

# cs168 — Computer Networks

John Jannotti  
jj@cs.brown.edu

# Networks class

- **Instructor: John Jannotti (jj)**
- **TAs: Claudiu Saftoiu (csaftoiu) and J. Rassi (jrassi)**
- **Email: [cs168tas@cs.brown.edu](mailto:cs168tas@cs.brown.edu) (entire staff)**
- **Goal: Teach the concepts underlying networks**
  - How do networks work? What can one do with them?
  - Give you the tools to understand new protocols & systems.
- **Prerequisites:**
  - CS 32, 36 (or equiv) – class assumes basic OS concepts (kernel/user, threads/processes, I/O, scheduling)
  - C programming or willingness to learn quickly. (explicit memory management, pthreads, locking)

# Administrivia

- **All assignments will be on the web page**  
`http://www.cs.brown.edu/courses/cs168/`
- **Text: Andrew Tanenbaum, *Computer Networks***
- **Consuly web page calendar for almost everything.**
  - Gives textbook chapters corresponding to lectures.
  - Later in the year, papers will augment the book.
  - Read the chapters & papers before class.
  - Handouts, due dates, *etc.*

# Grading

- **Quizzes: Midterm & Final**
- **Homework: Four written assignments — short answer and design questions**
- **Programming assignments (implemented in C/C++)**
  - User-level networking: a streaming music server.
  - IP, as an overlay, on top of UDP.
  - TCP, using your own IP.
  - Final. The *class* will design, specify, and implement a network service. (VoIP? File sharing? Tunneler? Proxy?)

# Topics

- **Network programming (sockets, RPC)**
- **Network (esp. Internet) architecture**
  - Switching, Routing, Congestion control, TCP/IP, Multicast, Wireless networks
- **Using the network**
  - Interface hardware & low-level implementation issues, Naming (DNS), Error detection, Compression
- **Higher level issues**
  - Encryption & Security, Caching & Replication, Overlays, P2P

# Networks

- **What is a network?**

- A system of lines/channels that interconnect
- *E.g.*, railroad, highway, plumbing, communication, telephone, social, **computer**

- **What is a *computer* network?**

- A form of communication network—moves information.
- Links may be wires, fiber optics, EM spectrum, or composite.
- Nodes are general-purpose computers (or close to it).

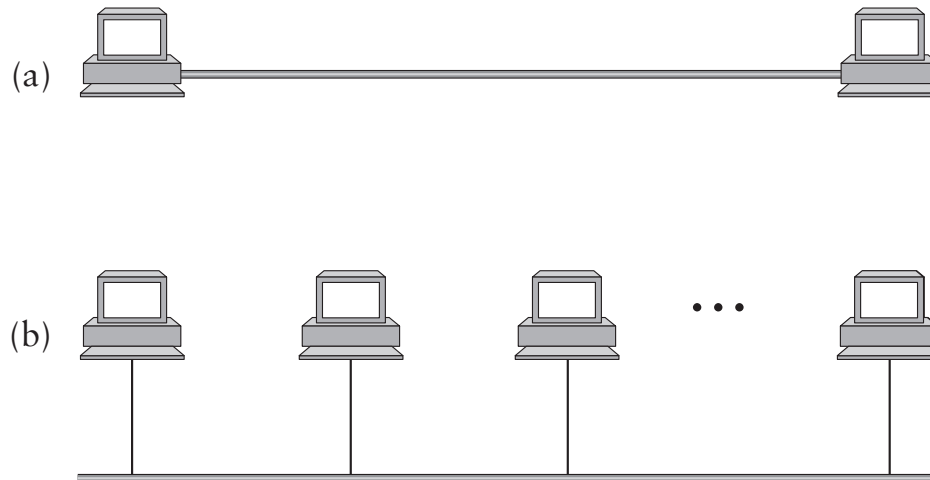
- **Why study computer networks?**

- Many nodes are general-purpose computers
- *You* can program the nodes
- Very easy to innovate and develop new uses of network

