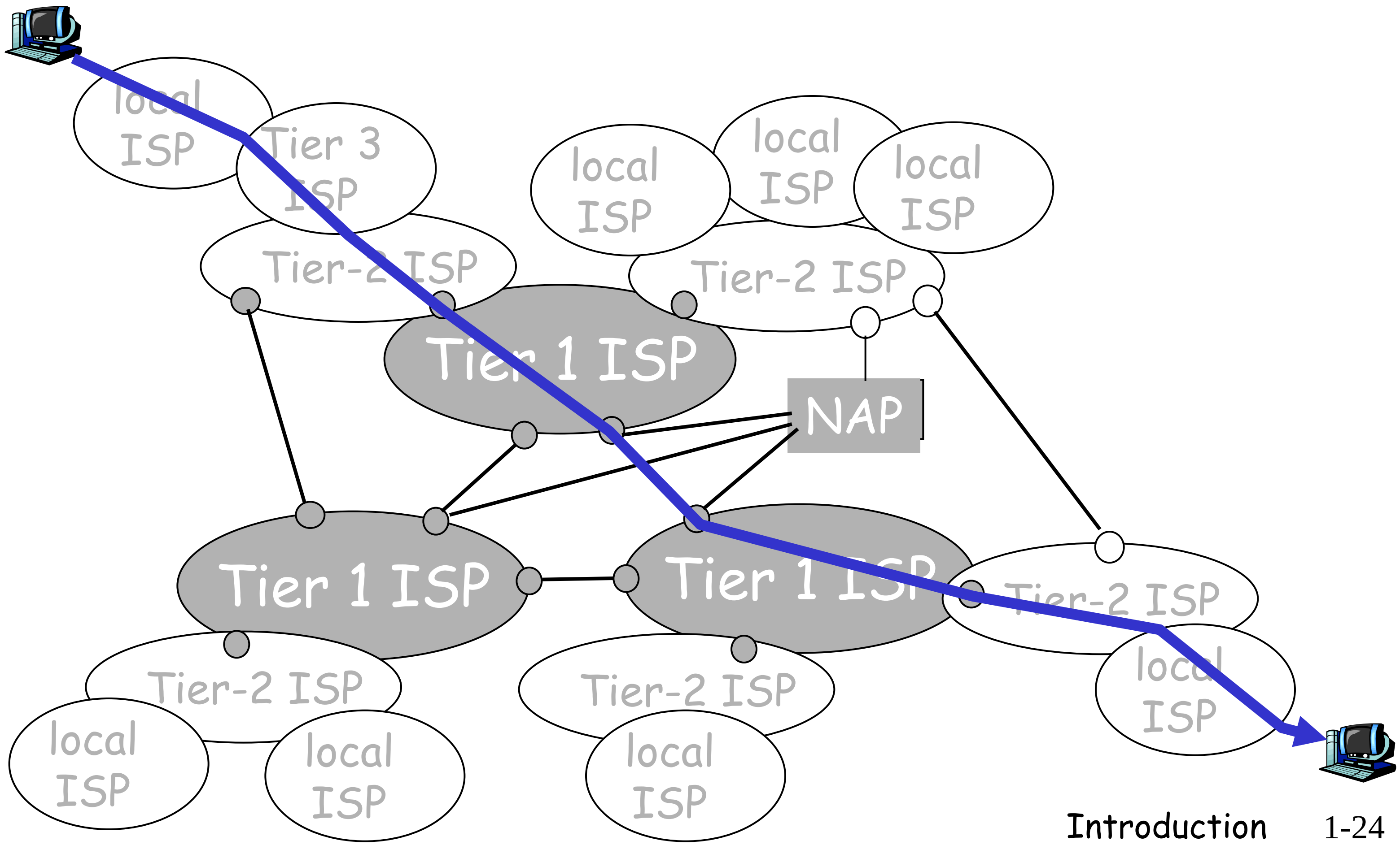
- "Tier-3" ISPs and local ISPs
  - last hop ("access") network (closest to end systems)

Local and tier-3 ISPs are *customers* of higher tier ISPs connecting them to rest of Internet



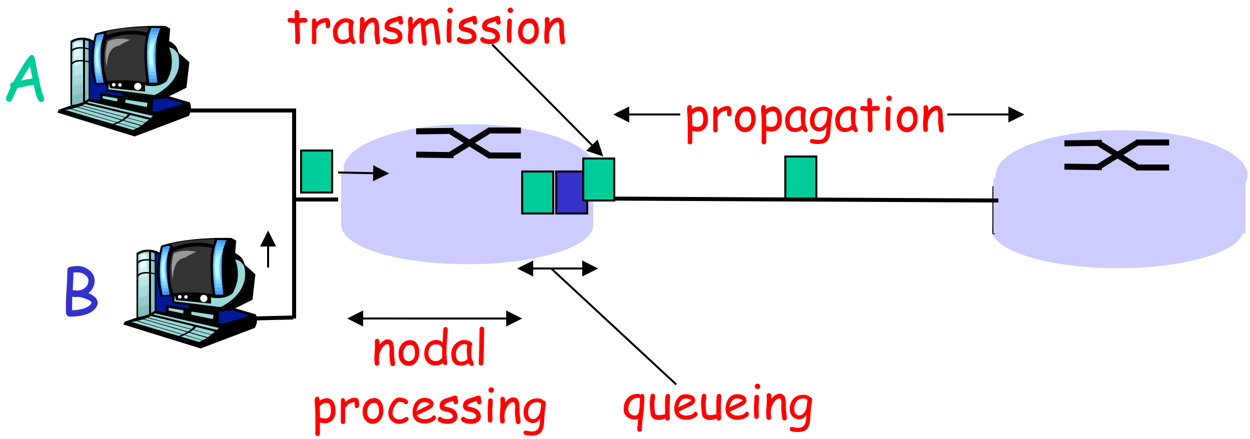
Internet Structure: Network of Networks

- a packet passes through many networks!



Four Sources of Packet Delay

- 1. nodal processing:
  - check bit errors
  - determine output link
- 2. queueing
  - time waiting at output link for transmission
  - depends on congestion level of router



Delay in Packet-Switched Networks

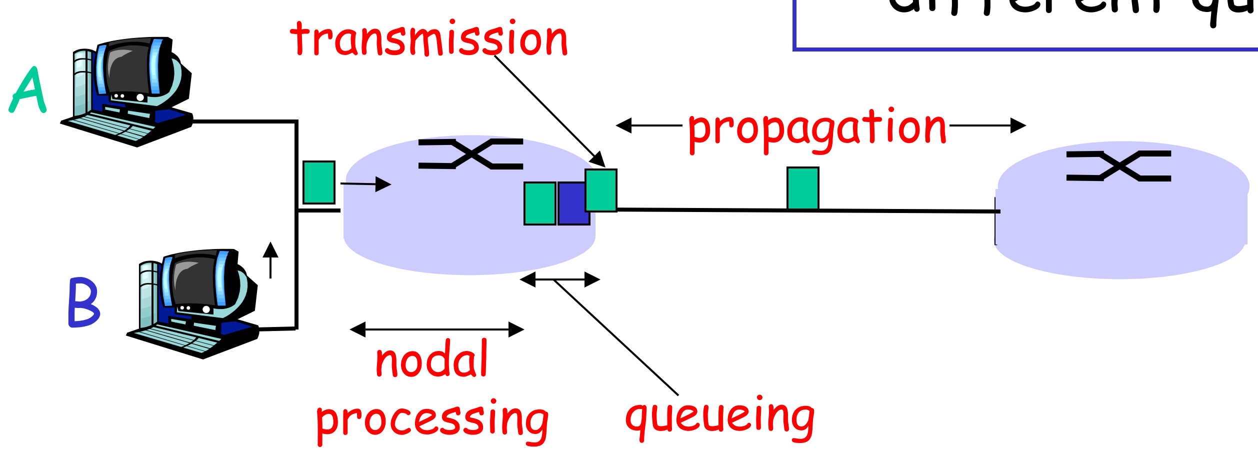
3. Transmission delay:

- R=link bandwidth (bps)
- L=packet length (bits)
- time to send bits into link = L/R

4. Propagation delay:

- d = length of physical link
- s = propagation speed in medium (~2×10^8 m/sec)
- propagation delay = d/s

Note: s and R are *very* different quantities!

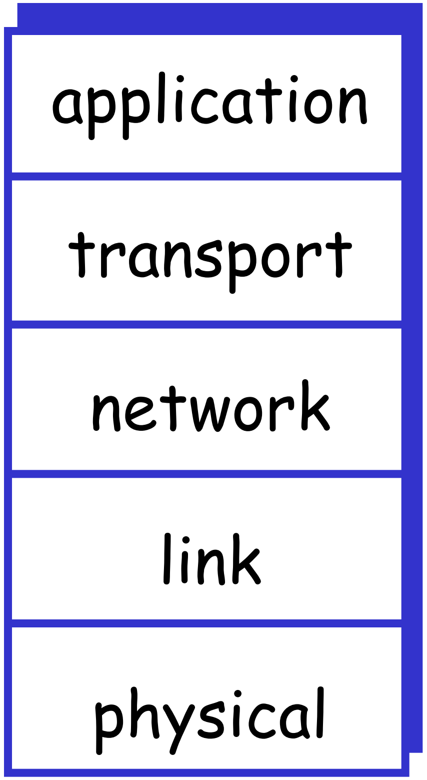


Packet Loss

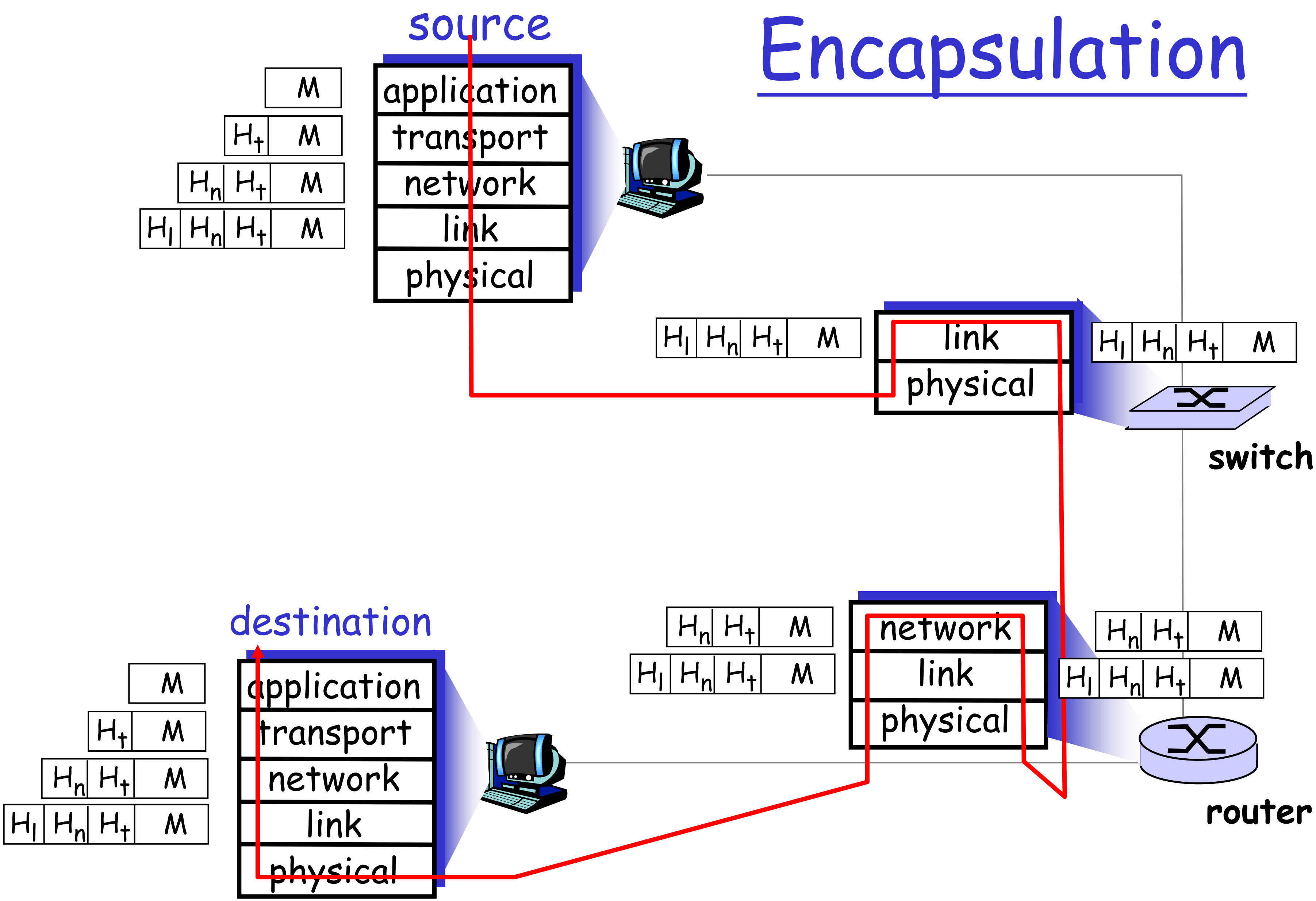
- A queue (a.k.a. buffer) preceding a link has finite capacity
- when packets arrive to a full queue, some packets have to be dropped (a.k.a. lost)
- lost packets may be retransmitted by the previous node, by source end system, or not retransmitted at all

