Chapter 1: 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

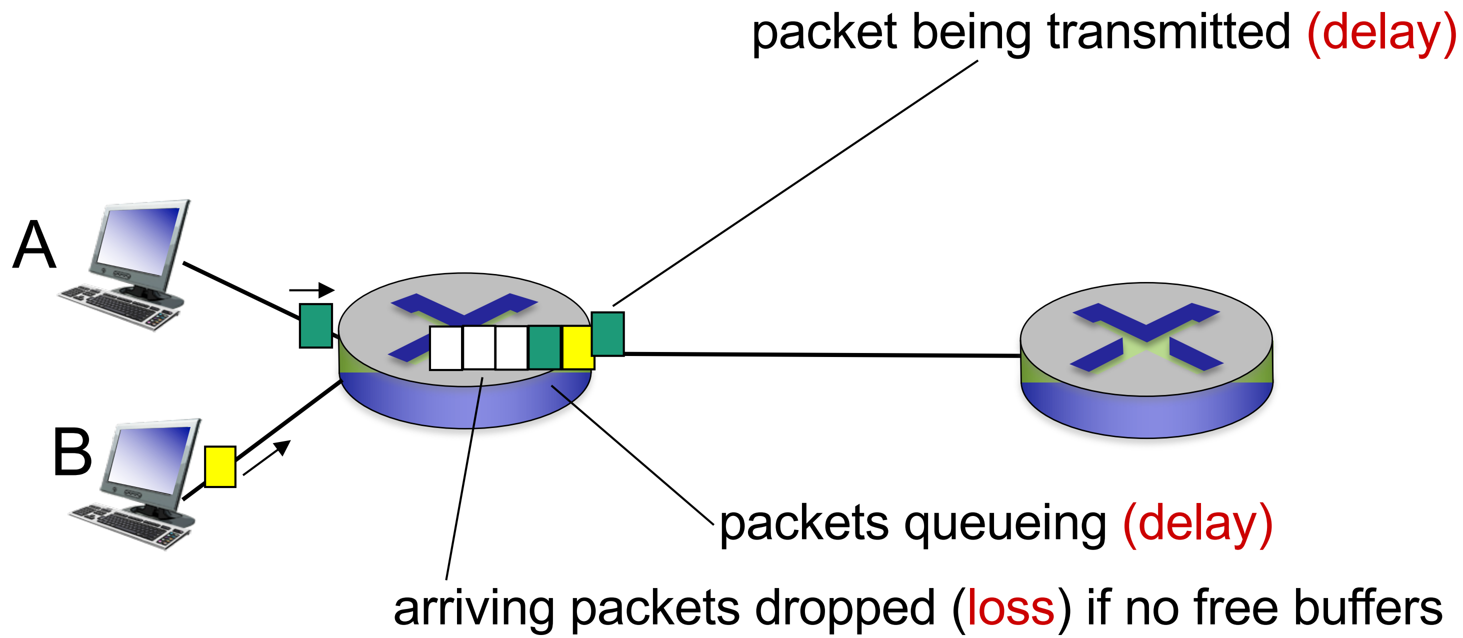
1.6 networks under attack: security

1.7 history

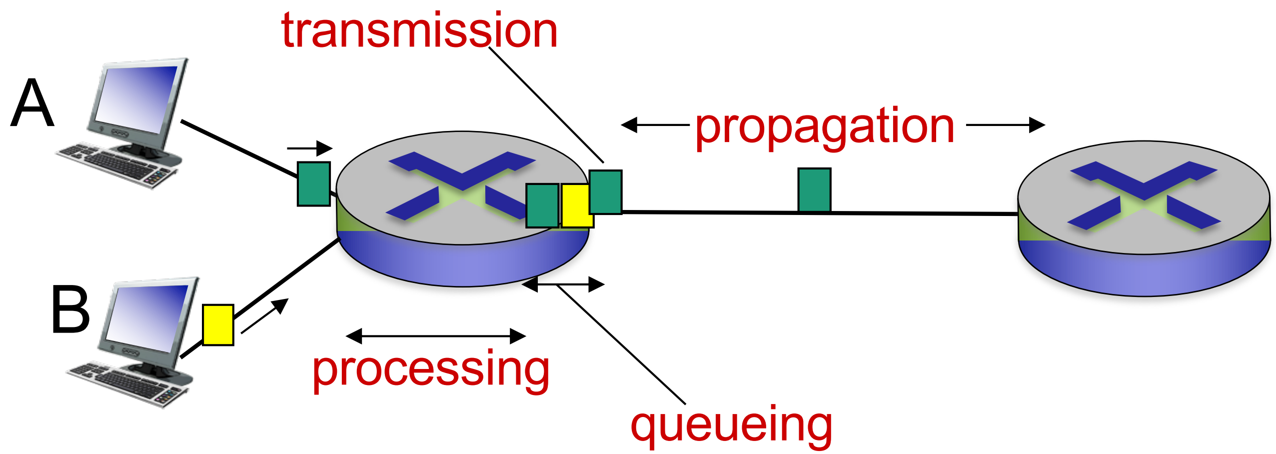
How do loss and delay occur?