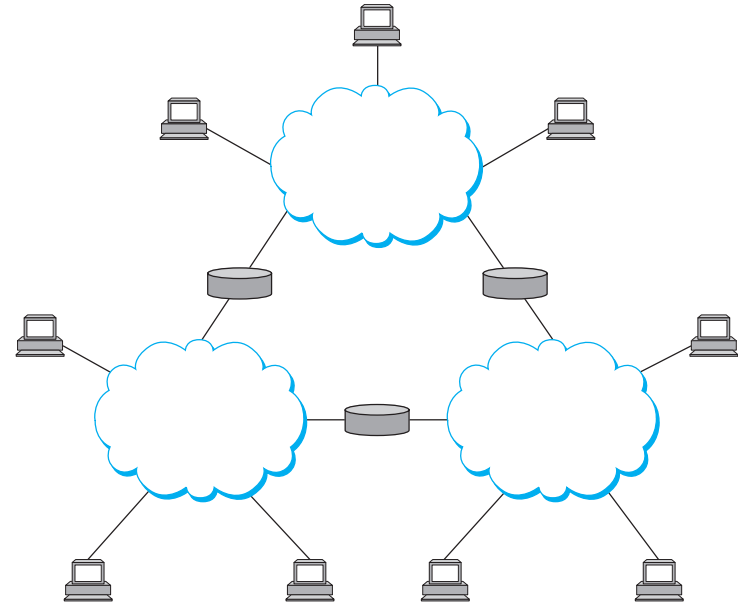
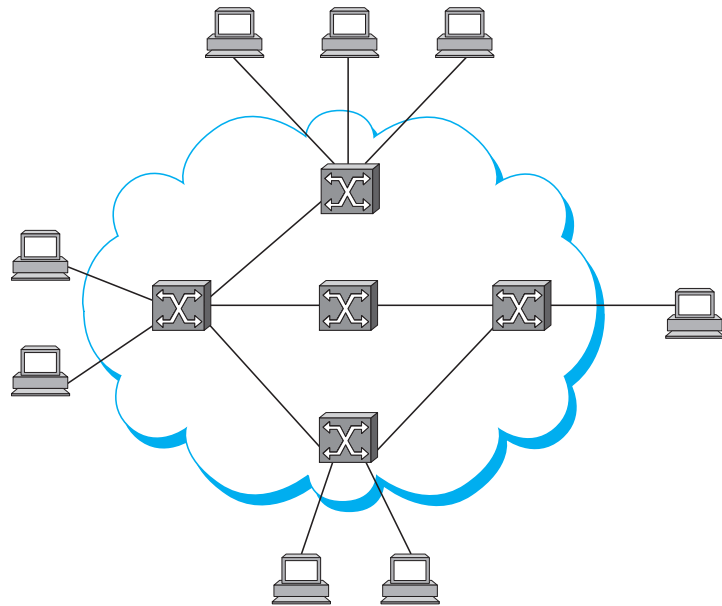
- Contrast: Telephone (POTS or cell) network—can't program most phones, need FCC/Carrier approval for new devices, *etc.*

# Building blocks

- **Nodes:** Computers, dedicated routers, ...
- **Links:** Coax, twisted pair, fibers, radio ...
  - (a) point-to-point
  - (b) multiple access – every node sees every packet



# Switched networks



- To scale to more nodes, use *switching*
  - nodes can connect multiple other nodes, or
  - Recursively, one node can connect multiple networks