Internet Protocol Stack

- application: supporting network applications
  - FTP, SMTP, HTTP
- transport: host-host data transfer
  - TCP, UDP
- network: routing of datagrams from source to destination
  - IP, routing protocols
- link: data transfer between neighboring network elements
  - PPP, Ethernet
- physical: bits "on the wire"



Encapsulation



What you Learnt!: Application Layer

Goals:

- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
- learnt about protocols by examining popular application-level protocols
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- programming network applications
  - socket API

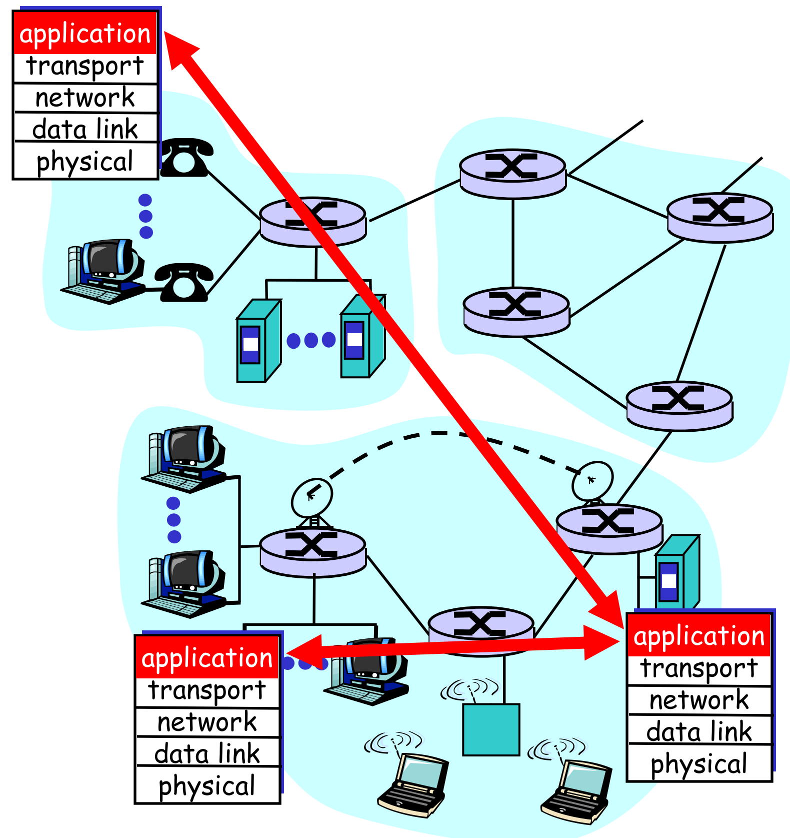
Creating a Network App

Write programs that

- run on different end systems and
- communicate over a network.
- e.g., Web: Web server software communicates with browser software

Almost NO app software written for devices in network core

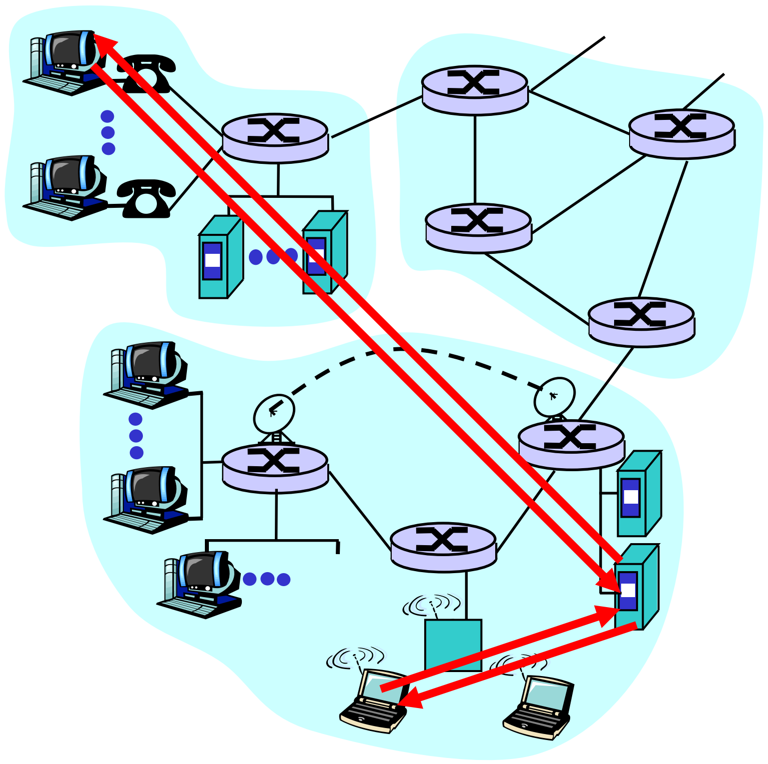
- Network core devices do not function at app layer
- This design allows for rapid app development



Application Architectures

- Client-server
- Peer-to-peer (P2P)
- Hybrid of client-server and P2P

Client-Server Archicture



server:

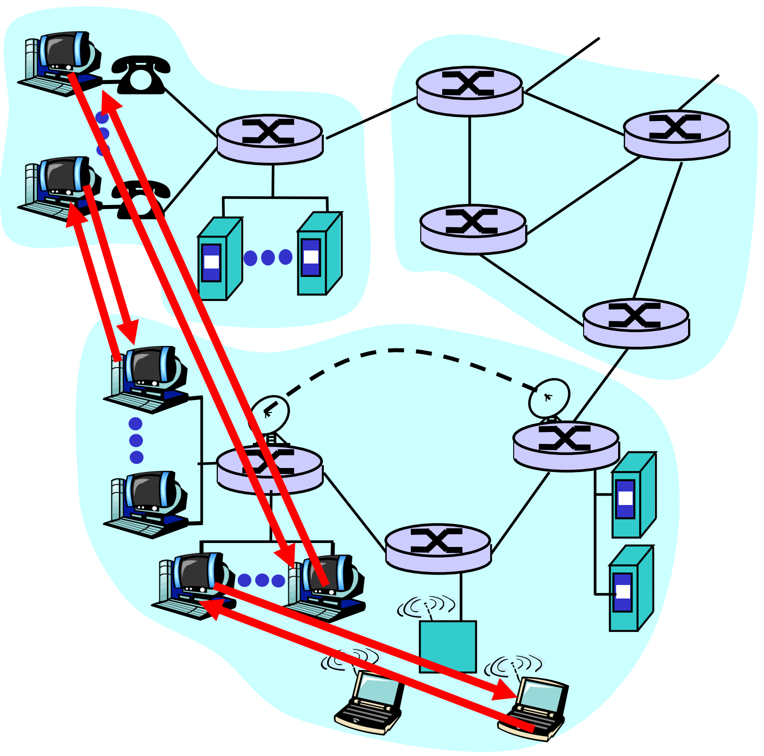
- always-on host
- permanent IP address
- server farms for scaling

clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

Pure P2P Architecture

- no (or very few) always-on servers
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses
- example: Gnutella



Highly scalable
But difficult to manage
Hard to be made reliable

Hybrid of Client-Server and P2P

Napster

- File transfer P2P
- File search centralized:
  - Peers register content at central server
  - Peers query same central server to locate content

Instant messaging

- Chatting between two users is often P2P
- Presence detection/location centralized:
  - User registers its IP address with central server when it comes online
  - User contacts central server to find IP addresses of buddies

DNS: Domain Name System

People: many identifiers:
- SSN, name, passport #

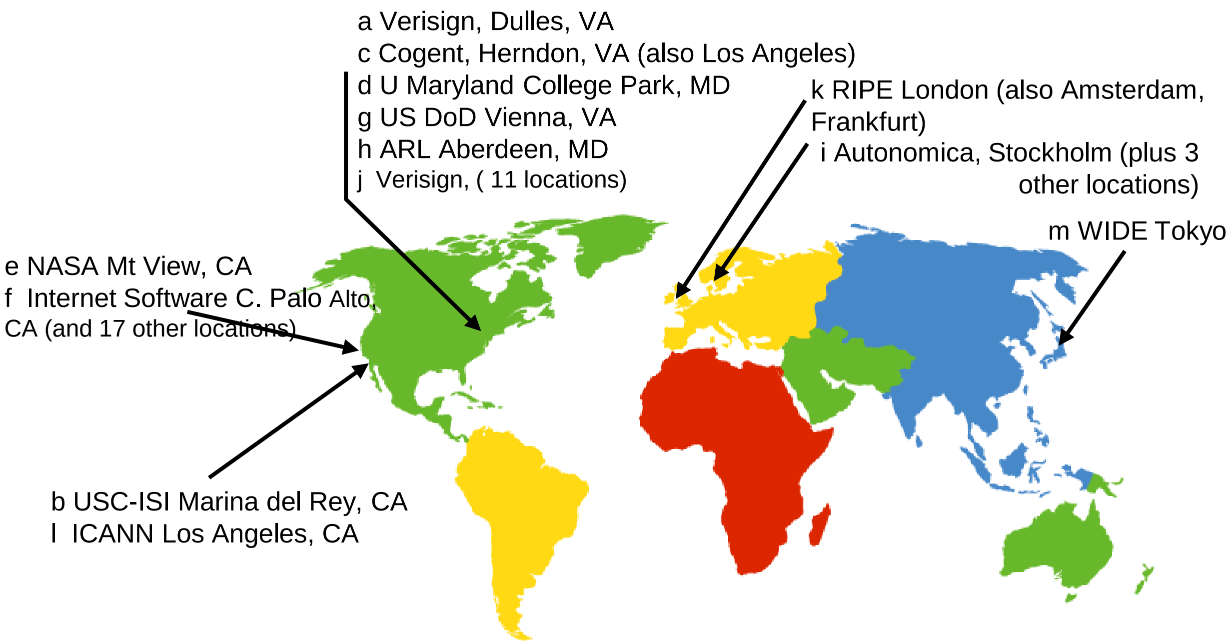
Internet hosts, routers:
- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.yahoo.com - used by humans

Q: map between IP addresses and name ?

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

DNS: Root Name Servers

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



13 root name servers worldwide

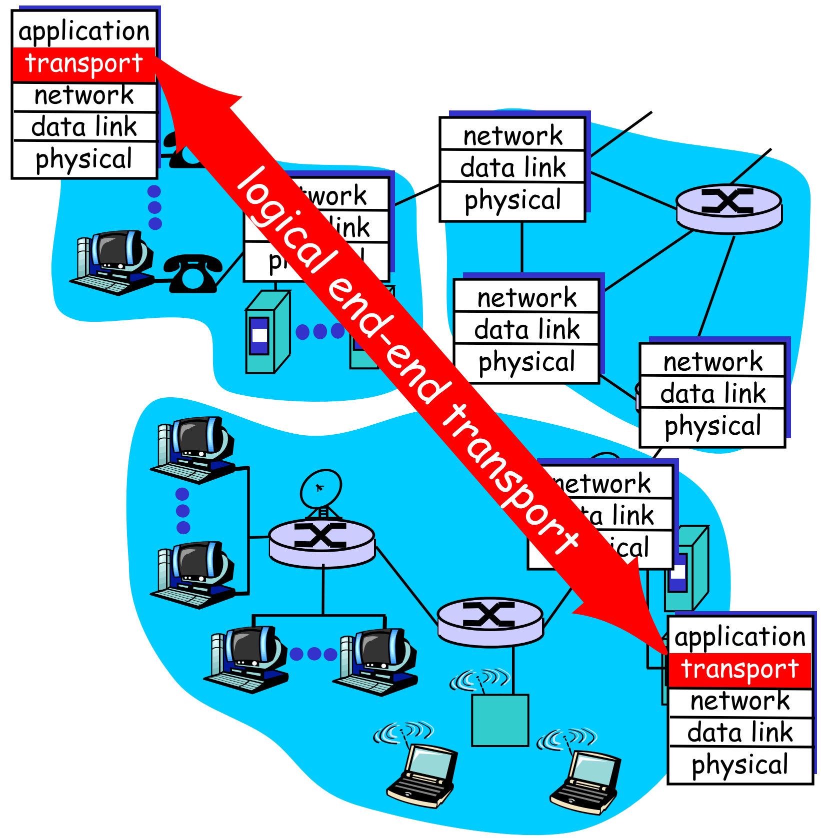
What you Learnt!: Transport Layer

Goals:

- understand principles behind transport layer services:
  - Multiplexing and demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- learnt about transport layer protocols in the Internet:
  - UDP: connectionless transport
  - TCP: connection-oriented transport
  - TCP congestion control

Transport Services and Protocols

- provide *logical communication* between app processes running on different hosts
- transport 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



Internet Transport Protocols Services

TCP service:

- *connection-oriented:* setup required between client and server processes
- *reliable transport* between sending and receiving processes
- *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 processes
- does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee

Q: why bother? Why is there a UDP?

