# Computer Communication Networks

## Foundation

UNIVERSITY AT ALBANY
State University of New York

ICEN/ICSI 416 – Fall 2016
Prof. Dola Saha

## Foundation

- Applications
- Requirements
- Network Architecture
- Implementing Network Software
- Performance

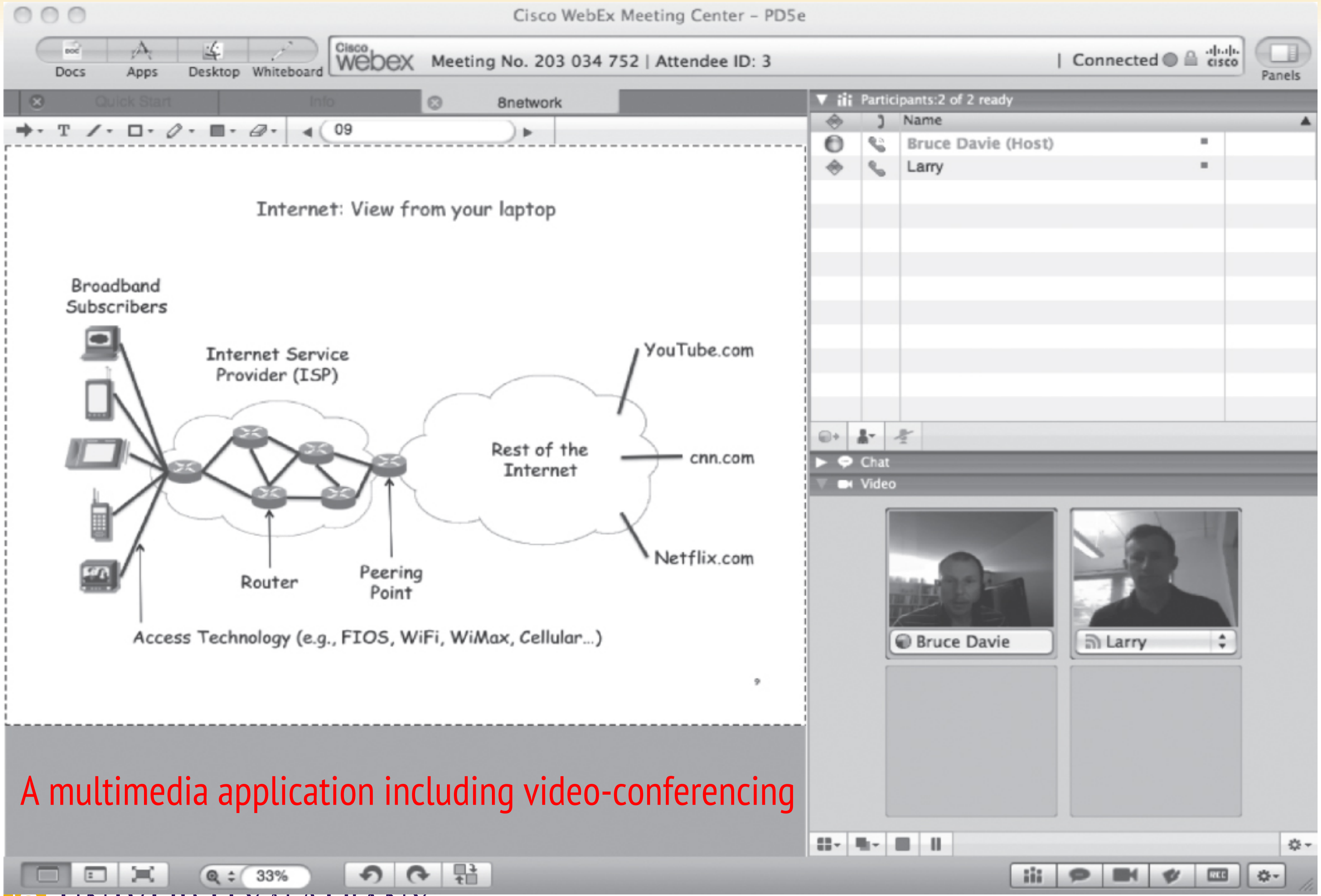
## Goals

- Exploring the requirements that different applications and different communities place on the computer network
- Introducing the idea of network architecture
- Introducing some key elements in implementing Network Software
- Define key metrics that will be used to evaluate the performance of computer network

## Applications

- Most people know about the Internet (a computer network) through applications
  - World Wide Web
  - Email
  - Online Social Network
  - Streaming Audio Video
  - File Sharing
  - Instant Messaging
  - …

# Example of an application



A multimedia application including video-conferencing

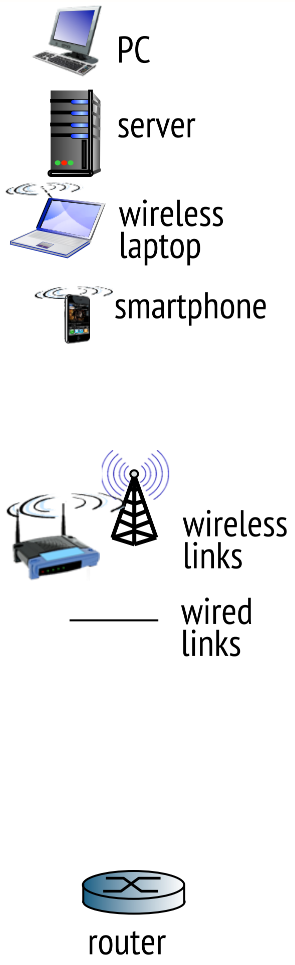
## Application Protocol

- URL
  - Uniform Resource Locater
  - http://www.albany.edu/faculty/dsaha/
- HTTP
  - Hyper Text Transfer Protocol
- TCP
  - Transmission Control Protocol
- 17 messages for one URL request
  - 6 to find the IP (Internet Protocol) address
  - 3 for connection establishment of TCP
  - 4 for HTTP request and acknowledgement
    - Request: I got your request and I will send the data
    - Reply: Here is the data you requested; I got the data
  - 4 messages for tearing down TCP connection

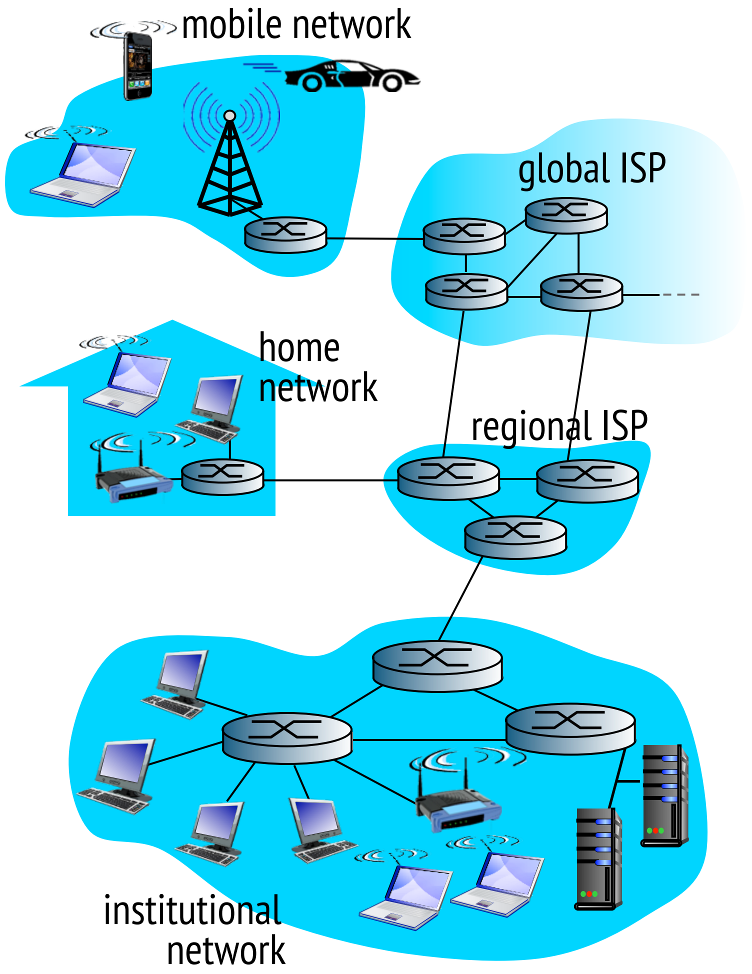
## Requirements

- Application Programmer
  - List the services that his application needs: delay bounded delivery of data
- Network Designer
  - Design a cost-effective network with sharable resources
- Network Provider
  - List the characteristics of a system that is easy to manage

## What's the Internet: "nuts and bolts" view

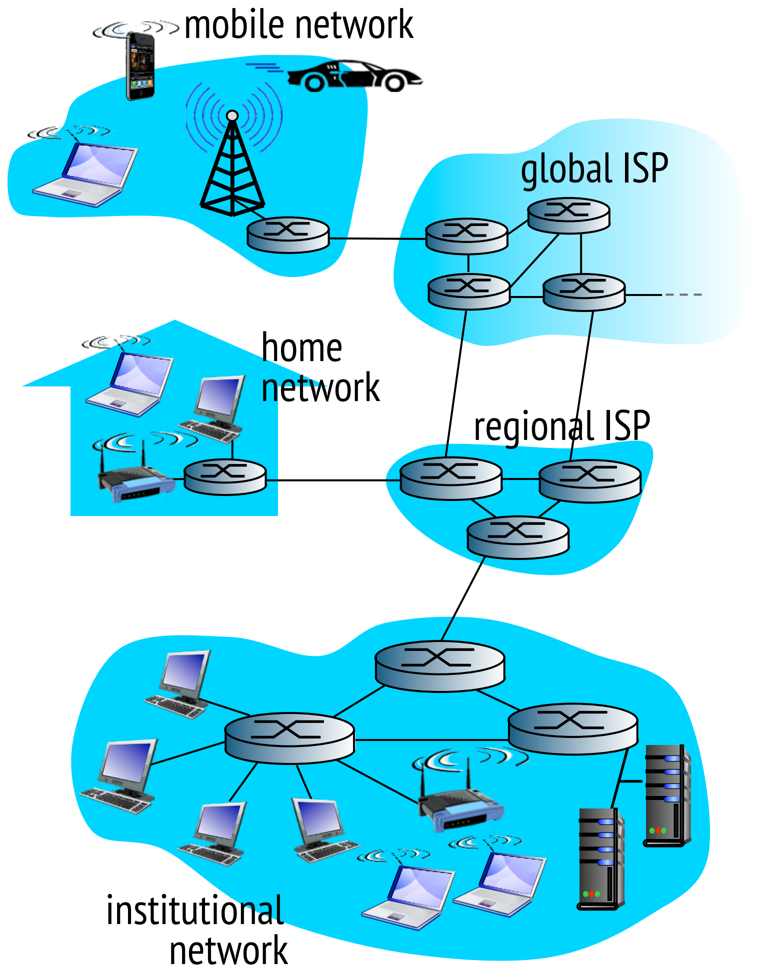


- Packet switches: forward packets (chunks of data)
  - routers and switches
- Millions of connected computing devices:
  - hosts = end systems
  - running network apps
- Communication links
  - Fiber, copper, radio, satellite
  - Transmission rate: bandwidth



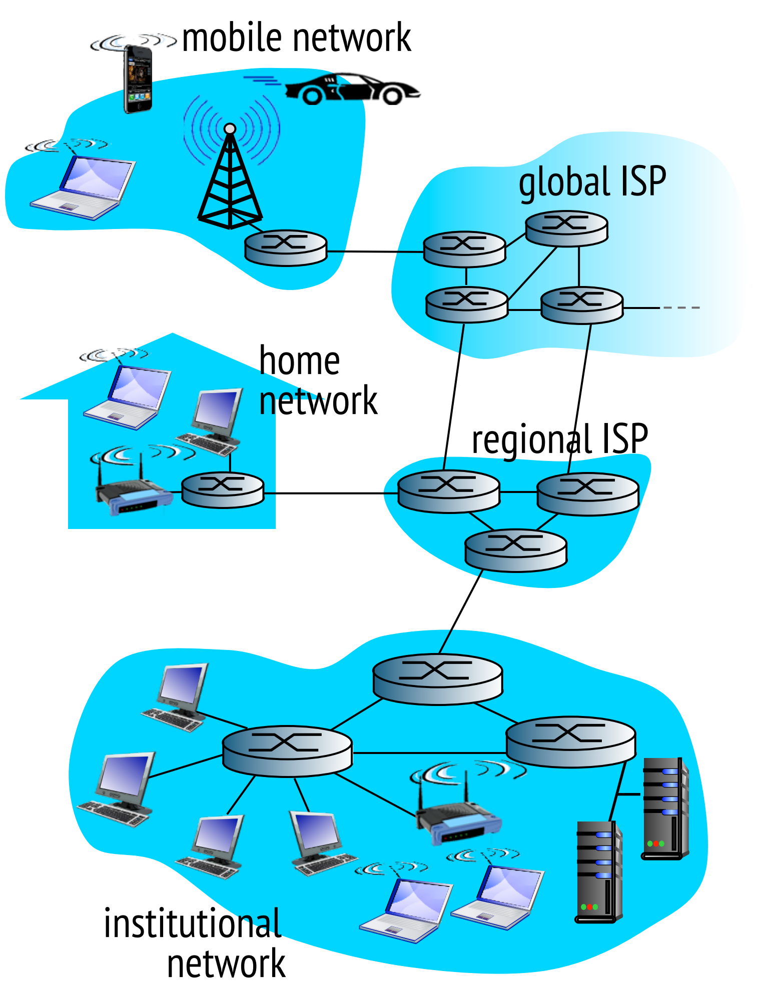
## What's the Internet: "nuts and bolts" view

- Internet: "network of networks"
  - Interconnected ISPs
- protocols control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, 802.11
- Internet standards
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force

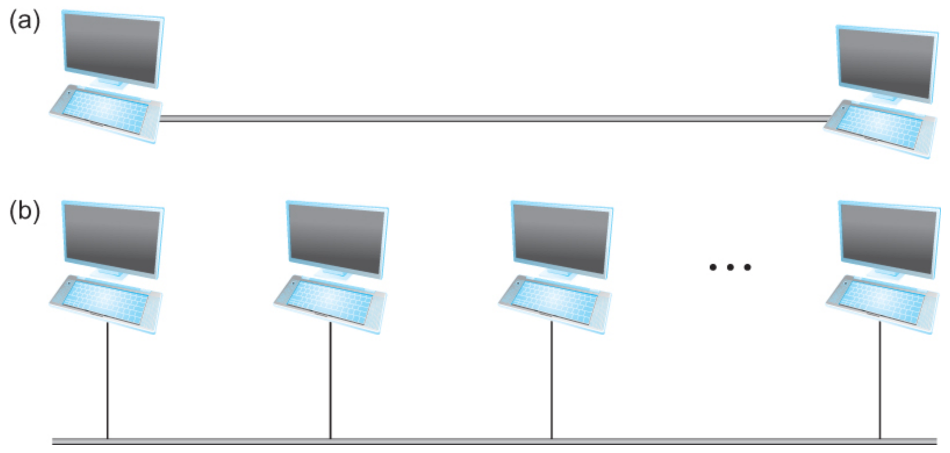


## What's the Internet: "service" view

- *Infrastructure that provides services to applications:*
  - Web, VoIP, email, games, e-commerce, social nets, …
- *provides programming interface to apps*
  - hooks that allow sending and receiving app programs to "connect" to Internet
  - provides service options, analogous to postal service



## Connectivity



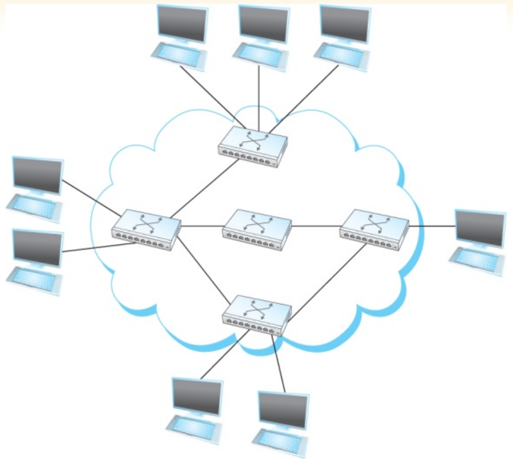
(a) Point-to-point

(b) Multiple access

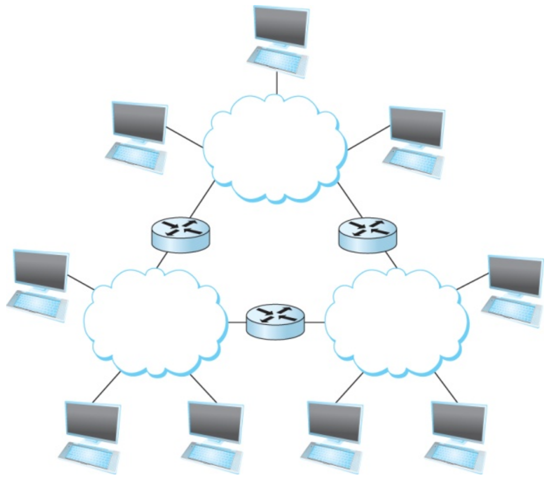
- Need to understand the following **terminologies**
  - Scale
  - Link
  - Nodes
  - Point-to-point
  - Multiple access
  - Switched Network
    - Circuit Switched
    - Packet Switched
  - Packet, message
  - Store-and-forward

## Connectivity

(a)



(b)



(a) A switched network

(b) Interconnection of networks

- Terminologies (contd.)
  - Cloud
  - Hosts
  - Switches
  - Internetwork
  - Router/gateway
  - Host-to-host connectivity
  - Address
  - Routing
  - Unicast/broadcast/multicast

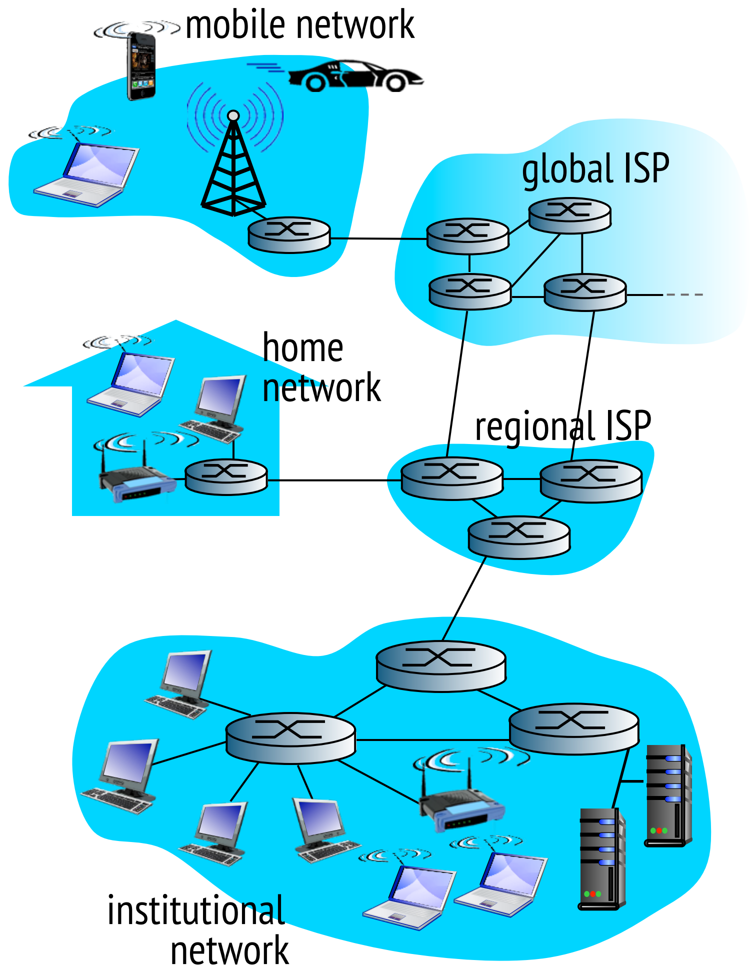
UNIVERSITY AT ALBANY
State University of New York

# A closer look at network structure:

- *network edge:*
  - hosts: clients and servers
  - servers often in data centers

- *access networks, physical media:* wired, wireless communication links

- *network core:*
  - interconnected routers
  - network of networks



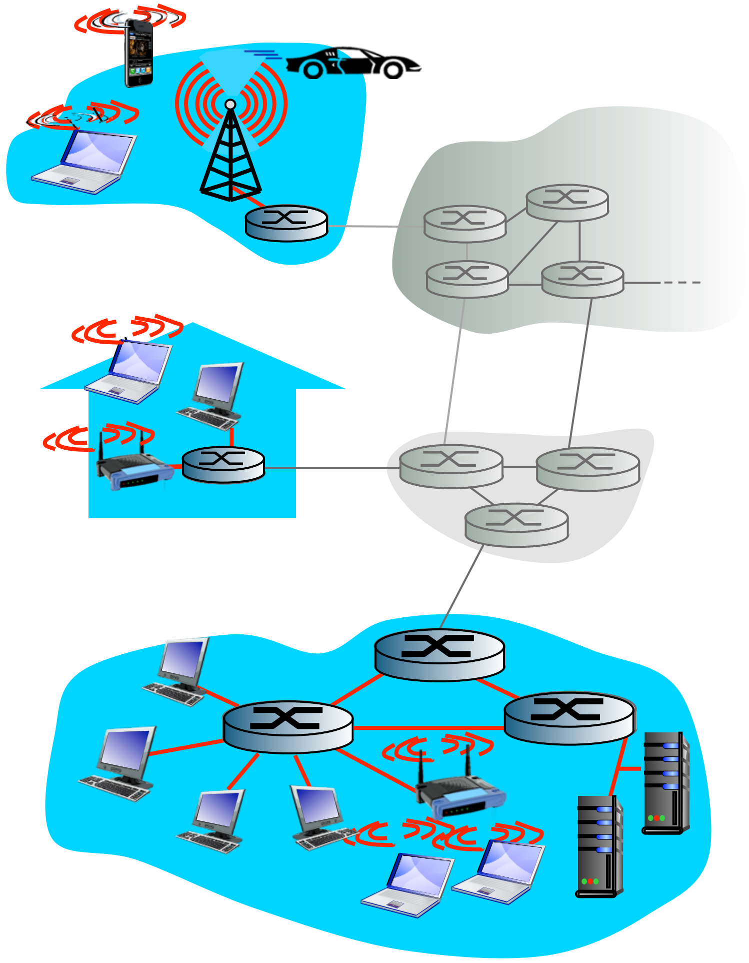
## Access networks and physical media

*Q: How to connect end systems to edge router?*

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

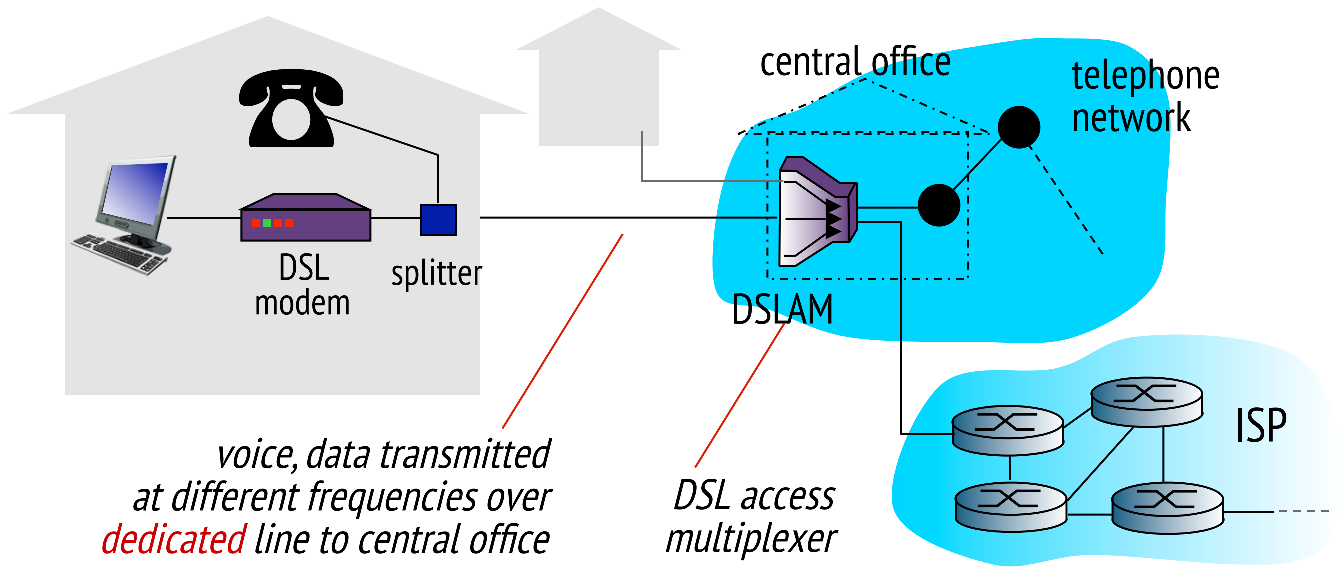
*keep in mind:*

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



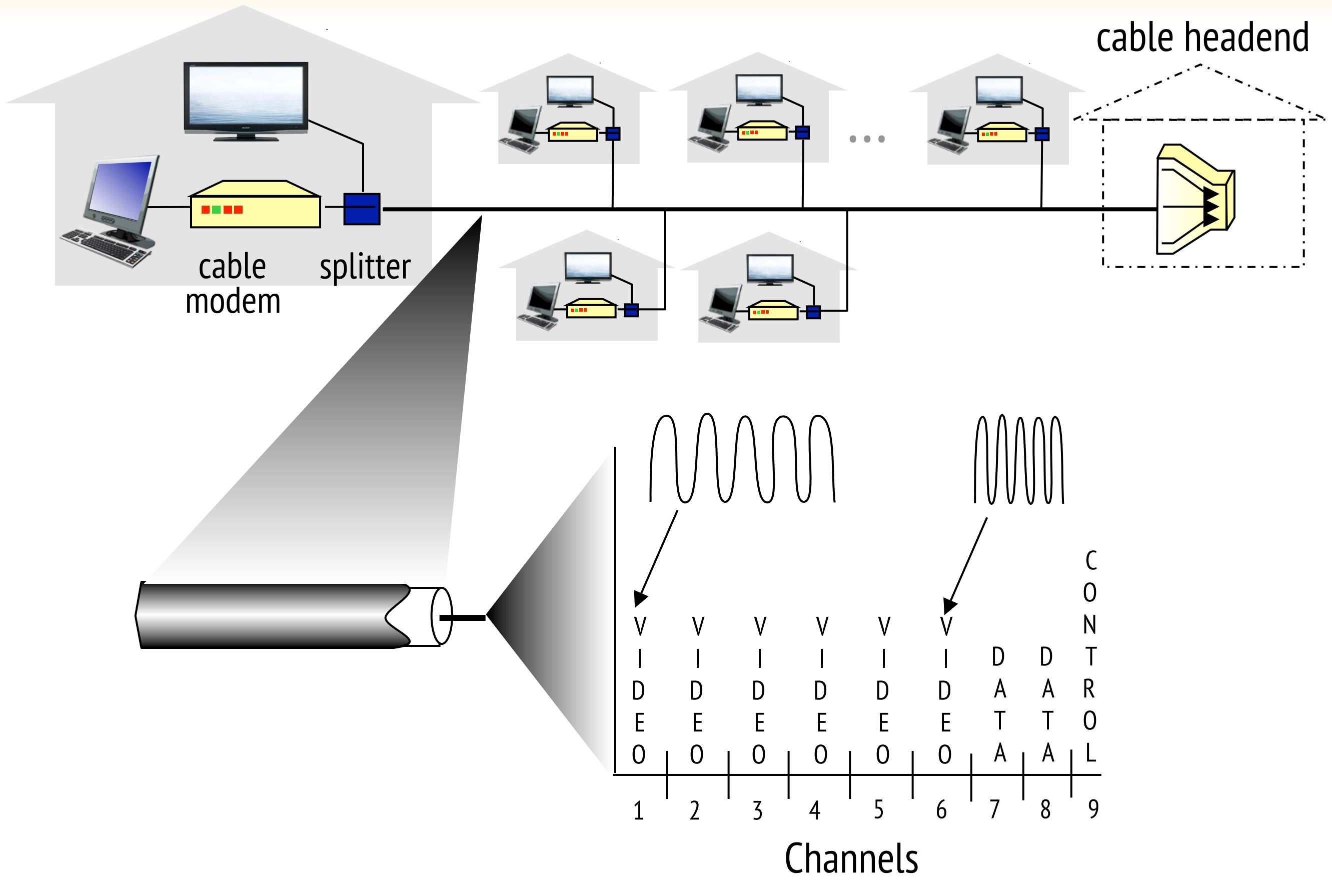
UNIVERSITY AT ALBANY
State University of New York