# Switching strategies

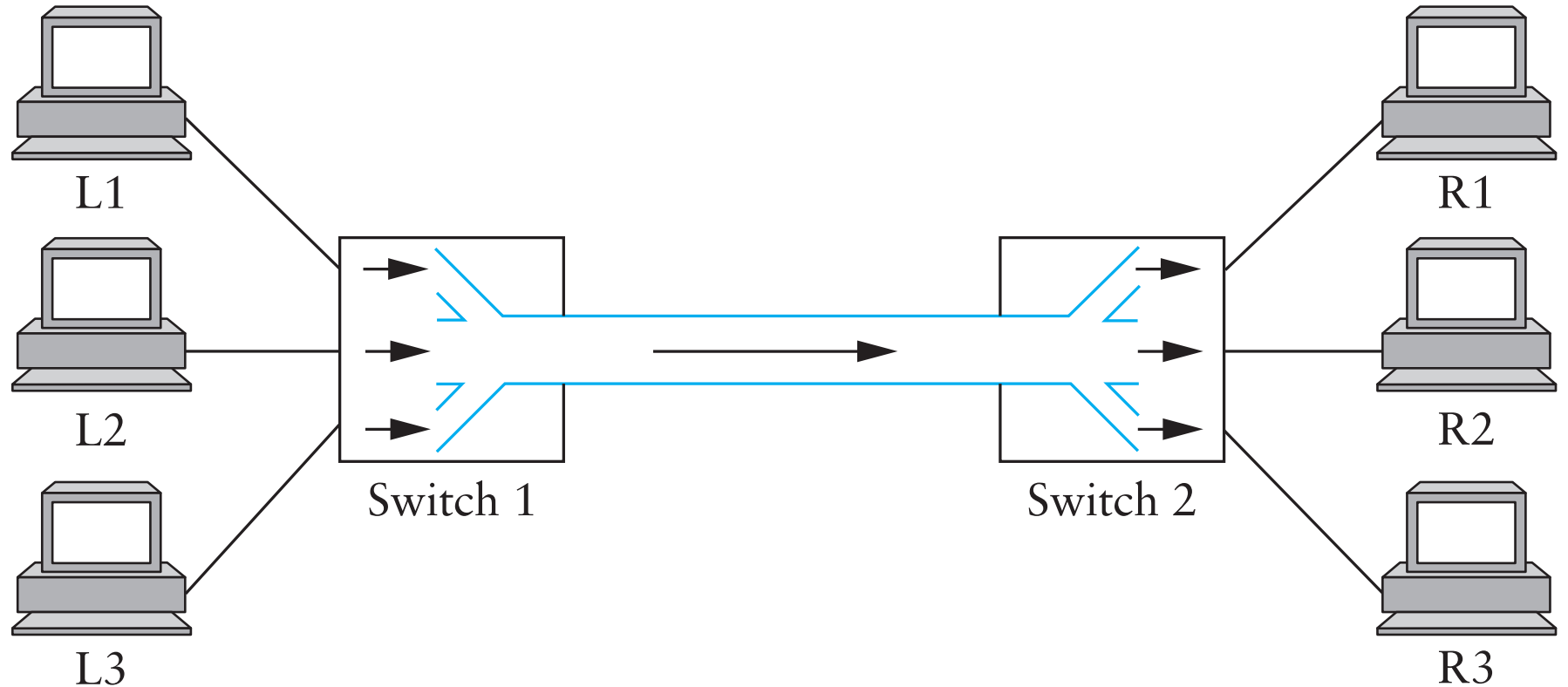
- **Circuit switching – virtual link between two nodes**
  - Set up circuit (*e.g.*, dialing, signaling) – may fail if busy.
  - Transfer data at known rate.
  - Tear down circuit.
- **Packet switching**
  - Forward bounded-size messages.
  - Each message can be between a different sender receiver.
  - This class will concentrate on packet switched networks.

Analogy: Circuit switching reserves the highway for a cross-country trip. Packet switching interleaves everyone's cars.

# Addressing

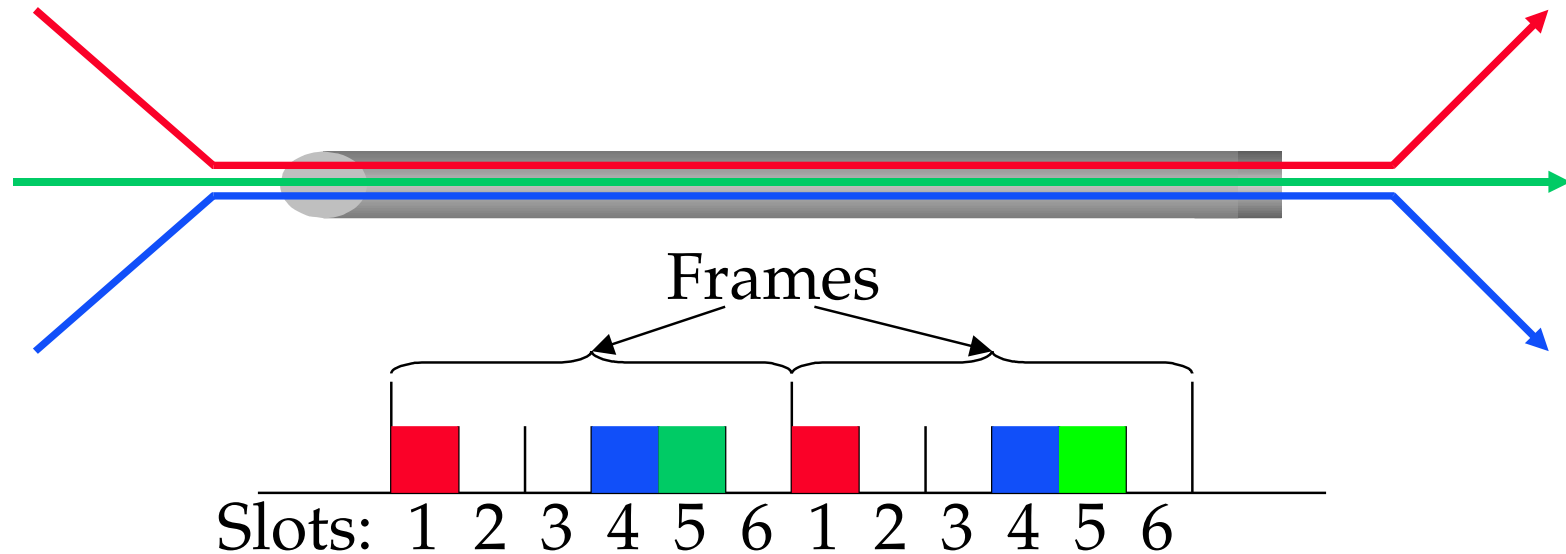
- **Each node typically has a unique *address*.**
  - (or at least is made to think it does in case of shortage)
- ***Routing* is the process of delivering data to destination.**
  - For packet switched networks, each packet must have destination address.
  - For circuit switched, use address to set up circuit.
- **Special addresses can exist for broadcast/multicast.**