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?

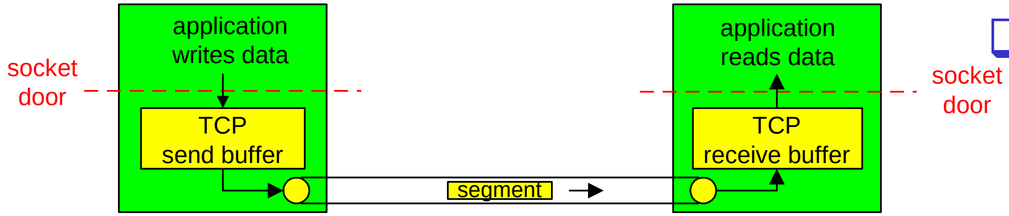
- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired

Attractive to applications requiring high bandwidth, low latency, and a best-effort service

TCP: Overview

RFCs: 793, 1122, 1323, 2018, 2581

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size
- *send & receive buffers*



- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) initiates sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

TCP Reliable Data Transfer

- On top of IP's unreliable service
- Pipelined segments
- Cumulative acks
- TCP uses single retransmission timer

- Retransmissions are triggered by:
  - timeout events
  - duplicate acks

TCP Sender Events:

data rcvd from app:

- Create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer if not already running (think of timer as for oldest unacked segment)
- expiration interval: `TimeOutInterval`

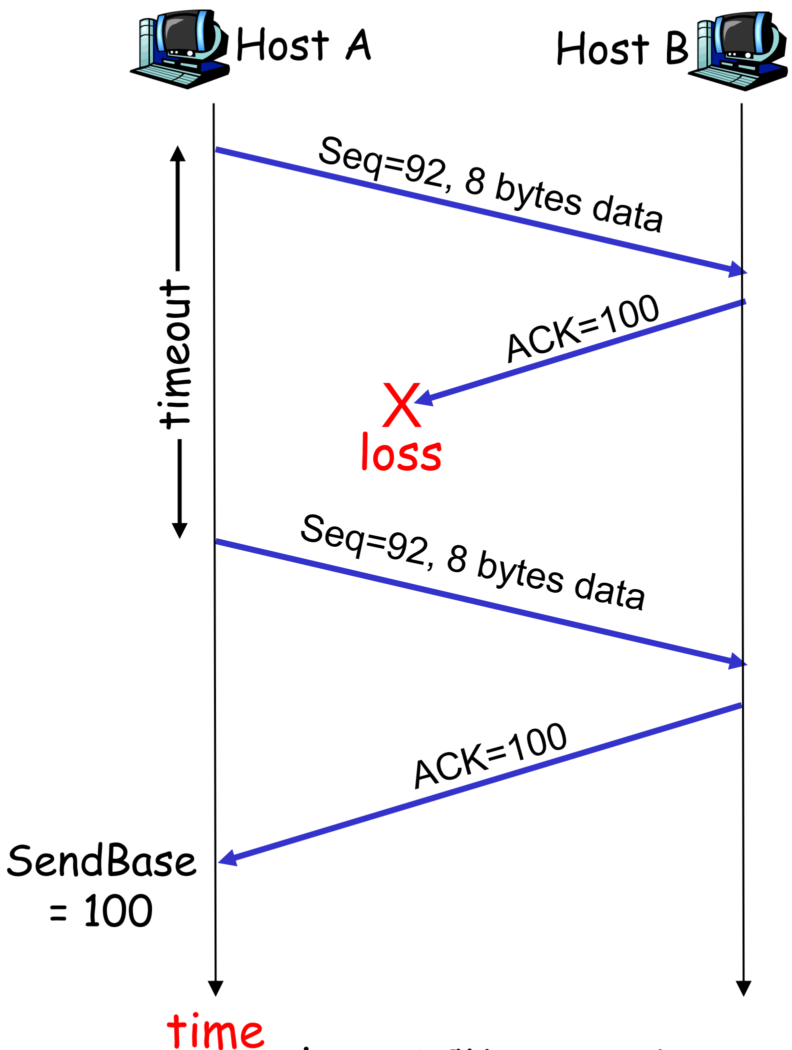
timeout:

- retransmit segment that caused timeout
- restart timer

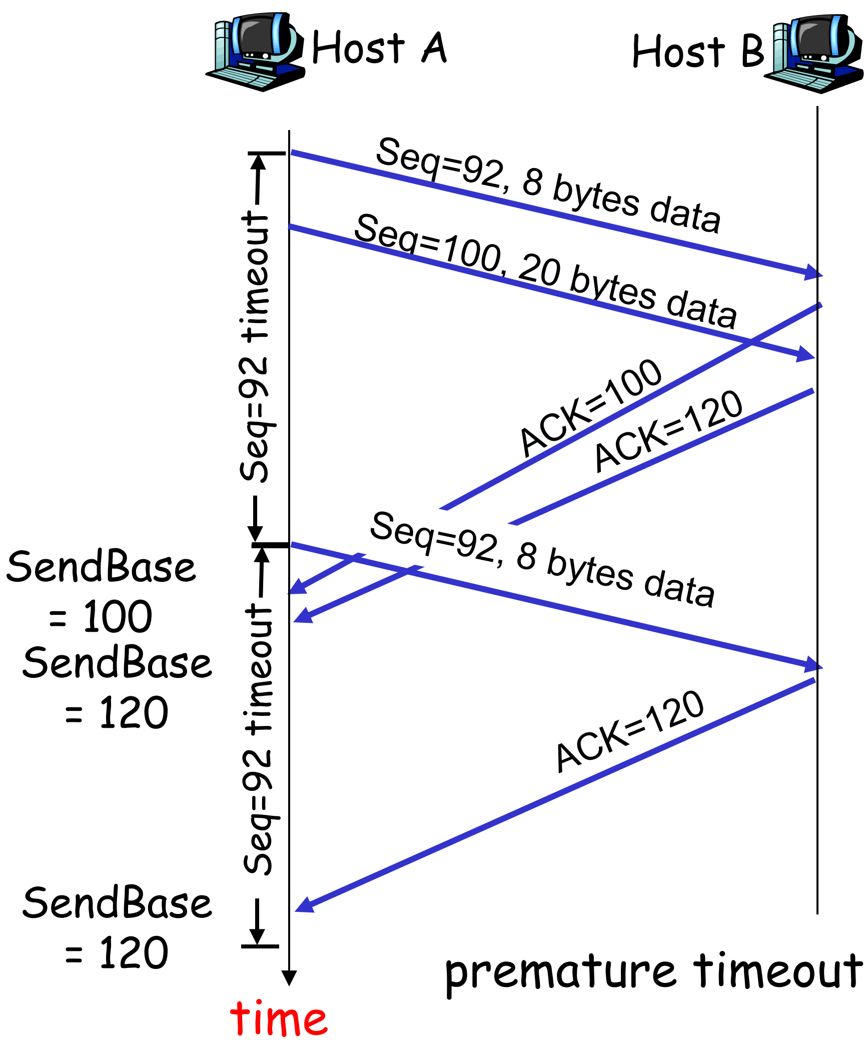
Ack rcvd:

- If acknowledges previously unacked segments
  - update what is known to be acked
  - start timer if there are outstanding segments

TCP: Retransmission Scenarios

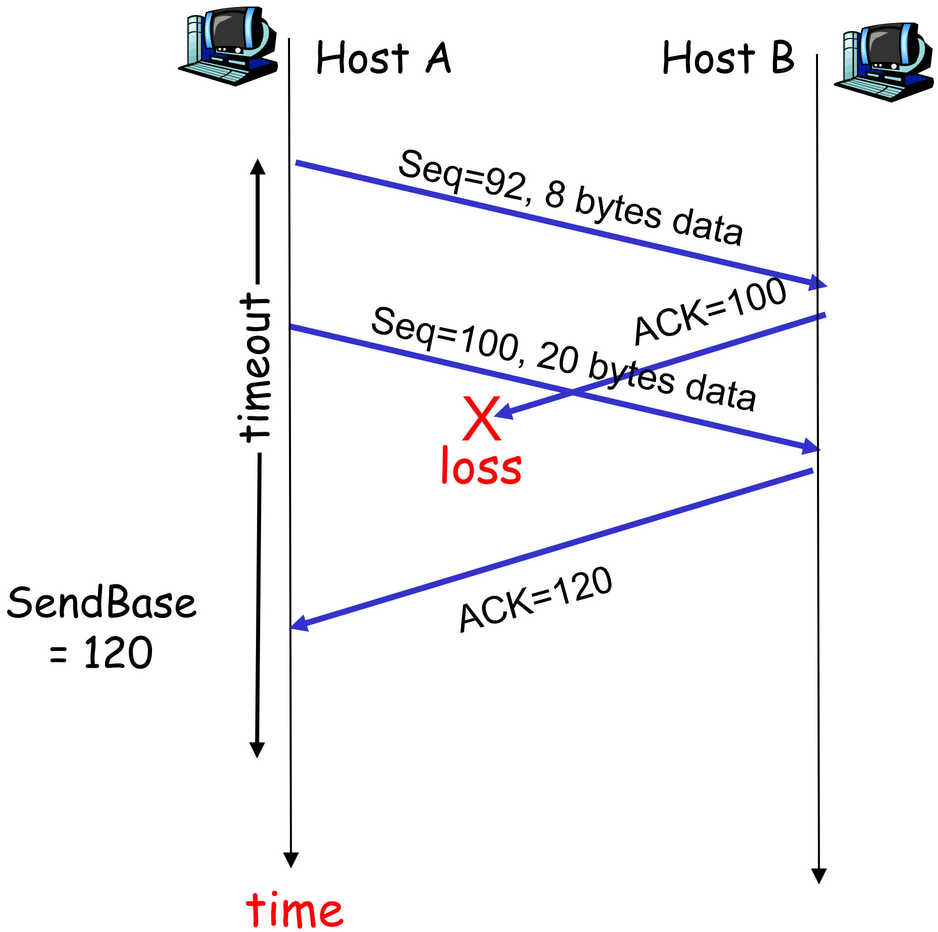


lost ACK scenario



premature timeout

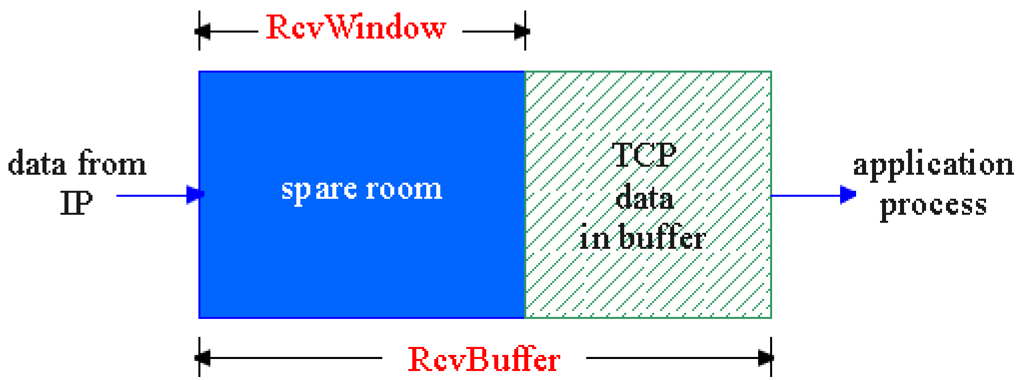
TCP Retransmission Scenarios (more)



Cumulative ACK scenario

TCP Flow Control

- receive side of TCP connection has a receive buffer:



- app process may be slow at reading from buffer

flow control

sender won't overflow receiver's buffer by transmitting too much, too fast

- speed-matching service: matching the send rate to the receiving app's drain rate

Principles of Congestion Control

Congestion:

- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)

What you Learnt!: Network Layer

Goals:

