Introduction: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

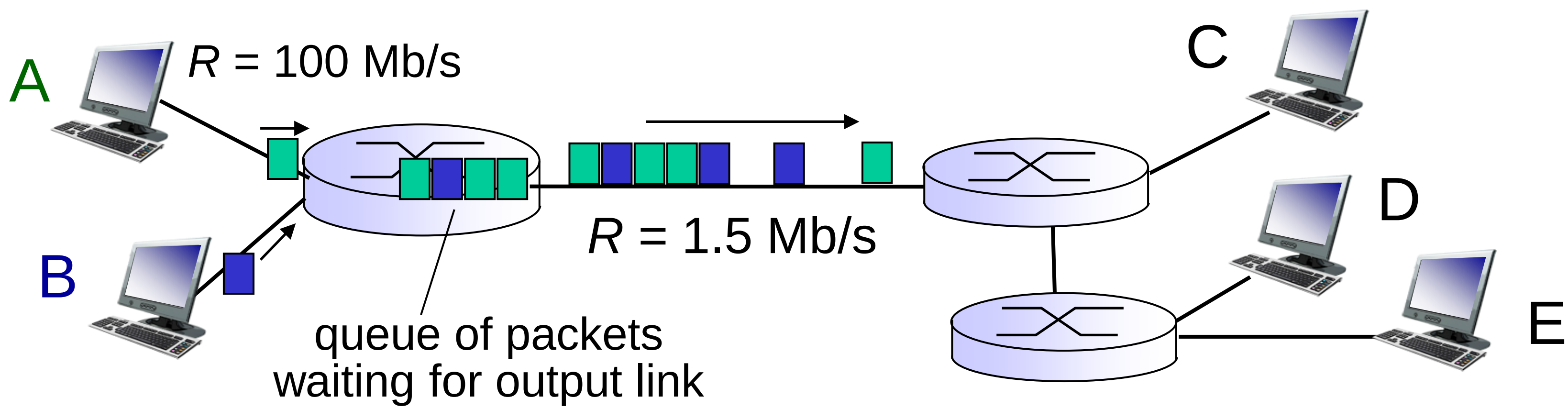
- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service model

Sources of packet delay and loss?

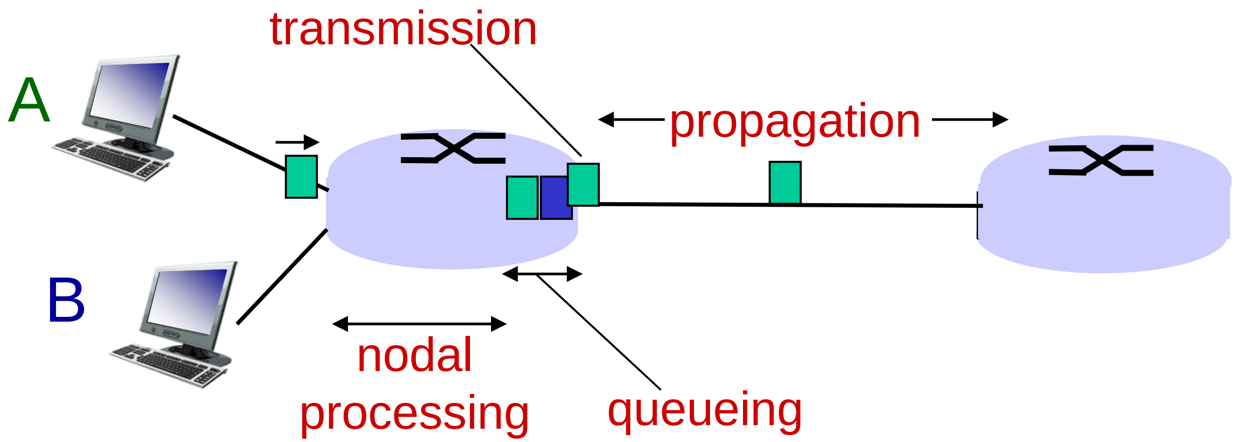
Packet Switching: queueing delay, loss



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
- Do you want to keep this buffer full, empty, or what?