## Access net: digital subscriber line (DSL)


central office
telephone network
DSL modem
splitter
DSLAM
ISP
voice, data transmitted at different frequencies over dedicated line to central office
DSL access multiplexer

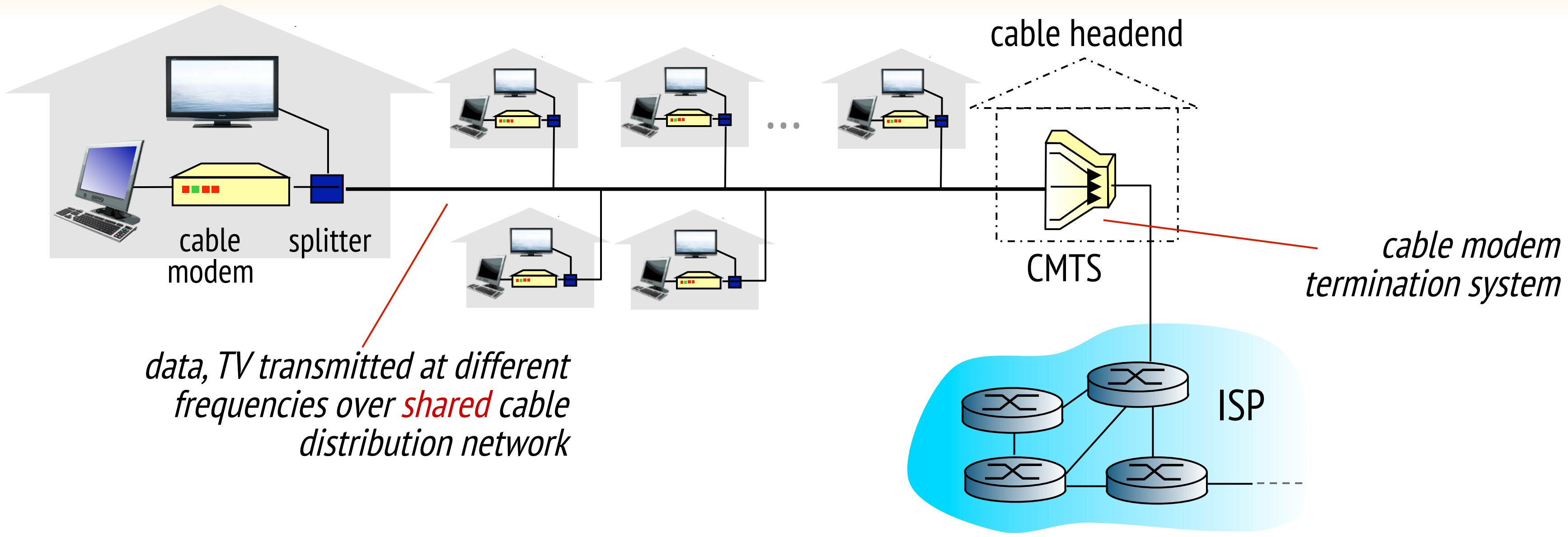
- use existing telephone line to central office DSLAM
  - data over DSL phone line goes to Internet
  - voice over DSL phone line goes to telephone net
- < 2.5 Mbps upstream transmission rate (typically < 1 Mbps)
- < 24 Mbps downstream transmission rate (typically < 10 Mbps)

# Access net: cable network



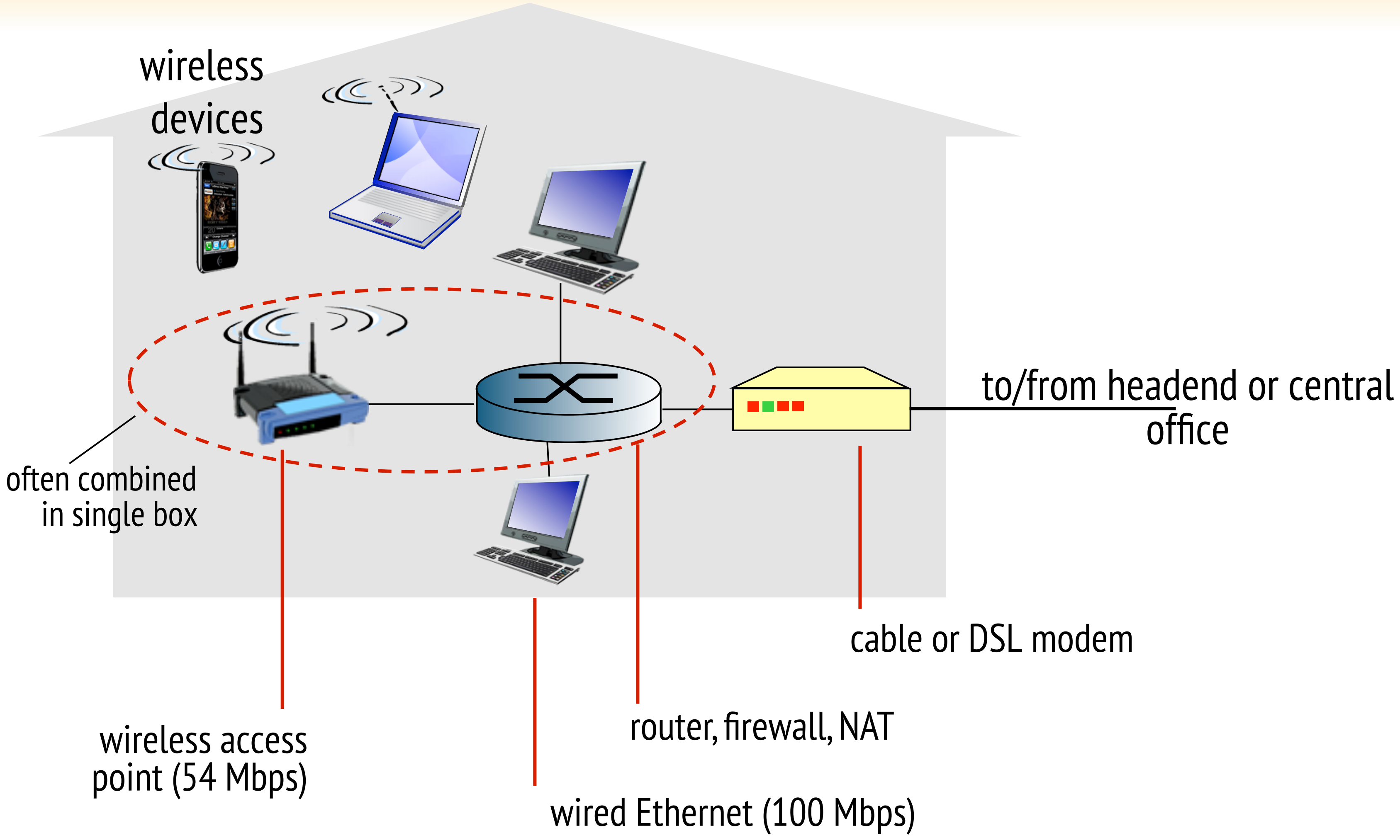
*frequency division multiplexing:* different channels transmitted in different frequency bands

# Access net: cable network

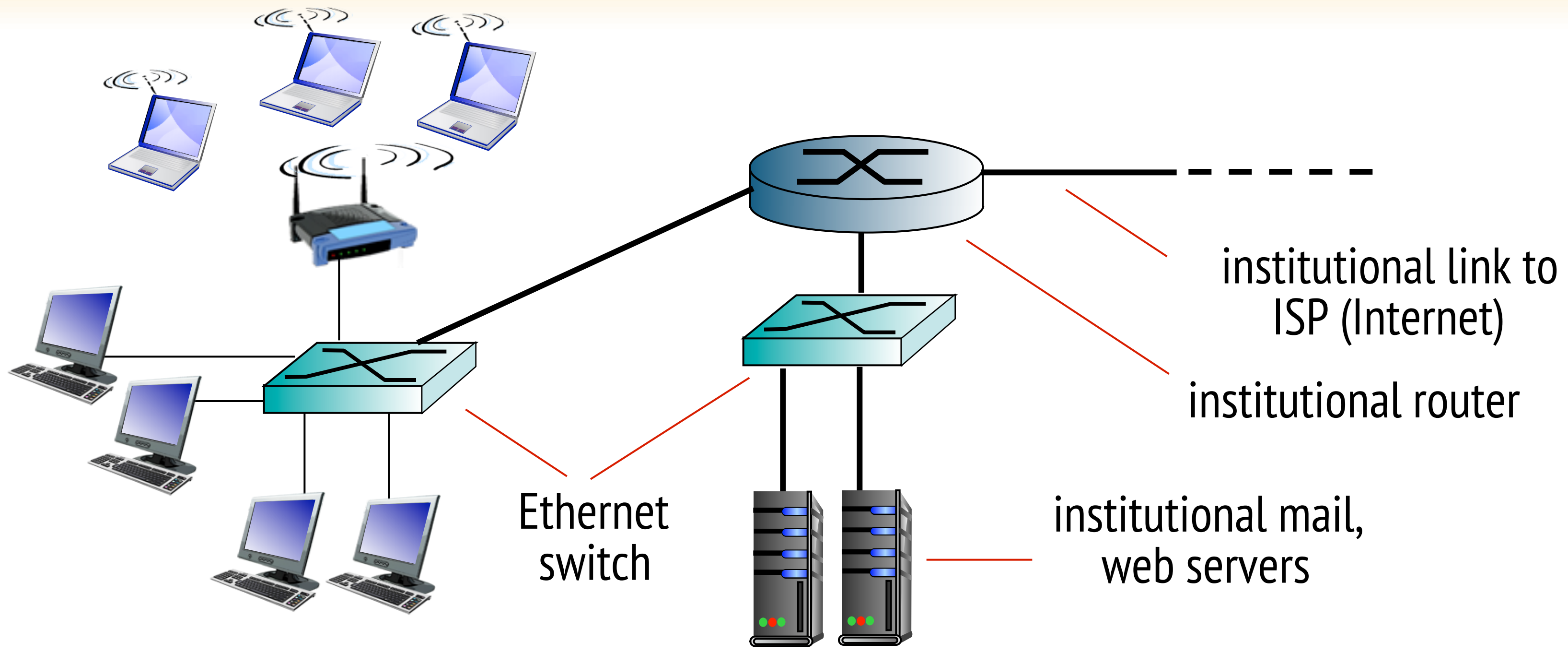

cable headend
cable modem
splitter
...
CMTS
cable modem termination system
data, TV transmitted at different frequencies over shared cable distribution network
ISP

- HFC: hybrid fiber coax
  - asymmetric: up to 30Mbps downstream transmission rate, 2 Mbps upstream transmission rate
- network of cable, fiber attaches homes to ISP router
  - homes *share access network* to cable headend
  - unlike DSL, which has dedicated access to central office

## Access net: home network


wireless devices
often combined in single box
to/from headend or central office
cable or DSL modem
router, firewall, NAT
wireless access point (54 Mbps)
wired Ethernet (100 Mbps)

## Enterprise access networks (Ethernet)


institutional link to ISP (Internet)
institutional router
Ethernet switch
institutional mail, web servers

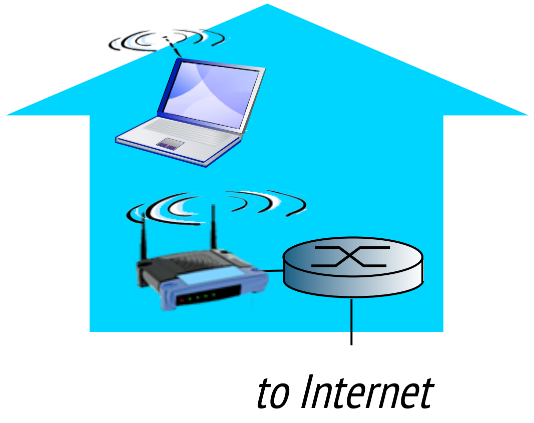
- typically used in companies, universities, etc
- 10 Mbps, 100Mbps, 1Gbps, 10Gbps transmission rates
- today, end systems typically connect into Ethernet switch

## Wireless access networks

- shared *wireless* access network connects end system to router
  - via base station aka "access point"

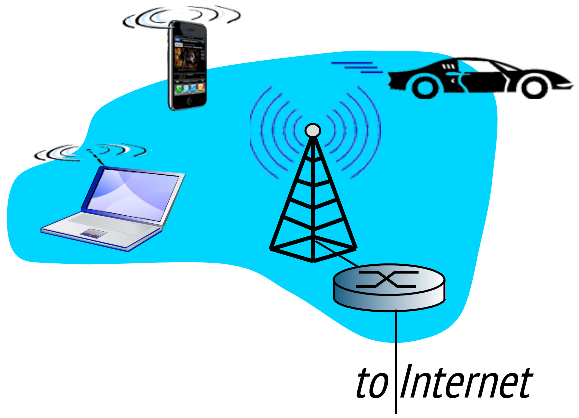
*wireless LANs:*

- within building (100 ft)
- 802.11b/g (WiFi): 11, 54 Mbps transmission rate



wide-area wireless access

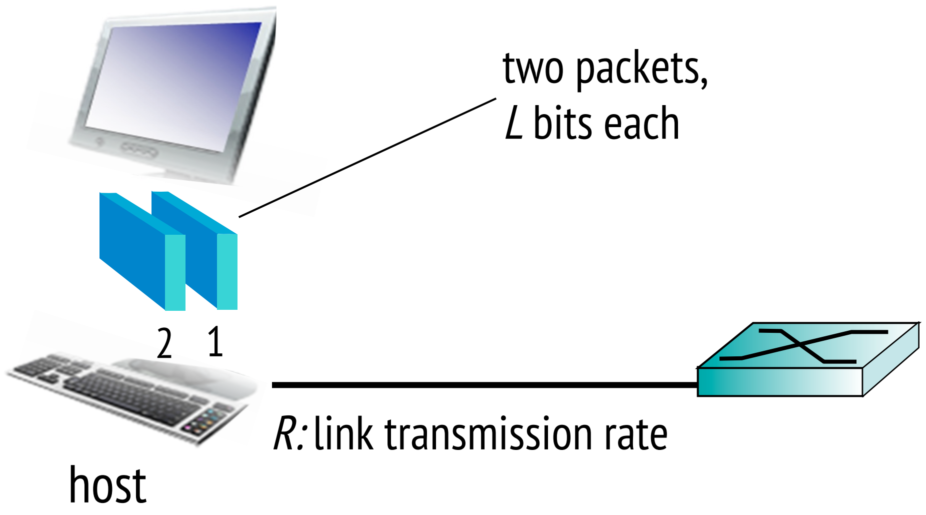
- provided by AT&T (cellular) operator, 10's km
- between 1 and 10 Mbps
- 3G, 4G: LTE



# Host: sends *packets* of data

host sending function:

- takes application message
- breaks into smaller chunks, known as *packets*, of length *L* bits
- transmits packet into access network at *transmission rate R*
    - link transmission rate, aka link *capacity, aka link bandwidth*



packet transmission delay = time needed to transmit *L*-bit packet into link = $\frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$

## Physical media

- bit: propagates between transmitter/receiver pairs
- physical link: what lies between transmitter & receiver
- guided media:
  - signals propagate in solid media: copper, fiber, coax
- unguided media:
  - signals propagate freely, e.g., radio

*twisted pair (TP)*

two insulated copper wires

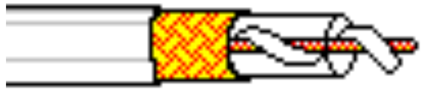
Category 5: 100 Mbps, 1 Gpbs Ethernet

Category 6: 10Gbps

## Physical media: coax, fiber

*coaxial cable:*

- two concentric copper conductors
- bidirectional
- broadband:
  - multiple channels on cable
  - HFC



*fiber optic cable:*

- glass fiber carrying light pulses, each pulse a bit
- high-speed operation:
  - high-speed point-to-point transmission (e.g., 10's-100's Gpbs transmission rate)
- low error rate:
  - repeaters spaced far apart
  - immune to electromagnetic noise

UNIVERSITY AT ALBANY
State University of New York

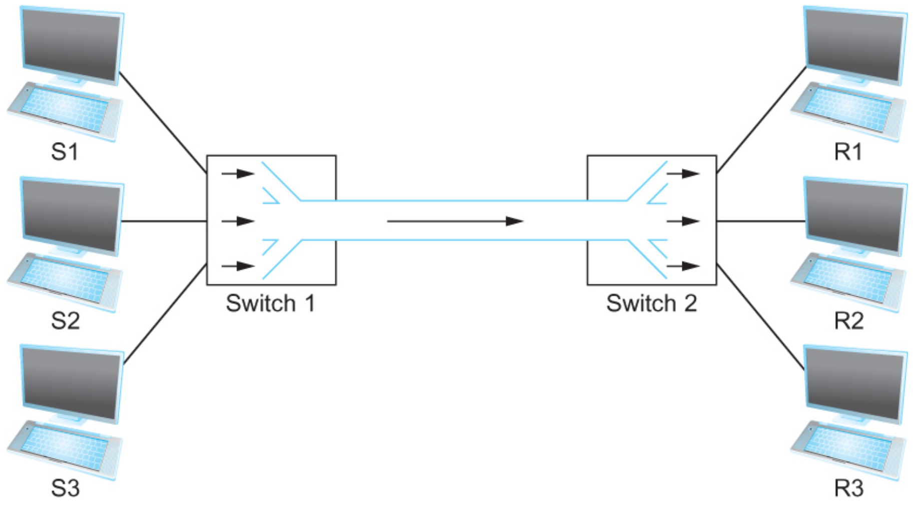
## Physical media: radio

- signal carried in electromagnetic spectrum
- no physical “wire”
- bidirectional
- propagation environment effects:
  - reflection
  - obstruction by objects
  - interference

*radio link types:*

- terrestrial  microwave
  - e.g. up to 45 Mbps channels
- LAN (e.g., WiFi)
  - 11Mbps, 54 Mbps
- wide-area (e.g., cellular)
  - 3G cellular: ~ few Mbps
- satellite
  - Kbps to 45Mbps channel (or multiple smaller channels)
  - 270 msec end-end delay
  - geosynchronous versus low altitude

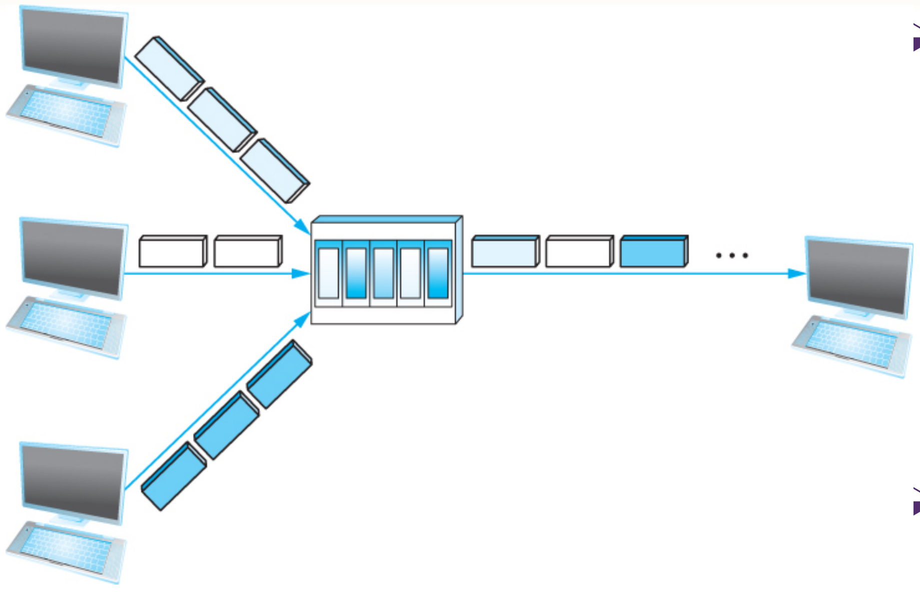
# Cost-Effective Resource Sharing



Multiplexing multiple logical flows over a single physical link

- Resource: links and nodes
- How to share a link?
  - Multiplexing
  - De-multiplexing
  - Synchronous Time-division Multiplexing
  - Time slots/data transmitted in predetermined slots

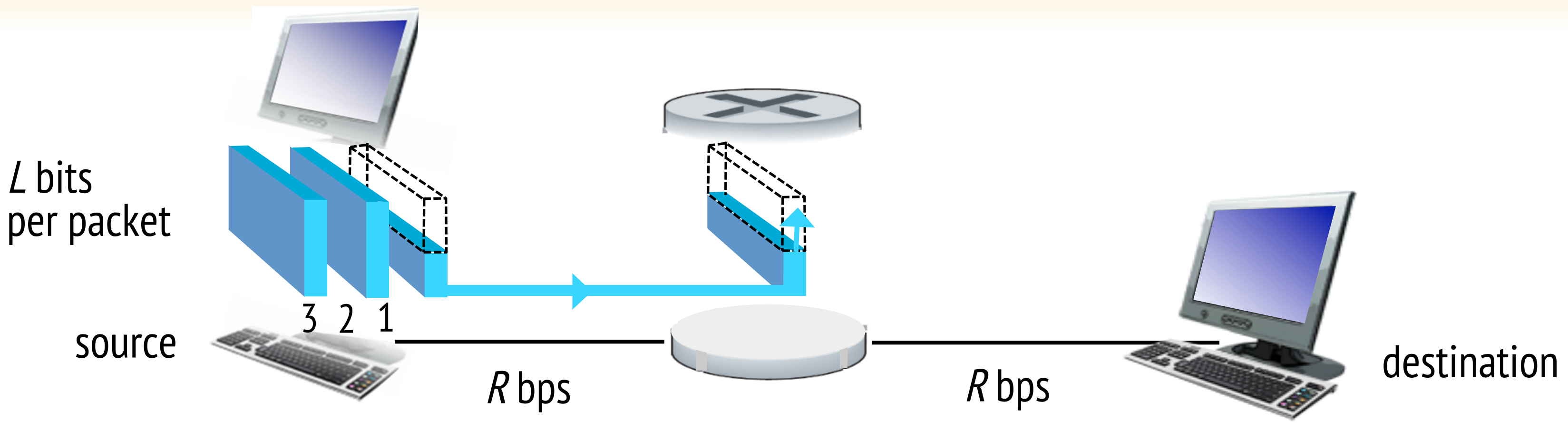
## Cost-Effective Resource Sharing



A switch multiplexing packets from multiple sources onto one shared link

- FDM: Frequency Division Multiplexing
  - Statistical Multiplexing
  - Data is transmitted based on demand of each flow.
- What is a flow?
  - Packets vs. Messages
  - FIFO, Round-Robin, Priorities (Quality-of-Service (QoS))
  - Congested?
  - LAN, MAN, WAN
  - SAN (System Area Networks)

## Packet-switching: store-and-forward



takes *L*/*R* seconds to transmit (push out) *L*-bit packet into link at *R* bps

*store and forward:* entire packet must arrive at router before it can be transmitted on next link