# Multiplexing



- What to do when multiple flows must share a link?

# STDM



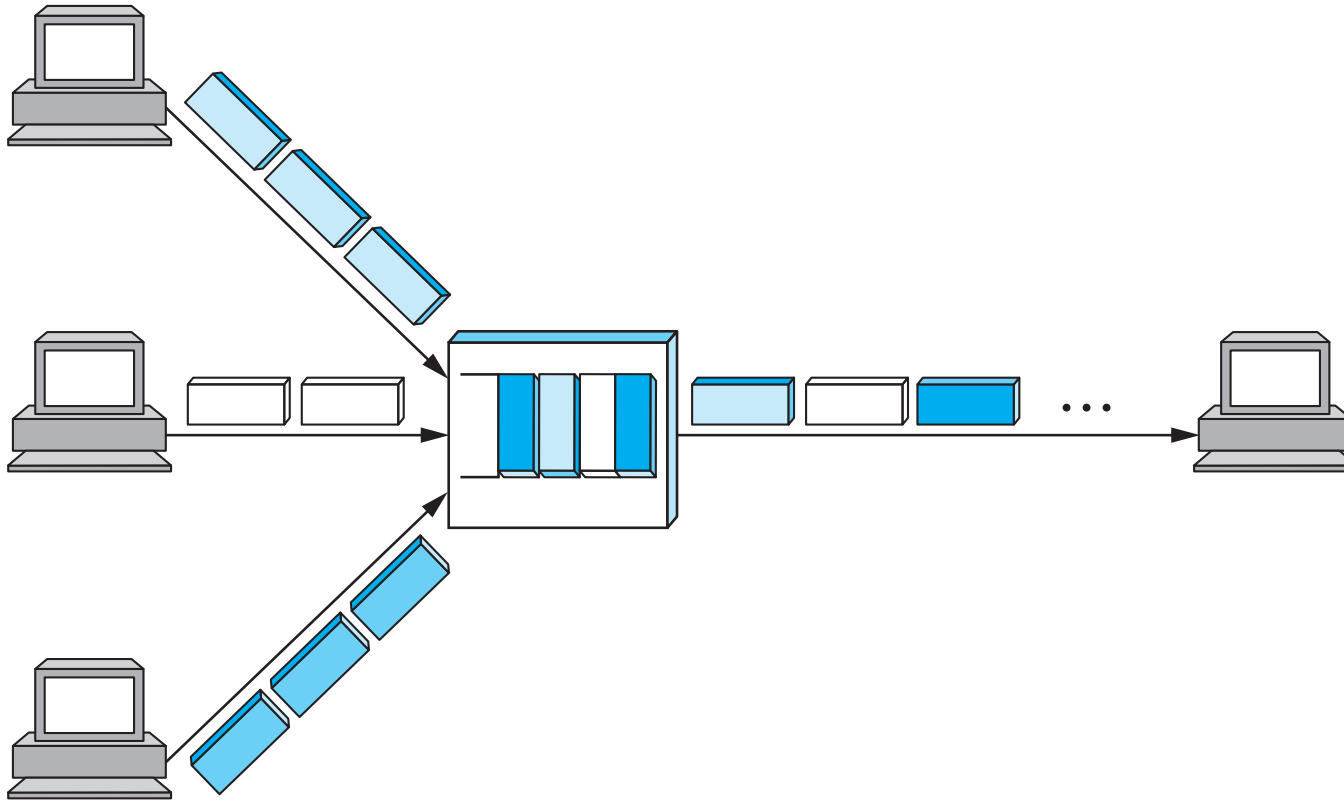
- **Synchronous time-division multiplexing**