Four sources of packet delay



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

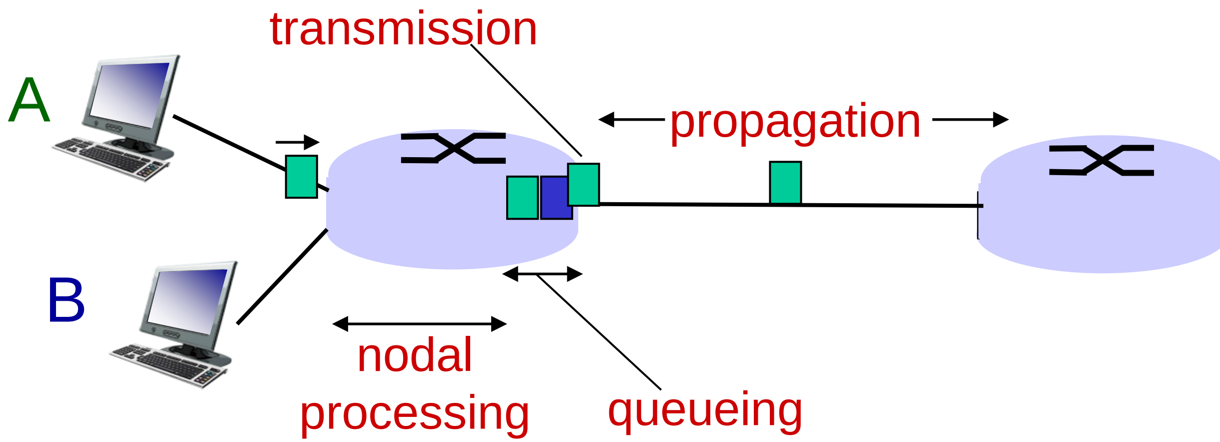
d_{proc}: nodal processing

- check bit errors
- determine output link
- typically < msec
- **Negligible?**

d_{queue}: queueing delay

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

Four sources of packet delay



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

d_{trans}: transmission delay:

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

d_{trans} and d_{prop}
very different

d_{prop}: propagation delay:

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

muhammadshahzad — -zsh — 80×24

```
64 bytes from 92.223.20.41: icmp_seq=15 ttl=46 time=162.168 ms
64 bytes from 92.223.20.41: icmp_seq=16 ttl=46 time=198.682 ms
64 bytes from 92.223.20.41: icmp_seq=17 ttl=46 time=155.489 ms
64 bytes from 92.223.20.41: icmp_seq=18 ttl=46 time=162.114 ms
64 bytes from 92.223.20.41: icmp_seq=19 ttl=46 time=169.983 ms
64 bytes from 92.223.20.41: icmp_seq=20 ttl=46 time=166.867 ms
64 bytes from 92.223.20.41: icmp_seq=21 ttl=46 time=168.033 ms
64 bytes from 92.223.20.41: icmp_seq=22 ttl=46 time=189.507 ms
64 bytes from 92.223.20.41: icmp_seq=23 ttl=46 time=184.747 ms
64 bytes from 92.223.20.41: icmp_seq=24 ttl=46 time=169.925 ms
64 bytes from 92.223.20.41: icmp_seq=25 ttl=46 time=174.059 ms
64 bytes from 92.223.20.41: icmp_seq=26 ttl=46 time=158.222 ms
64 bytes from 92.223.20.41: icmp_seq=27 ttl=46 time=166.997 ms
64 bytes from 92.223.20.41: icmp_seq=28 ttl=46 time=162.546 ms
64 bytes from 92.223.20.41: icmp_seq=29 ttl=46 time=169.475 ms
64 bytes from 92.223.20.41: icmp_seq=30 ttl=46 time=161.843 ms
64 bytes from 92.223.20.41: icmp_seq=31 ttl=46 time=174.667 ms
64 bytes from 92.223.20.41: icmp_seq=32 ttl=46 time=183.142 ms
64 bytes from 92.223.20.41: icmp_seq=33 ttl=46 time=194.294 ms
64 bytes from 92.223.20.41: icmp_seq=34 ttl=46 time=164.461 ms
64 bytes from 92.223.20.41: icmp_seq=35 ttl=46 time=155.234 ms
64 bytes from 92.223.20.41: icmp_seq=36 ttl=46 time=165.632 ms
64 bytes from 92.223.20.41: icmp_seq=37 ttl=46 time=171.418 ms
^C
```