packets *queue* in router buffers

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



Four sources of packet delay



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

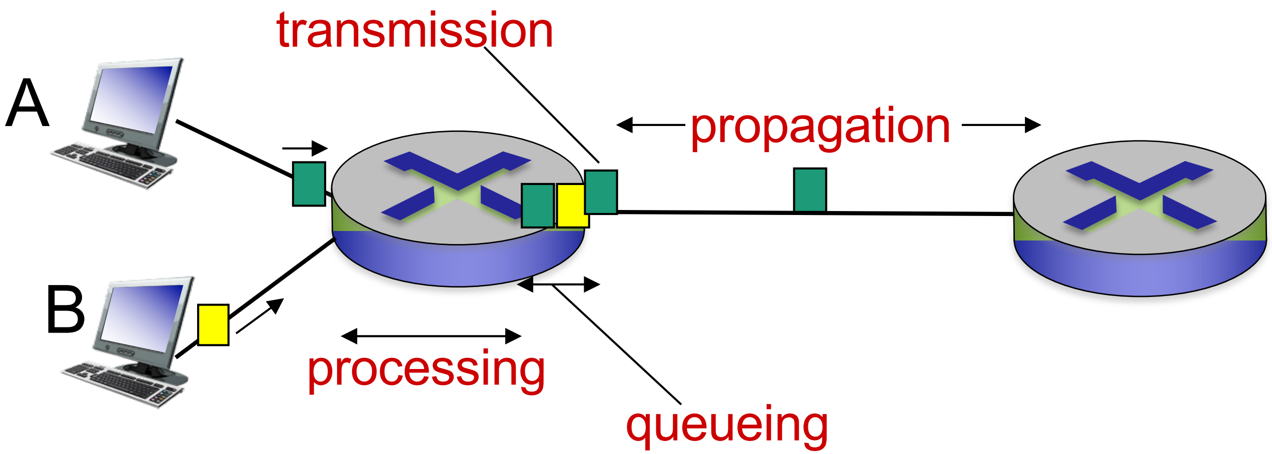
d_{proc}: nodal processing

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

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)*
- $d_{trans} = L/R$

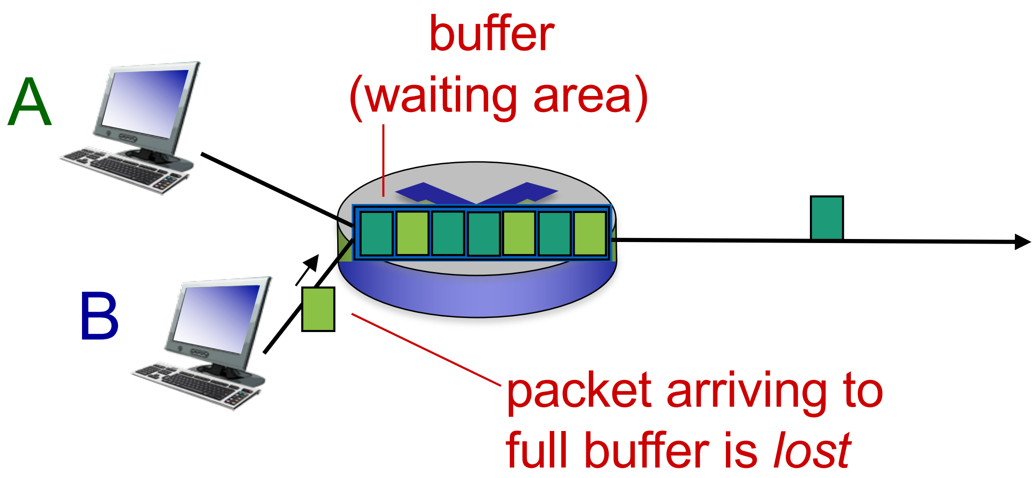
d_{trans} and d_{prop} *very* different

d_{prop}: propagation delay:

- *d*: length of physical link
- *s*: propagation speed (~2x10^8 m/sec)
- $d_{prop} = d/s$

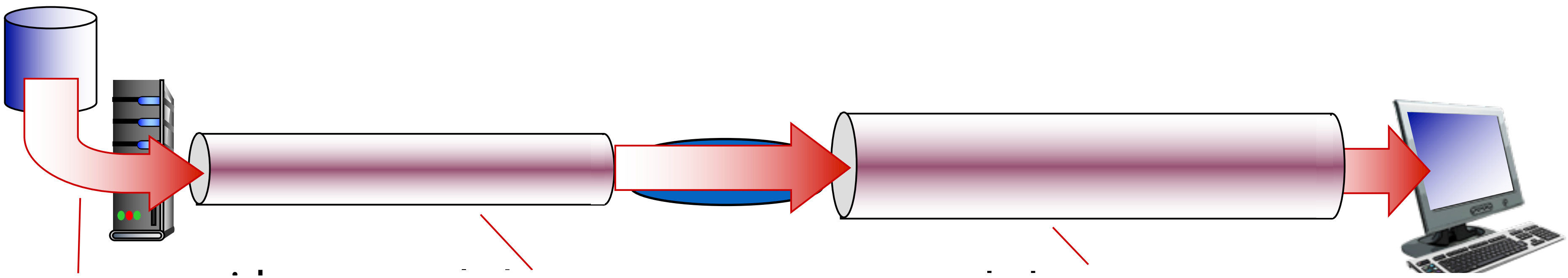
Packet loss

- queue (aka buffer) in buffer has limited capacity
- packet dropped if arriving to full queue (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
  - *Real-time:* rate at a given time point
  - *Average:* rate over longer period of time



server sends bits (fluid) into pipe

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

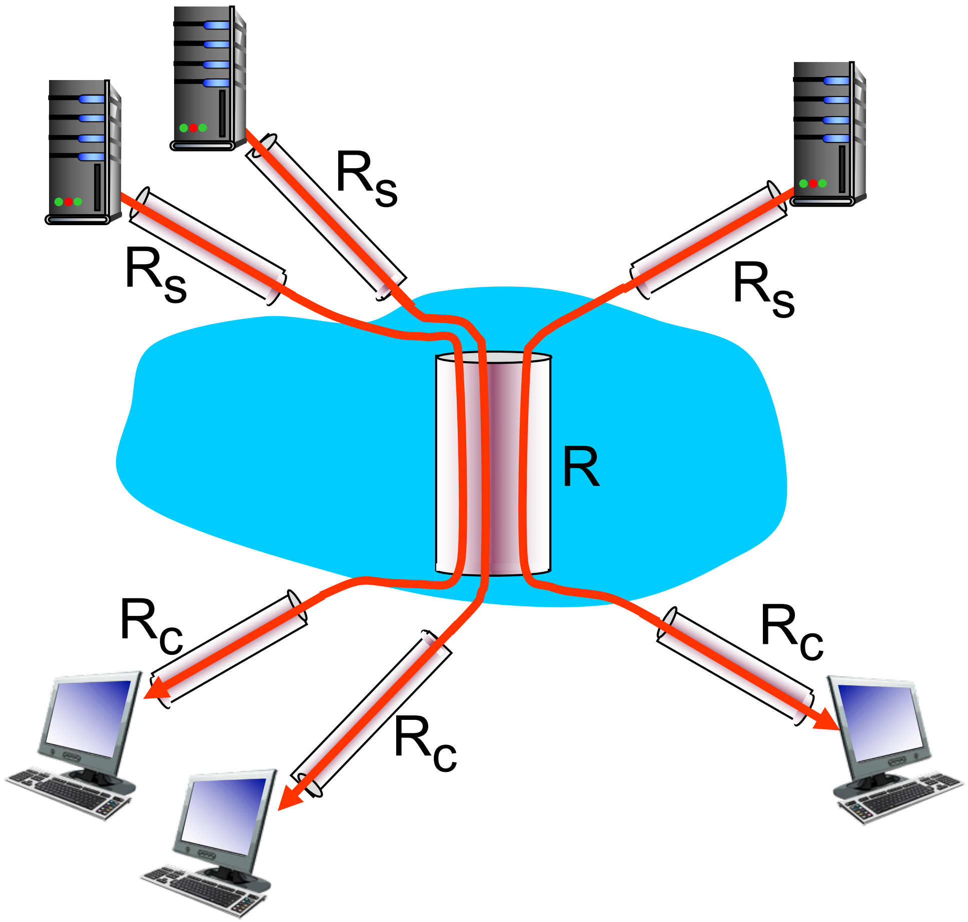
pipe that can carry fluid at rate R_c bits/sec)

