CSCI-1680 - Computer Networks

Chen Avin (avin)

Based partly on lecture notes by David Mazières, Phil Levis, John Jannotti, Peterson & Davie, Rodrigo Fonseca

Administrivia

- **Sign and hand in Collaboration Policy**
- **Signup for Snowcast milestone**
  - Thursday from 8pm to 11pm
  - See Piazza for links
- **Github**

Today

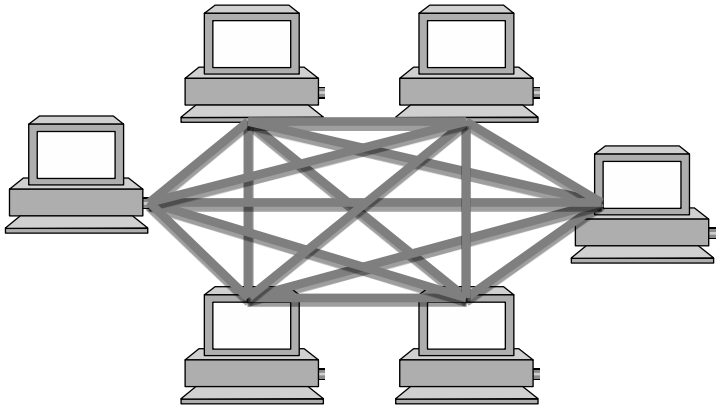
- **Switching, Multiplexing**
- **Layering and Encapsulation**
- **Intro to IP, TCP, UDP**

Building Blocks

- **Nodes: Computers (hosts), dedicated routers, …**
- **Links: Coax, twisted pair, fiber, radio, …**

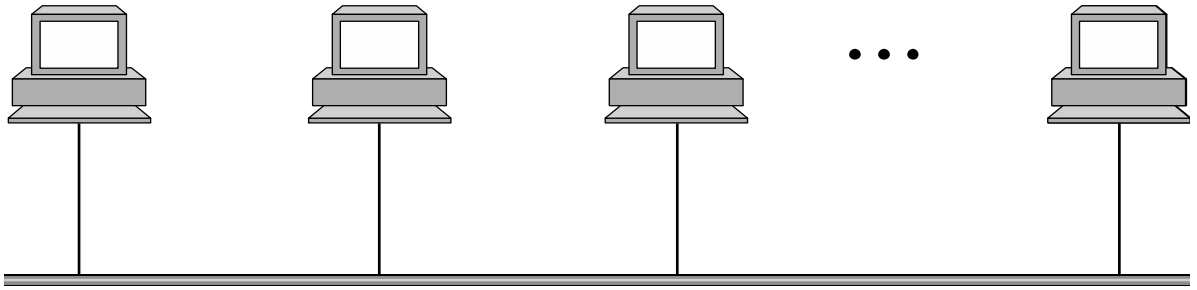
Host **Host**

How to connect more nodes?