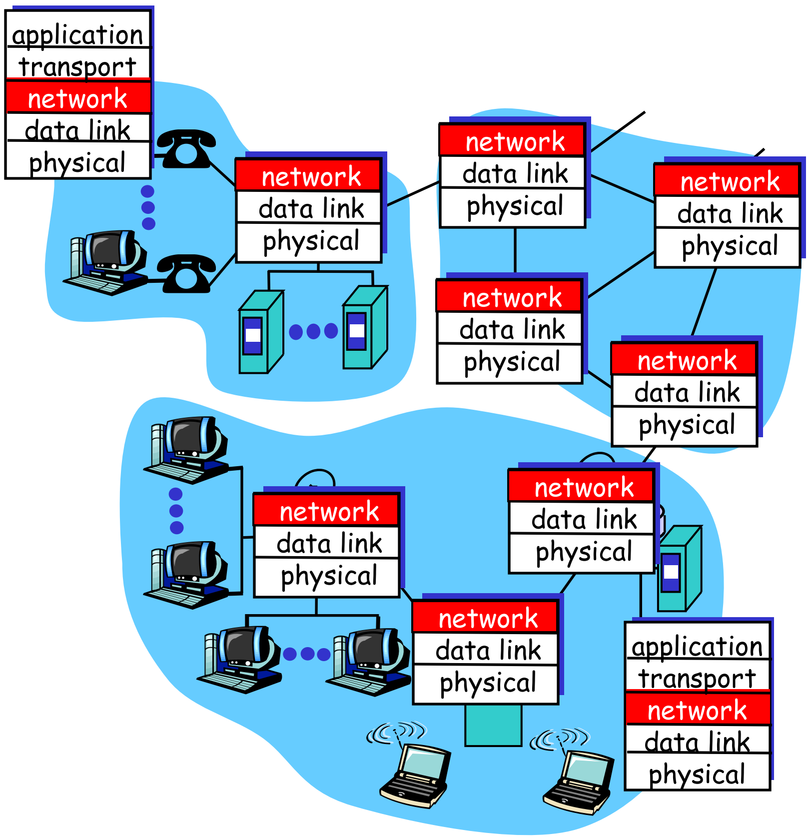
- understand principles behind network layer services:
  - routing (path selection)
  - dealing with scale
  - how a router works
  - advanced topics: IPv6
- instantiation and implementation in the Internet

Overview:

- network layer services
- routing principles: path selection
- hierarchical routing
- IP
- Internet routing protocols
  - intra-domain
  - inter-domain
- what's inside a router?
- IPv6

Network Layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on rcving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- Router examines header fields in all IP datagrams passing through it



Key Network-Layer Functions

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to dest.
  - *Routing algorithms*

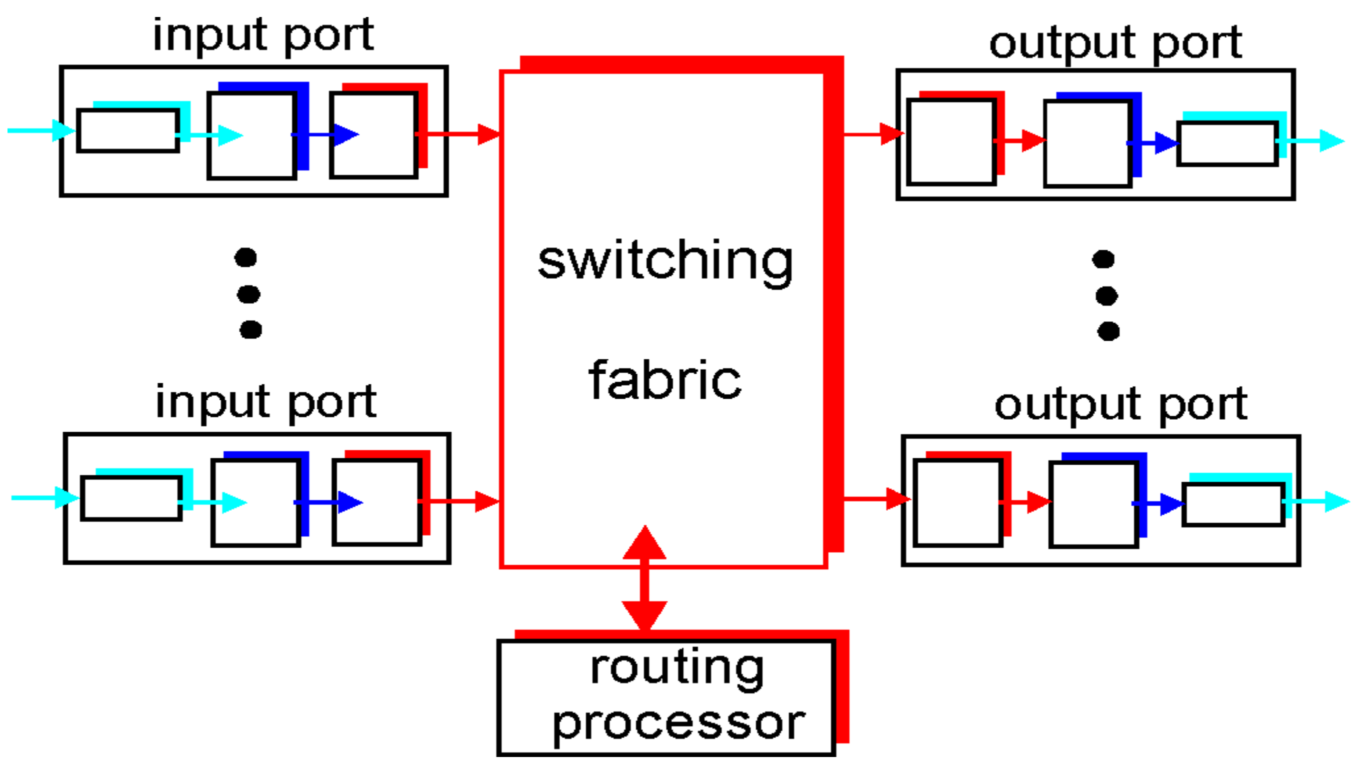
analogy:

- routing: process of planning trip from source to dest
- forwarding: process of getting through single interchange

Router Architecture Overview

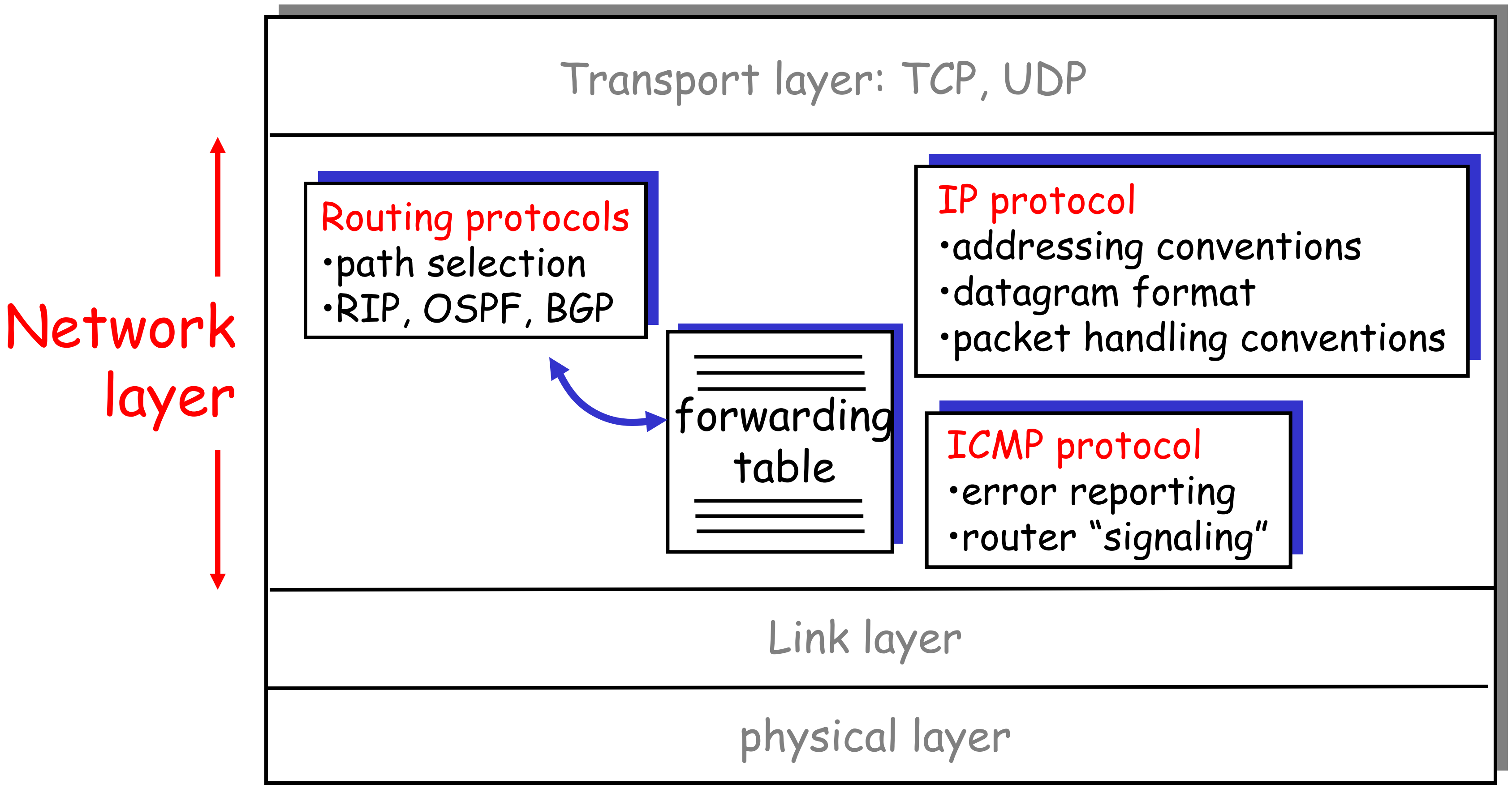
Two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



The Internet Network layer

Host, router network layer functions:



IP Datagram Format

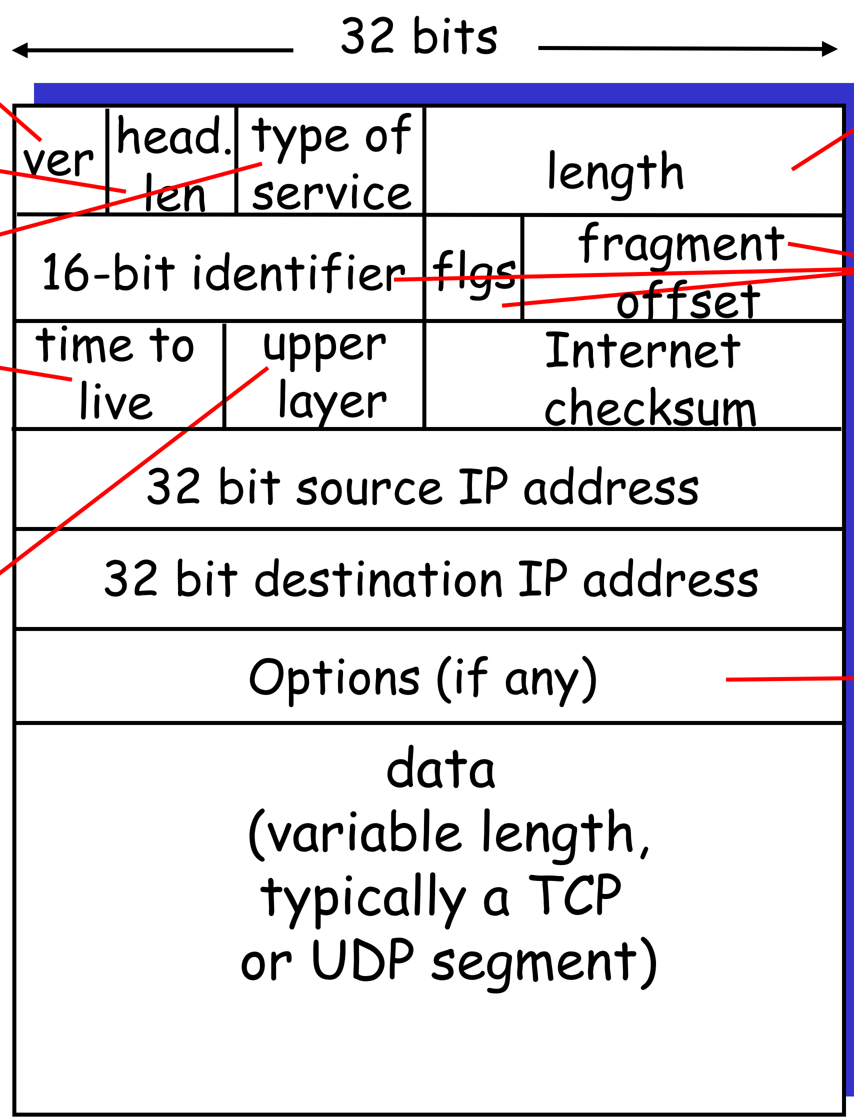
IP protocol version number

header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to



total datagram length (bytes)