*one-hop numerical example:*

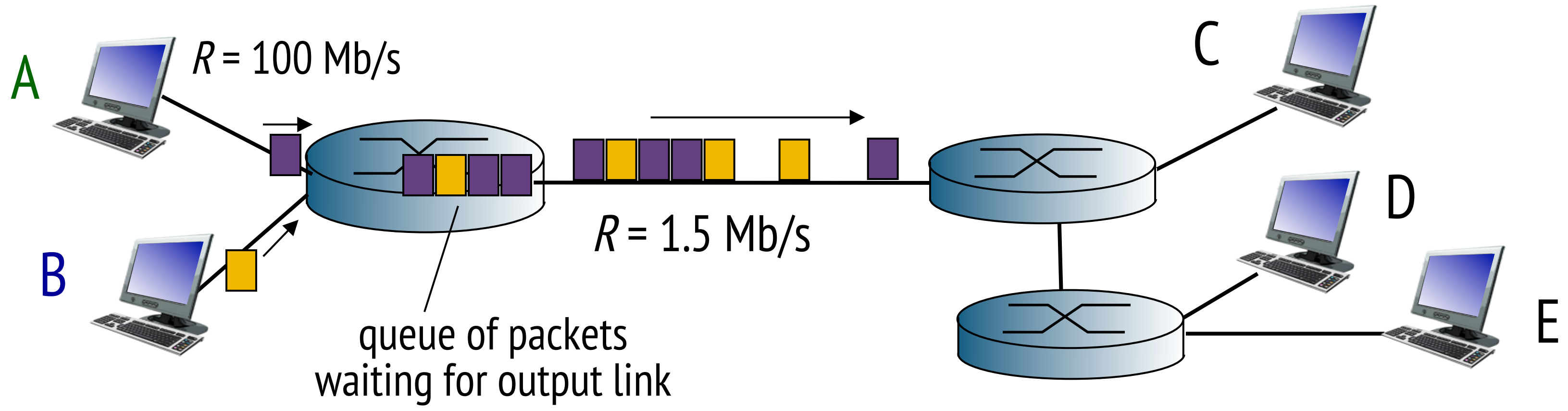
- *L* = 7.5 Mbits
- *R* = 1.5 Mbps
- one-hop transmission delay = 5 sec

- end-end delay = 2*L*/*R* (assuming zero propagation delay)

more on delay shortly …

UNIVERSITY AT ALBANY
State University of New York

# Packet Switching: queueing delay, loss



queuing and loss:

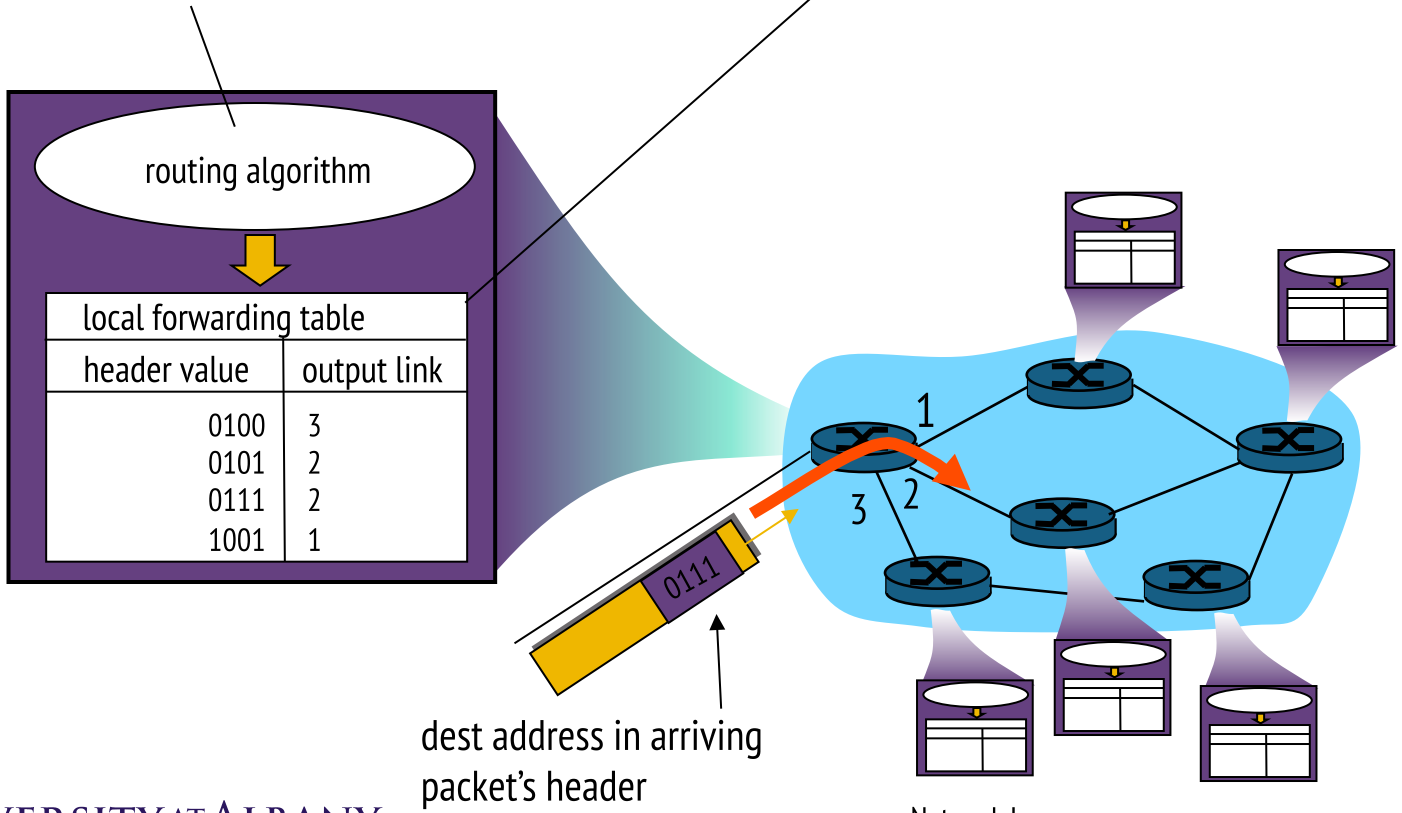
- If arrival rate (in bits) to link exceeds transmission rate of link for a period of time:
  - packets will queue, wait to be transmitted on link
  - packets can be dropped (lost) if memory (buffer) fills up

# Two key network-core functions

*routing:* determines source-destination route taken by packets

- *routing algorithms*

*forwarding:* move packets from router's input to appropriate router output

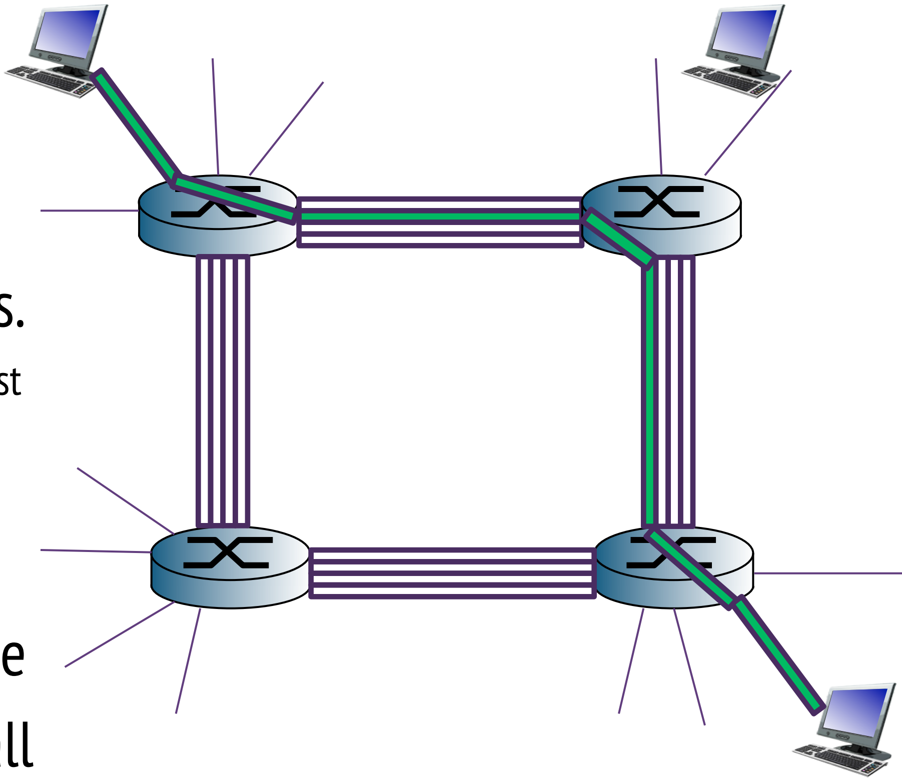


| header value | output link |
|---|---|
| 0100 | 3 |
| 0101 | 2 |
| 0111 | 2 |
| 1001 | 1 |

## Alternative core: circuit switching

end-end resources allocated to, reserved for "call" between source & dest:

- In diagram, each link has four circuits.
  - call gets 2nd circuit in top link and 1st circuit in right link.
- dedicated resources: no sharing
  - circuit-like (guaranteed) performance
- circuit segment idle if not used by call *(no sharing)*
- Commonly used in traditional telephone networks

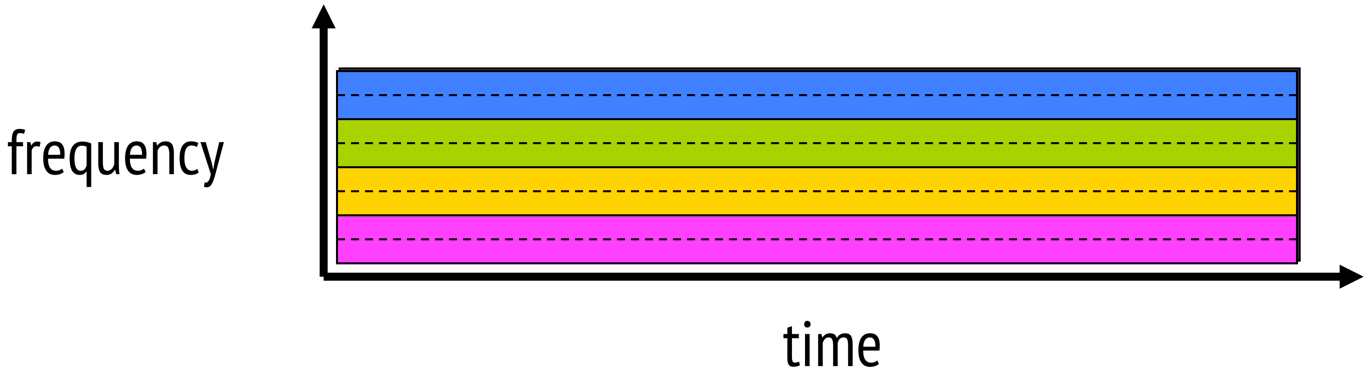


## Circuit switching: FDM versus TDM

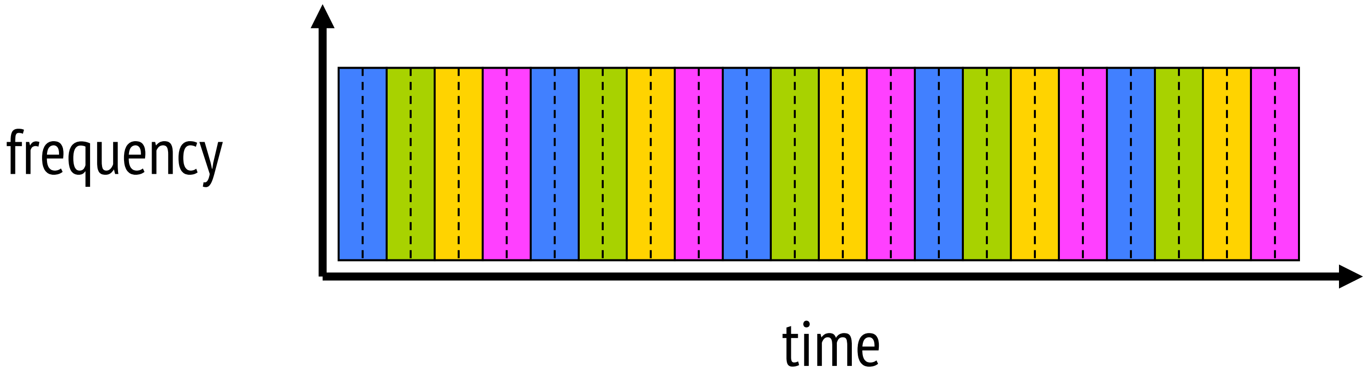
Example:

4 users 🟦 🟩 🟨 🟪

FDM



TDM

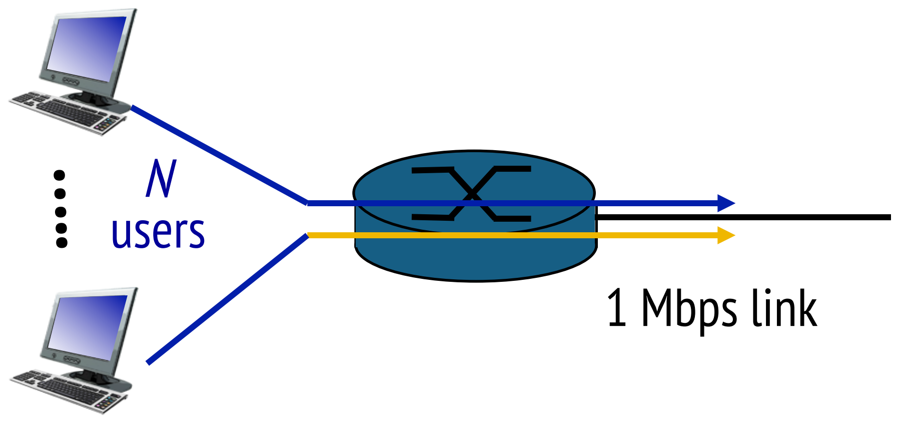


## Packet switching versus circuit switching

*packet switching allows more users to use network!*

example:

- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time



*circuit-switching:*

10 users

*packet switching:*

with 35 users, probability > 10 active at same time is less than .0004 *

*Q:* how did we get value 0.0004?

*Q:* what happens if > 35 users ?

# Packet switching versus circuit switching

is packet switching a "slam dunk winner?"

- great for bursty data
- resource sharing
- simpler, no call setup
- excessive congestion possible: packet delay and loss
- protocols needed for reliable data transfer, congestion control
- *Q:* How to provide circuit-like behavior?
- bandwidth guarantees needed for audio/video apps
- still an unsolved problem

*Q:* human analogies of reserved resources (circuit switching) versus on-demand allocation (packet-switching)?

## Internet structure: network of networks

- End systems connect to Internet via access ISPs (Internet Service Providers)
  - Residential, company and university ISPs

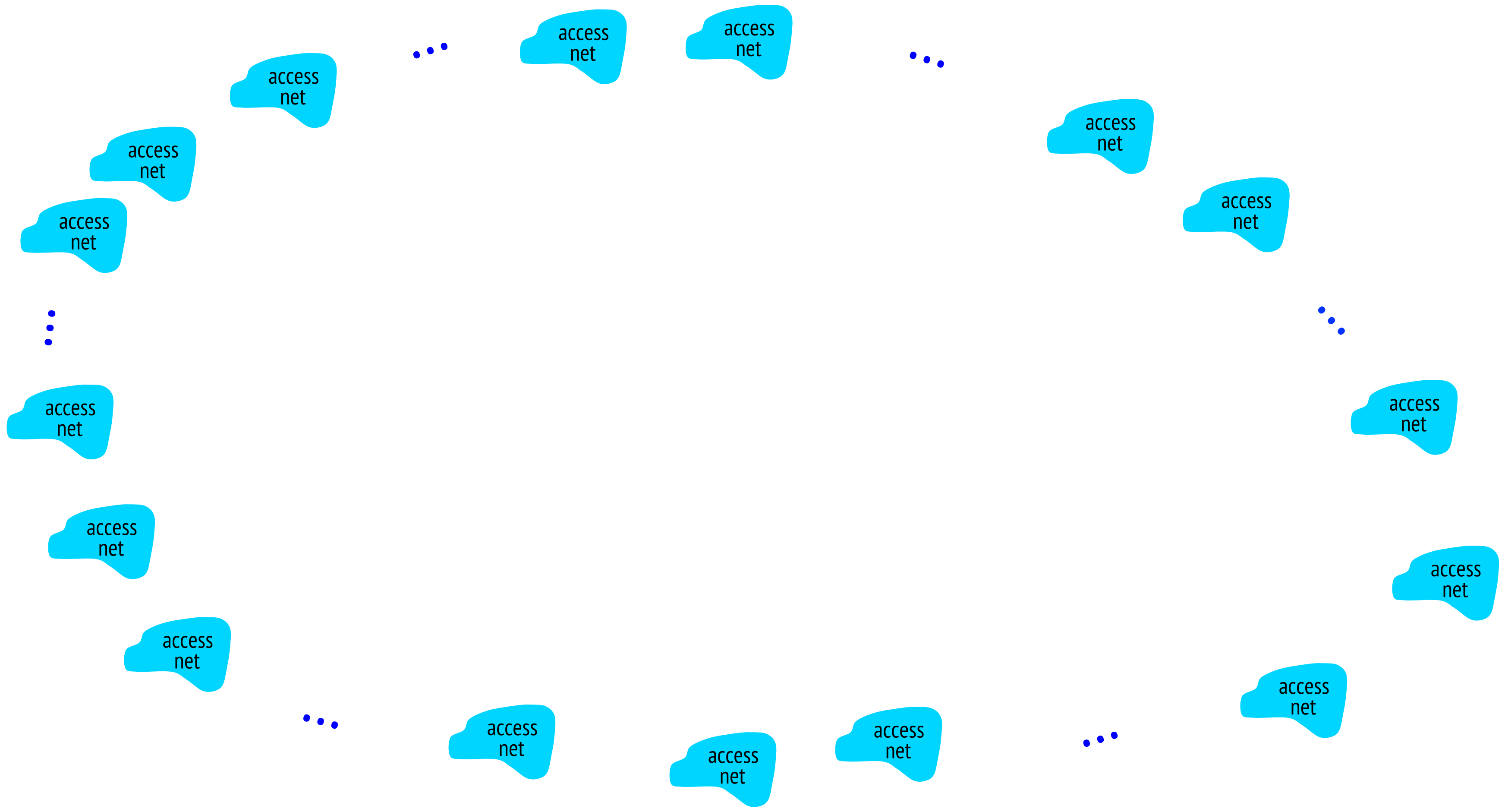
- Access ISPs in turn must be interconnected.
  - So that any two hosts can send packets to each other

- Resulting network of networks is very complex
  - Evolution was driven by economics and national policies

- Let's take a stepwise approach to describe current Internet structure

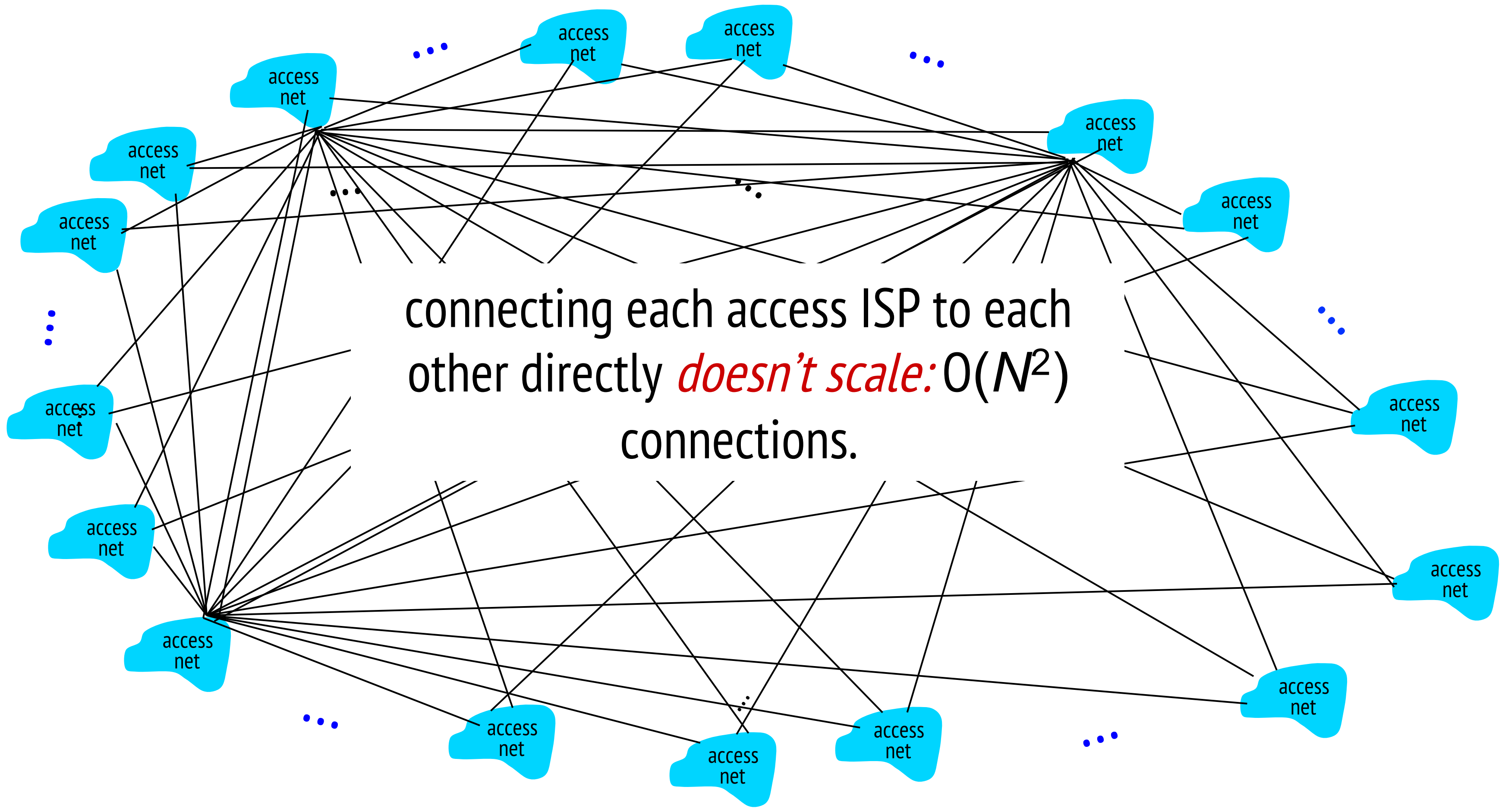
# Internet structure: network of networks

*Question:* given *millions* of access ISPs, how to connect them together?



# Internet structure: network of networks

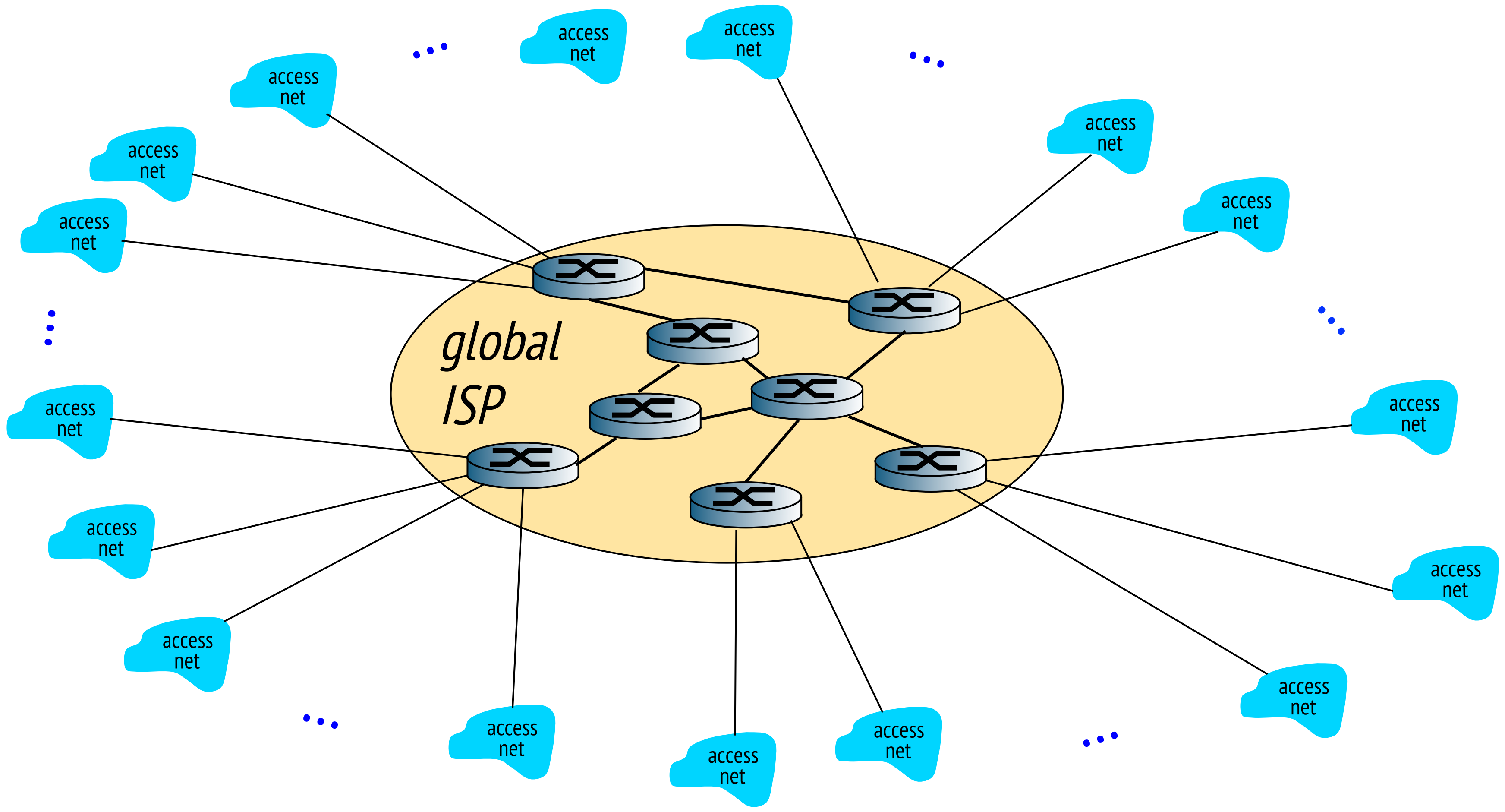
*Option: connect each access ISP to every other access ISP?*



connecting each access ISP to each other directly *doesn't scale:* $O(N^2)$ connections.