A game server in Europe

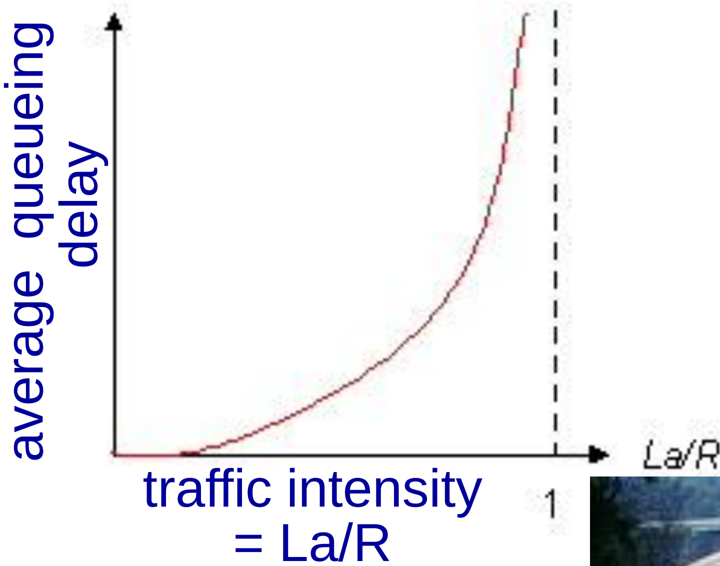
muhammadshahzad — -zsh — 74×24

```
Request timeout for icmp_seq 2
64 bytes from 209.142.68.29: icmp_seq=3 ttl=118 time=81.927 ms
64 bytes from 209.142.68.29: icmp_seq=4 ttl=118 time=41.956 ms
64 bytes from 209.142.68.29: icmp_seq=5 ttl=118 time=60.169 ms
64 bytes from 209.142.68.29: icmp_seq=6 ttl=118 time=49.446 ms
64 bytes from 209.142.68.29: icmp_seq=7 ttl=118 time=66.815 ms
64 bytes from 209.142.68.29: icmp_seq=8 ttl=118 time=54.070 ms
64 bytes from 209.142.68.29: icmp_seq=9 ttl=118 time=66.603 ms
64 bytes from 209.142.68.29: icmp_seq=10 ttl=118 time=55.843 ms
64 bytes from 209.142.68.29: icmp_seq=11 ttl=118 time=54.960 ms
64 bytes from 209.142.68.29: icmp_seq=12 ttl=118 time=62.267 ms
64 bytes from 209.142.68.29: icmp_seq=13 ttl=118 time=138.473 ms
64 bytes from 209.142.68.29: icmp_seq=14 ttl=118 time=53.609 ms
64 bytes from 209.142.68.29: icmp_seq=15 ttl=118 time=75.197 ms
64 bytes from 209.142.68.29: icmp_seq=16 ttl=118 time=59.527 ms
64 bytes from 209.142.68.29: icmp_seq=17 ttl=118 time=71.403 ms
64 bytes from 209.142.68.29: icmp_seq=18 ttl=118 time=54.427 ms
64 bytes from 209.142.68.29: icmp_seq=19 ttl=118 time=51.863 ms
64 bytes from 209.142.68.29: icmp_seq=20 ttl=118 time=71.081 ms
64 bytes from 209.142.68.29: icmp_seq=21 ttl=118 time=47.509 ms
64 bytes from 209.142.68.29: icmp_seq=22 ttl=118 time=37.661 ms
64 bytes from 209.142.68.29: icmp_seq=23 ttl=118 time=42.590 ms
64 bytes from 209.142.68.29: icmp_seq=24 ttl=118 time=42.442 ms
^C
```

A server in Chicago

Queueing delay (revisited)

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



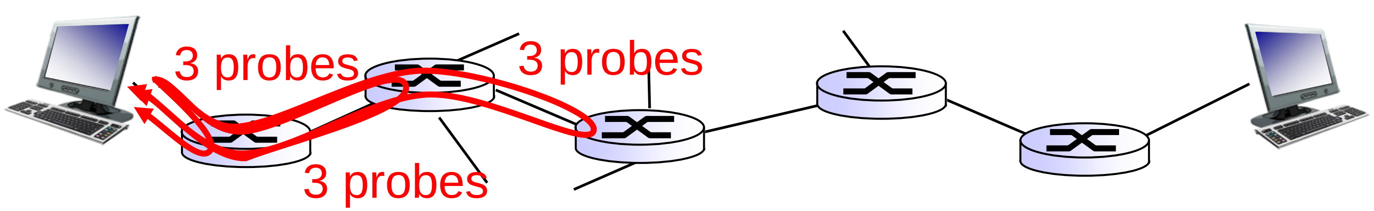
La/R ~ 0

- *La/R* ~ 0: avg. queueing delay = ?
- *La/R* > 1: avg. queueing delay = ?
- *La/R* ≈ 1: avg. queueing delay = ? (nature of traffic)

La/R > 1

"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

traceroute: gaia.cs.umass.edu to www.eurecom.fr

3 delay measurements from
gaia.cs.umass.edu to cs-gw.cs.umass.edu

```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 ms 18 ms
6  abilene-vbns.abilene.ucaid.edu (198.32.11.9)  22 ms  18 ms  22 ms
7  nycm-wash.abilene.ucaid.edu (198.32.8.46)  22 ms  22 ms  22 ms
8  62.40.103.253 (62.40.103.253)  104 ms 109 ms 106 ms
9  de2-1.de1.de.geant.net (62.40.96.129)  109 ms 102 ms 104 ms
10  de.fr1.fr.geant.net (62.40.96.50)  113 ms 121 ms 114 ms
11  renater-gw.fr1.fr.geant.net (62.40.103.54)  112 ms  114 ms  112 ms
12  nio-n2.cssi.renater.fr (193.51.206.13)  111 ms  114 ms  116 ms
13  nice.cssi.renater.fr (195.220.98.102)  123 ms  125 ms  124 ms
14  r3t2-nice.cssi.renater.fr (195.220.98.110)  126 ms  126 ms  124 ms
15  eurecom-valbonne.r3t2.ft.net (193.48.50.54)  135 ms  128 ms  133 ms
16  194.214.211.25 (194.214.211.25)  126 ms  128 ms  126 ms
17  * * *
18  * * *
19  fantasia.eurecom.fr (193.55.113.142)  132 ms  128 ms  136 ms
```