for fragmentation/ reassembly

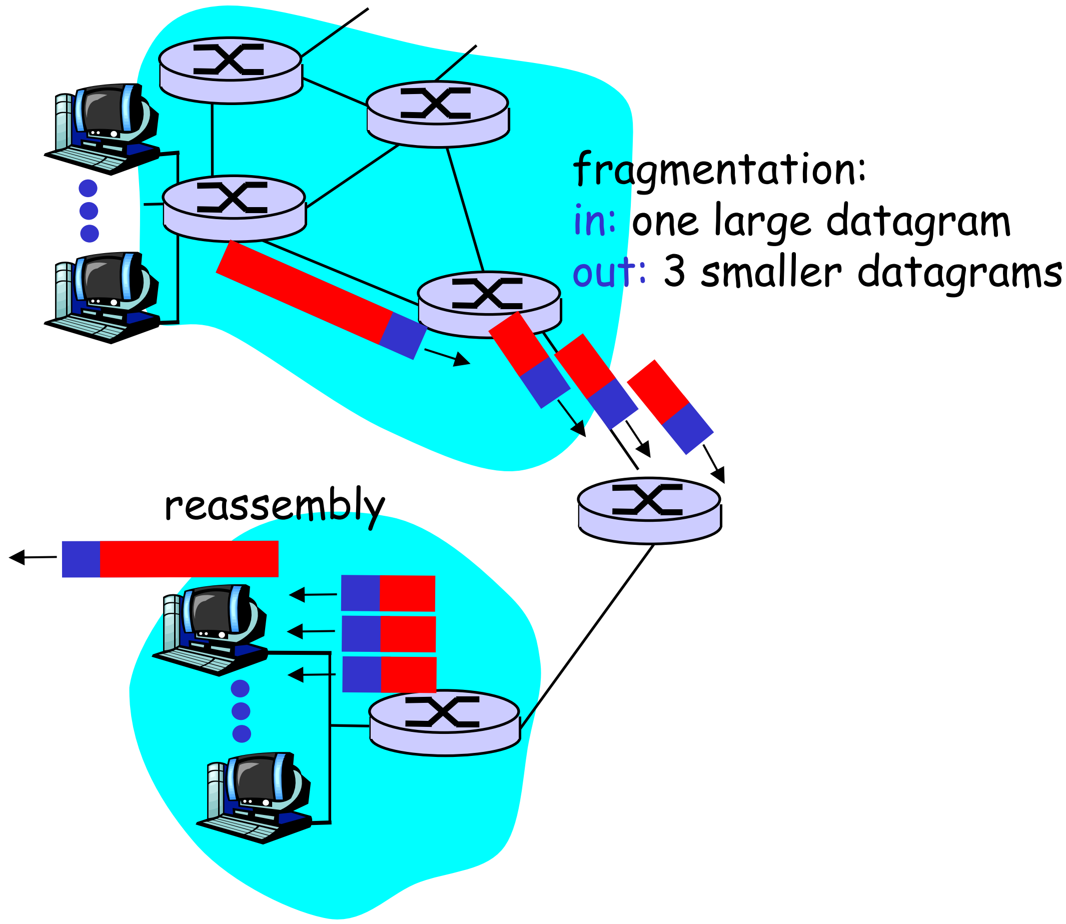
E.g. timestamp, record route taken, specify list of routers to visit.

<u>how much overhead with TCP?</u>

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

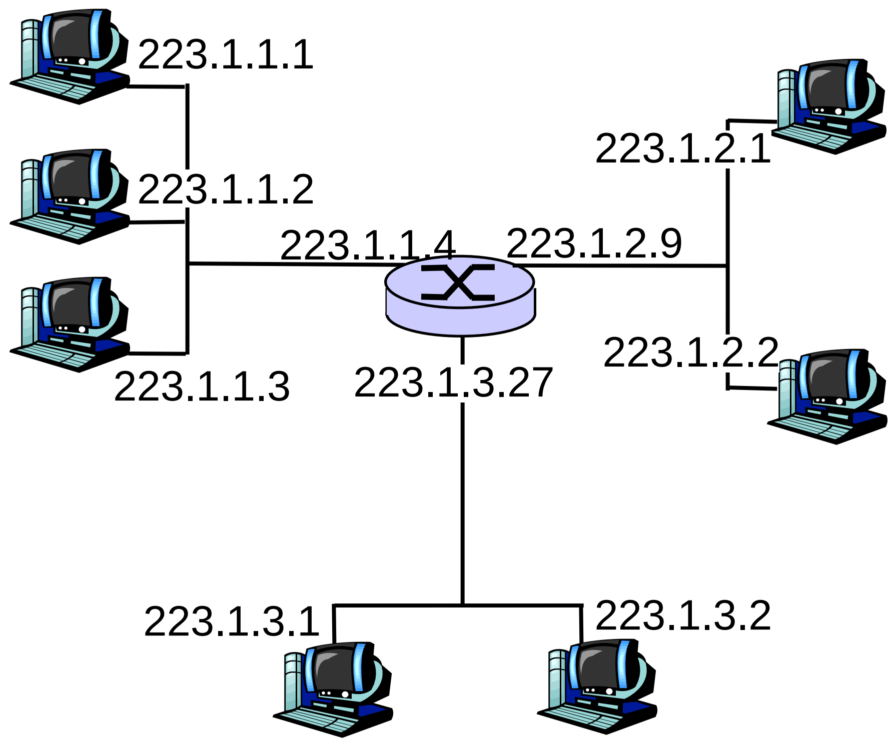
IP Fragmentation & Reassembly

- network links have MTU (Maximum Transmission Unit) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP Addressing: Introduction

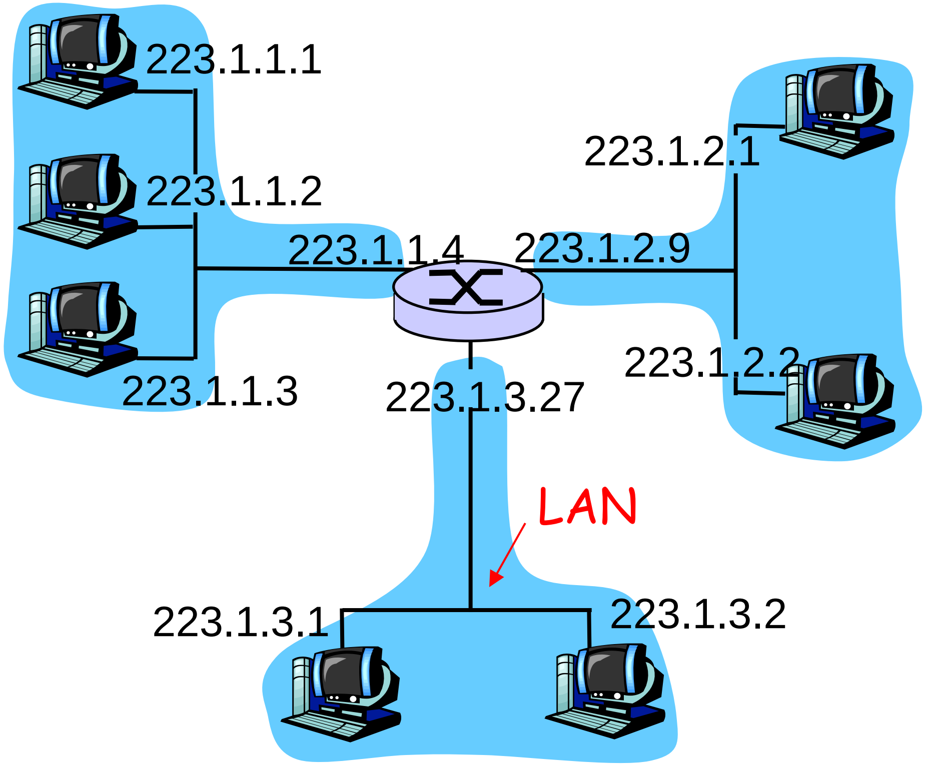
- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with each interface



$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

Subnets

- IP address:
  - subnet part (high order bits)
  - host part (low order bits)
- *What's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router



network consisting of 3 subnets

IP Addresses: How to Get One?

Q: How does *host* get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->TCP/IP->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

(more in next chapter)

IP Addresses: How to Get One?

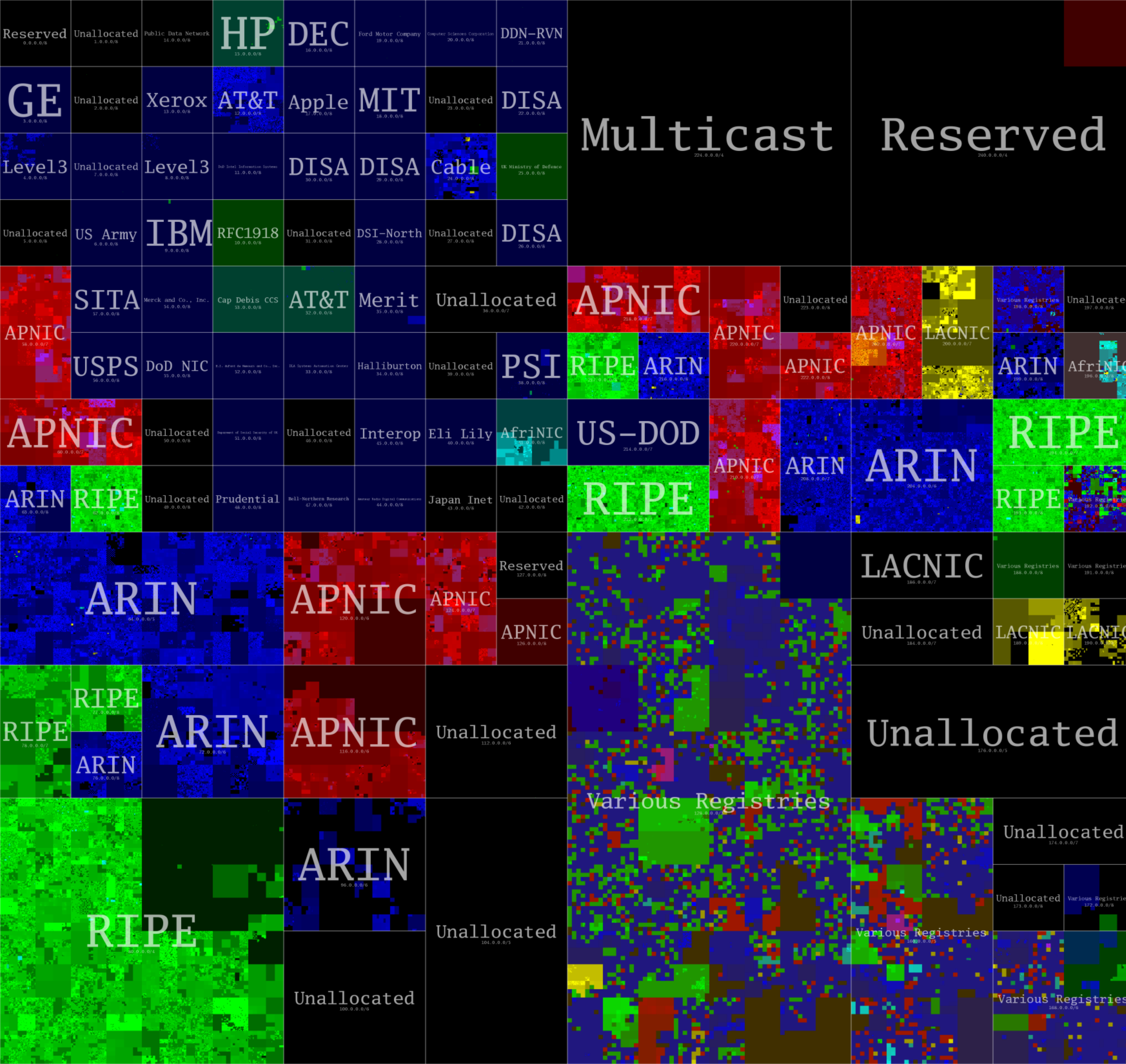
Q: How does *network* get a "subnet" of IP addr?

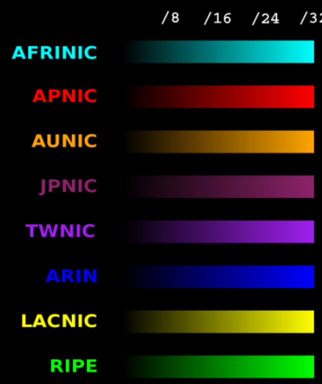
A: gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000 00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010 00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100 00000000	200.23.20.0/23
...	....	....
Organization 7	11001000 00010111 00011110 00000000	200.23.30.0/23

RIRs IPv4 Whois Map

October 2007





This map shows the characteristics of IPv4 "whois" data from the Regional Internet Registries.

Each pixel in the full-size image represents a single /24 network containing up to 256 hosts. The color of each pixel corresponds to the RIR(s) where the address space is listed.

Areas are shaded with varying levels of transparency. Larger blocks of address space are more transparent, which also makes them darker. Thus, varying shades of a color indicate different-sized blocks of addresses appearing in the whois data.