# Internet structure: network of networks

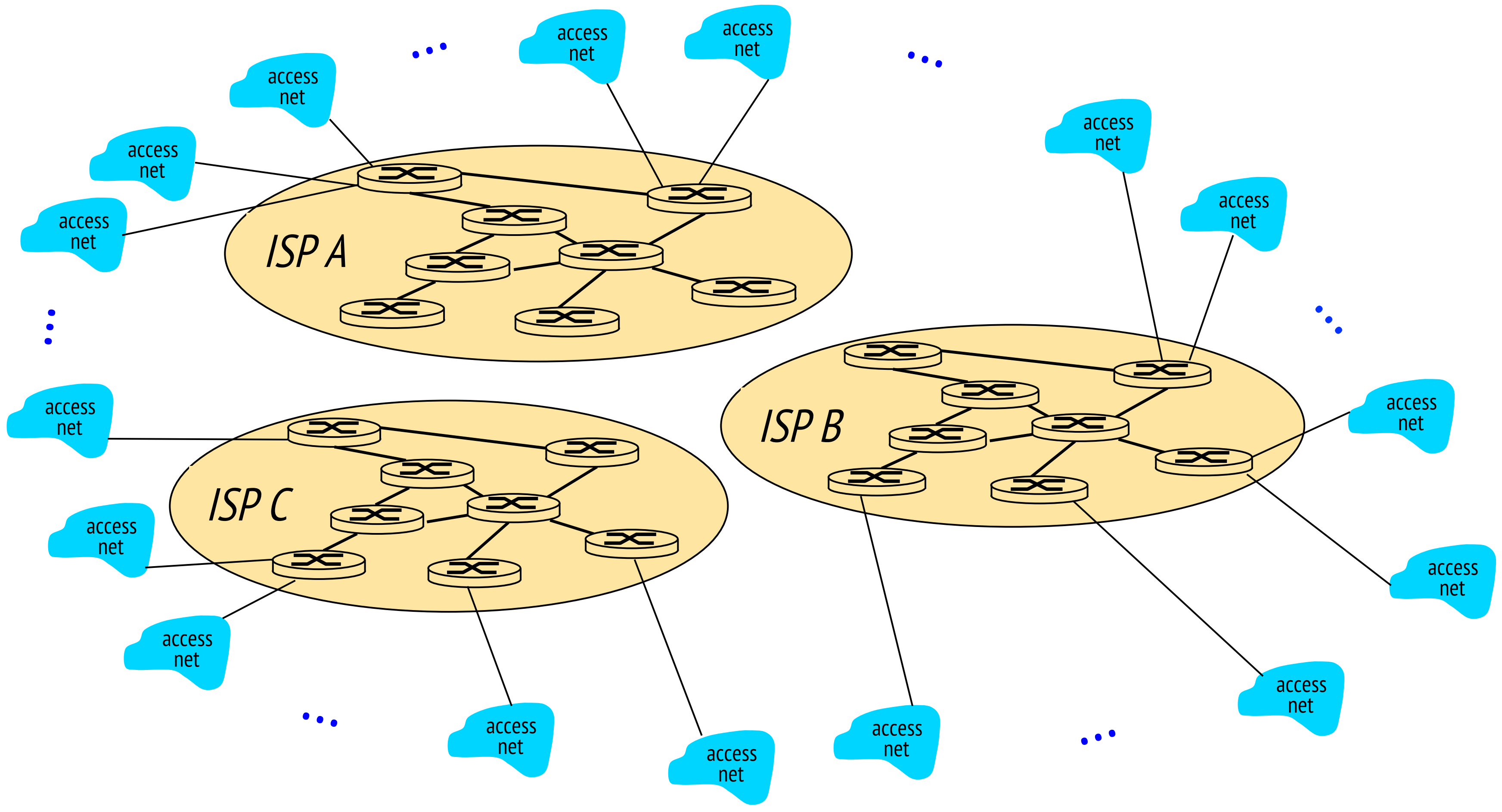
*Option: connect each access ISP to a global transit ISP? Customer and provider ISPs have economic agreement.*



UNIVERSITY AT ALBANY
State University of New York

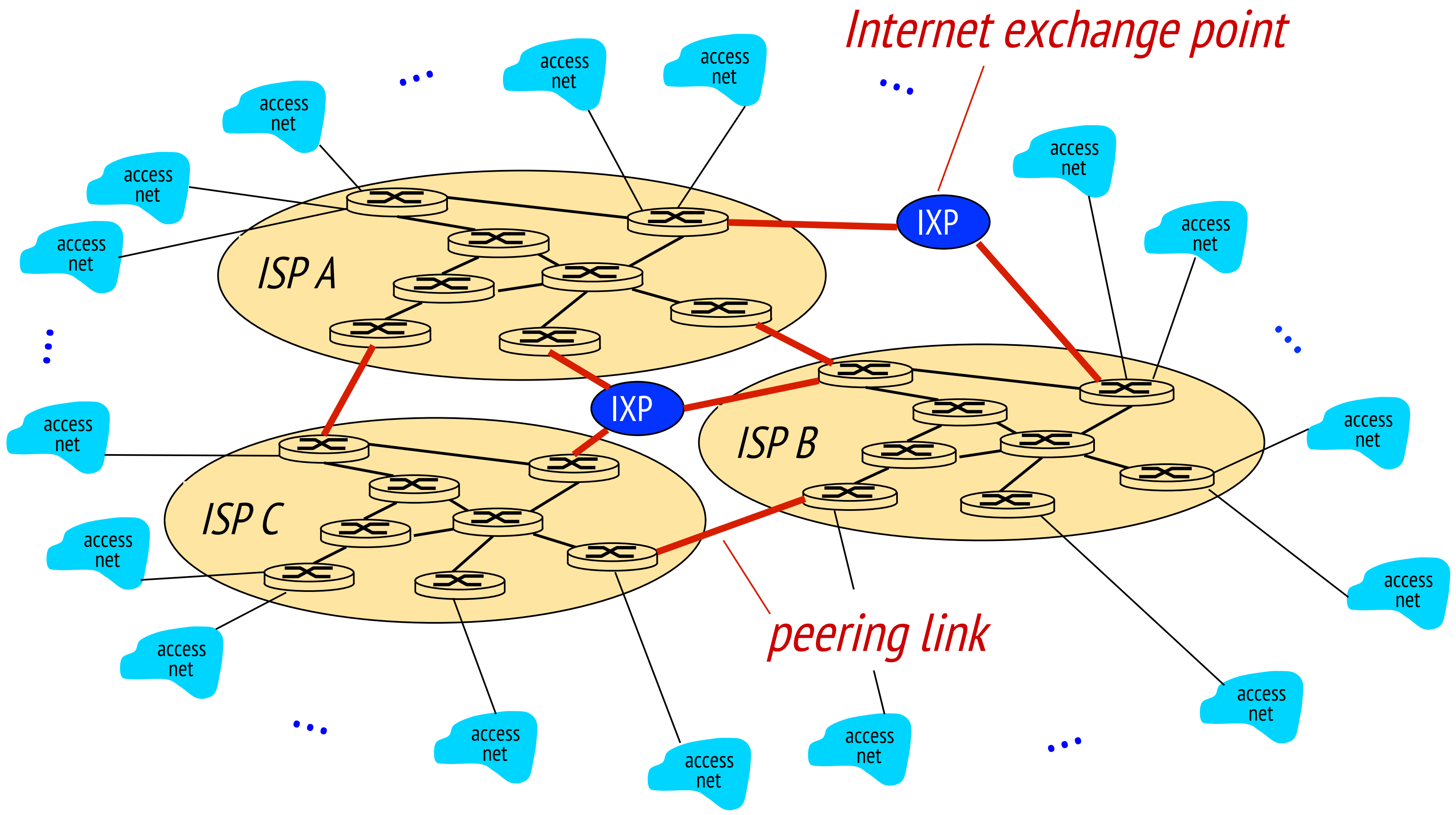
# Internet structure: network of networks

But if one global ISP is viable business, there will be competitors ….



# Internet structure: network of networks

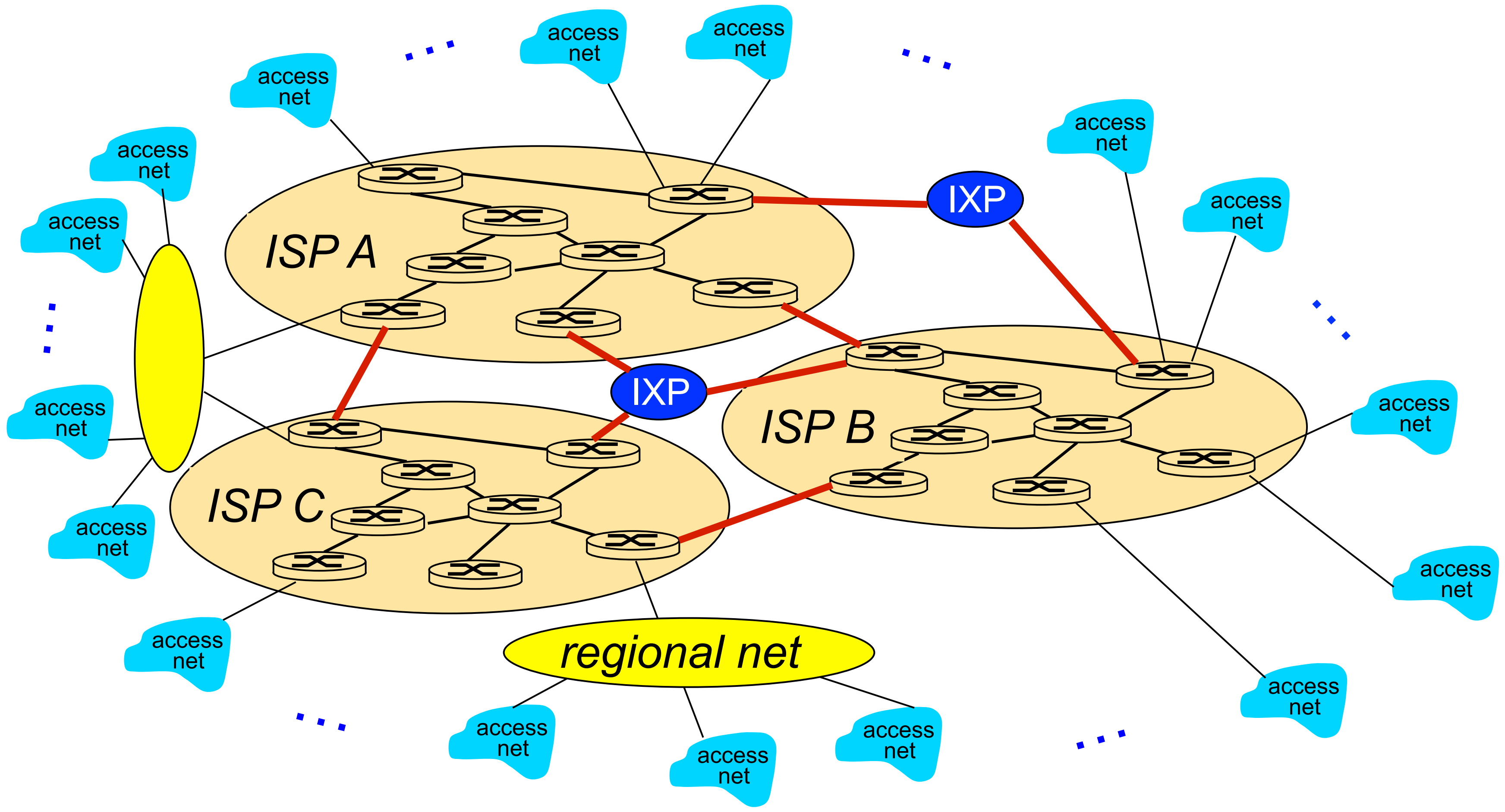
But if one global ISP is viable business, there will be competitors .... which must be interconnected



UNIVERSITY AT ALBANY
State University of New York

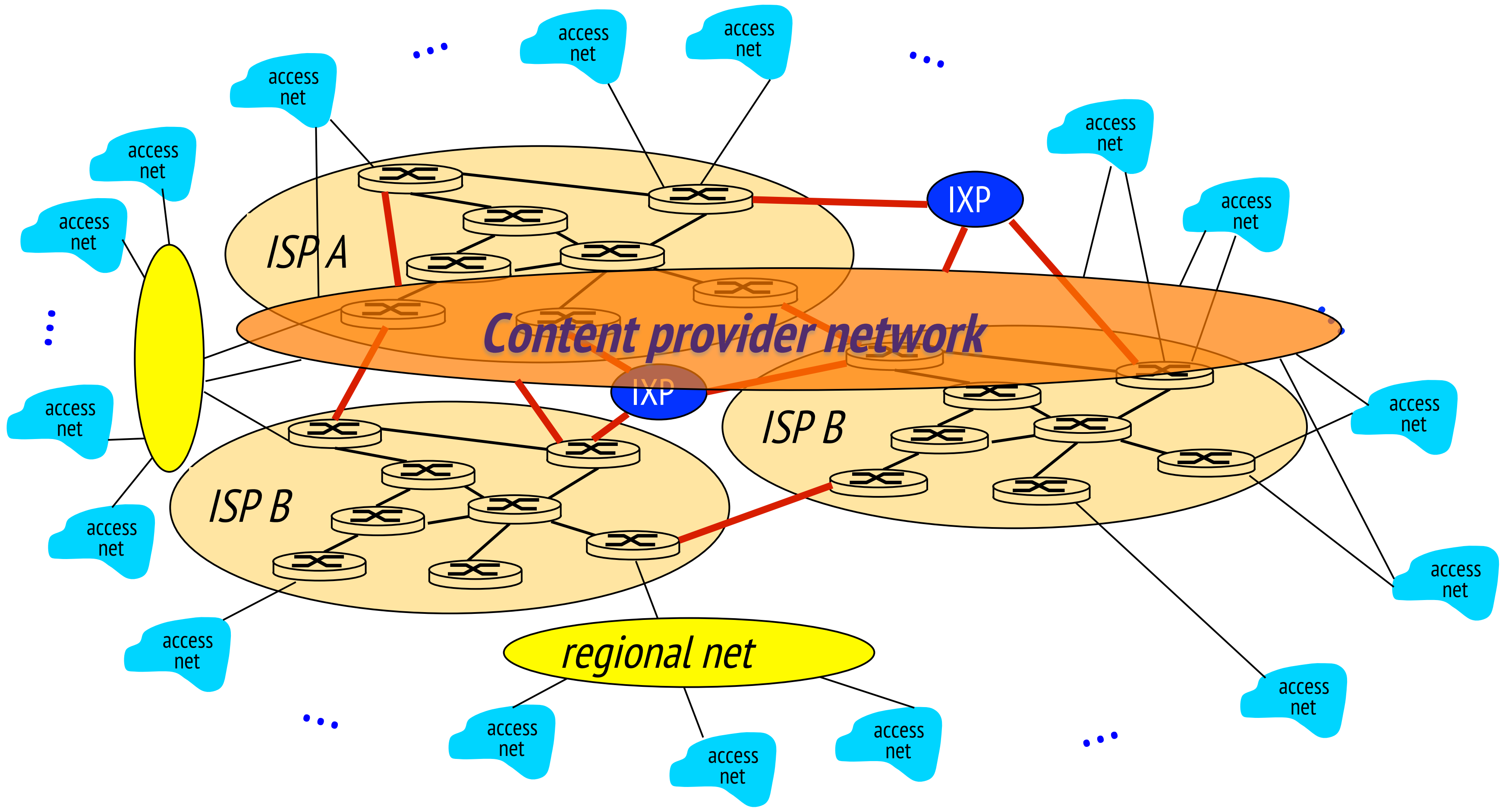
# Internet structure: network of networks

… and regional networks may arise to connect access nets to ISPS

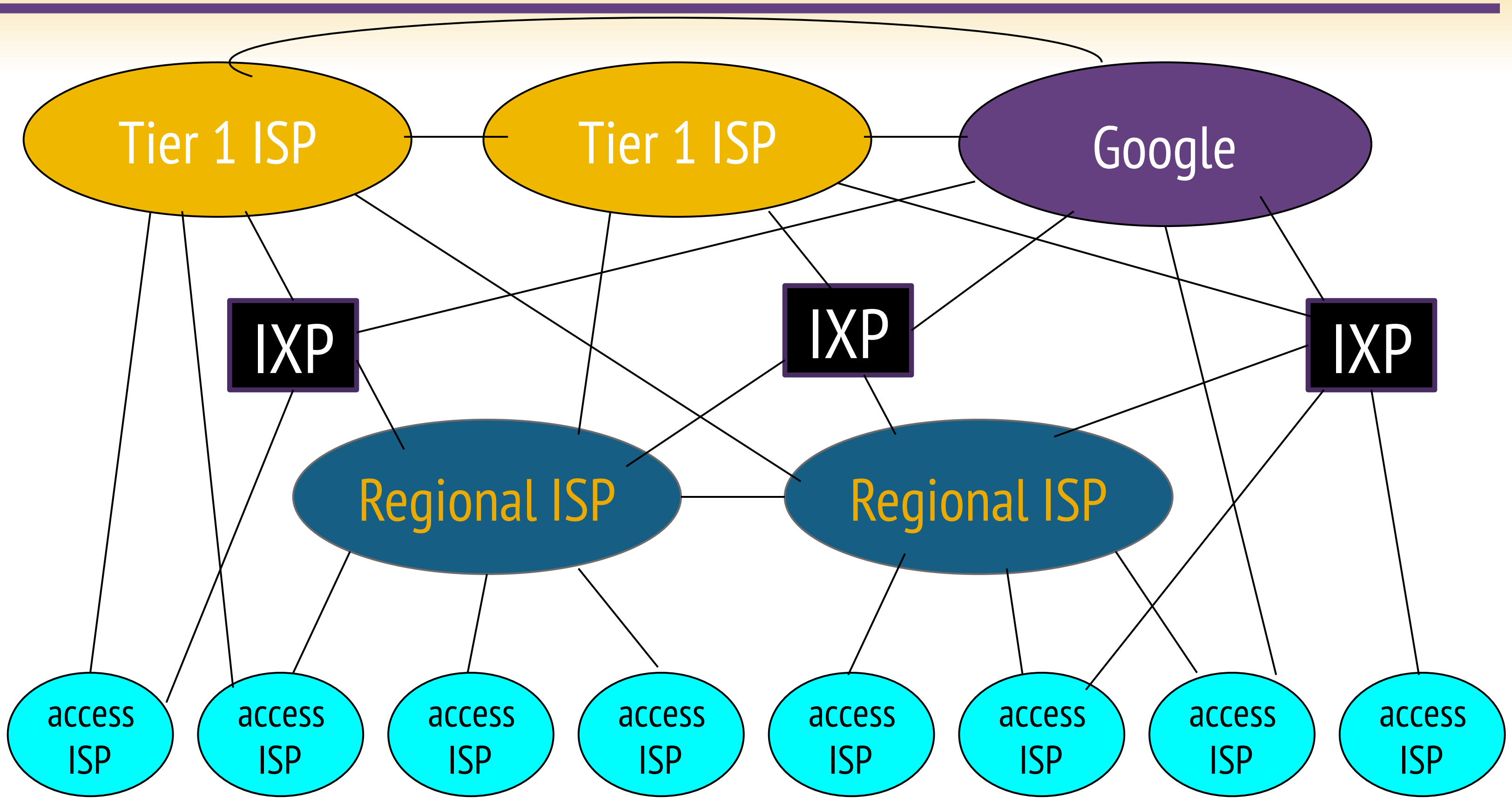


# Internet structure: network of networks

… and content provider networks (e.g., Google, Microsoft, Akamai ) may run their own network, to bring services, content close to end users

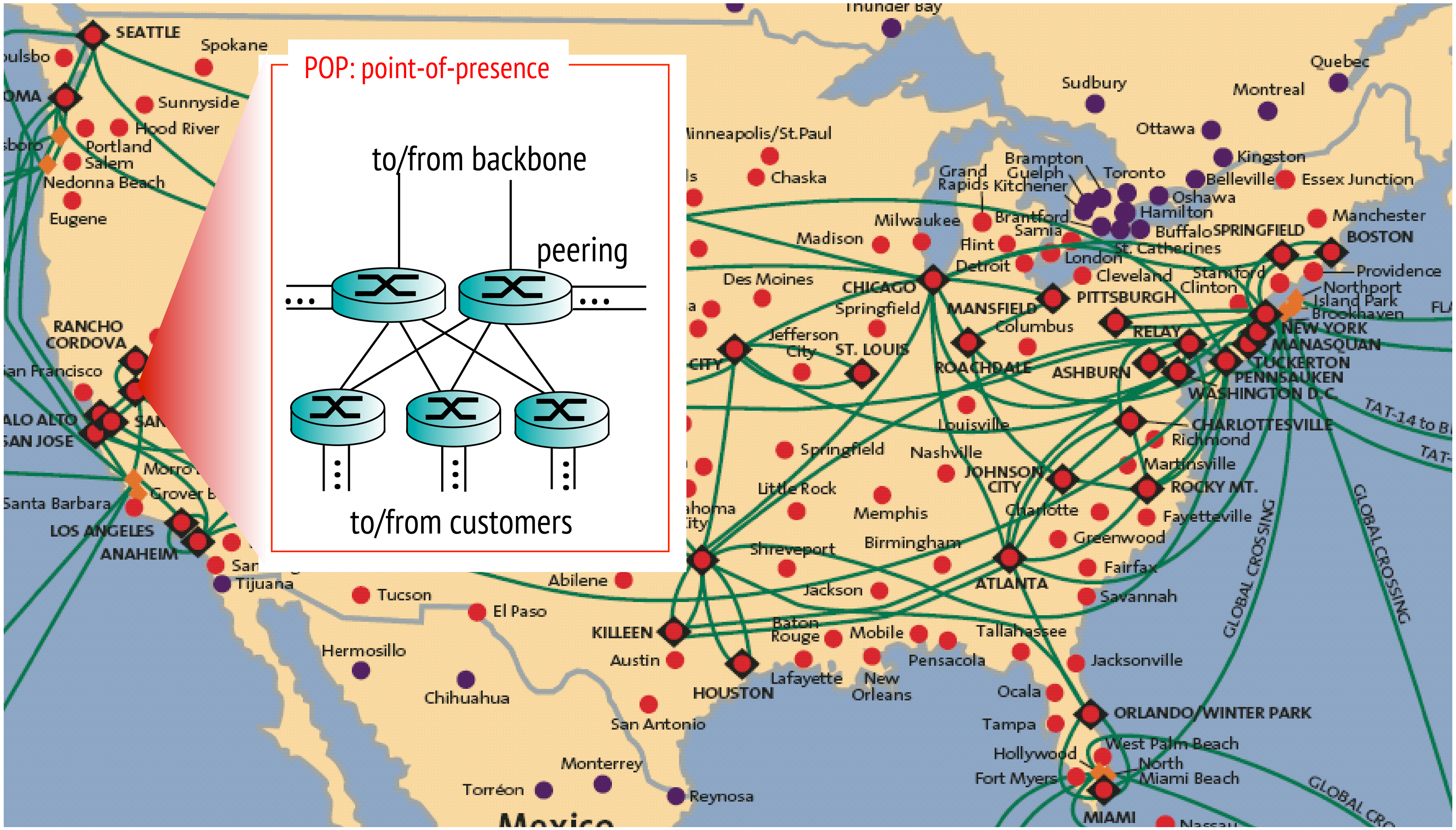


# Internet structure: network of networks



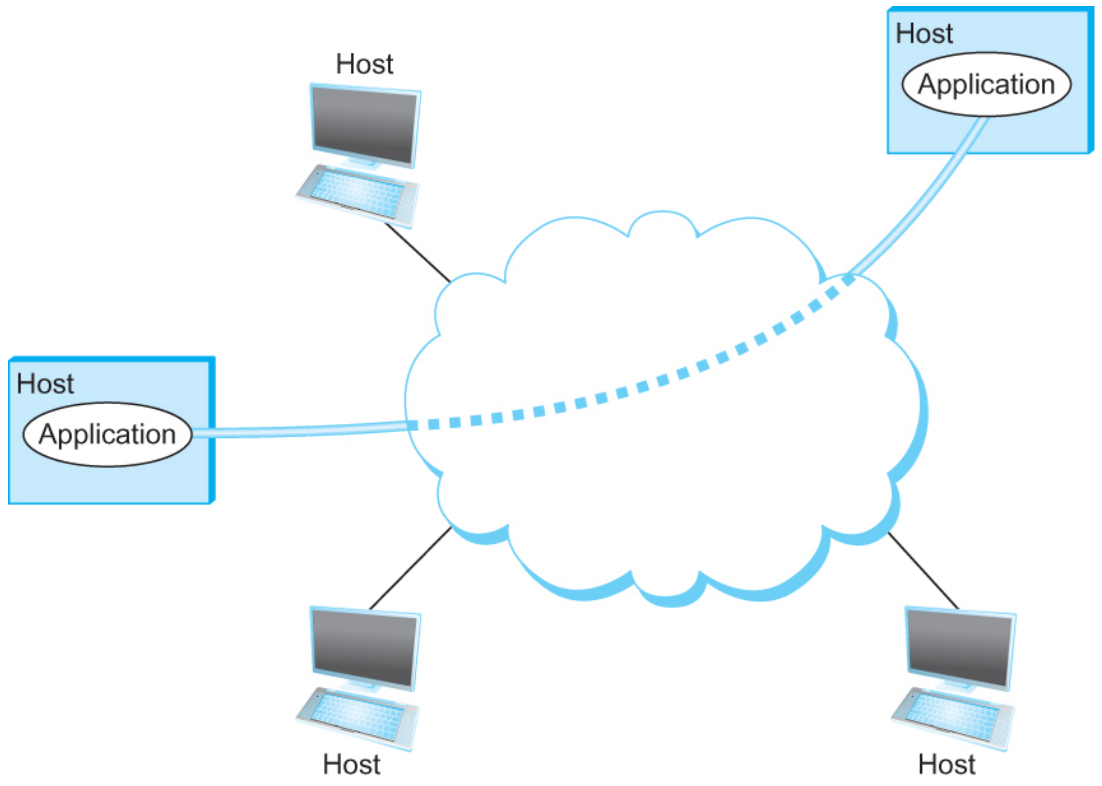
- at center: small # of well-connected large networks
  - “tier-1” commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT), national & international coverage
  - content provider network (e.g, Google): private network that connects it data centers to Internet, often bypassing tier-1, regional ISPs