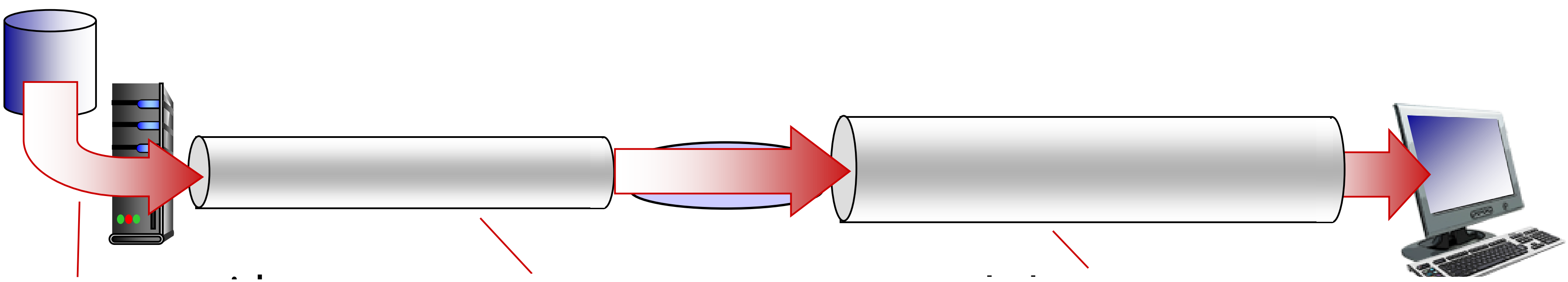
trans-oceanic link

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

* Do some traceroutes from exotic countries at www.traceroute.org

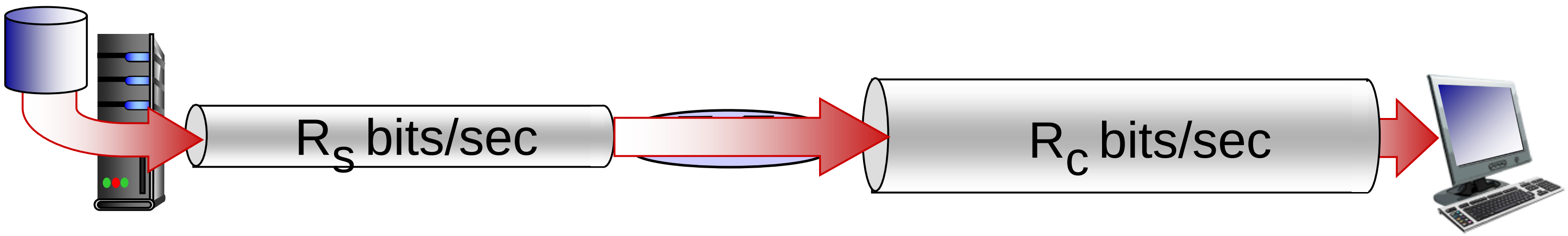
Throughput

- *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

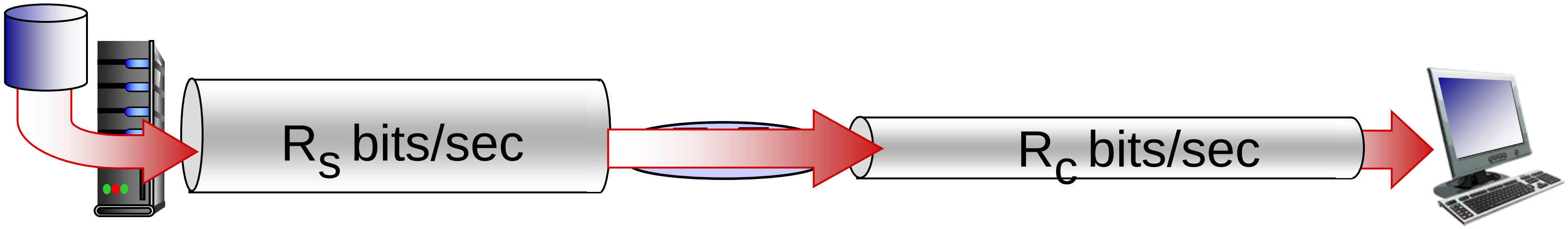


Throughput (more)