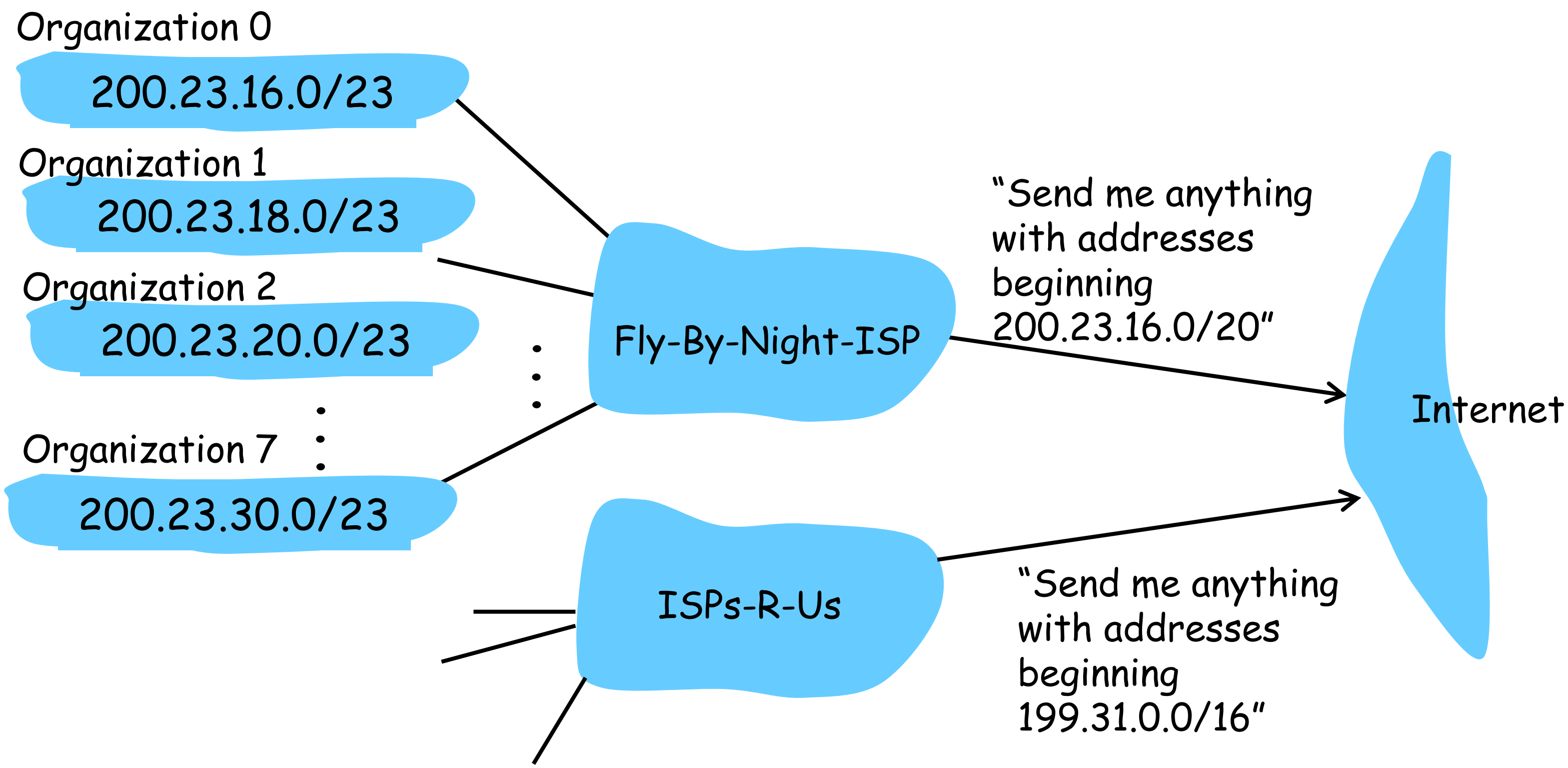
Furthermore, some parts of the address space are listed in multiple RIRs, for various reasons. When this happens, new colors may be created. This is especially evident in the areas labeled "Various Registries" where you can see large areas of brown, created by combining red (ARIN) and green (RIPE) together.

caida

The Measurement Factory

Hierarchical Addressing: Route Aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Routing Algorithm Classification

Global or decentralized information?

Global:
- all routers have complete topology, link cost info
- "link state" algorithms

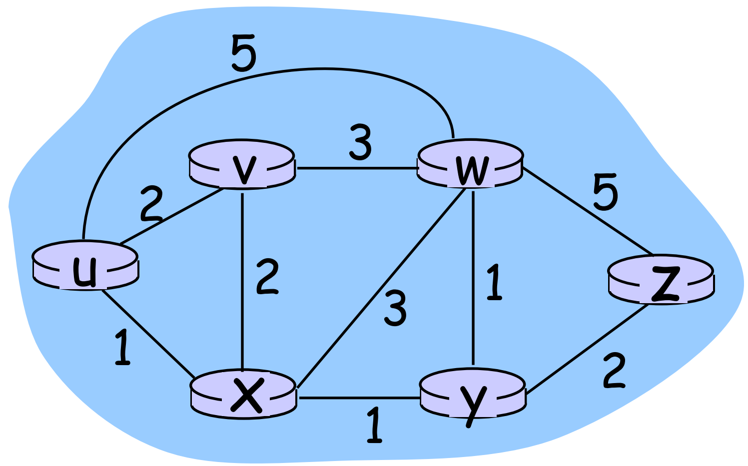
Decentralized:
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:
- routes change slowly over time

Dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes

Graph Abstraction



Graph: G = (N,E)

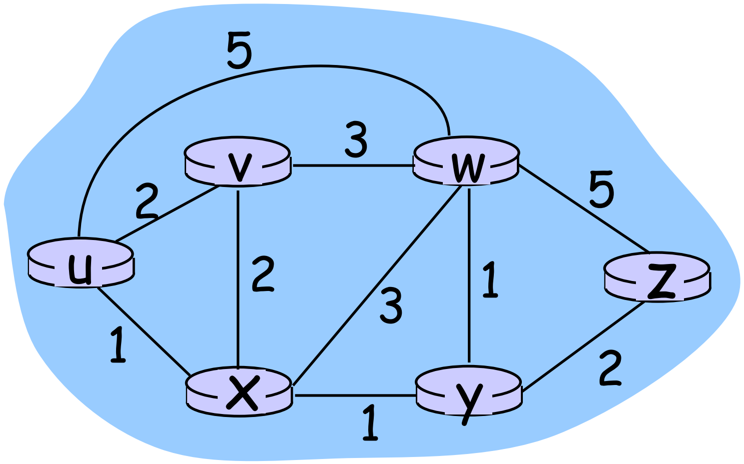
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph Abstraction: Costs



• $c(x,x')$ = cost of link (x,x')

- e.g., $c(w,z) = 5$

• cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Virtual Circuits

"source-to-dest path behaves much like telephone circuit"

- performance-wise
- network actions along source-to-dest path

- call setup/teardown for each call *before*/after the data flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC

What you Learnt!: The Data Link Layer

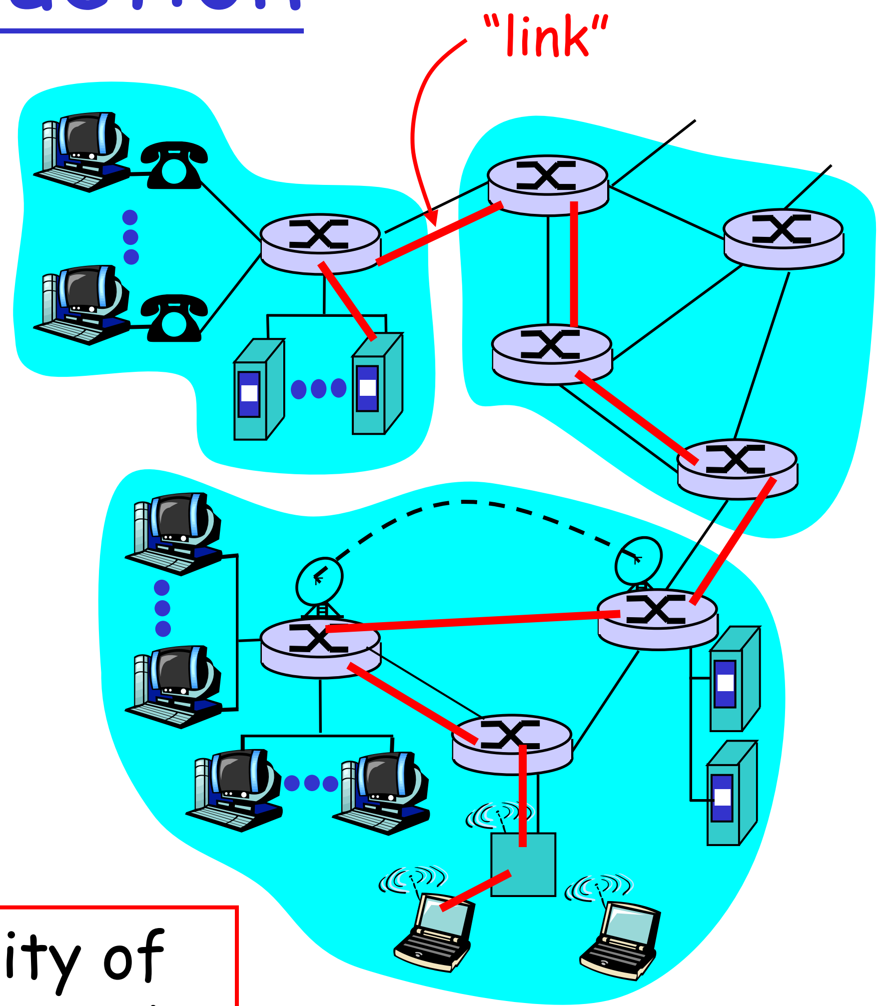
Goals:

- understand principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control: *done!*
- instantiation and implementation of various link layer technologies

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

Link Layer Services

- Framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- Reliable delivery between adjacent nodes
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?

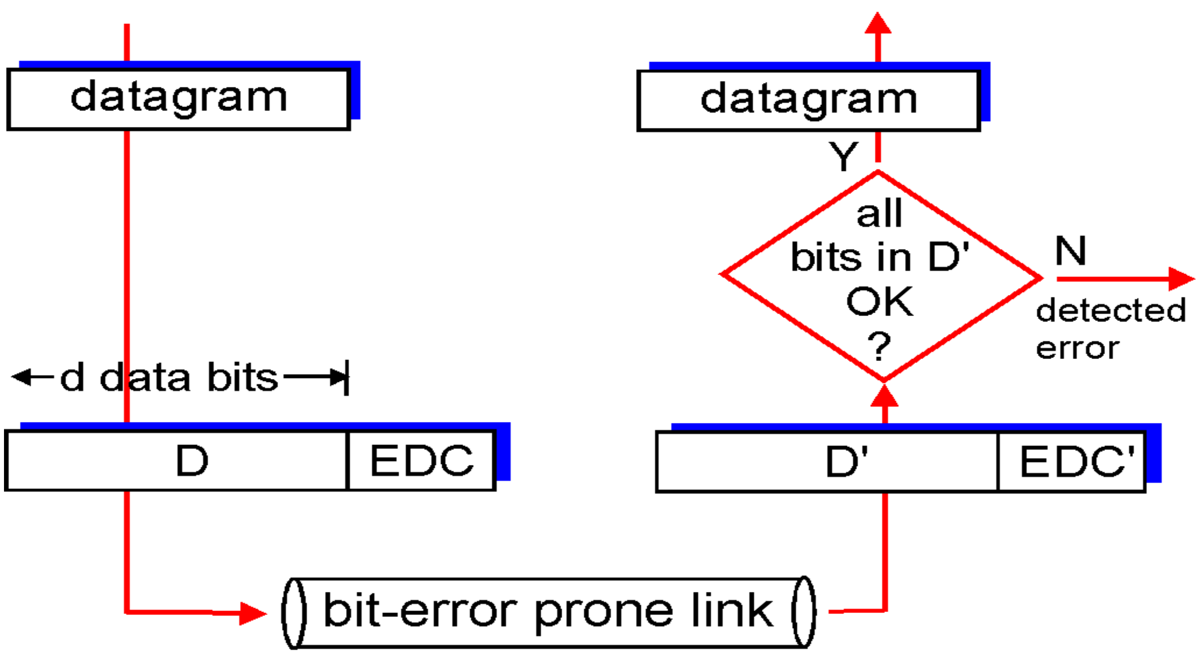
Link Layer Services (more)

- *Flow Control:*
  - pacing between adjacent sending and receiving nodes
- *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
- *Half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

Error Detection

EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

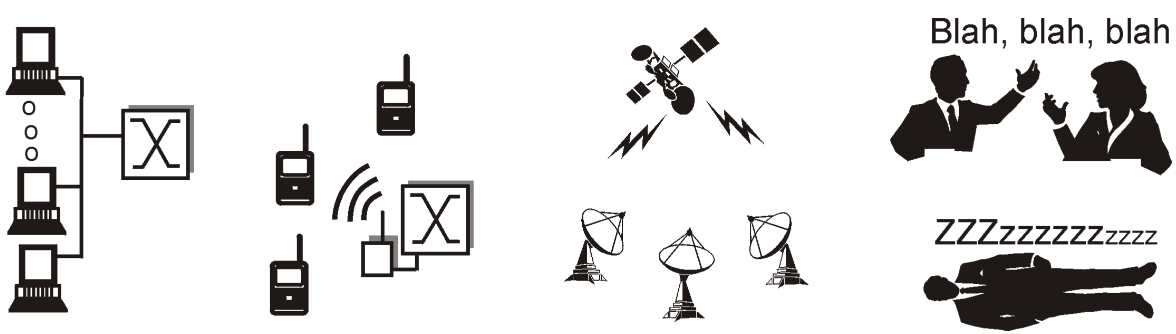
- Error detection not 100% reliable!
    - protocol may miss some errors, but rarely
    - larger EDC field yields better detection and correction



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)
  - traditional Ethernet
  - upstream HFC
  - 802.11 wireless LAN



shared wire (e.g. Ethernet) | shared wireless (e.g. Wavelan) | satellite | cocktail party

Multiple Access Protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - collision if node receives two or more signals at the same time

multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

Ideal Mulitple Access Protocol

Broadcast channel of rate R bps

1. When one node wants to transmit, it can send at rate R.
2. When M nodes want to transmit, each can send at average rate R/M
3. Fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. Simple

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!