Throughput: Internet scenario

- per-connection end-end throughput:

  min(R_c, R_s, $R/10$)

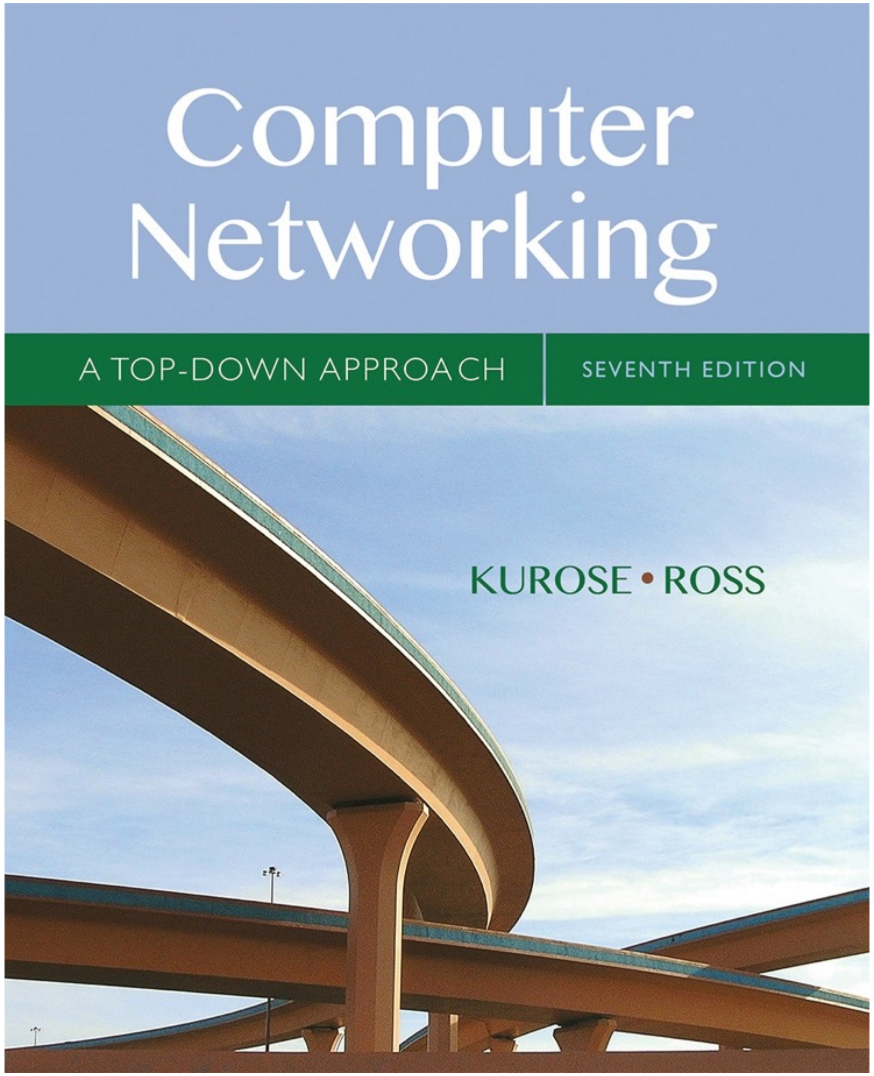
- in practice: R_c or R_s is often bottleneck



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

RUTGERS

CS 352: Internet Technology (Section 3, 4, 5)