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



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

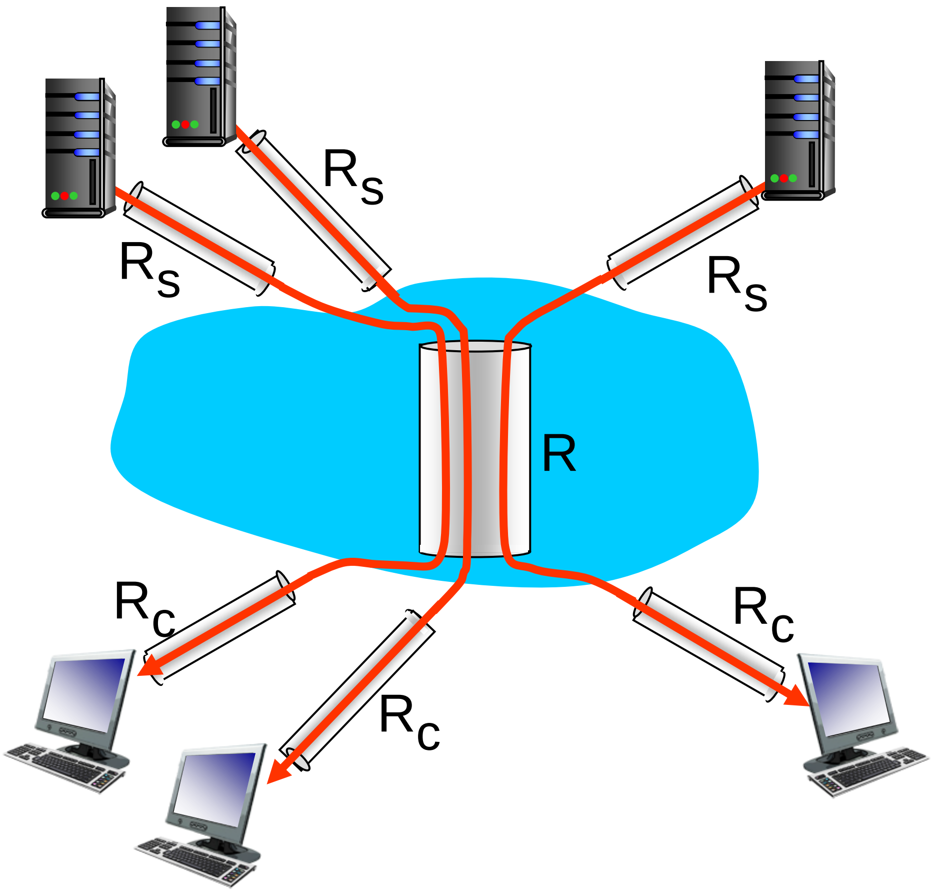


bottleneck link

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

Introduction: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

Protocol "layers"

Networks are complex, with many "pieces":

- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

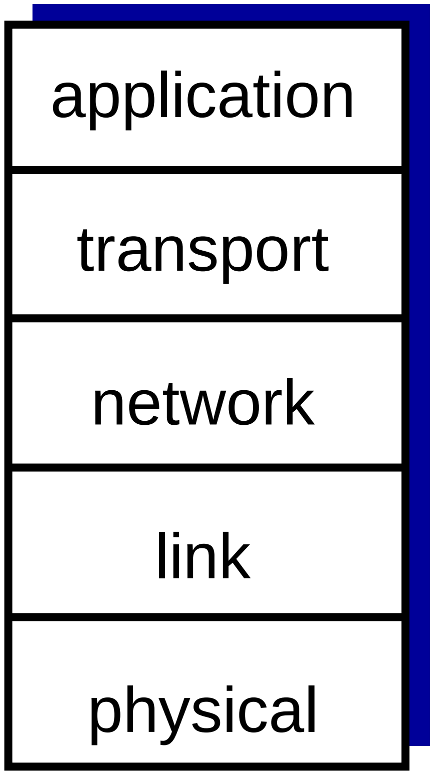
Question:

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

.... or at least our discussion of networks?

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"
- Which layers of protocol stack should be implemented by who?
- Which layers should be in HW and which in SW?