# Tier-1 ISP: e.g., Sprint



## Support for Common Services

- Logical Channels
  - Application-to-Application communication path or a pipe



Process communicating over an abstract channel

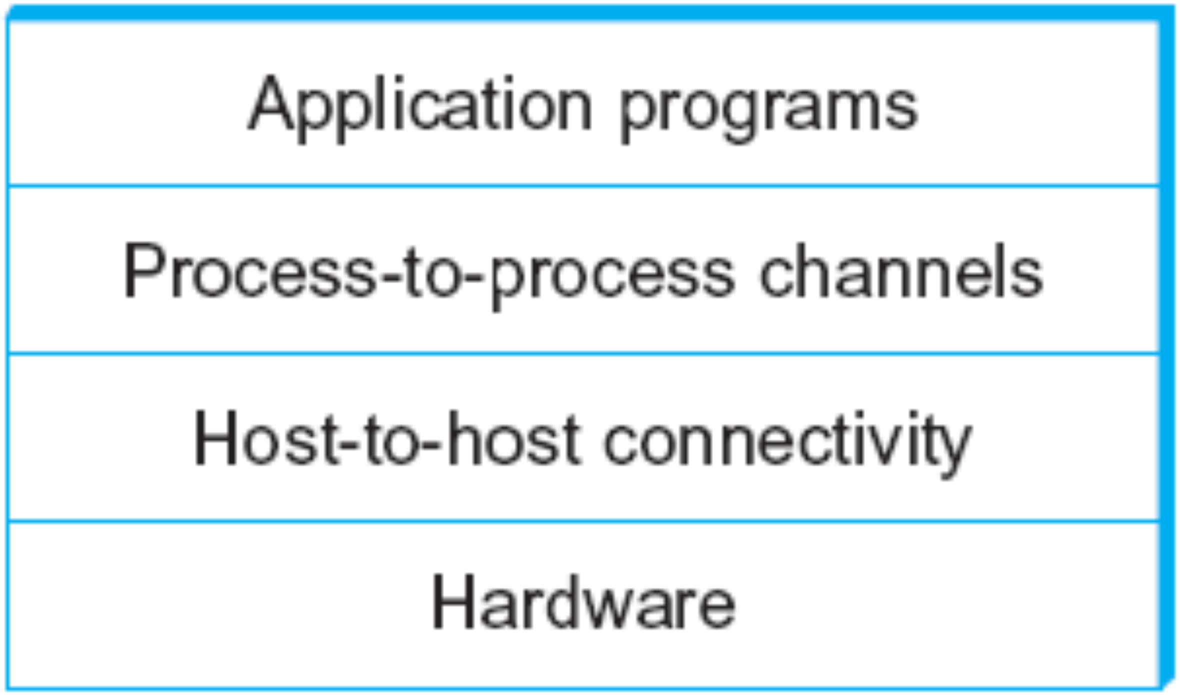
## Common Communication Patterns

- Client/Server
- Two types of communication channel
  - Request/Reply Channels
  - Message Stream Channels

## Reliability

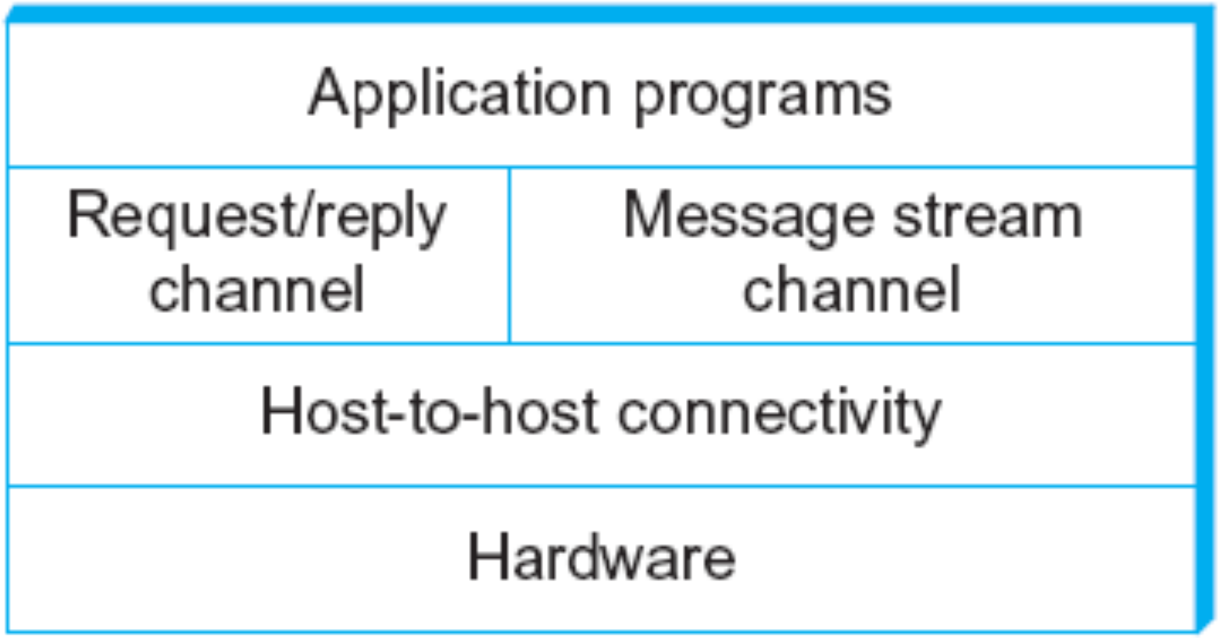
- Network should hide the errors
- Bits are lost
  - Bit errors (1 to a 0, and vice versa)
  - Burst errors – several consecutive errors
- Packets are lost (Congestion)
- Links and Node failures
- Messages are delayed
- Messages are delivered out-of-order
- Third parties eavesdrop

## Network Architecture



Example of a layered network system

## Network Architecture



Layered system with alternative abstractions available at a given layer

## What is a protocol?

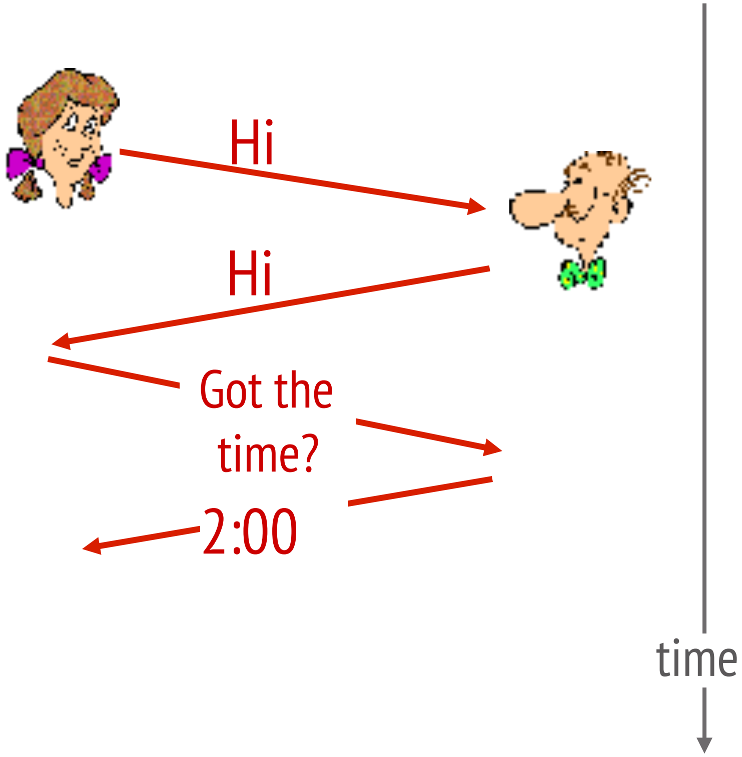
- Human Protocols:
  - What's the time?
  - I have a question….
  - Hi, I am XYZ….
  - Hello, how are you?

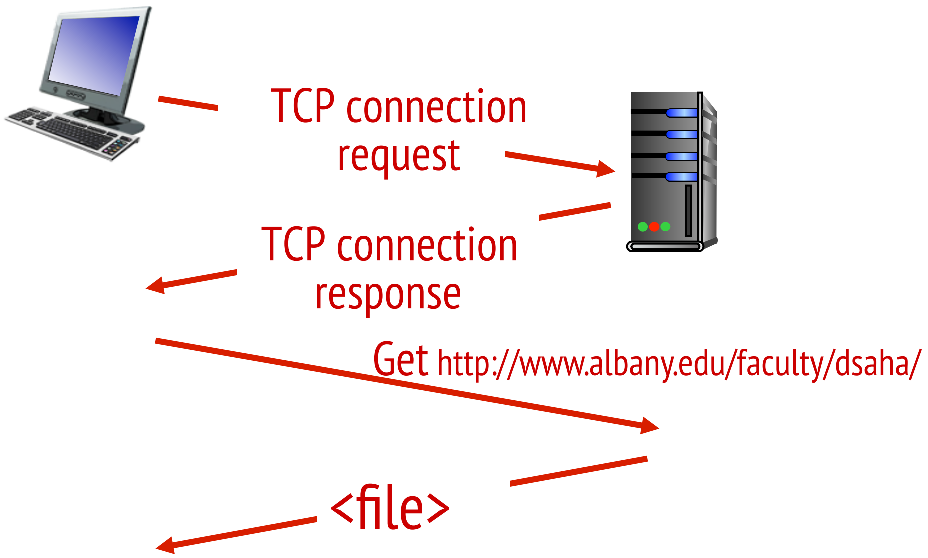
- Network Protocols:
  - Nodes / machines in the network participate
  - Communication activity in Internet is governed by Network Protocols
  - Are you alive?
  - Do you know the route to node X?

Protocols define format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

# What is a protocol?

- Human Protocols:
- Network Protocols:

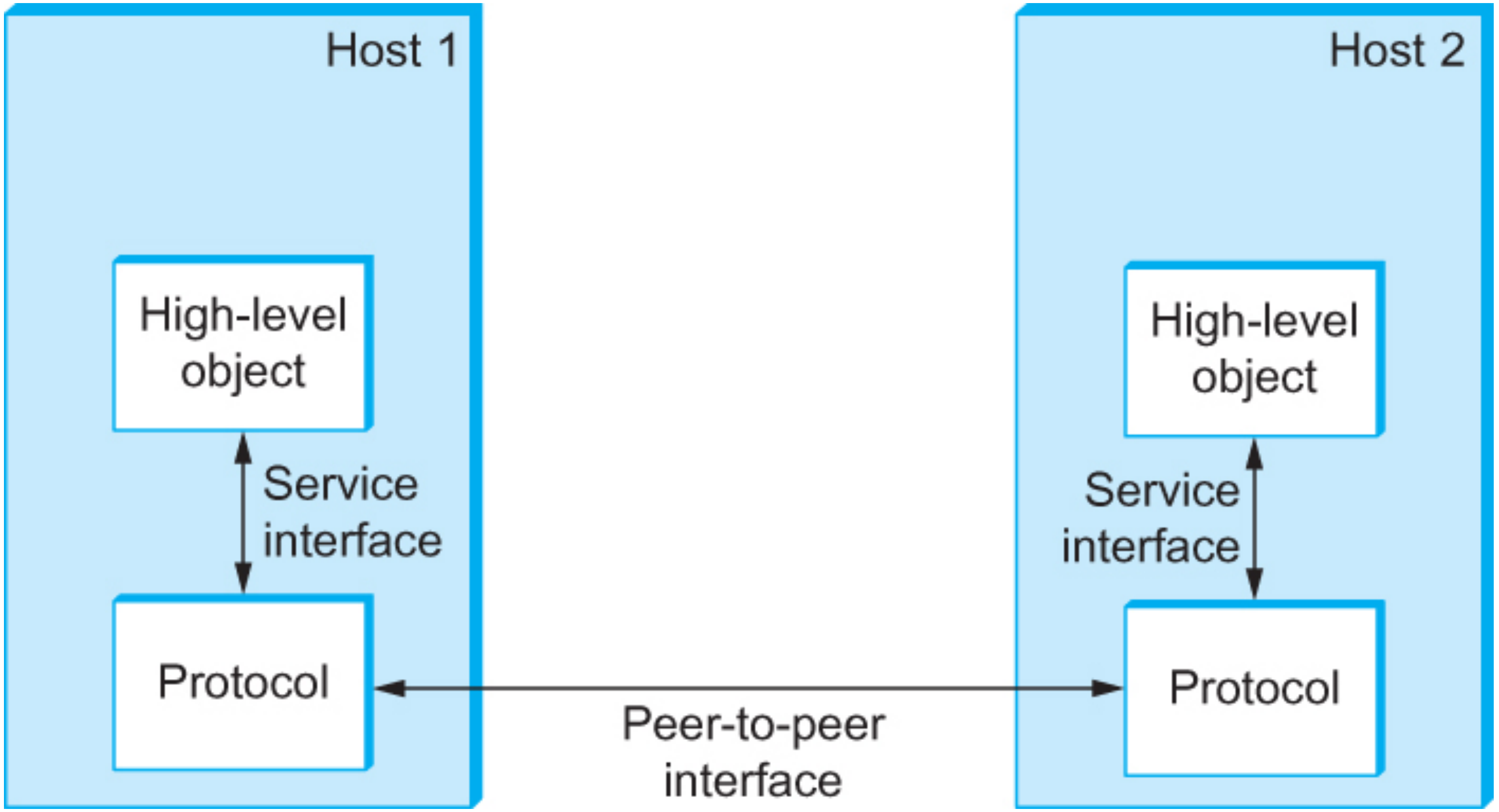




## Protocols

- Protocol defines the interfaces between the layers in the same system and with the layers of peer system
- Building blocks of a network architecture
- Each protocol object has two different interfaces
  - service interface: operations on this protocol
  - peer-to-peer interface: messages exchanged with peer
- Term “protocol” is overloaded
  - specification of peer-to-peer interface
  - module that implements this interface

# Interfaces

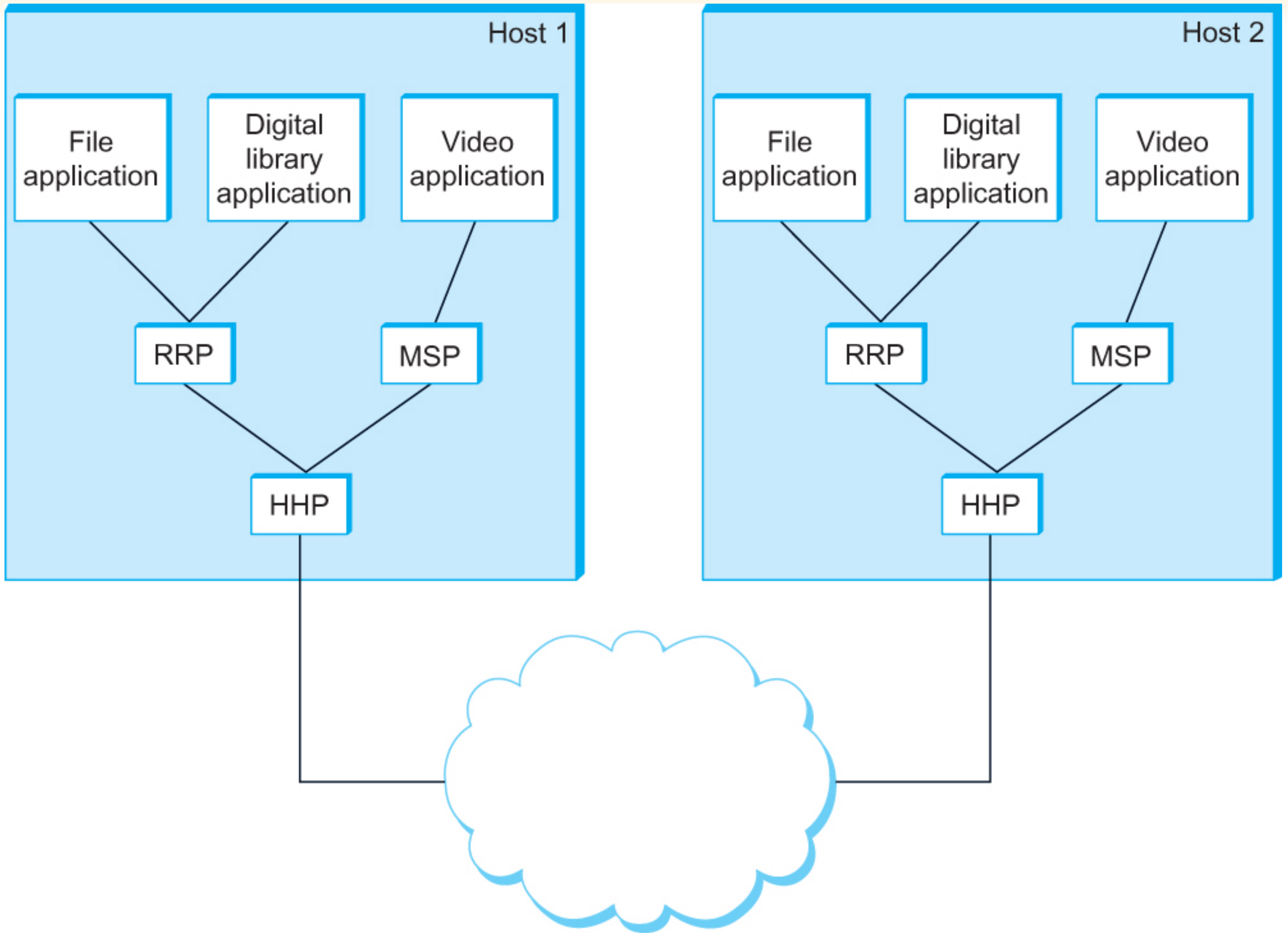


Service and Peer Interfaces

## Protocols

- Protocol Specification: prose, pseudo-code, state transition diagram
- Interoperable: when two or more protocols that implement the specification accurately
- IETF: Internet Engineering Task Force

## Protocol Graph



Example of a protocol graph nodes are the protocols and links the "depends-on" relation

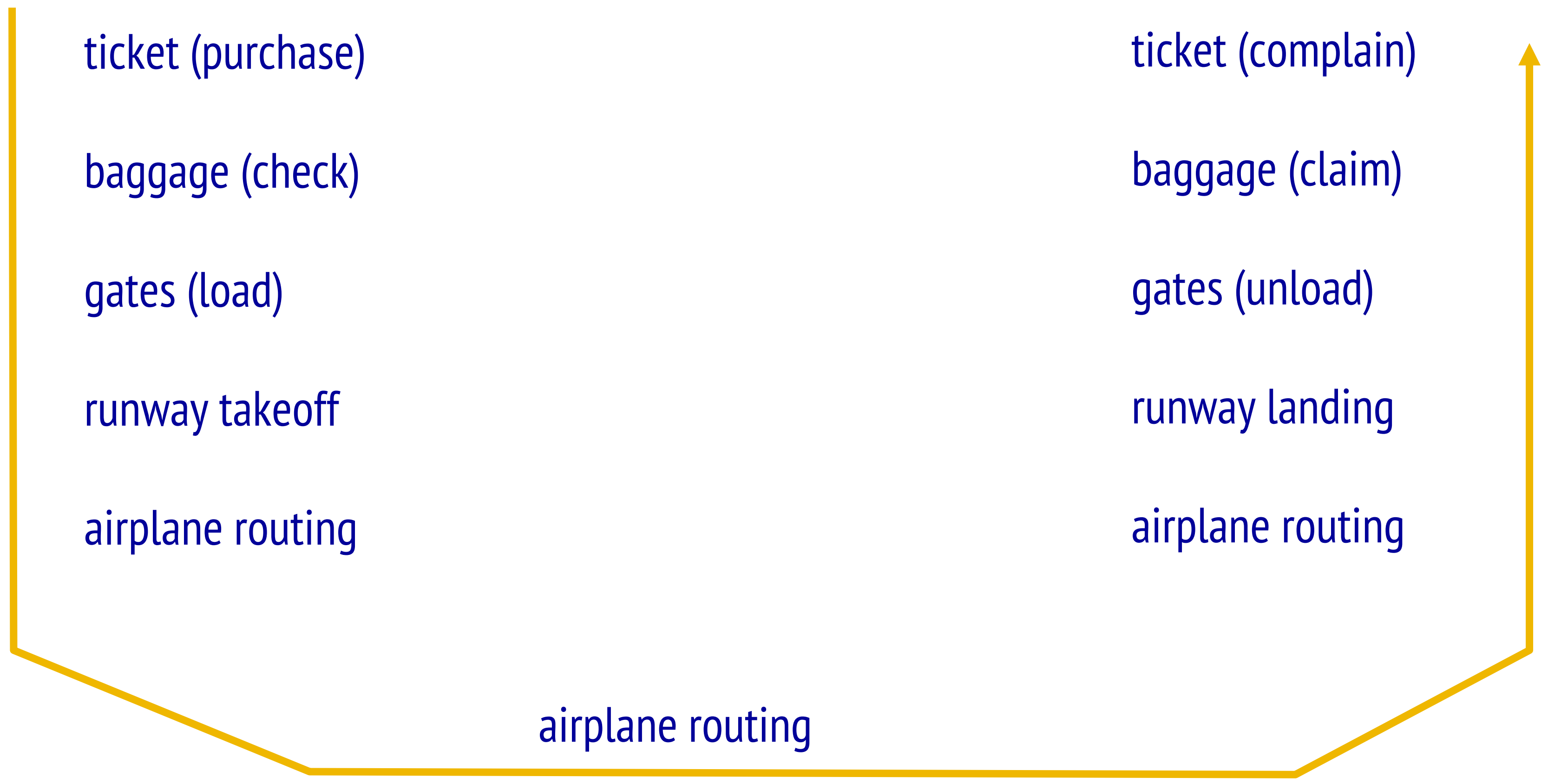
## Protocol Layers

- Networks are complex, with many "pieces":
  - hosts
  - routers
  - links of various media
  - applications
  - protocols
  - hardware, software

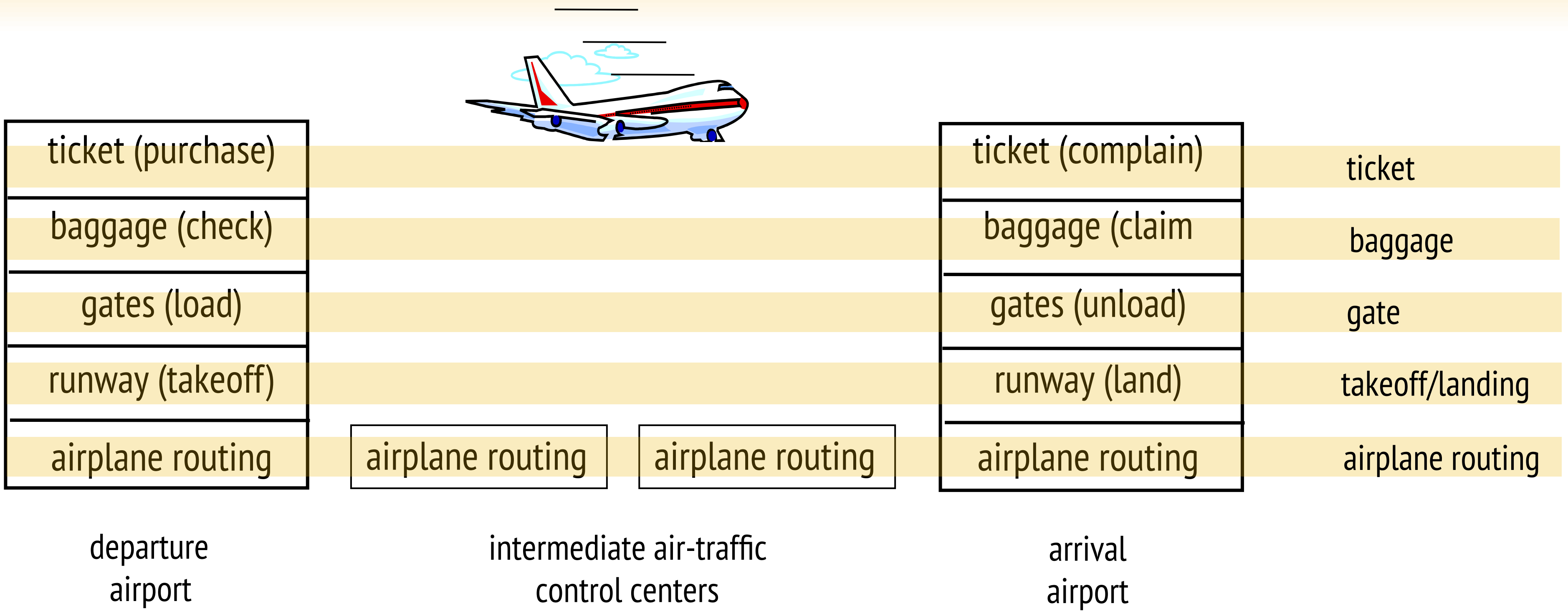
*Question:*

is there any hope of *organizing* structure of network?