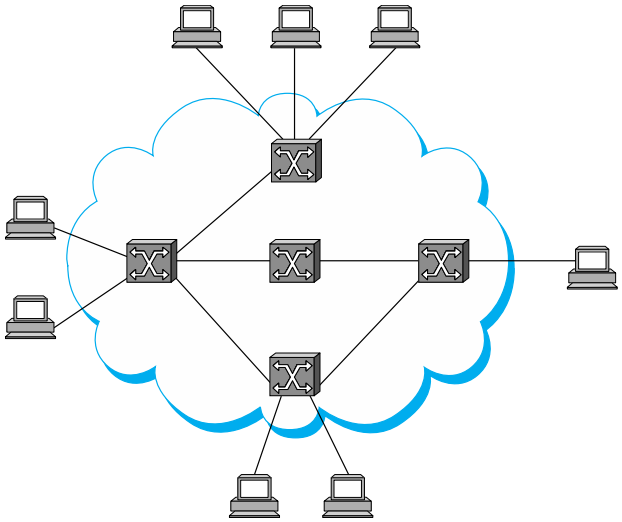


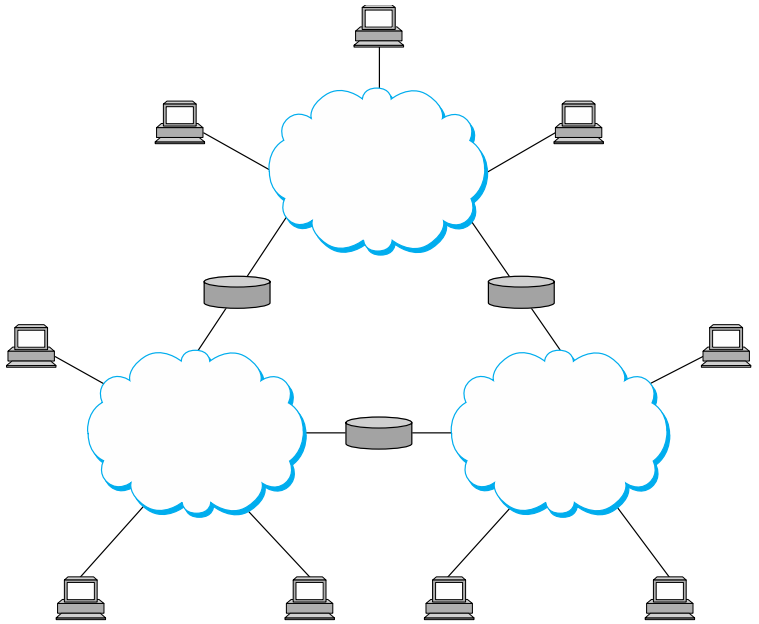
Multiple wires

Shared medium



From Links to Networks



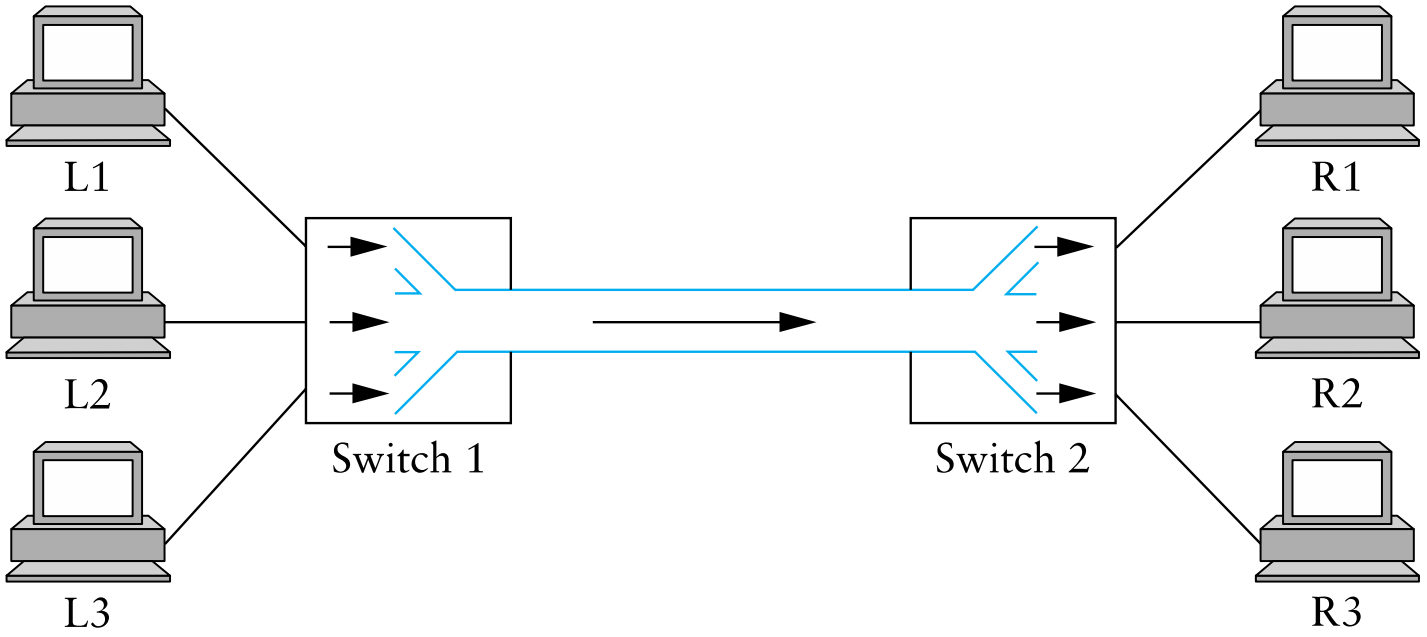


- **To scale to more nodes, use *switching***
  - Nodes can connect to multiple other nodes
  - Recursively, one node can connect to multiple networks