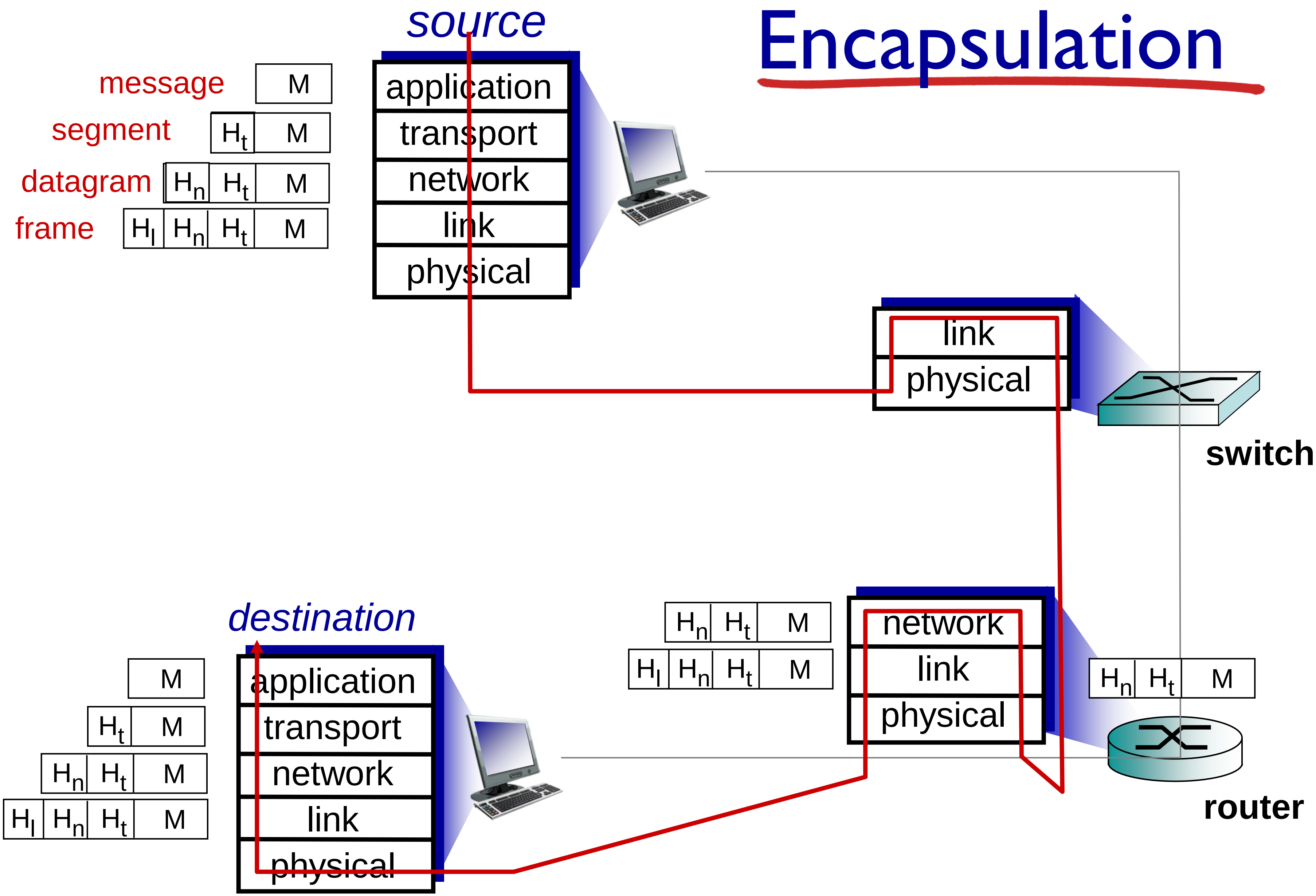


Why layering?

dealing with complex systems:

- explicit structure allows identification, relationship of complex system's pieces
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
- layering considered harmful?
  - Replication of functionality

Encapsulation



App-Layer: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 electronic mail

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P applications

2.7 socket programming with UDP and TCP

Some network apps?

- e-mail
- web
- text messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)
- voice over IP (e.g., Skype)
- real-time video conferencing
- social networking
- search
- …
- …

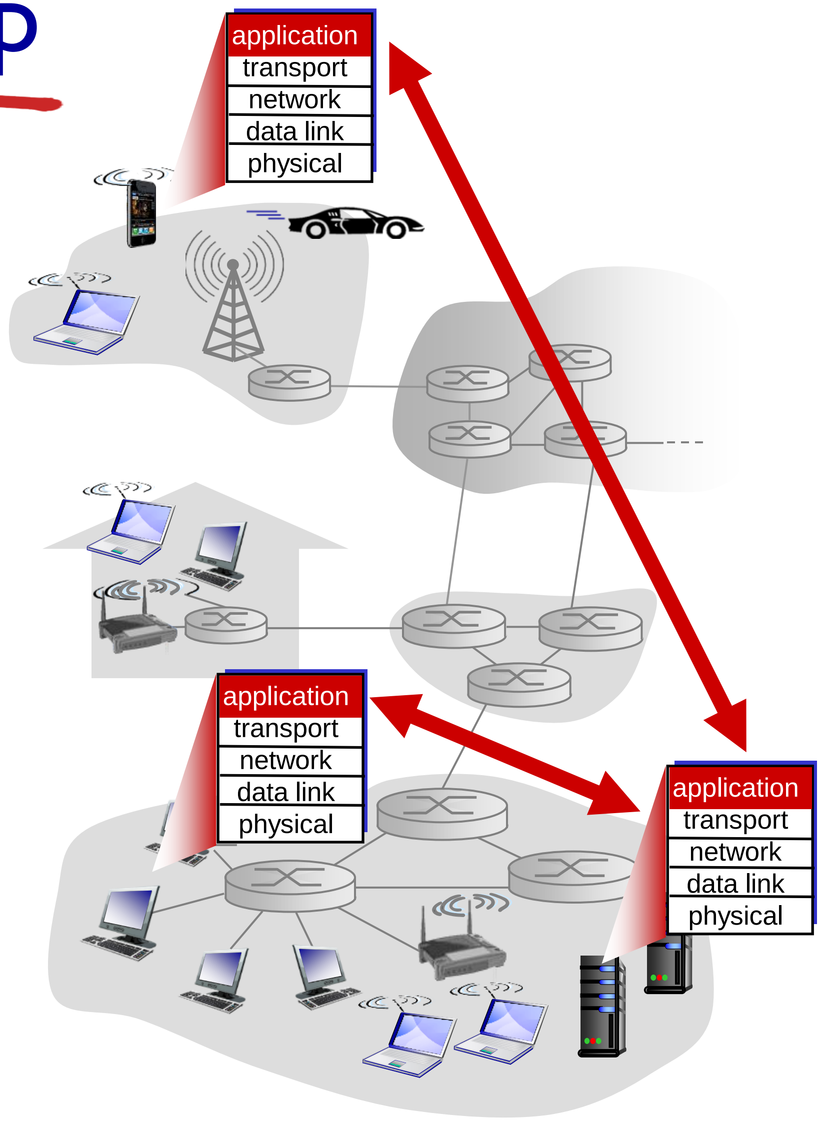
Creating a network app

write programs that:

- run on (different) *end systems*
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation

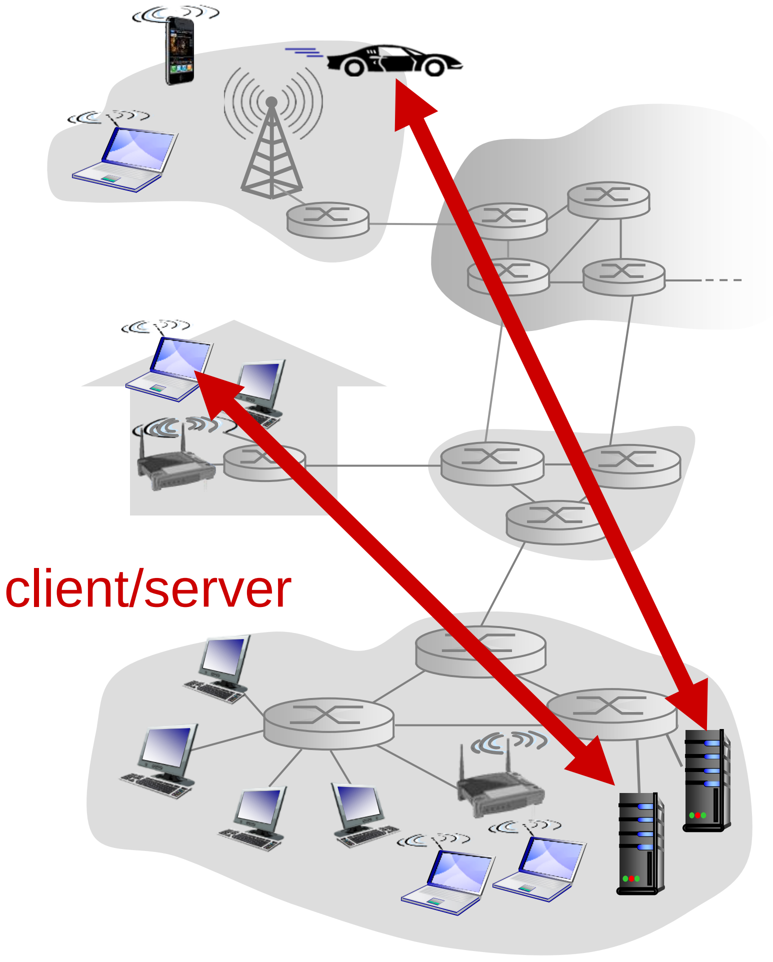


Application architectures

possible structure of applications:

- client-server
- peer-to-peer (P2P)

Client-server architecture

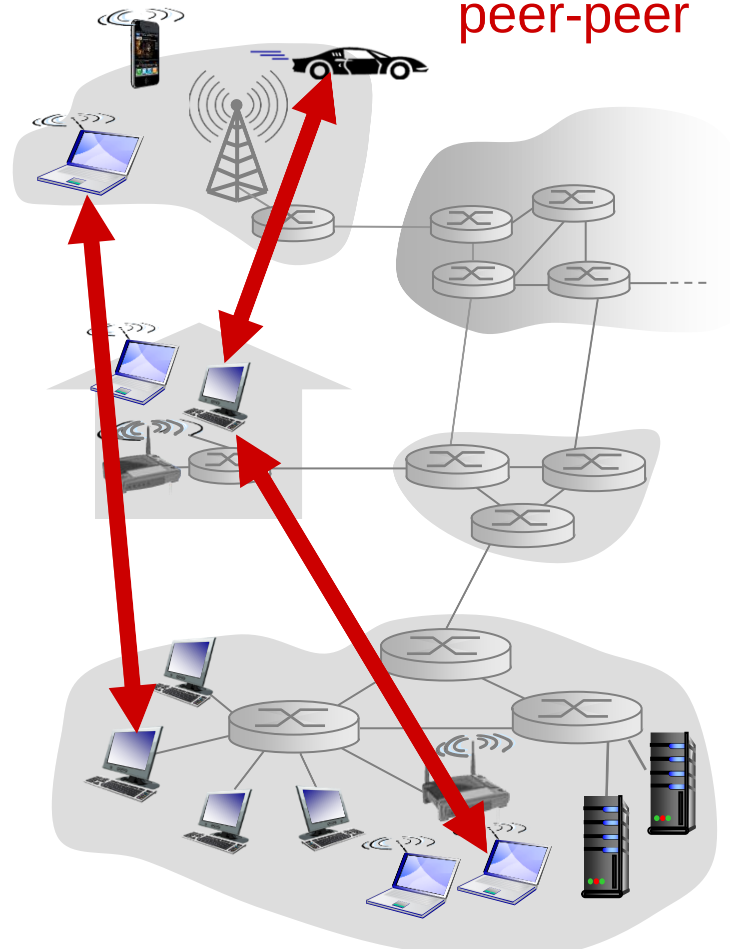


server:
- always-on host
- permanent IP address
- data centers for scaling

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

P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
  - complex management



Processes communicating

process: program running within a host

- within same host, two processes communicate using inter-process communication (defined by OS)
- processes in different hosts communicate by exchanging messages over a network