- Divide time into equal sized quanta, round-robin
- Provide illusion of direct link for circuit switched net
- But also wastes capacity if not enough flows
- And doesn't degrade gracefully when run out of slots

# FDM

- **Frequency-division multiplexing: Everybody transmits on a different frequency**
  - Same way radio and TV stations each have separate frequency
- **Similar drawbacks to STDM**
  - Wastes bandwidth if someone not sending (ok for TV and radio with constant transmission rate, but might want to download audio faster than you can play).
  - Can run out of spectrum (*e.g.*, limited # of radio stations)

# Statistical multiplexing

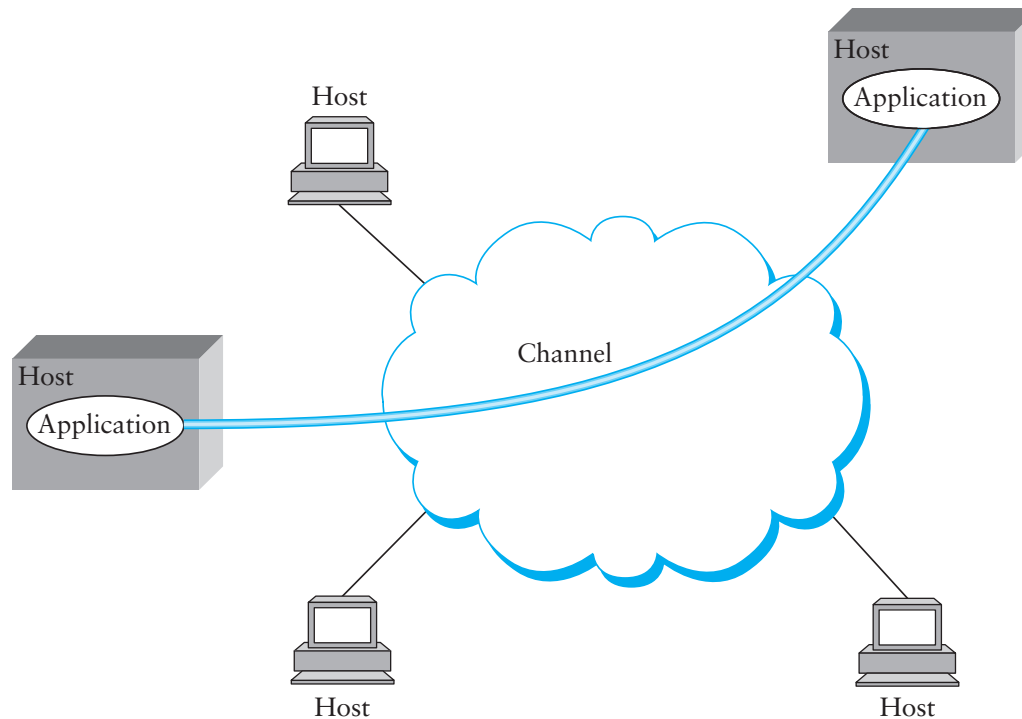


- **Idea: Like STDM w/o pre-determined time slots**
- **Maximizes link utilization**
  - Link never goes idle when packet to send

# Internet Protocol (IP)

- **Used by most computer networks today.**
  - Runs over a variety of physical networks, so can connect Ethernet, Wireless, people behind modem lines, etc.
- **Every host has a unique 4-byte IP address**
  - *E.g.*, `www.cs.brown.edu` → 128.148.32.110
  - The *network* knows how to route a packet to any address.
- **More is needed to build something like the web.**
  - Need naming (look up `www.cs.brown.edu`) – DNS (later lecture)
  - Need interface for browser & server software (next lecture)
  - Need demultiplexing within a host—*E.g.*, which packets are for web server, which for mail server, etc.?