## Organization of air travel



# Layering of airline functionality



- **layers**: each layer implements a service
  - via its own internal-layer actions
  - relying on services provided by layer below

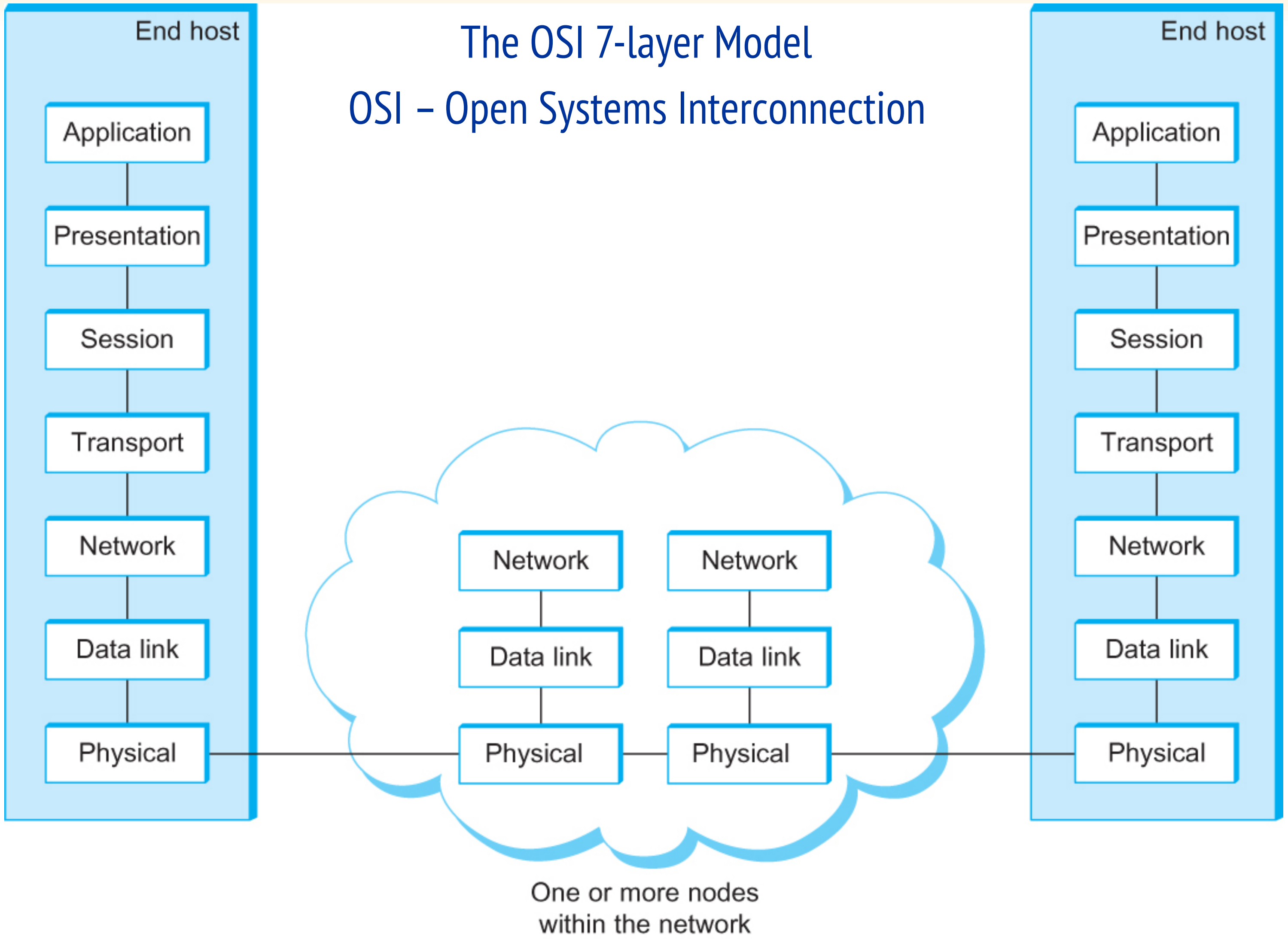
## Why layering?

- dealing with complex systems:
  - explicit structure allows identification, relationship of complex system’s pieces
    - layered reference model for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer’s service transparent to rest of system
  - e.g., change in gate procedure doesn’t affect rest of system
- layering considered harmful?

# OSI Architecture

The OSI 7-layer Model

OSI – Open Systems Interconnection



## Description of Layers

- Physical Layer
  - Handles the transmission of raw bits over a communication link
- Data Link Layer
  - Collects a stream of bits into a larger aggregate called a frame
  - Network adaptor along with device driver in OS implement the protocol in this layer
  - Frames are actually delivered to hosts
- Network Layer
  - Handles routing among nodes within a packet-switched network
  - Unit of data exchanged between nodes in this layer is called a packet

*The lower three layers are implemented on all network nodes*

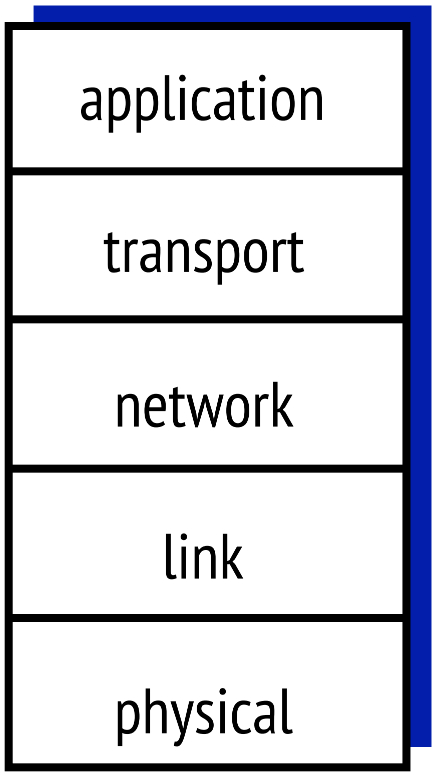
## Description of Layers

- Transport Layer
  - Implements a process-to-process channel
  - Unit of data exchanges in this layer is called a *message*
- Session Layer
  - Provides a name space that is used to tie together the potentially different transport streams that are part of a single application
- Presentation Layer
  - Concerned about the format of data exchanged between peers
- Application Layer
  - Standardize common type of exchanges

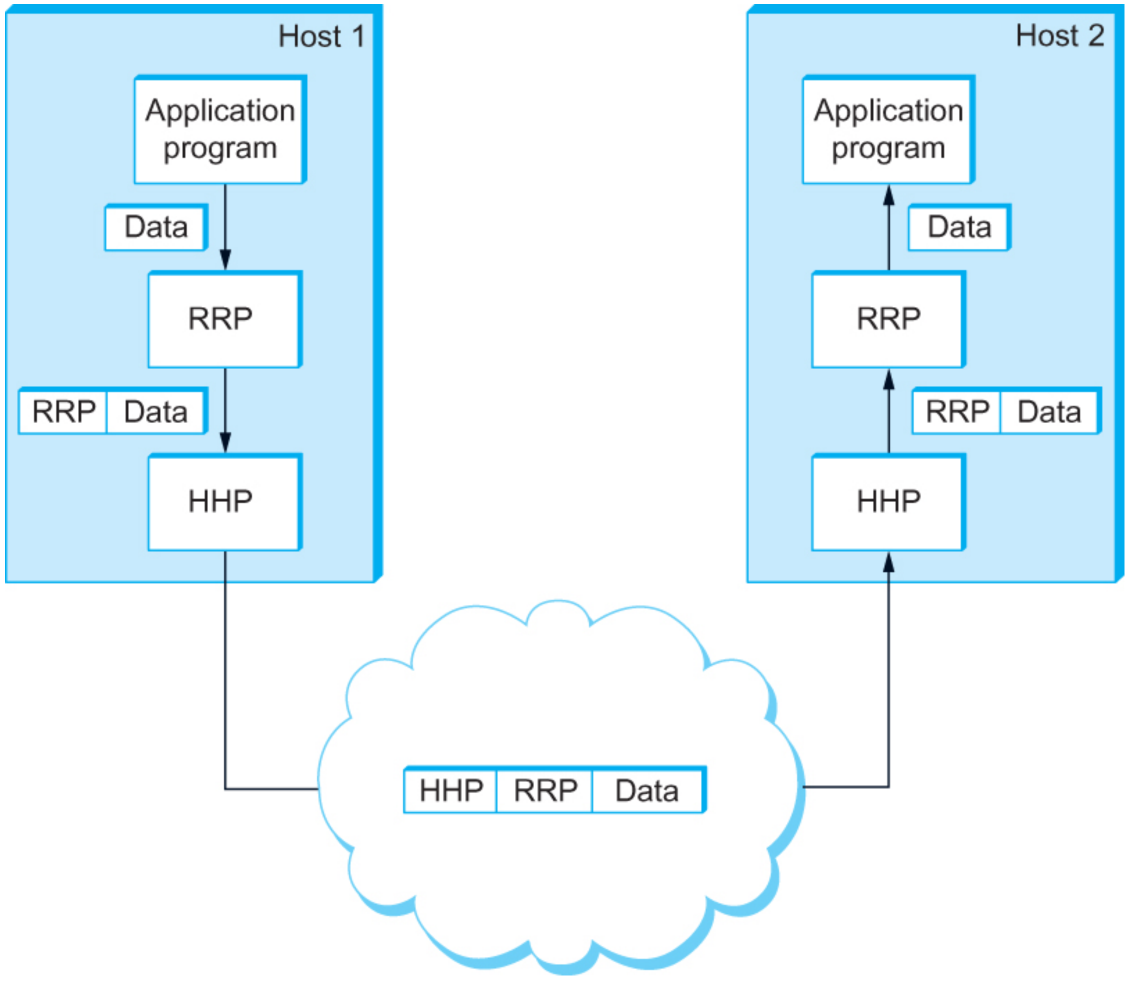
*The transport layer and the higher layers typically run only on end-hosts and not on the intermediate switches and routers*

## Internet Protocol Stack

- application: supporting network applications
  - FTP, SMTP, HTTP
- transport: process-process data transfer
  - TCP, UDP
- network: routing of datagrams from source to destination
  - IP, routing protocols
- link: data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi)
- physical: bits "on the wire" / "over the air"

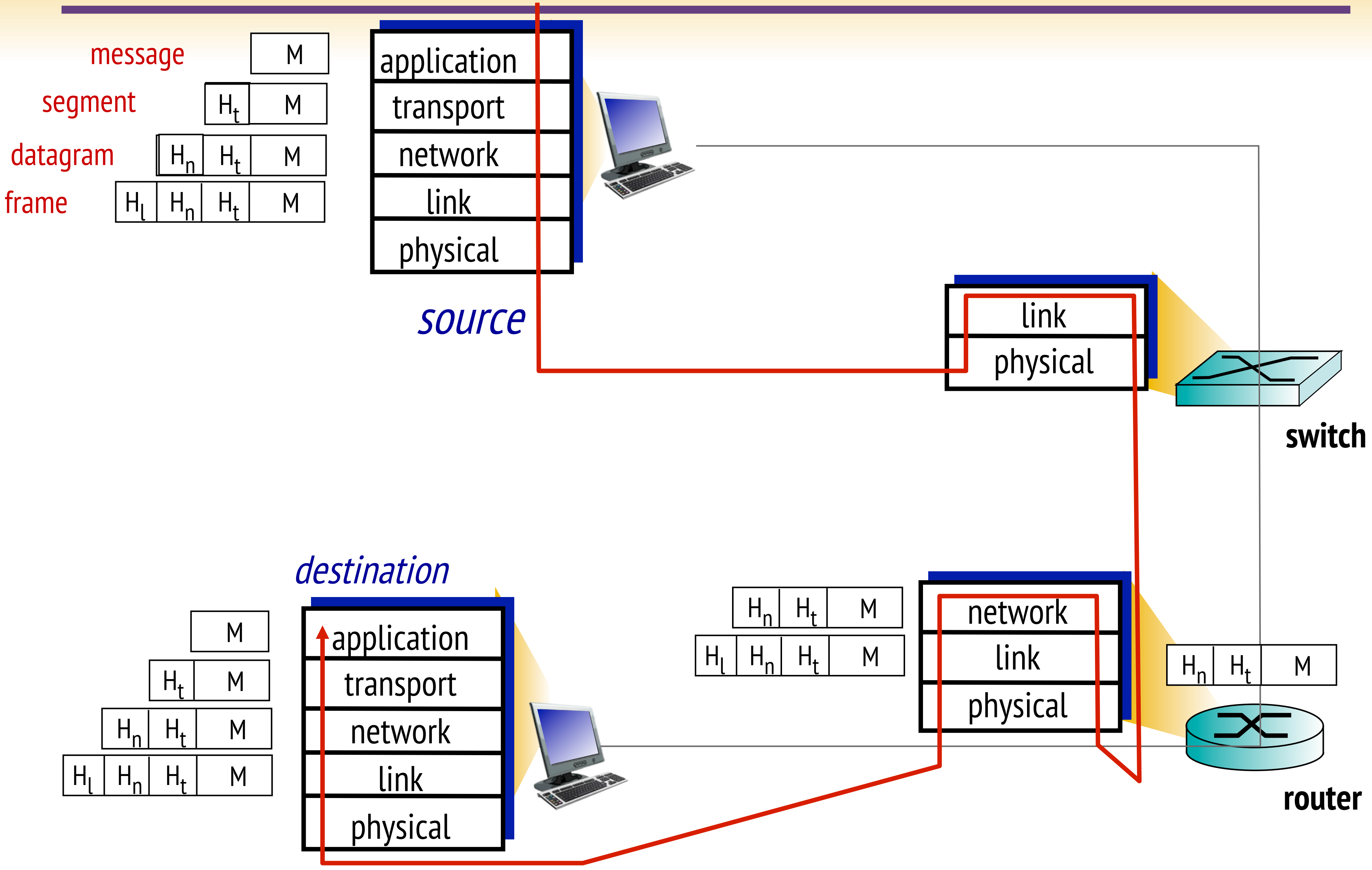


# Encapsulation

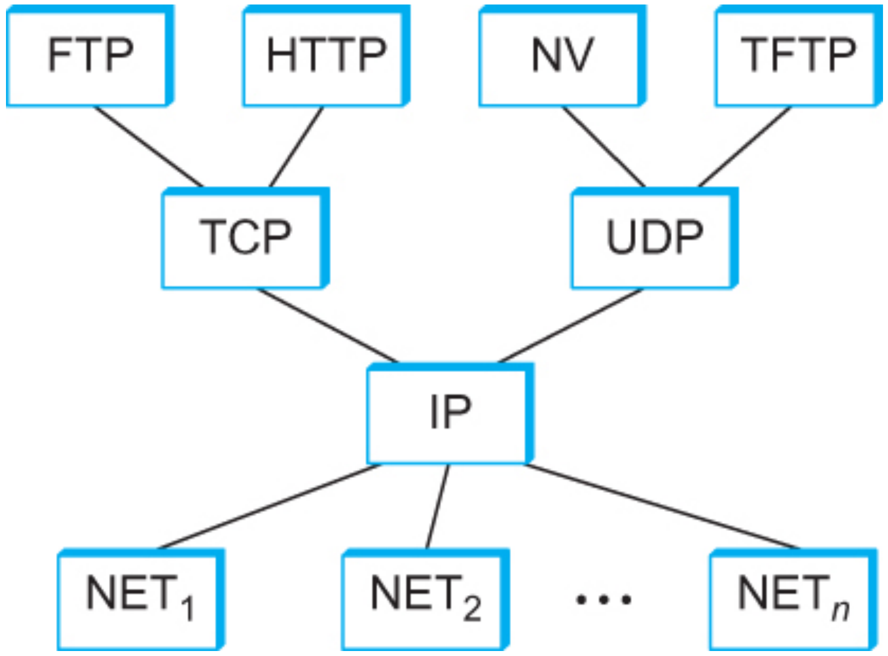


High-level messages are encapsulated inside of low-level messages

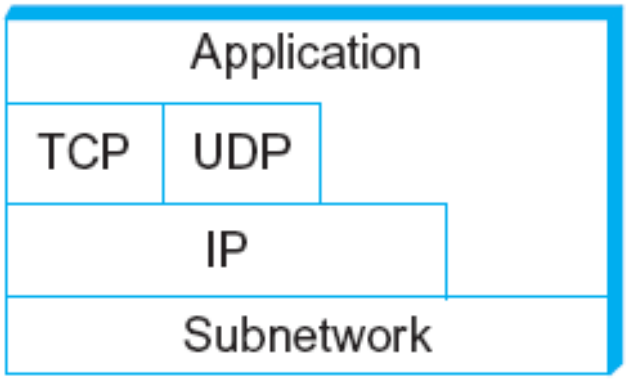
## Encapsulation



## Internet Architecture



Internet Protocol Graph



Alternative view of the Internet architecture. The "Network" layer shown here is sometimes referred to as the "sub-network" or "link" layer.

UNIVERSITY AT ALBANY
State University of New York

## Internet Architecture

- Defined by IETF (The Internet Engineering Task Force)

- Three main features
  - Does not imply strict layering. The application is free to bypass the defined transport layers and to directly use IP or other underlying networks
  - An hour-glass shape – wide at the top, narrow in the middle and wide at the bottom. IP serves as the focal point for the architecture
  - In order for a new protocol to be officially included in the architecture, there needs to be both a protocol specification and at least one (and preferably two) representative implementations of the specification

## Application Programming Interface

- Interface exported by the network

- Since most network protocols are implemented (those in the high protocol stack) in software and nearly all computer systems implement their network protocols as part of the operating system, when we refer to the interface "*exported by the network*", we are generally referring to the interface that the OS provides to its networking subsystem

- The interface is called the network Application Programming Interface (API)

## Application Programming Interface (Sockets)

- Socket Interface was originally provided by the Berkeley distribution of Unix
  - Now supported in virtually all operating systems

- Each protocol provides a certain set of *services*, and the API provides a syntax by which those services can be invoked in this particular OS

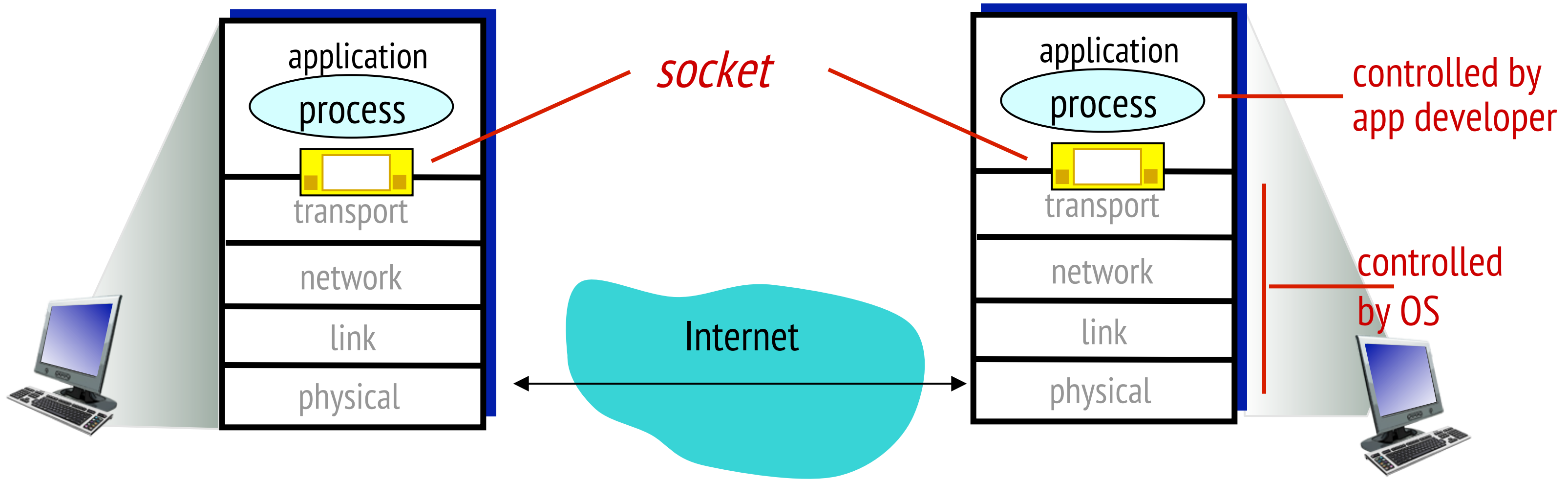
## Socket

- What is a socket?
  - The point where a local application process attaches to the network
  - An interface between an application and the network
  - An application creates the socket

- The interface defines operations for
  - Creating a socket
  - Attaching a socket to the network
  - Sending and receiving messages through the socket
  - Closing the socket

## Sockets

- process sends/receives messages to/from its socket
- socket analogous to door between application process & end-end-transport
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process



## Socket programming

*Two socket types for two transport services:*

- *UDP:* unreliable datagram
- *TCP:* reliable, byte stream-oriented

*Application Example:*

1. client reads a line of characters (data) from its keyboard and sends data to server
2. server receives the data and converts characters to uppercase
3. server sends modified data to client
4. client receives modified data and displays line on its screen

## Socket programming *with UDP*

UDP: no "connection" between client & server

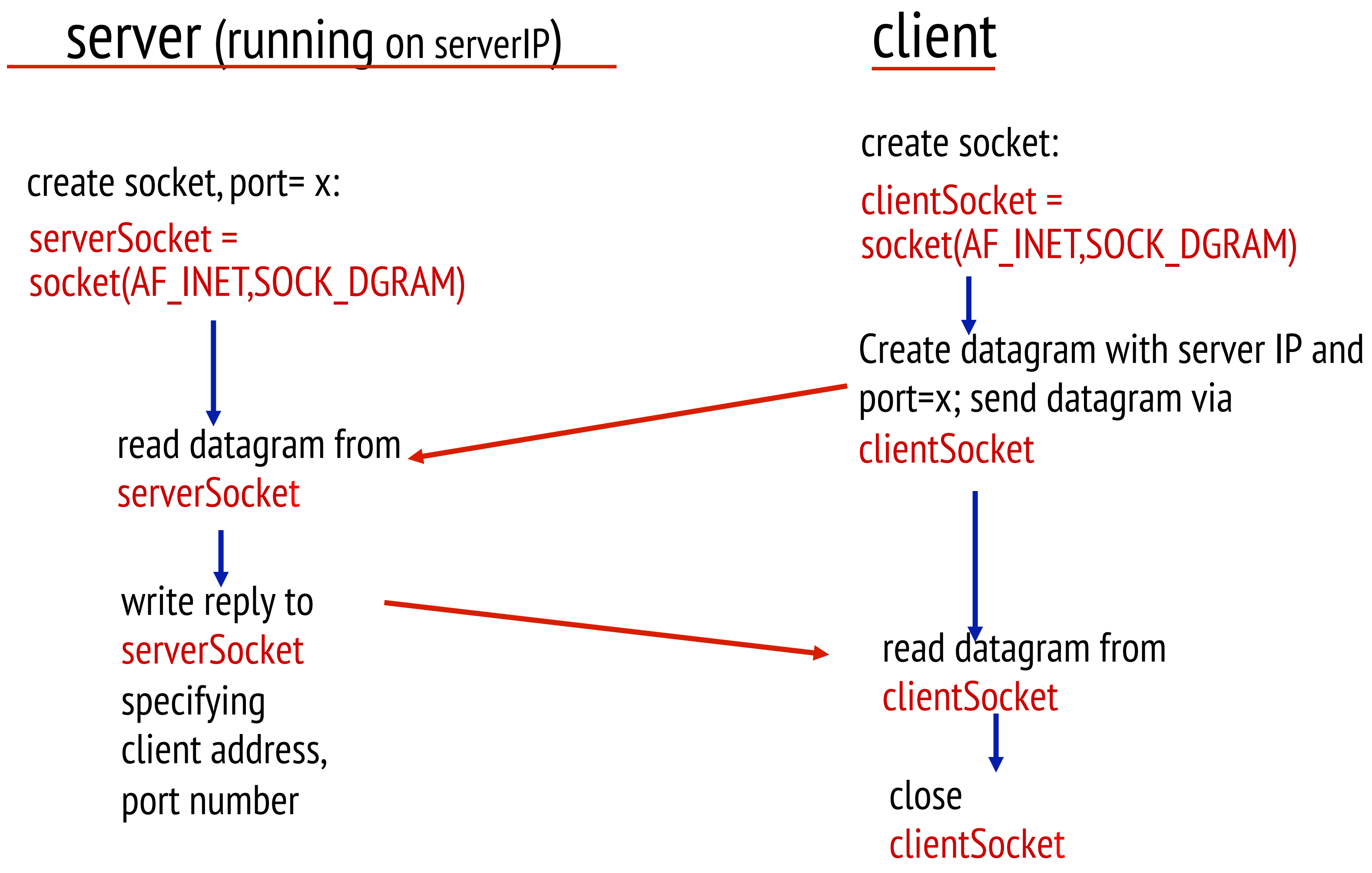
- no handshaking before sending data
- sender explicitly attaches IP destination address and port # to each packet
- receiver extracts sender IP address and port# from received packet

UDP: transmitted data may be lost or received out-of-order

Application viewpoint:

- UDP provides *unreliable* transfer of groups of bytes ("datagrams") between client and server

# Client/server socket interaction: UDP



# Example app: UDP client

*Python UDPClient*

include Python's socket library →

```
from socket import *
serverName = 'hostname'
serverPort = 12000
```

create UDP socket for server →

```
clientSocket = socket(AF_INET,
                      SOCK_DGRAM)
```

get user keyboard input →

```
message = raw_input('Input lowercase sentence:')
```

Attach server name, port to message; send into socket →

```
clientSocket.sendto(message.encode(),
                    (serverName, serverPort))
```

read reply characters from socket into string →

```
modifiedMessage, serverAddress =
                    clientSocket.recvfrom(2048)
```

print out received string and close socket →

```
print modifiedMessage.decode()
clientSocket.close()
```