CSMA/CD (Collision Detection)

CSMA/CD: carrier sensing, deferral as in CSMA
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs
    - The transmitting node cannot "listen" during a transmission
    - Collision occurs at the receive side

"Taking Turns" MAC protocols

channel partitioning MAC protocols:

- share channel efficiently and fairly at high load
- inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

Random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

"taking turns" protocols

look for best of both worlds!

LAN Technologies

Data link layer so far:

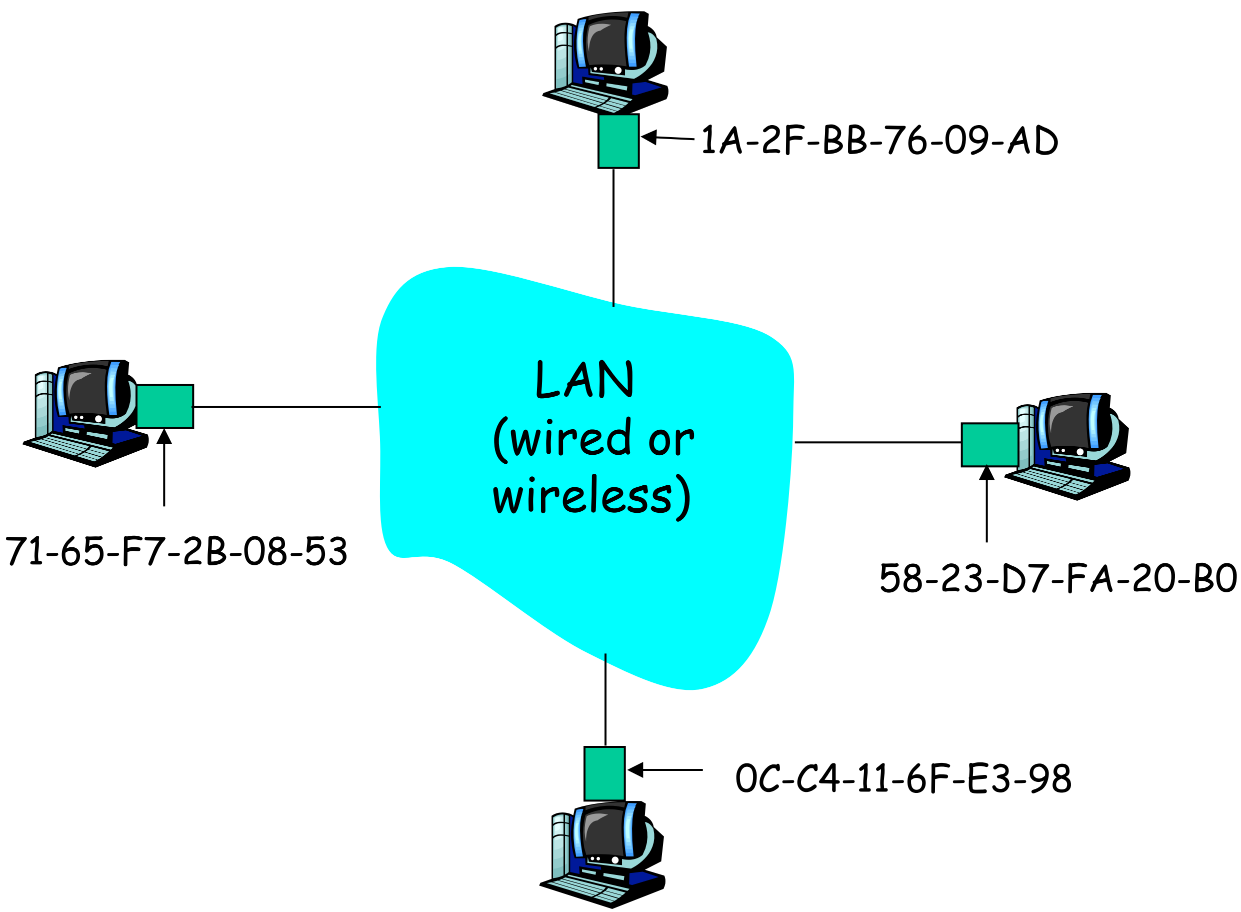
- services, error detection/correction, multiple access

Next: LAN technologies

- addressing
- Ethernet
- hubs, switches
- PPP

LAN Addresses and ARP

Each adapter on LAN has unique LAN address



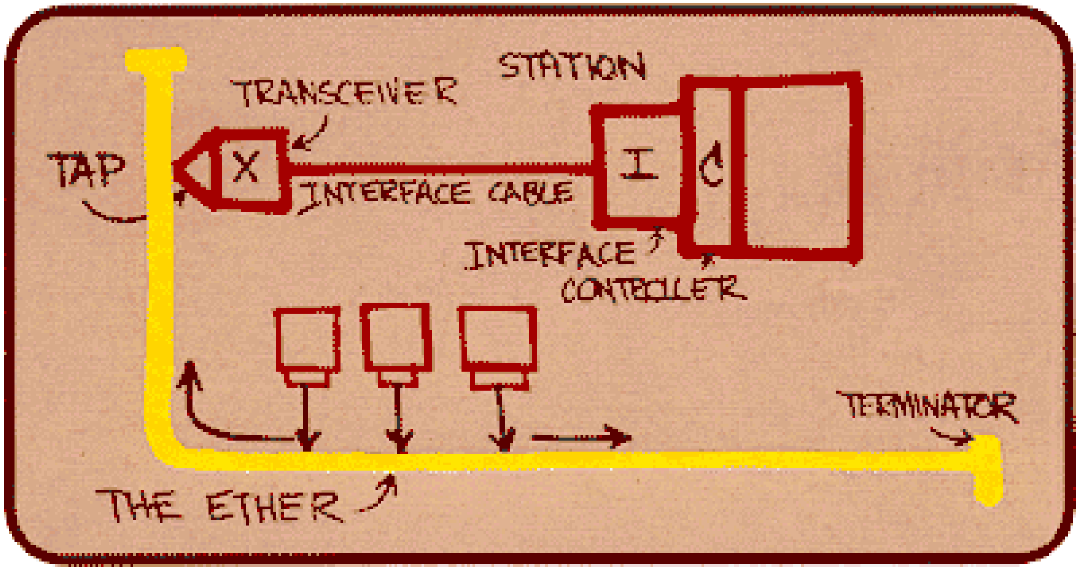
Broadcast address =
FF-FF-FF-FF-FF-FF

= adapter

Ethernet

"dominant" wired LAN technology:

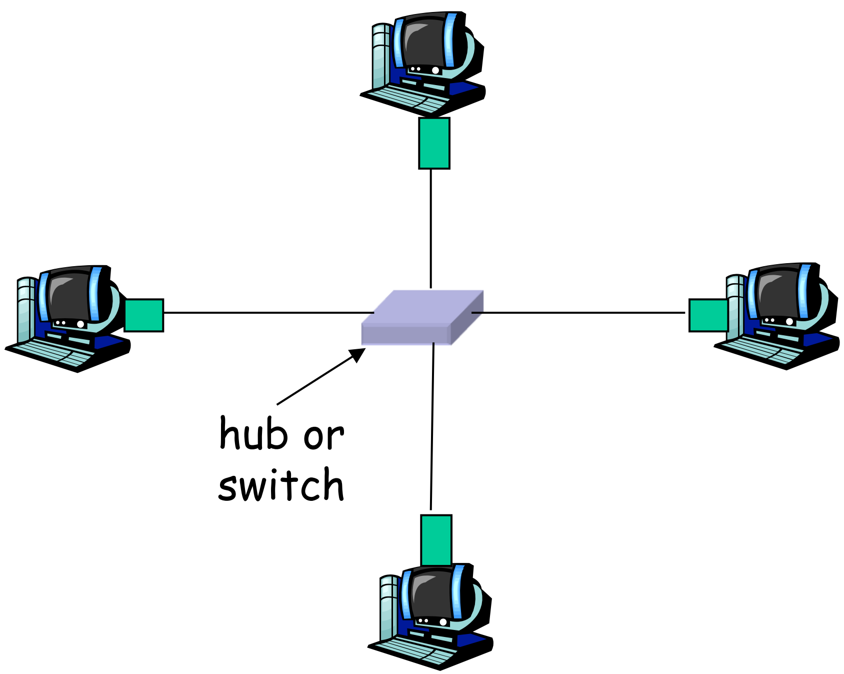
- cheap $20 for 100Mbs!
- first widely used LAN technology
- Simpler, cheaper than token ring LANs and ATM
- Kept up with speed race: 10 Mbps – 10 Gbps



Metcalfe's Ethernet sketch

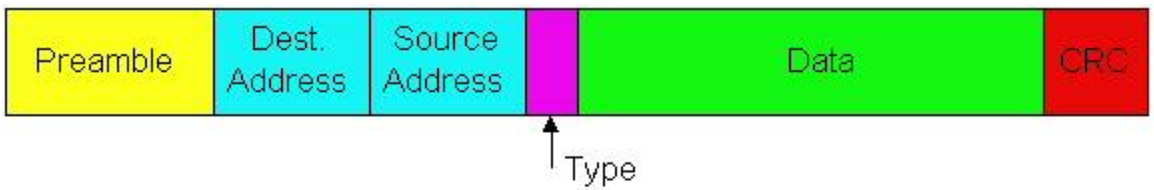
Star Topology

- Bus topology popular through mid 90s
- Now star topology prevails
- Connection choices: hub or switch (more later)



Ethernet Frame Structure

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

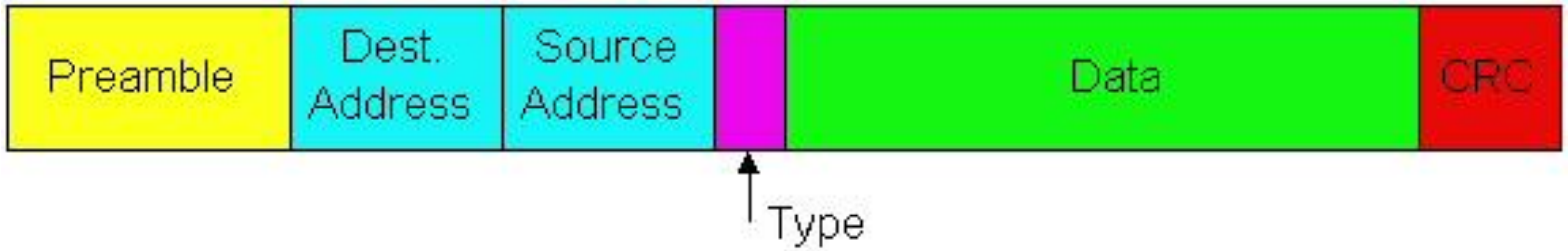


Preamble:

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

Ethernet Frame Structure (more)