# Inter-process communication



- Want abstraction of inter-process (not just inter-node) communication
- Solution: *Encapsulate* another protocol within IP

# UDP and TCP

- **UDP and TCP most popular protocols on IP**
  - Both use 16-bit *port* number as well as 32-bit IP address
  - Applications *bind* a port & receive traffic to that port
- **UDP – user (unreliable) datagram protocol**
  - Exposes packet-switched nature of Internet
  - Sent packets may be dropped, reordered, even duplicated (but generally not corrupted)
- **TCP – transmission control protocol**
  - Provides illusion of a reliable “pipe” or “stream” between two processes on two (probably) different machines
  - Handles congestion & flow control (much later lecture)

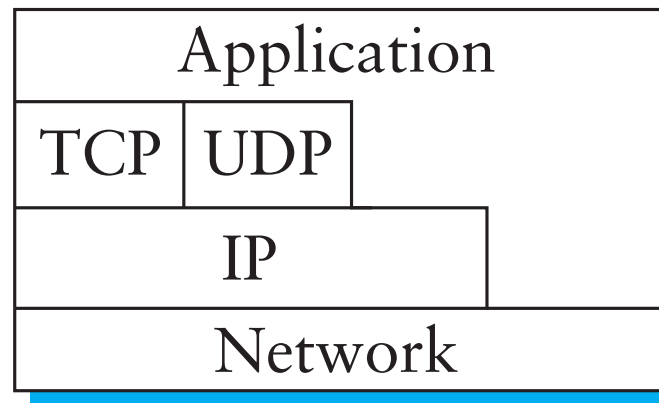
# Failure

- **Several types of error can affect packet delivery**
  - Bit errors (*e.g.*, electrical interference, cosmic rays)
  - Packet loss (overload)
  - Link and node failure
- **In addition, properly delivered packets can be delayed and reordered**

# Uses of TCP

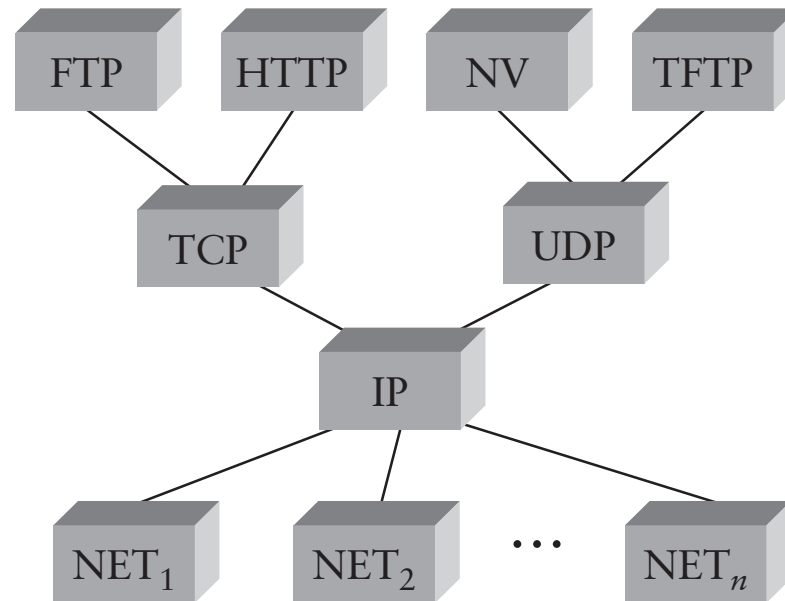
- **Most applications use TCP**
  - Easier interface to program. (reliability is convenient)
  - Automatically avoids congestion (don't need to worry about taking down network)
- **Servers typically listen on well-known ports**
  - SSH: 22
  - SMTP (email): 25
  - Finger: 79
  - HTTP (web): 80