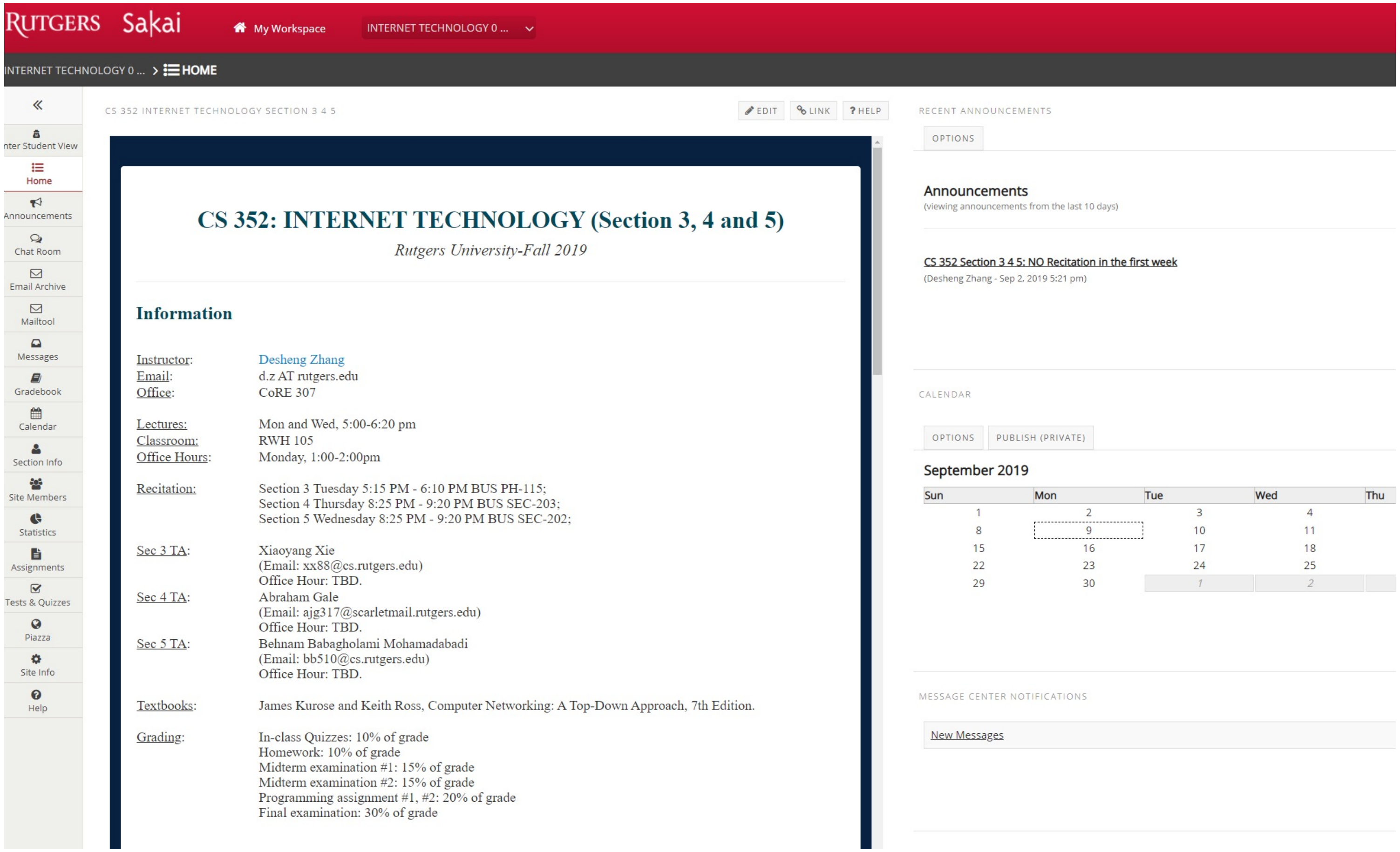
Computer Science
Rutgers University

Fall 2019

SPN Requests

- **For all students who need a SPN**
  - **We are currently overwhelmed with SPN requests**
  - **We are constrained by classroom capacity, TA resources, class size, etc**
  - **We already issued a few SPNs to some students based on the department policy**
  - **We may be able to issue a few more if some students drop this course, but the chance is low.**
  - **For these who did not get SPNs so far, please try to register next semester**
- **All SPN questions should be sent to Prof. Zhang at d.z@rutgers.edu**

Class Website



- on Sakai
- also https://www.cs.rutgers.edu/~dz220/CS352F2019.html

Chapter 1: 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

1.6 networks under attack: security

1.7 history

Protocol "layers"

Networks are complex, with many "pieces":

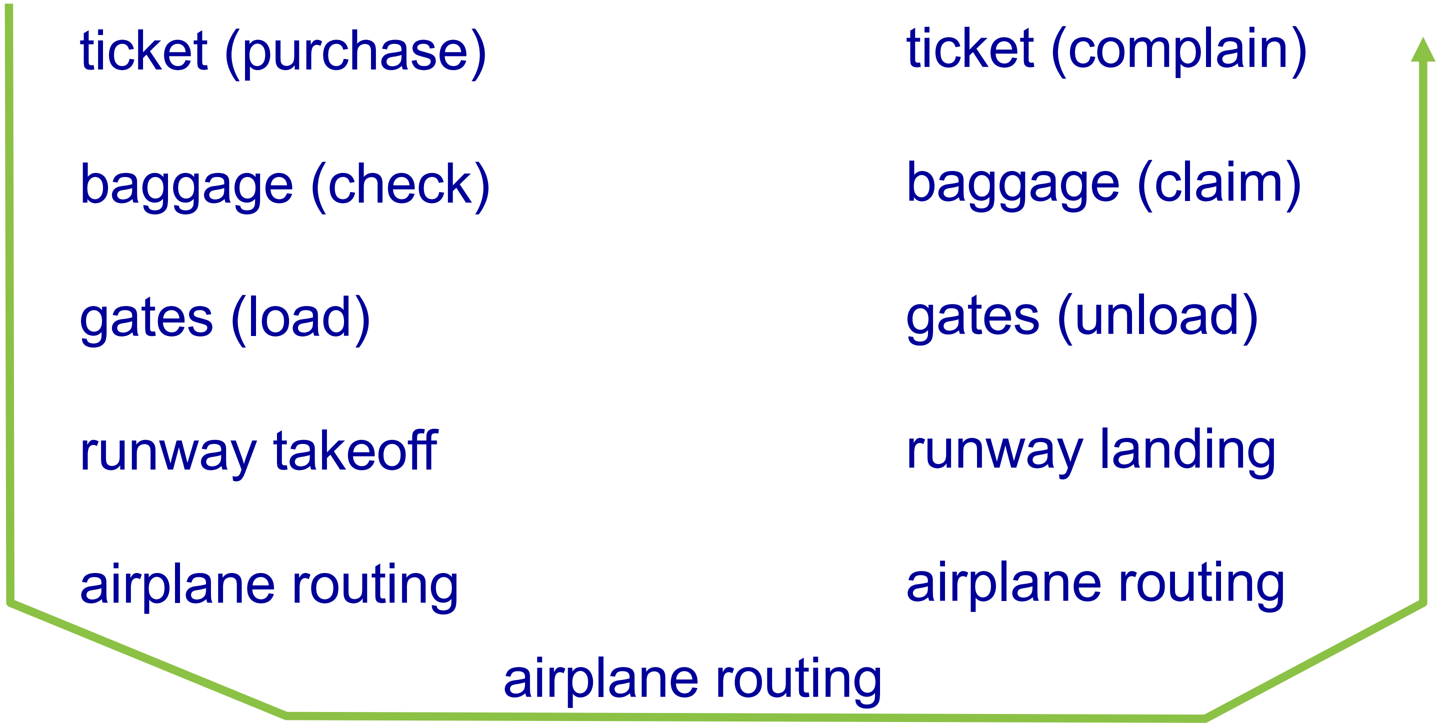
- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:

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

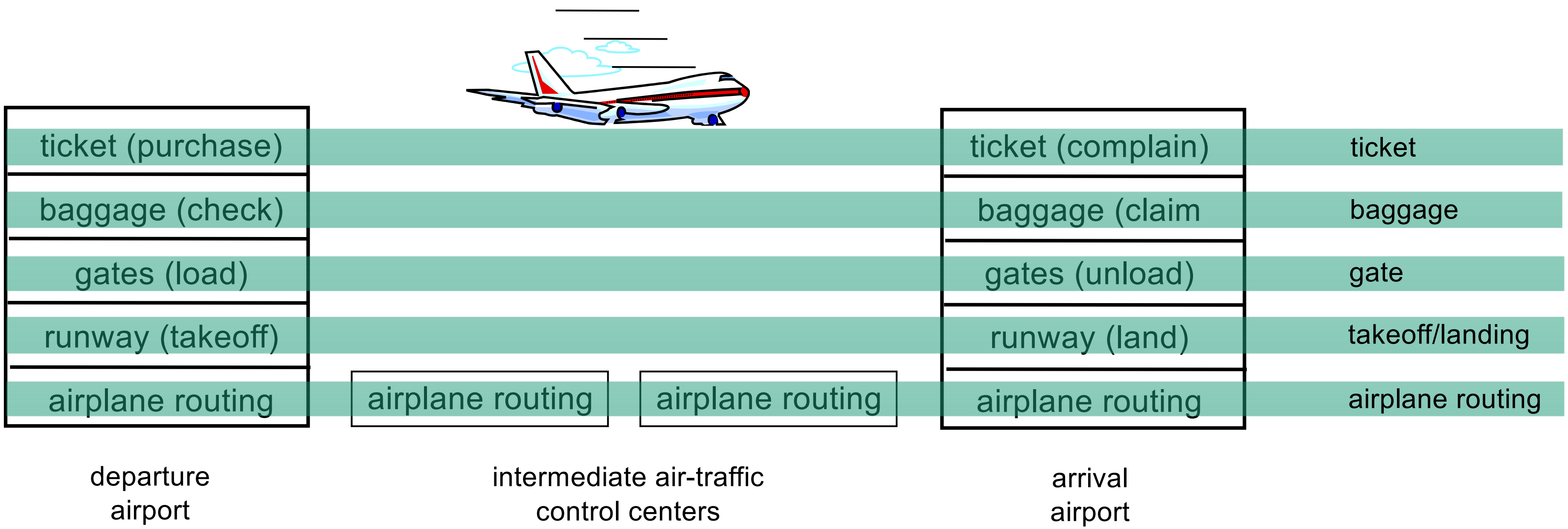
…. or at least our discussion of networks?