UNIVERSITY AT ALBANY
State University of New York

# Example app: UDP server

*Python UDPServer*

create UDP socket

bind socket to local port number 12000

loop forever

Read from UDP socket into message, getting client's address (client IP and port)

send upper case string back to this client

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print ("The server is ready to receive")
while True:
   message, clientAddress = serverSocket.recvfrom(2048)
   modifiedMessage = message.decode().upper()
   serverSocket.sendto(modifiedMessage.encode(),
                            clientAddress)
```

UNIVERSITY AT ALBANY
State University of New York

# Socket programming *with TCP*

client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client's contact

client contacts server by:

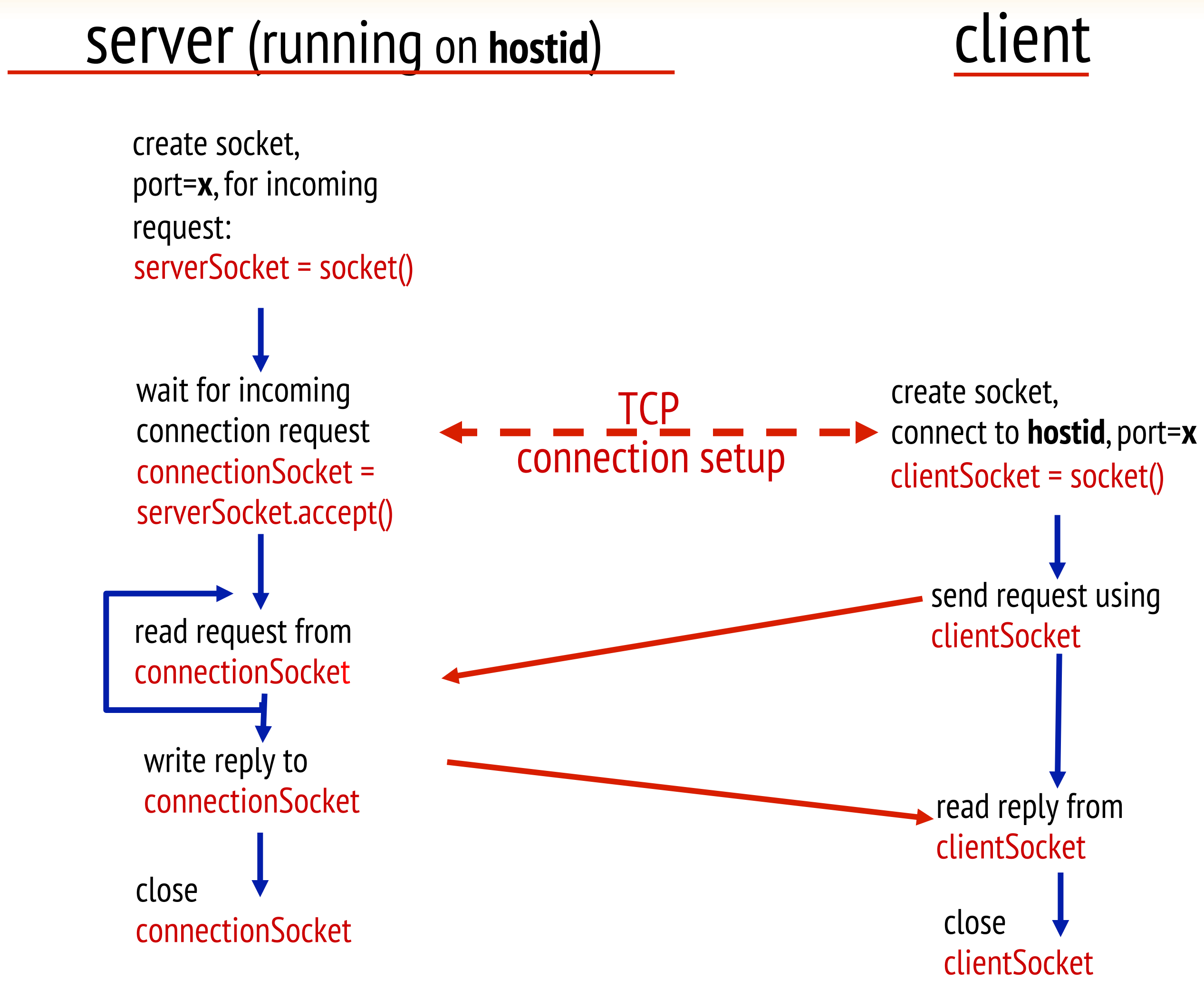
- Creating TCP socket, specifying IP address, port number of server process
- *when client creates socket:* client TCP establishes connection to server TCP

- when contacted by client, *server TCP creates new socket* for server process to communicate with that particular client
  - allows server to talk with multiple clients
  - source port numbers used to distinguish clients (more in Chap 3)

application viewpoint:

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server

# Client/server socket interaction: TCP



server (running on **hostid**)

client

create socket,
port=**x**, for incoming
request:
serverSocket = socket()

wait for incoming
connection request
connectionSocket =
serverSocket.accept()

TCP
connection setup

create socket,
connect to **hostid**, port=**x**
clientSocket = socket()

send request using
clientSocket

read request from
connectionSocket

write reply to
connectionSocket

read reply from
clientSocket

close
connectionSocket

close
clientSocket

UNIVERSITY AT ALBANY
State University of New York

## Example app: TCP client

*Python TCPClient*

create TCP socket for server, remote port 12000

No need to attach server name, port

```
from socket import *
serverName = ’servername’
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input(‘Input lowercase sentence:’)
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print (‘From Server:’, modifiedSentence.decode())
clientSocket.close()
```

UNIVERSITY AT ALBANY
State University of New York

## Example app: TCP server

*Python TCPServer*

create TCP welcoming socket

server begins listening for incoming TCP requests

loop forever

server waits on accept() for incoming requests, new socket created on return

read bytes from socket (but not address as in UDP)

close connection to this client (but *not* welcoming socket)

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((‘’,serverPort))
serverSocket.listen(1)
print ‘The server is ready to receive’
while True:
    connectionSocket, addr = serverSocket.accept()

    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.
                                        encode())
    connectionSocket.close()
```

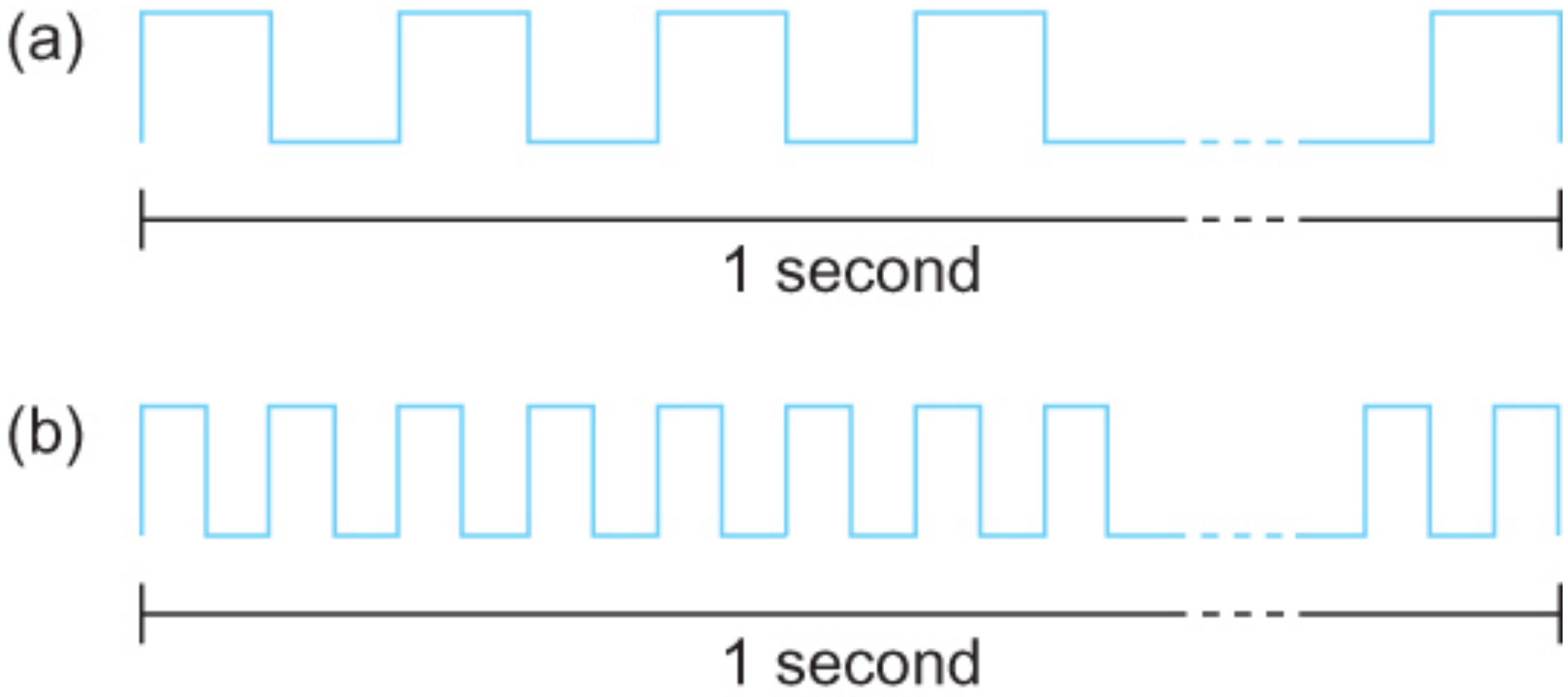
## Port Numbers

- Commonly used:
  - *21*: File Transfer Protocol (FTP)
  - *22*: Secure Shell (SSH)
  - *23*: Telnet remote login service
  - *25*: Simple Mail Transfer Protocol (SMTP)
  - *80*: Hypertext Transfer Protocol (HTTP) used in the World Wide Web
  - *110*: Post Office Protocol (POP3)
  - *123*: Network Time Protocol (NTP)
  - *143*: Internet Message Access Protocol (IMAP)
  - *443*: HTTP Secure (HTTPS)
- The registered ports are those from 1024 through 49151. IANA maintains the official list of registered ports.
- The dynamic or private ports are those from 49152 through 65535.

## Performance

- Bandwidth
  - Width of the frequency band
  - Number of bits per second that can be transmitted over a communication link
- 1 Mbps: 1 x $10^6$ bits/second = $1x2^{20}$ bits/sec
- 1 x $10^{-6}$ seconds to transmit each bit or imagine that a timeline, now each bit occupies 1 micro second space.
- On a 2 Mbps link the width is 0.5 micro second.
- Smaller the width more will be transmission per unit time.

## Bandwidth

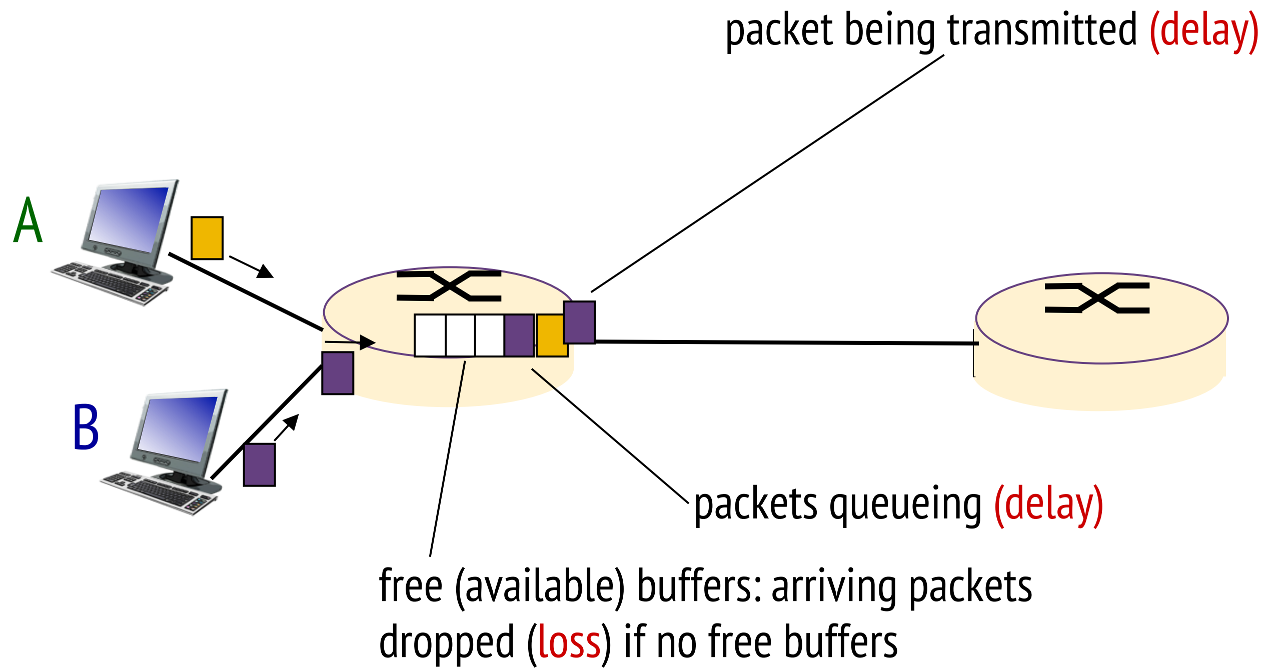


Bits transmitted at a particular bandwidth can be regarded as having some width:
(a) bits transmitted at 1Mbps (each bit 1 μs wide);
(b) bits transmitted at 2Mbps (each bit 0.5 μs wide).

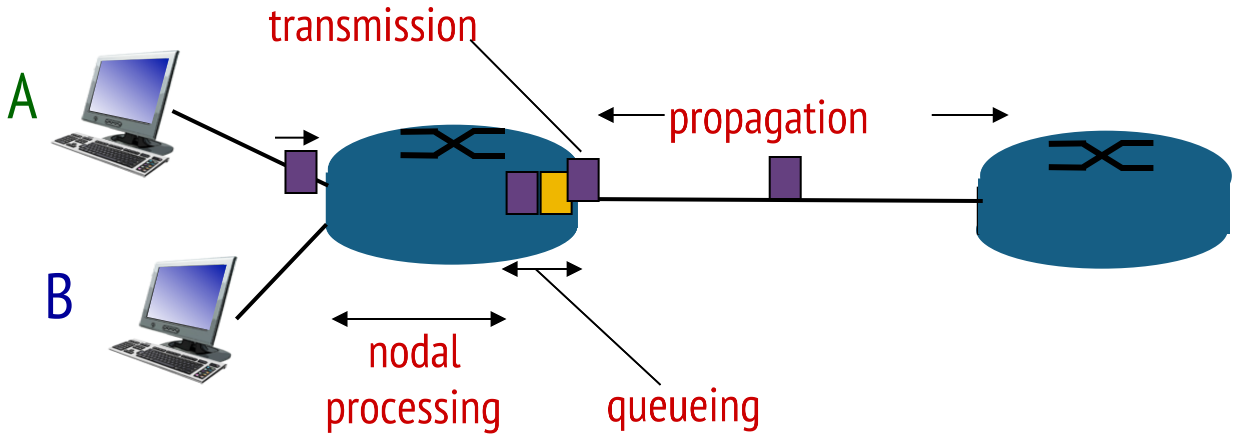
## How do loss and delay occur?

packets *queue* in router buffers

- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn


packet being transmitted (delay)
A
B
packets queueing (delay)
free (available) buffers: arriving packets dropped (loss) if no free buffers

# Four sources of packet delay



$$\boxed{d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}}$$

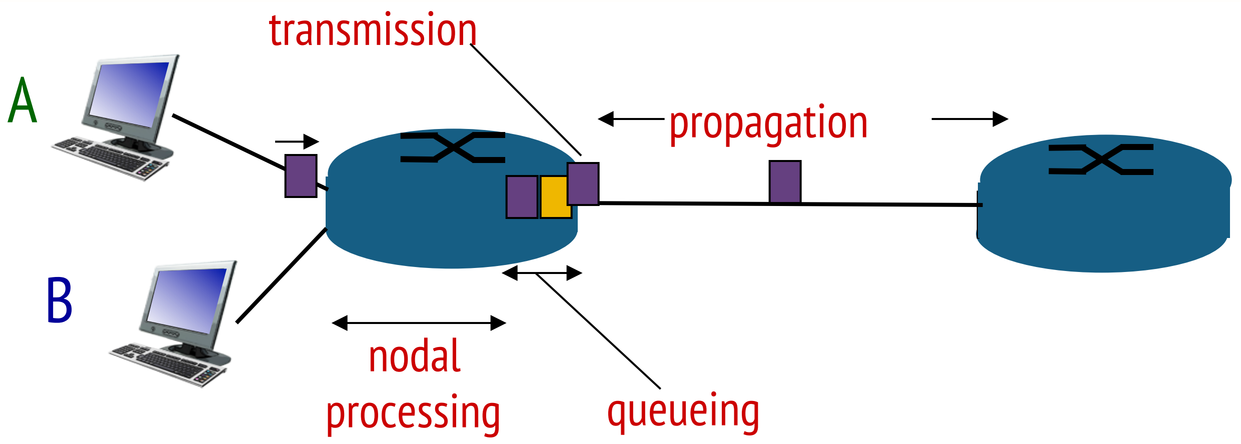
## $d_{\text{proc}}$: nodal processing

- check bit errors
- determine output link
- typically < msec

## $d_{\text{queue}}$: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

# Four Sources of Packet Delay



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

$d_{\text{trans}}$: transmission delay:

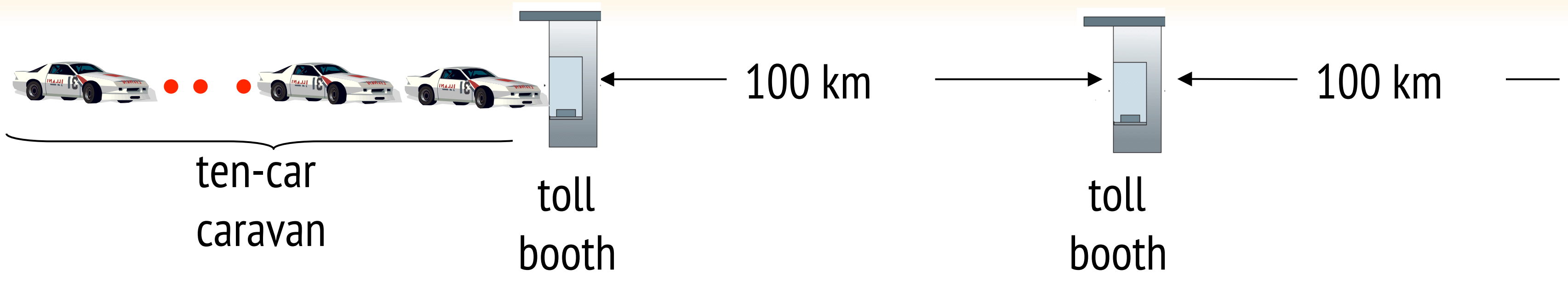
- $L$: packet length (bits)
- $R$: link *bandwidth (bps)*
- $d_{\text{trans}} = L/R$

$d_{\text{prop}}$: propagation delay:

- $d$: length of physical link
- $s$: propagation speed in medium (~$2 \times 10^8$ m/sec)
- $d_{\text{prop}} = d/s$

$d_{\text{trans}}$ and $d_{\text{prop}}$ *very* different

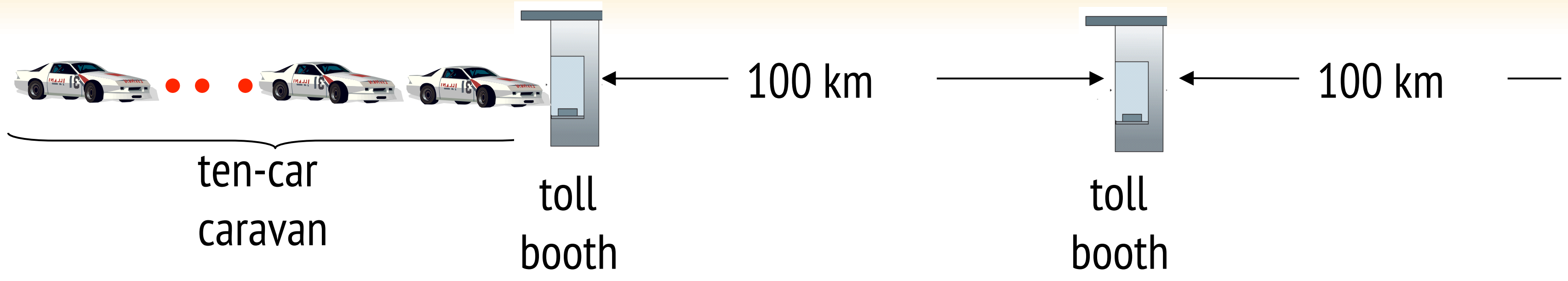
## Caravan analogy



- cars "propagate" at 100 km/hr
- toll booth takes 12 sec to service car (bit transmission time)
- car~bit; caravan ~ packet
- *Q:* How long until caravan is lined up before 2nd toll booth?

- time to "push" entire caravan through toll booth onto highway = 12*10 = 120 sec
- time for last car to propagate from 1st to 2nd toll both: 100km/(100km/hr)= 1 hr
- *A:* 62 minutes

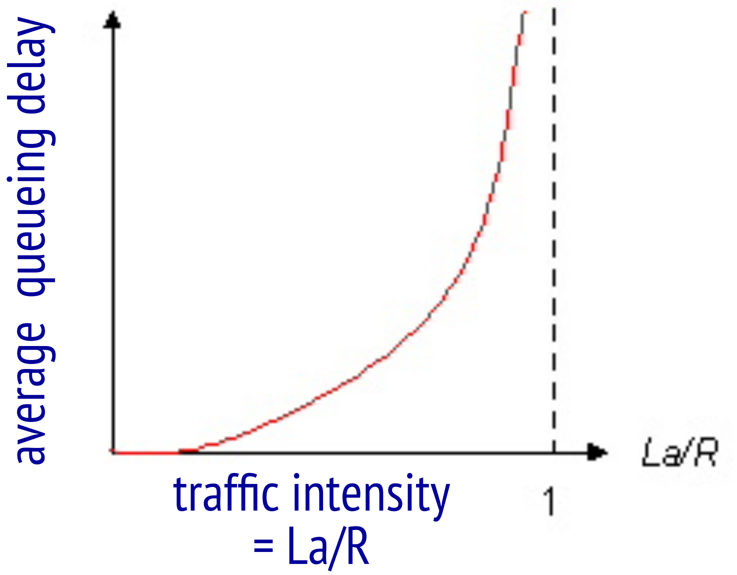
## Caravan analogy (more)



- suppose cars now "propagate" at 1000 km/hr
- and suppose toll booth now takes one min to service a car
- *Q:* Will cars arrive to 2nd booth before all cars serviced at first booth?
    - *A: Yes!* after 7 min, 1st car arrives at second booth; three cars still at 1st booth.

# Queueing delay (revisited)

- *R:* link bandwidth (bps)
- *L:* packet length (bits)
- a: average packet arrival rate



- *La/R* ~ 0: avg. queueing delay small
- *La/R* -> 1: avg. queueing delay large
- *La/R* > 1: more "work" arriving

  than can be serviced, average delay infinite!

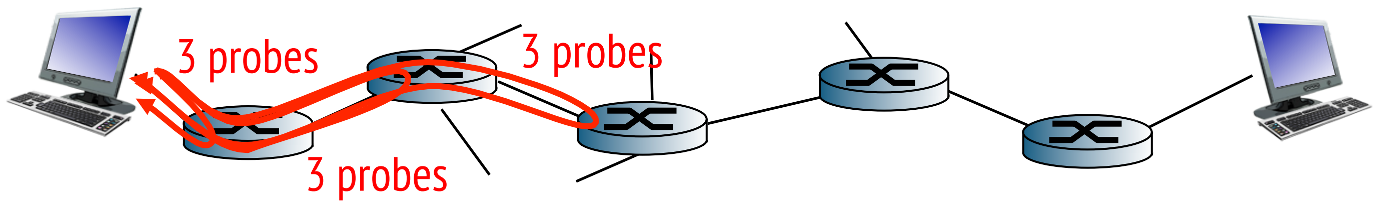
La/R ~ 0

La/R -> 1

UNIVERSITY AT ALBANY
State University of New York

## “Real” Internet delays and routes

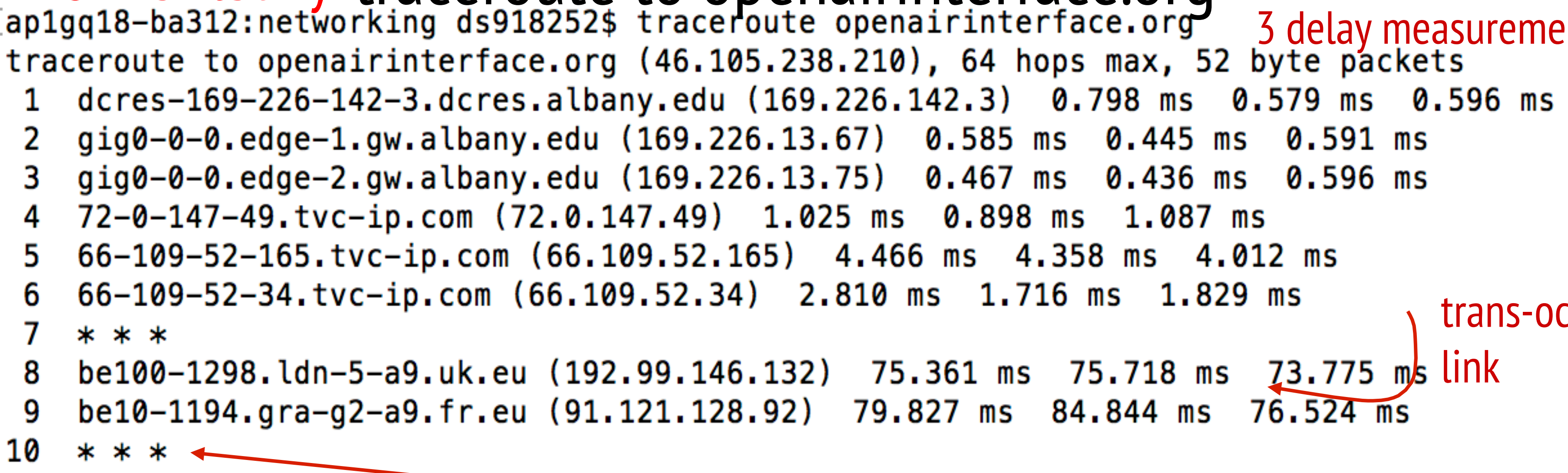
- what do “real” Internet delay & loss look like?
- traceroute program: provides delay measurement from source to router along end-end Internet path towards destination. For all *i:*
  - sends three packets that will reach router *i* on path towards destination
  - router *i* will return packets to sender
  - sender times interval between transmission and reply.