clients, servers

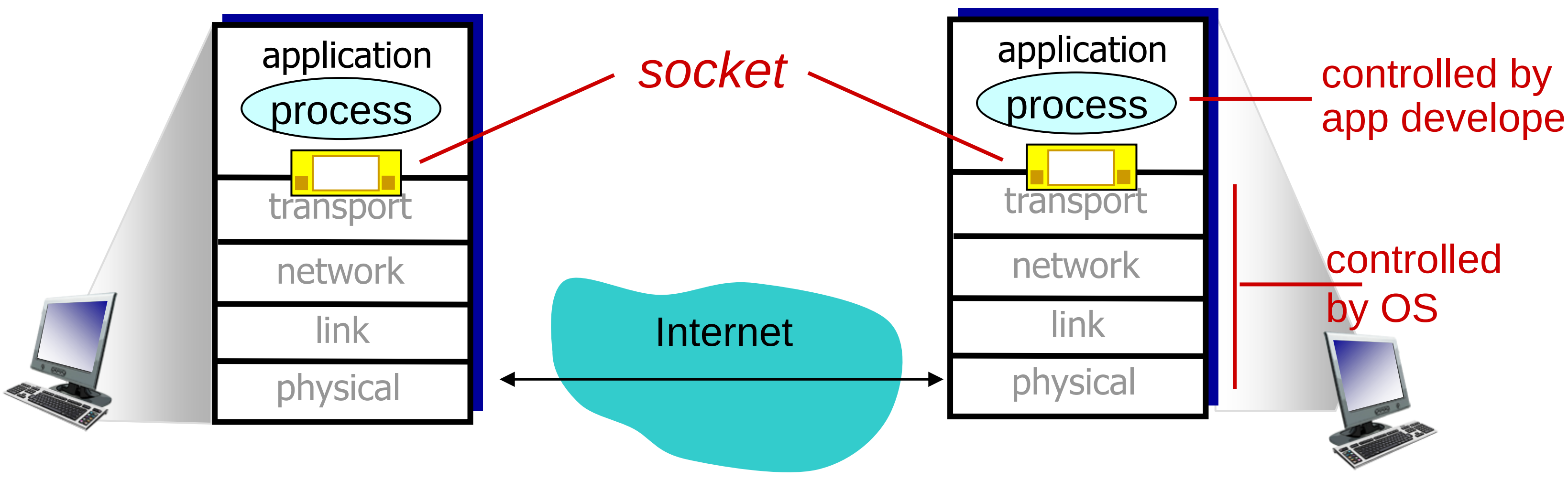
client process: process that initiates communication

server process: process that waits to be contacted

- Applications with P2P architectures have client processes & server processes?

Sockets

- process sends/receives messages to/from its socket
- socket analogous to door
  - 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



Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host on which process runs suffice for identifying the process?
  - *A:* no, *many* processes can be running on same host

- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to msu.edu web server:
  - IP address: 35.9.247.69
  - port number: 80

App-layer protocol defines

- Types of messages exchanged,
  - e.g., request, response
- message syntax:
  - what fields in messages & how fields are delineated
- message semantics
  - meaning of information in fields
- rules for when and how processes send & respond to messages

open protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols:

- e.g., Skype

- Difference between Network application and application layer protocol?

What transport service does an app need?

data integrity

- some apps require 100% reliable data transfer; other apps can tolerate some loss
- Examples?
  - File transfer, web transactions; audio

timing

- some apps require low delay to be "effective"; others don't
- Examples?
  - Internet telephony, interactive games

throughput

- some apps require minimum amount of throughput to be "effective"; others don't
- Examples?
  - Multimedia

security

- encryption, data integrity, …

Transport service requirements: common apps

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100’s msec
stored audio/video	loss-tolerant	same as above	yes
interactive games	loss-tolerant	few kbps up	yes, 100’s msec
text messaging	no loss	elastic	

Internet transport protocols services

TCP service:

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

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, orconnection setup,

Q: why bother? Why is there a UDP?

Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Securing TCP

TCP & UDP

- no encryption
- cleartext passwds sent into socket traverse Internet in cleartext

SSL

- provides encrypted TCP connection
- data integrity
- end-point authentication

SSL is at app layer

- Apps use SSL libraries, which "talk" to TCP

SSL socket API

- cleartext passwds sent into socket traverse Internet encrypted

Q: Why not run SSL over UDP?