Organization of air travel



- a series of steps

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:

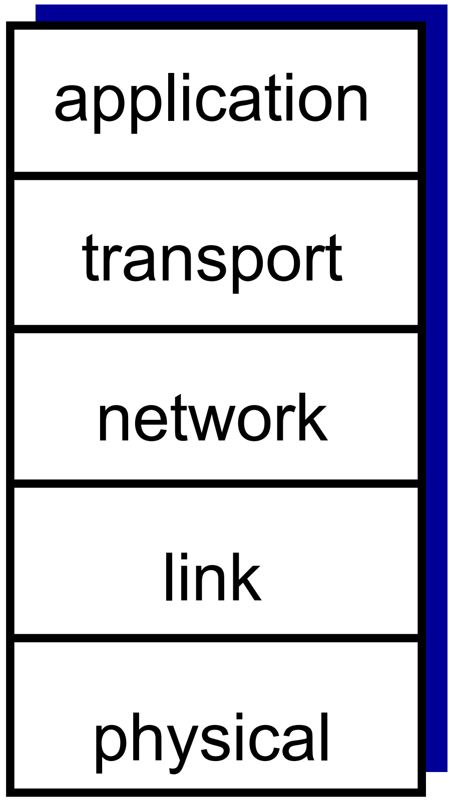
- Layering structure allows understanding of relationship of complex system's pieces

- Layering eases maintaining and 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?

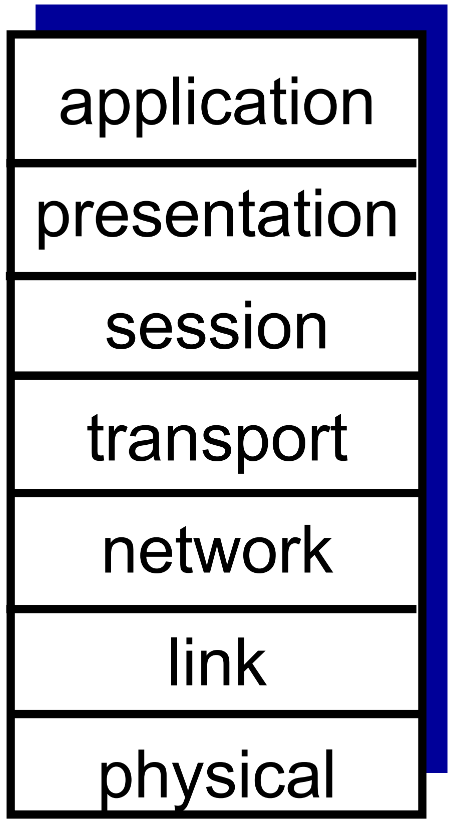
Internet protocol stack

- *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 devices
  - Ethernet, 802.11 (WiFi)
- *physical:* bits "on the wire"



ISO/OSI reference model

- *Open Systems Interconnections*
- *presentation:* allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application



Chapter 1: 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

1.6 networks under attack: security

1.7 history

Network security

- field of network security:
  - how bad guys can **attack** computer networks
  - how we can **defend** networks against attacks
  - how to **design architectures** that are immune to attacks

- Internet not originally designed with (much) security in mind
  - *original vision:* "a group of **mutually trusting** users attached to a transparent network" ☺
  - security considerations in all layers!

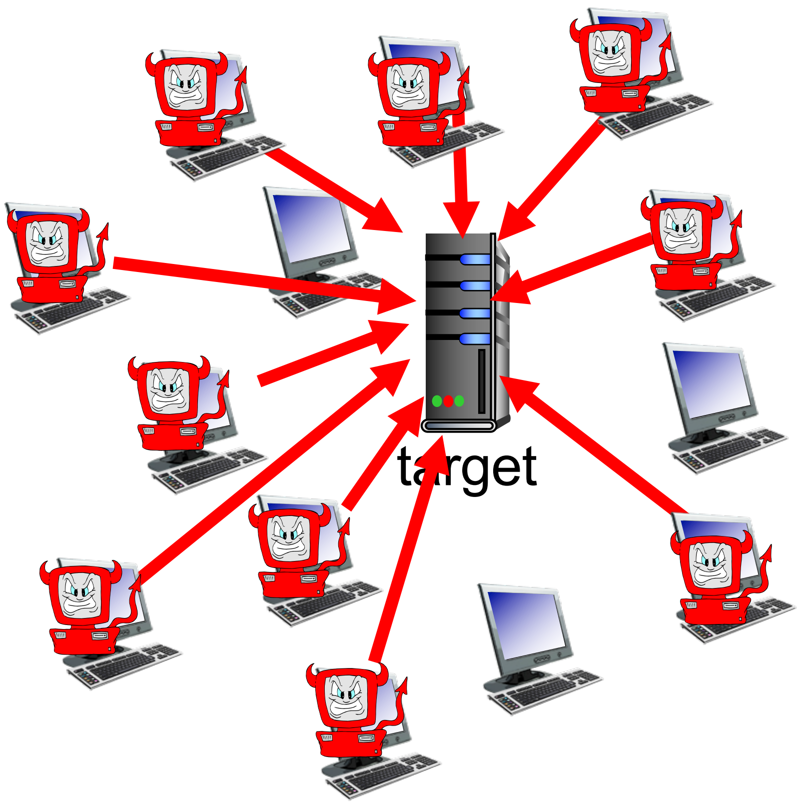
Bad guys: put malware into hosts via Internet