# Real Internet Delays, routes

From UAlbany traceroute to openairinterface.org

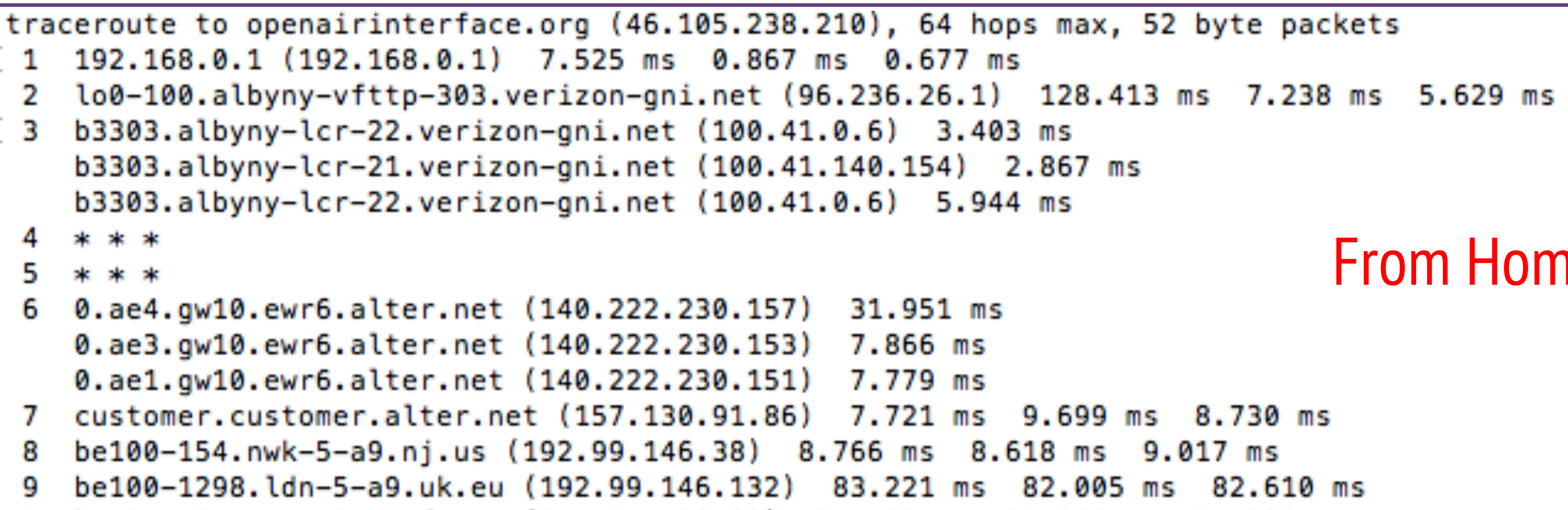
3 delay measurements



```
ap1gq18-ba312:networking ds918252$ traceroute openairinterface.org
traceroute to openairinterface.org (46.105.238.210), 64 hops max, 52 byte packets
 1  dcres-169-226-142-3.dcres.albany.edu (169.226.142.3)  0.798 ms  0.579 ms  0.596 ms
 2  gig0-0-0.edge-1.gw.albany.edu (169.226.13.67)  0.585 ms  0.445 ms  0.591 ms
 3  gig0-0-0.edge-2.gw.albany.edu (169.226.13.75)  0.467 ms  0.436 ms  0.596 ms
 4  72-0-147-49.tvc-ip.com (72.0.147.49)  1.025 ms  0.898 ms  1.087 ms
 5  66-109-52-165.tvc-ip.com (66.109.52.165)  4.466 ms  4.358 ms  4.012 ms
 6  66-109-52-34.tvc-ip.com (66.109.52.34)  2.810 ms  1.716 ms  1.829 ms
 7  * * *
 8  be100-1298.ldn-5-a9.uk.eu (192.99.146.132)  75.361 ms  75.718 ms  73.775 ms
 9  be10-1194.gra-g2-a9.fr.eu (91.121.128.92)  79.827 ms  84.844 ms  76.524 ms
10  * * *
```

trans-oceanic link

* means no response (probe lost, router not replying)

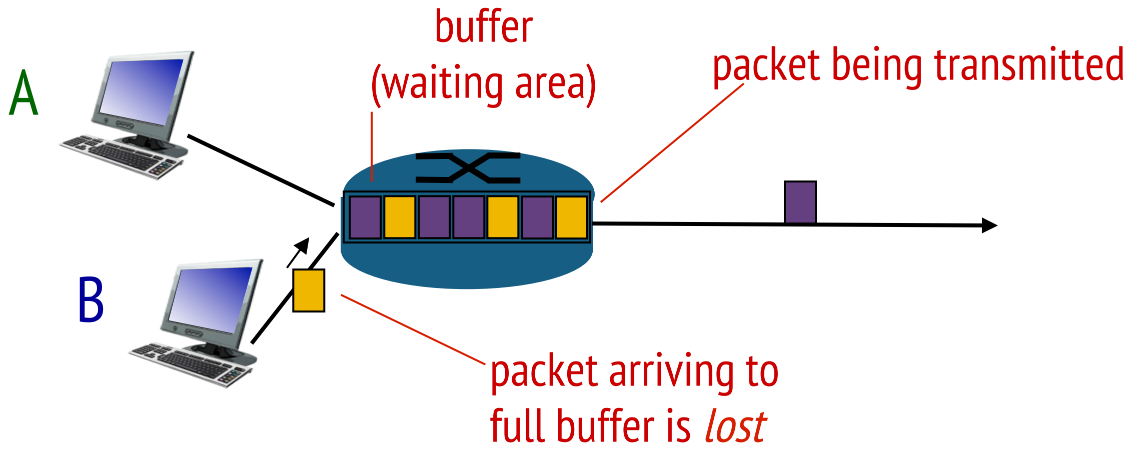


```
traceroute to openairinterface.org (46.105.238.210), 64 hops max, 52 byte packets
 1  192.168.0.1 (192.168.0.1)  7.525 ms  0.867 ms  0.677 ms
 2  lo0-100.albyny-vfttp-303.verizon-gni.net (96.236.26.1)  128.413 ms  7.238 ms  5.629 ms
 3  b3303.albyny-lcr-22.verizon-gni.net (100.41.0.6)  3.403 ms
    b3303.albyny-lcr-21.verizon-gni.net (100.41.140.154)  2.867 ms
    b3303.albyny-lcr-22.verizon-gni.net (100.41.0.6)  5.944 ms
 4  * * *
 5  * * *
 6  0.ae4.gw10.ewr6.alter.net (140.222.230.157)  31.951 ms
    0.ae3.gw10.ewr6.alter.net (140.222.230.153)  7.866 ms
    0.ae1.gw10.ewr6.alter.net (140.222.230.151)  7.779 ms
 7  customer.customer.alter.net (157.130.91.86)  7.721 ms  9.699 ms  8.730 ms
 8  be100-154.nwk-5-a9.nj.us (192.99.146.38)  8.766 ms  8.618 ms  9.017 ms
 9  be100-1298.ldn-5-a9.uk.eu (192.99.146.132)  83.221 ms  82.005 ms  82.610 ms
```

From Home

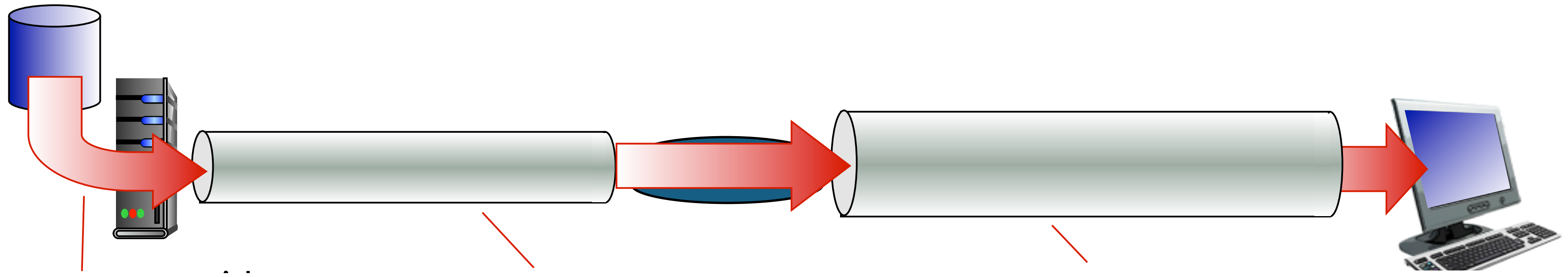
## Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



## Throughput

- *throughput:* rate (bits/time unit) at which bits transferred between sender/receiver
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time

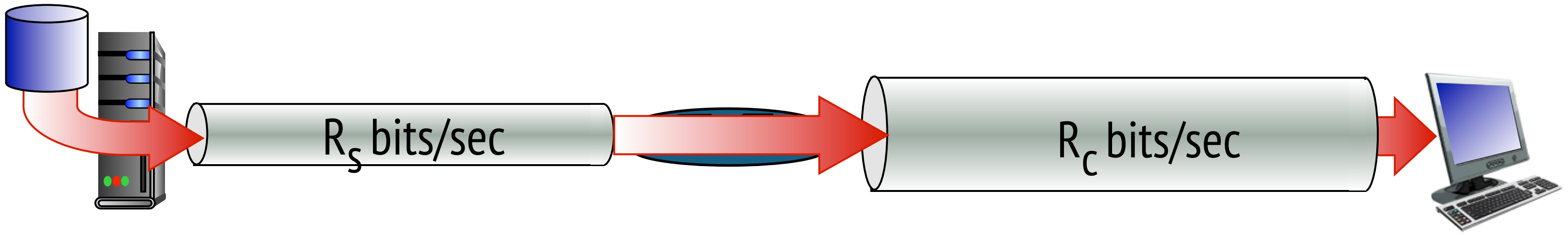

server sends bits (fluid) into pipe

pipe that can carry fluid at rate $R_s$ bits/sec)

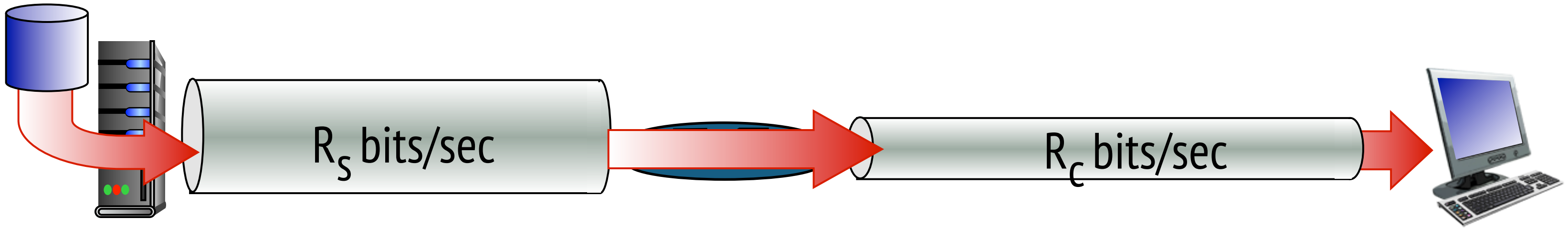
pipe that can carry fluid at rate $R_c$ bits/sec)

## Throughput (more)

- $R_s < R_c$ What is average end-end throughput?



- $R_s > R_c$ What is average end-end throughput?

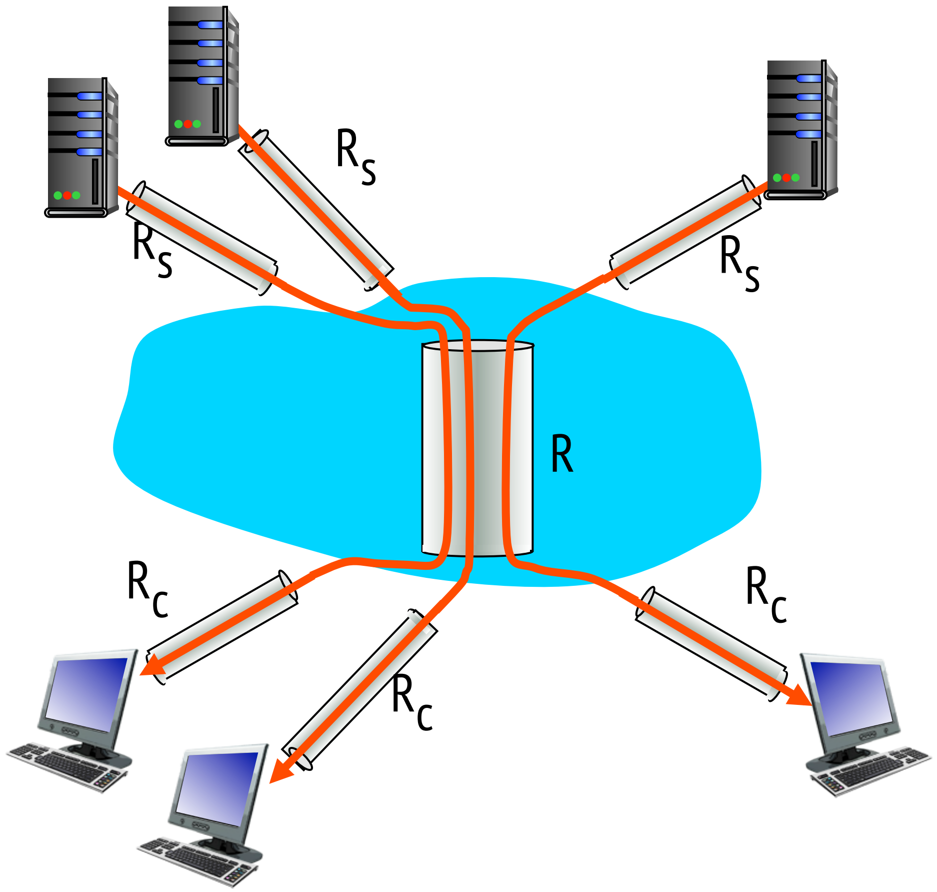


*bottleneck link*

link on end-end path that constrains end-end throughput

## Throughput: Internet scenario

- per-connection end-end throughput: $\min(R_c, R_s, R/10)$
- in practice: $R_c$ or $R_s$ is often bottleneck



10 connections (fairly) share backbone bottleneck link R bits/sec

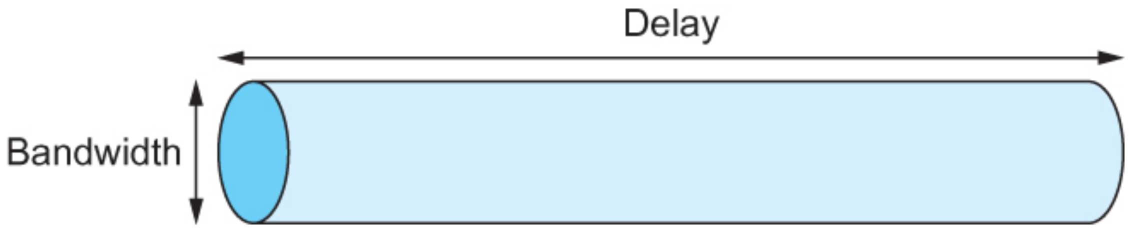
## Performance

- Latency = Propagation + processing + transmit + queue
- Propagation = distance/speed of light
- Transmit = size/bandwidth
- Processing = depends on the node (hardware + software)

- One bit transmission => propagation is important
- Large bytes transmission => bandwidth is important

## Delay X Bandwidth

- We think the channel between a pair of processes as a hollow pipe
- Latency (delay) length of the pipe and bandwidth the width of the pipe
- Delay of 50 ms and bandwidth of 45 Mbps
  - $50 \times 10^{-3}$ seconds x $45 \times 10^{6}$ bits/second
  - $2.25 \times 10^{6}$ bits = 280 KB data.



Network as a pipe

## Delay X Bandwidth

- Relative importance of bandwidth and latency depends on application
  - For large file transfer, bandwidth is critical
  - For small messages (HTTP, NFS, etc.), latency is critical
  - Variance in latency (jitter) can also affect some applications (*e.g.*, audio/video conferencing)

## Delay X Bandwidth

- How many bits the sender must transmit before the first bit arrives at the receiver if the sender keeps the pipe full

- Takes another one-way latency to receive a response from the receiver

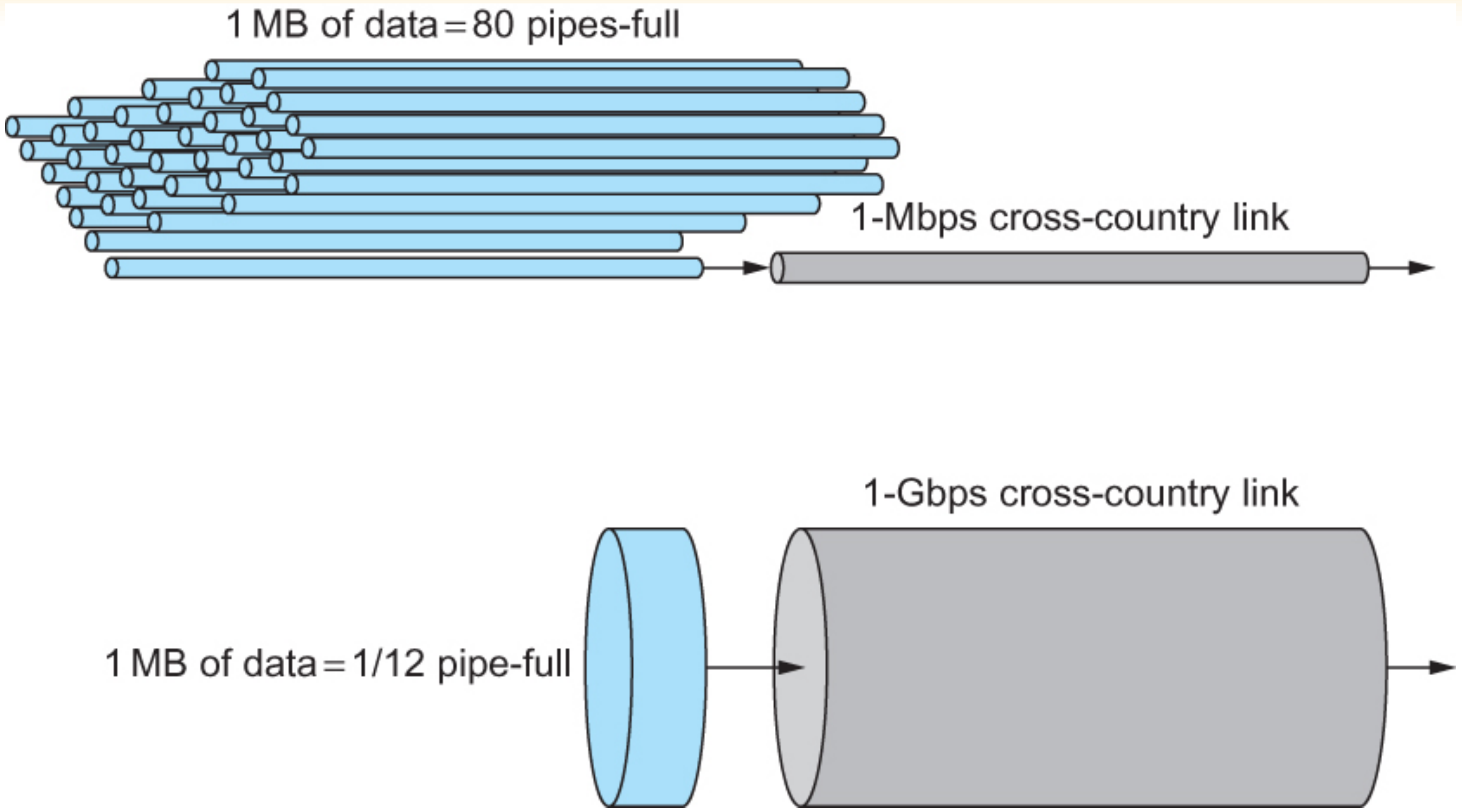
- If the sender does not fill the pipe—send a whole delay × bandwidth product's worth of data before it stops to wait for a signal—the sender will not fully utilize the network

## Delay X Bandwidth

- Infinite bandwidth
  - RTT dominates
  - Throughput = TransferSize / TransferTime
  - TransferTime = RTT + 1/Bandwidth x TransferSize

- Its all relative
  - 1-MB file to 1-Gbps link looks like a 1-KB packet to 1-Mbps link

## Relationship between bandwidth and latency


1 MB of data = 80 pipes-full
1-Mbps cross-country link
1-Gbps cross-country link
1 MB of data = 1/12 pipe-full

A 1-MB file would fill the 1-Mbps link 80 times,

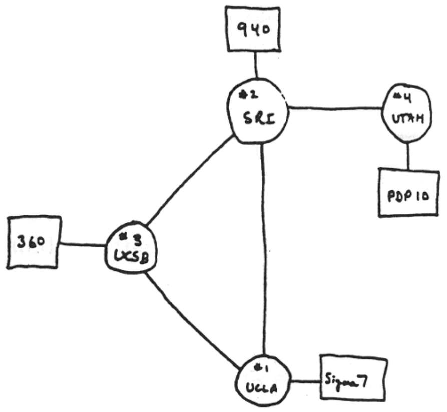
but only fill the 1-Gbps link 1/12 of one time

# Internet History

*1961-1972: Early packet-switching principles*

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational

- 1972:
  - ARPAnet public demo
  - NCP (Network Control Protocol) first host-host protocol
  - first e-mail program
  - ARPAnet has 15 nodes



THE ARPA NETWORK

## Internet History

*1972-1980: Internetworking, new and proprietary nets*

- 1970: ALOHAnet satellite network in Hawaii
- 1974: Cerf and Kahn - architecture for interconnecting networks
- 1976: Ethernet at Xerox PARC
- late70's: proprietary architectures: DECnet, SNA, XNA
- late 70's: switching fixed length packets (ATM precursor)
- 1979: ARPAnet has 200 nodes

Cerf and Kahn's internetworking principles:

- minimalism, autonomy - no internal changes required to interconnect networks
- best effort service model
- stateless routers
- decentralized control

define today's Internet architecture

## Internet History

*1980-1990: new protocols, a proliferation of networks*

- 1983: deployment of TCP/IP
- 1982: smtp e-mail protocol defined
- 1983: DNS defined for name-to-IP-address translation
- 1985: ftp protocol defined
- 1988: TCP congestion control

- new national networks: Csnet, BITnet, NSFnet, Minitel
- 100,000 hosts connected to confederation of networks

# Internet History

*1990, 2000's: commercialization, the Web, new apps*

- early 1990's: ARPAnet decommissioned
- 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- early 1990s: Web
  - hypertext [Bush 1945, Nelson 1960's]
  - HTML, HTTP: Berners-Lee
  - 1994: Mosaic, later Netscape
  - late 1990's: commercialization of the Web

late 1990's – 2000's:

- more killer apps: instant messaging, P2P file sharing
- network security to forefront
- est. 50 million host, 100 million+ users
- backbone links running at Gbps

## Internet History

*2005-present*

- ~750 million hosts
  - Smartphones and tablets
- Aggressive deployment of broadband access
- Increasing ubiquity of high-speed wireless access
- Emergence of online social networks:
  - Facebook: 1.71 billion active users
- Service providers (Google, Microsoft) create their own networks
  - Bypass  Internet, providing "instantaneous" access to search, email, etc.
- E-commerce, universities, enterprises running their services in "cloud" (eg, Amazon EC2)

## Summary

- We have identified what we expect from a computer network
- We have defined a layered architecture for computer network that will serve as a blueprint for our design
- We have discussed the socket interface which will be used by applications for invoking the services of the network subsystem
- We have discussed two performance metrics using which we can analyze the performance of computer networks

# Chapter 1, Problem 4

- Calculate the total time required to transfer a 1.5MB file in following cases with assumptions a) RTT is 80ms, b) packet size is 1KB, c) initial 2xRTT is required for handshaking before transmission:
  - Bandwidth is 10Mbps and data packets can be sent continuously

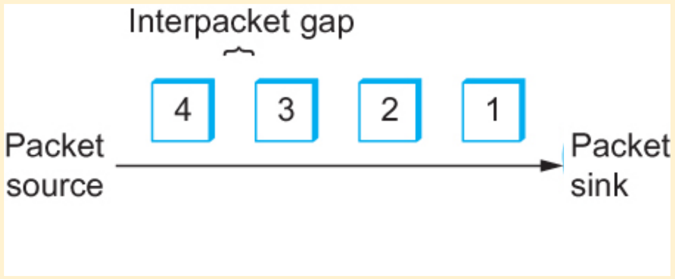
- Convert everything in same unit
  - 1.5MB = 1.5x1024KB = 1.5x1024x1024x8 bits = 12582912bits
  - 10Mbps = 10000000bps
- Total Time = initial 2RTT + Transmit Delay + Propagation Delay (RTT/2)
  - 80x2/1000 + 12582912/10000000 + 40/1000 secs
  - 1.458 secs

UNIVERSITY AT ALBANY
State University of New York

## Chapter 1, Problem 4

- Calculate the total time required to transfer a 1.5MB file in following cases with assumptions a) RTT is 80ms, b) packet size is 1KB, c) initial 2xRTT is required for handshaking before transmission:
  - Bandwidth is 10Mbps, but wait for one RTT between packets

- Number of packets
  - 1.5MB / 1KB = 1.5x1024 = 1536
- 1535 interpacket gaps between 1536 packets
- Add 1535xRTT to previous solution
  - 1535x80/1000secs + 1.458secs
  - 124.258 secs



## Chapter 1, Problem 4

- Calculate the total time required to transfer a 1.5MB file in following cases with assumptions a) RTT is 80ms, b) packet size is 1KB, c) initial 2xRTT is required for handshaking before transmission:
  - Link allows infinitely fast transmit, but limits bandwidth – 20 packets can be sent in one RTT

- Transmit Time = 0
- 1536 packets / 20 = 76.8 batches (76 full batch and 1 partial batch)
- Propagation delay for the first batch = RTT/2
- Initial Setup = RTTx2
- 76 batches in 76xRTT time
- Total = 80x2/1000 + 40/1000 + 76x80/1000 secs = 6.28 secs

## Chapter 1, Problem 4

- Calculate the total time required to transfer a 1.5MB file in following cases with assumptions a) RTT is 80ms, b) packet size is 1KB, c) initial 2xRTT is required for handshaking before transmission:
  - Zero transmit time, but limits 1 packet in first RTT, 2 in second, $2^{3-1}$ in third RTT and so on.

- $1 + 2 + 2^{3-1} + \ldots\ldots + 2^n = 2^{n+1} - 1$.
  - At n=9, total packets = 1023
  - At n=10, total packets = 2047
  - Thus, we can send all in 11 batches
- Propagation delay for the first batch = RTT/2, Initial Setup = RTTx2
- 10 batches in 10xRTT time
- Total = 80x2/1000 + 40/1000 + 10x80/1000 secs = 1sec