# IP layering



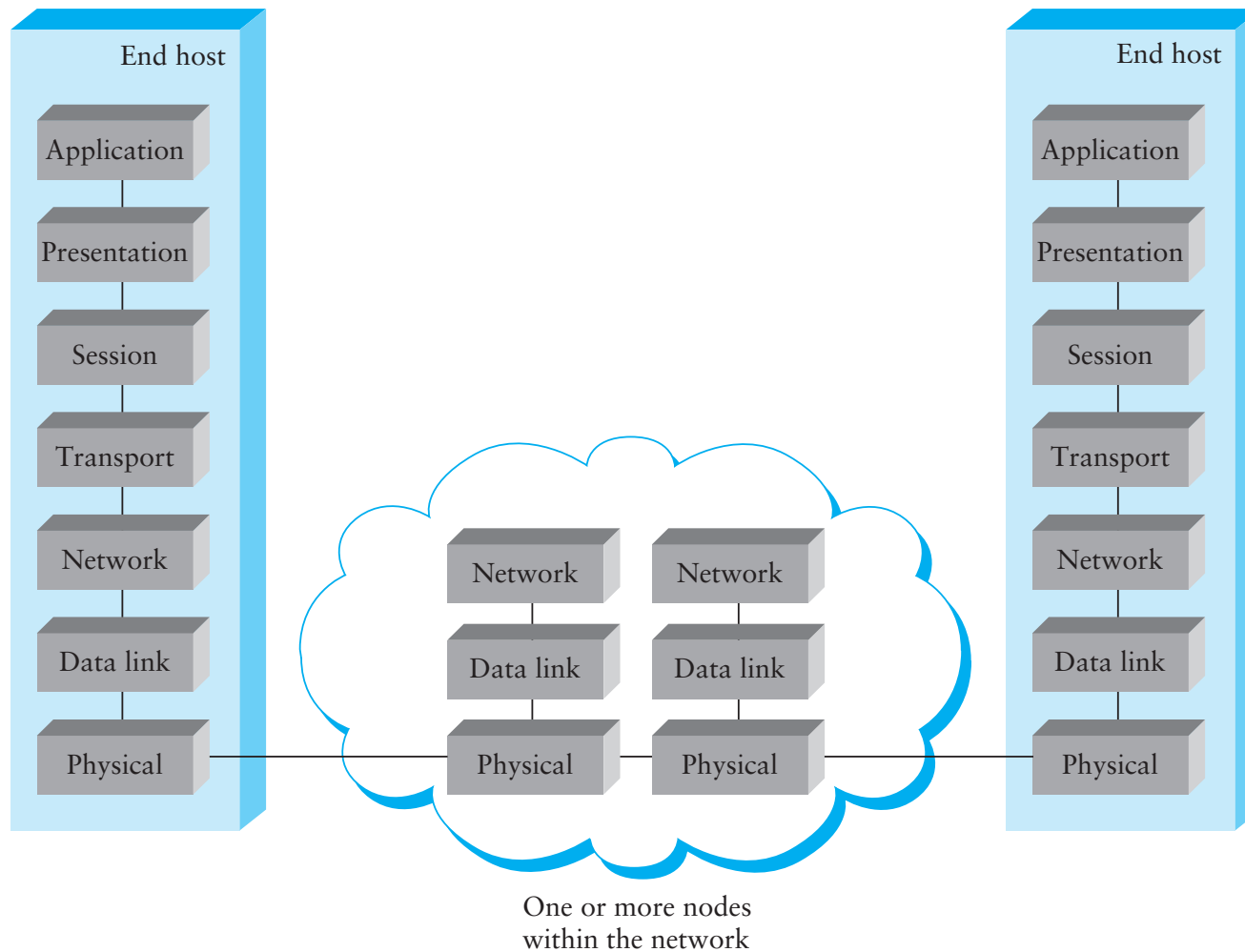
- **Can view network encapsulation as a stack**
  - Each layer produces packets that become the payload of the lower-layer's packets
  - This is almost correct, but TCP/UDP “cheat” to detect certain errors in IP-level information like address.

# Hourglass



- **Many application protocols over TCP & UDP**
- **IP works over many types of network**
- **This is the “Hourglass” philosophy of the Internet**
  - If every network supports IP, applications run over many different networks.

# OSI layers



- Layers typically fall into 1 of 7 categories.

# OSI layers

1. Physical – sends individual bits
2. Data link – sends *frames*; media access control
3. Network – delivers packets to hosts using *routing*
4. Transport – delivers data to apps; reliability & flow control
5. Session – can tie together multiple streams (*e.g.*, audio & video)
6. Presentation – crypto, representation conversion
7. Application – what end user gets, *e.g.*, HTTP (web)

Layers 1-3 are the “communications subnet.”

Layers 4-7 are often combined.