- malware can get in host from:
  - *virus:* self-copying (self-replicating) infection by receiving/executing object (e.g., e-mail attachment)
  - *worm:* self-copying infection by passively receiving object that gets itself executed
- spyware malware can record
  - keystrokes
  - web sites visited
  - upload info to collection site

Bad guys: attack server, network infrastructure

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts



Chapter 1: 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

1.6 networks under attack: security

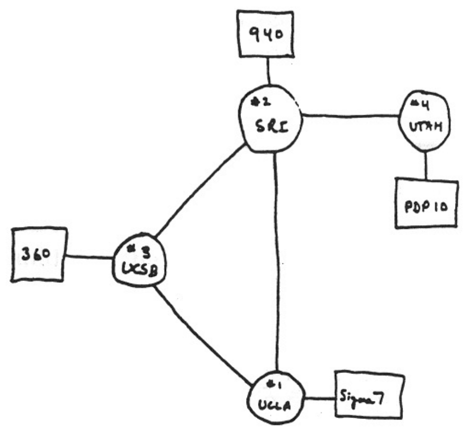
1.7 history

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
- est. 50 million host, 100 million+ users
- backbone links running at Gbps

Internet history

2005-present

- ~5B devices attached to Internet (2016)
  - smartphones and tablets
- aggressive deployment of broadband access
- increasing ubiquity of high-speed wireless access
- emergence of online social networks:
  - Facebook: ~ one billion users
- service providers (Google, Microsoft) create their own networks
  - bypass Internet, providing "instantaneous" access to search, video content, email, etc.
- e-commerce, universities, enterprises running their services in "cloud" (e.g., Amazon EC2)

Introduction: summary

covered a "ton" of material!

- Internet overview
- what's a protocol?
- network edge, core, access network
  - packet-switching versus circuit-switching
  - Internet structure
- performance: loss, delay, throughput
- layering, service models
- security
- history

you now have:

- context, overview, "feel" of networking
- more depth, detail *to follow!*

Chapter 2
Application Layer

App-layer protocol defines

- Types of messages exchanged,
  - e.g., request, response
- Message format:
  - Syntax :what fields in messages
  - Semantics: meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:

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

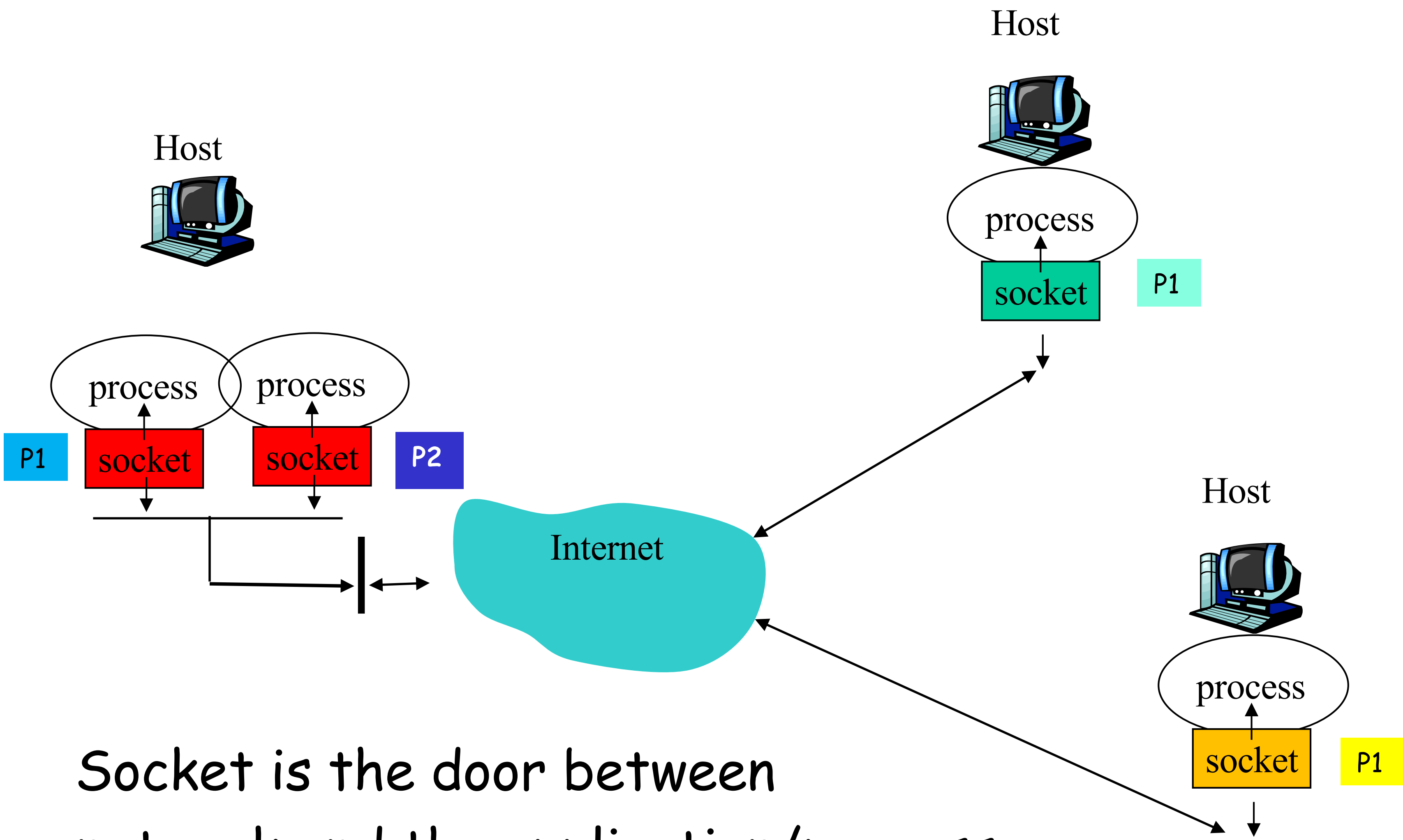
Proprietary protocols:

- e.g., Skype, Hangout

Network application

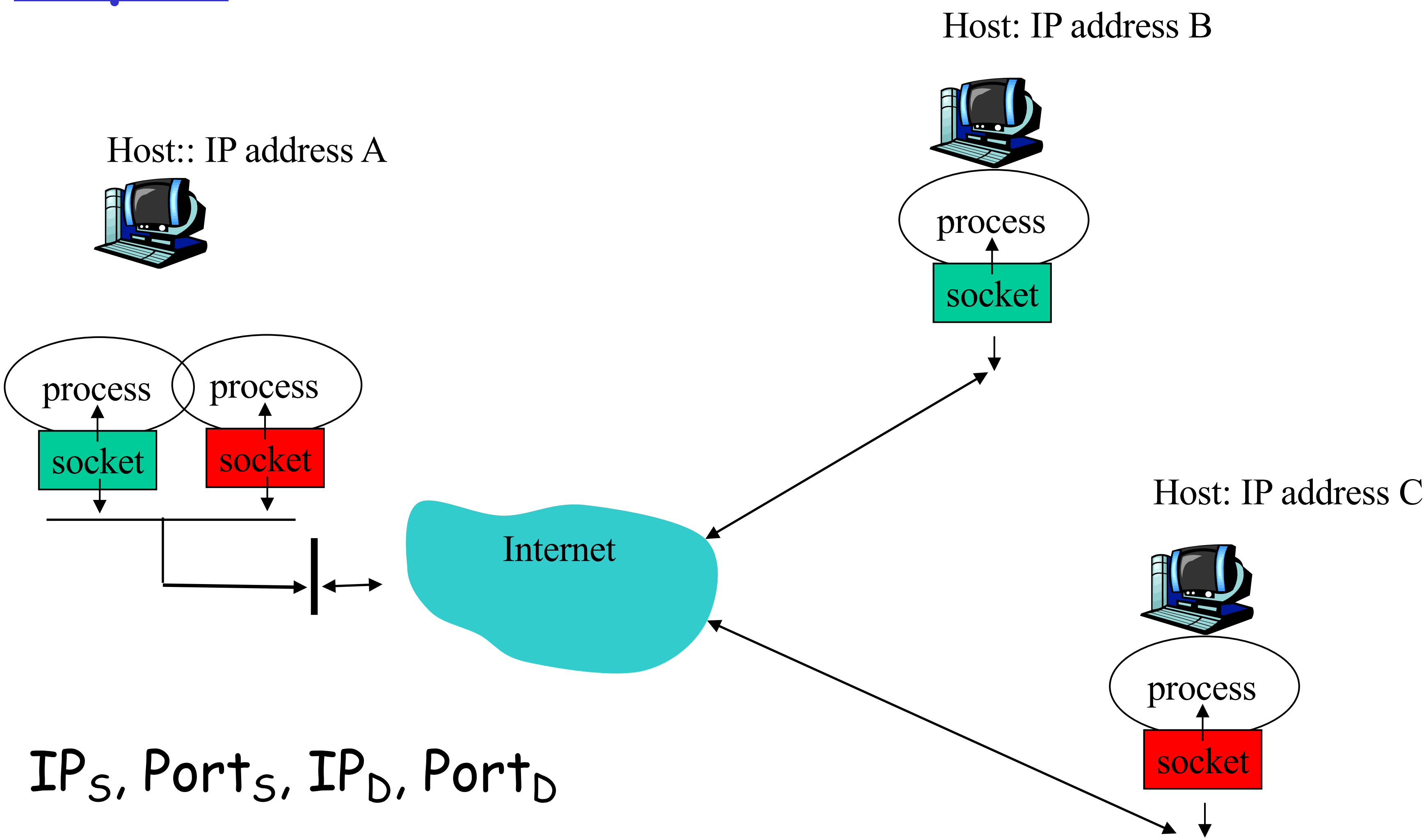
- Consider: Two applications on 2 different hosts connected by a network
- In order to communicate, need to identify the parties
- Phone network: phone number (10 digits)
- Computer network: **IP address**
  - IPv4 (32 bits) 128.6.24.78
  - IPv6 (128 bits) 2001:4000:A000:C000:6000:B001:412A:8000
- In addition to find host address, we need one more.
- More than one program executing on a host
- Which Program to talk to ?
- We need another identity : port #

IP address & port number



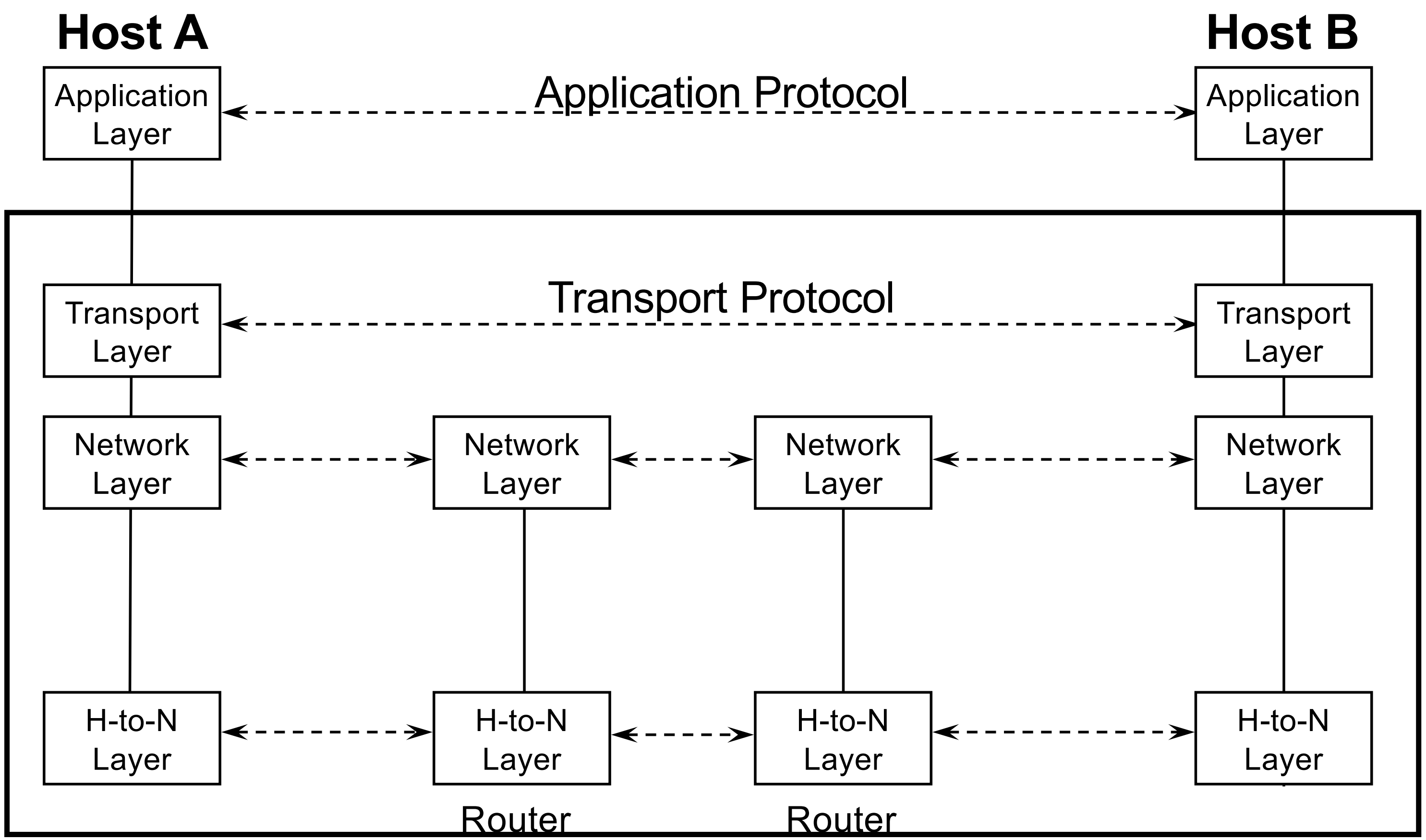
Socket is the door between network and the application/process