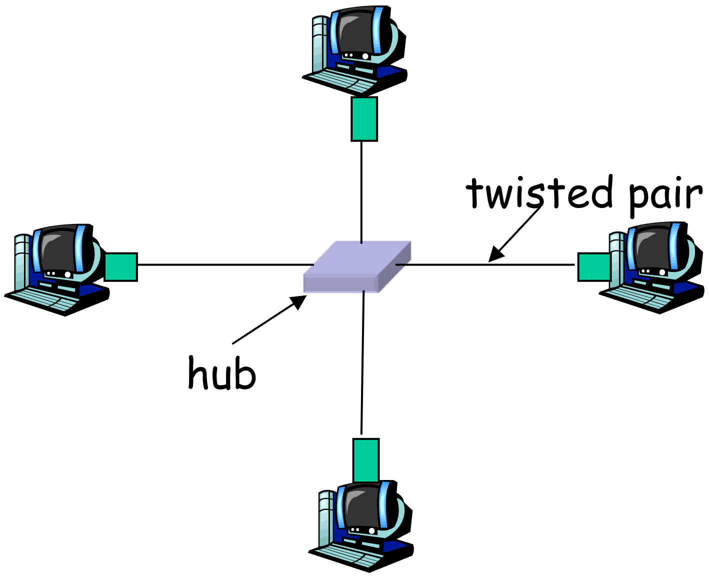
- Addresses: 6 bytes
  - if adapter receives frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to net-layer protocol
  - otherwise, adapter discards frame
- Type: indicates the higher layer protocol (mostly IP but others may be supported such as Novell IPX and AppleTalk)
- CRC: checked at receiver, if error is detected, the frame is simply dropped



Hubs

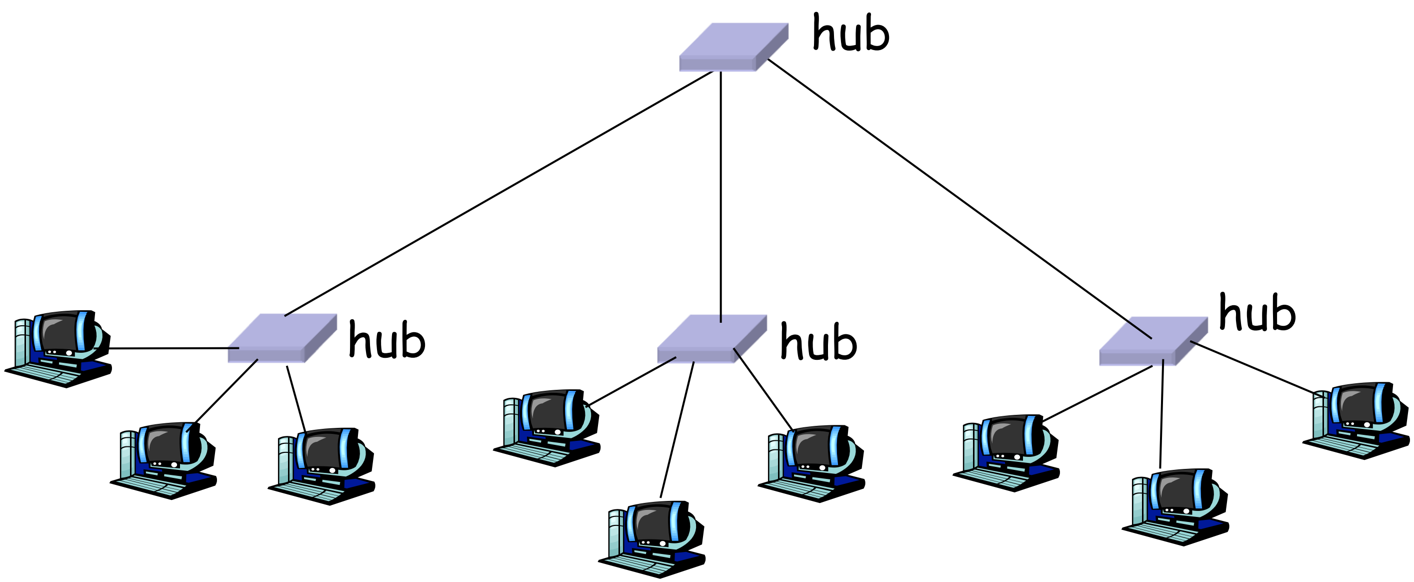
Hubs are essentially physical-layer repeaters:

- bits coming from one link go out all other links
- at the same rate
- no frame buffering
- no CSMA/CD at hub: adapters detect collisions
- provides net management functionality



Interconnecting with Hubs

- Backbone hub interconnects LAN segments
- Extends max distance between nodes
- But individual segment collision domains become one large collision domain
- Can't interconnect 10BaseT & 100BaseT

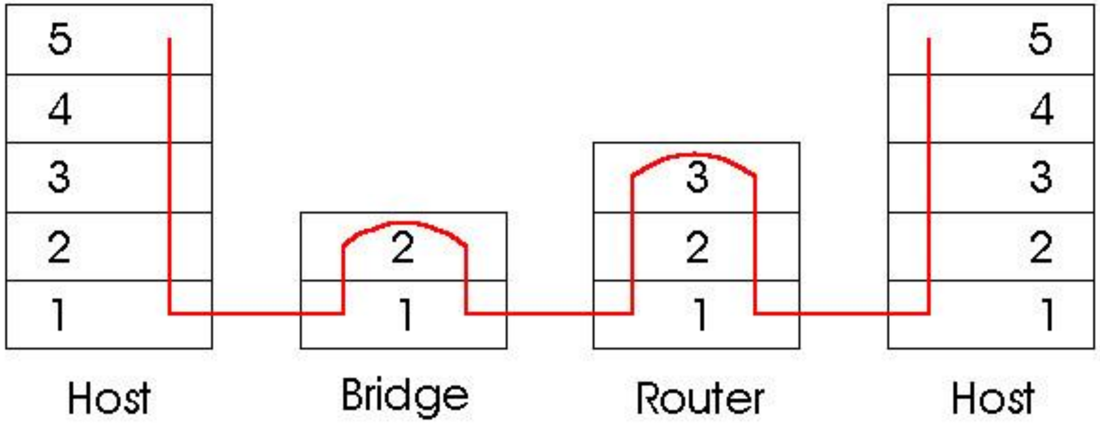


Switch

- Link layer device
  - stores and forwards Ethernet frames
  - examines frame header and selectively forwards frame based on MAC dest address
  - 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

Switches vs. Routers

- both store-and-forward devices
  - routers: network layer devices (examine network layer headers)
  - switches are link layer devices
- routers maintain routing tables, implement routing algorithms
- switches maintain switch tables, implement filtering, learning algorithms



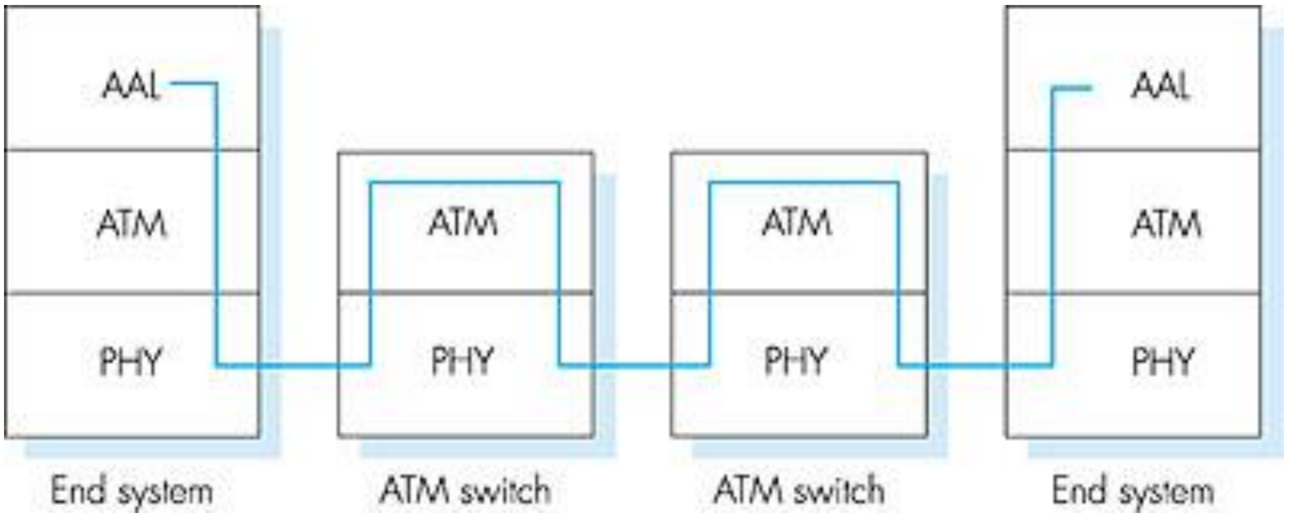
ATM and MPLS

- ATM, MPLS separate networks in their own right
  - different service models, addressing, routing from Internet
- viewed by Internet as logical link connecting IP routers
  - just like dialup link is really part of separate network (telephone network)
- ATM, MPSL: of technical interest in their own right

Asynchronous Transfer Mode: ATM

- **1990's/00 standard for high-speed** (155Mbps to 622 Mbps and higher) *Broadband Integrated Service Digital Network* architecture
- Goal: *integrated, end-end transport of carry voice, video, data*
  - meeting timing/QoS requirements of voice, video (versus Internet best-effort model)
  - "next generation" telephony: technical roots in telephone world
  - packet-switching (fixed length packets, called "cells") using virtual circuits

ATM Architecture

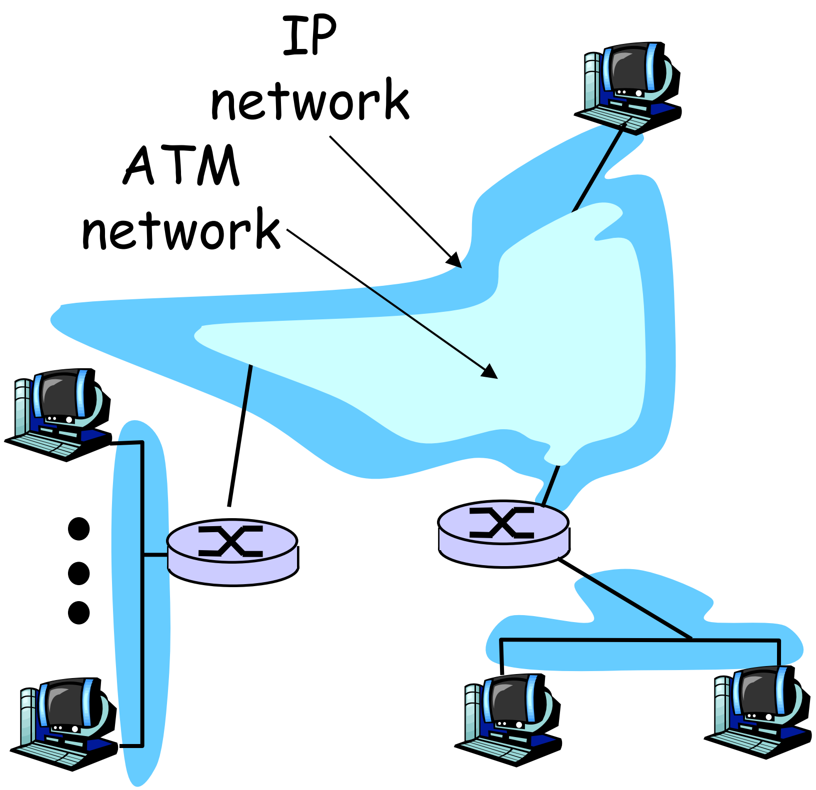


- adaptation layer: only at edge of ATM network
  - data segmentation/reassembly
  - roughly analagous to Internet transport layer
- ATM layer: "network" layer
  - cell switching, routing
- physical layer

ATM: Network or Link Layer?

Vision: end-to-end transport: "ATM from desktop to desktop"
- ATM *is* a network technology

Reality: used to connect IP backbone routers
- "IP over ATM"
- ATM as switched link layer, connecting IP routers



ATM Layer

Service: transport cells across ATM network

- analogous to IP network layer
- very different services than IP network layer

Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

ATM Layer: Virtual Circuits

- VC transport: cells carried on VC from source to dest
  - call setup, teardown for each call
  - each packet carries VC identifier (not destination ID)
  - *every* switch on source-dest path maintain "state" for each passing connection
  - link,switch resources (bandwidth, buffers) may be *allocated* to VC: to get circuit-like performance
- Permanent VCs (PVCs)
  - long lasting connections
  - typically: "permanent" route between to IP routers
- Switched VCs (SVC):
  - dynamically set up on per-call basis

What Else is Left to Learn?

- Why am I taking this course? I know everything already!
- Not quite yet!

What Else is Left to Learn?

- Multicast (1 week)
- Peer-to-Peer networking (1.5 weeks)
- Wireless and Mobile Networks (2.5 weeks)
- Mid-term
- Multimedia Networking (2 weeks)
- Network Security and Wireless Security (2.5 weeks)
- Sensor and Senor Networks (1.5 weeks)