A network connection is a 4-tuple

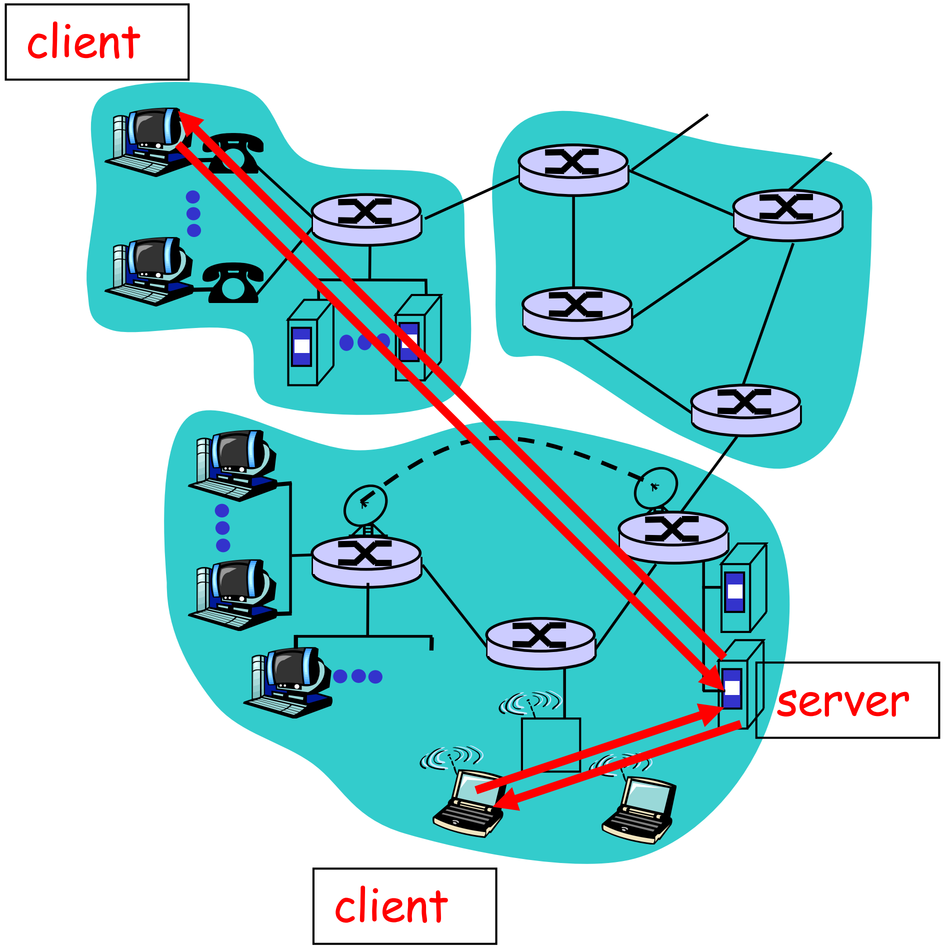


IP_S, $Port_S$, IP_D, $Port_D$

Recall: Services provided by lower layers



Client-server architecture



server:

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

clients:

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

DNS

Why?

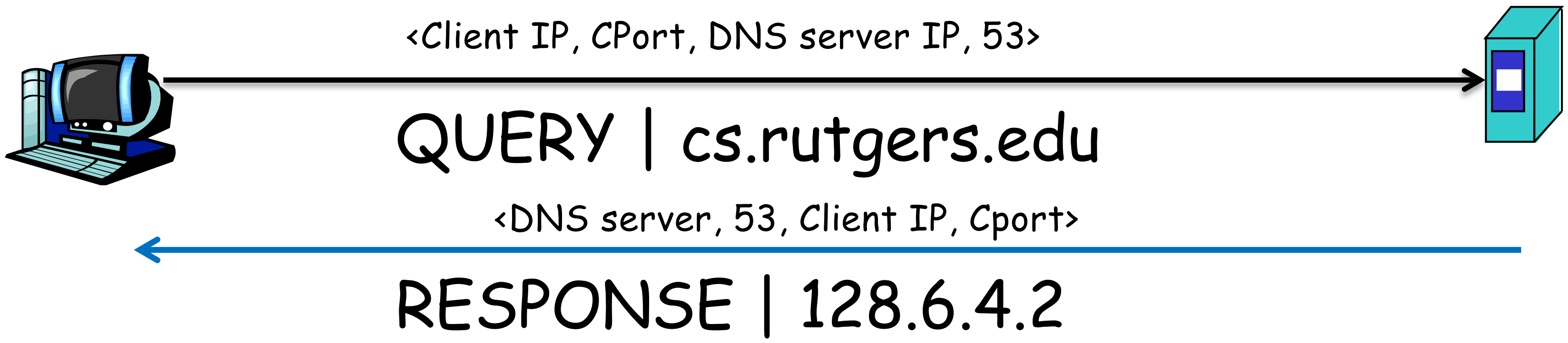
For any networked application, we need to know the IP address of a host given its name

Domain Name System (DNS)

- Problem statement:
  - Average brain can easily remember 7 digits for a few names
  - On average, IP addresses have 12 digits
  - We need an easier way to remember IP addresses
- Solution:
  - Use names to refer to hosts
  - Just as a contact or telephone book (white pages)
  - Add a service (called DNS) to map between host names and binary IP addresses
  - We call this *Address Resolution*

Simple DNS

DOMAIN NAME	IP ADDRESS
WWW.YAHOO.COM	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86



- Simple but does not scale
- Every new host needs to be entered in this table
- Performance? Failure?

DNS

Centralize DNS?

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

doesn't *scale!*