Switching Strategies

- **Circuit Switching – virtual link between two nodes**
  - Set up circuit (e.g. dialing, signaling) – may fail: busy
  - Transfer data at known rate
  - Tear down circuit
- **Packet Switching**
  - Forward bounded-size messages.
  - Each message can have different senders/receivers
  - Focus of this course
- **Analogy: circuit switching reserves the highway for a cross-country trip. Packet switching interleaves everyone's cars.**

Multiplexing



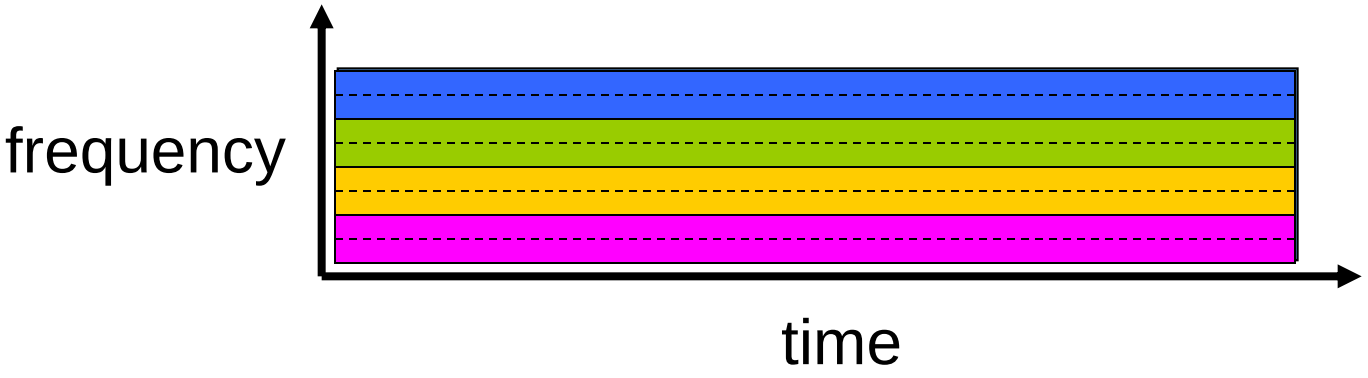
- **What to do when multiple flows must share a link?**

Circuit switching: FDM versus TDM

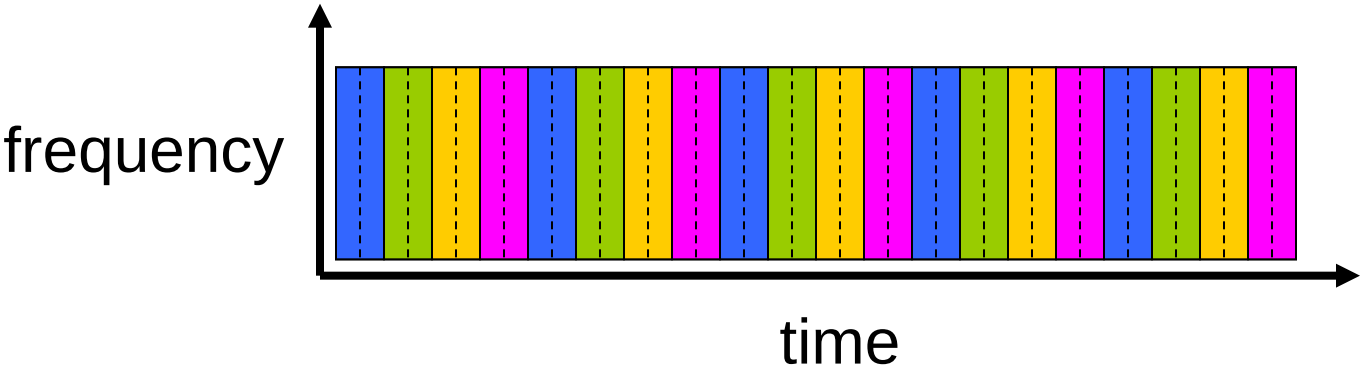
Example:

4 users

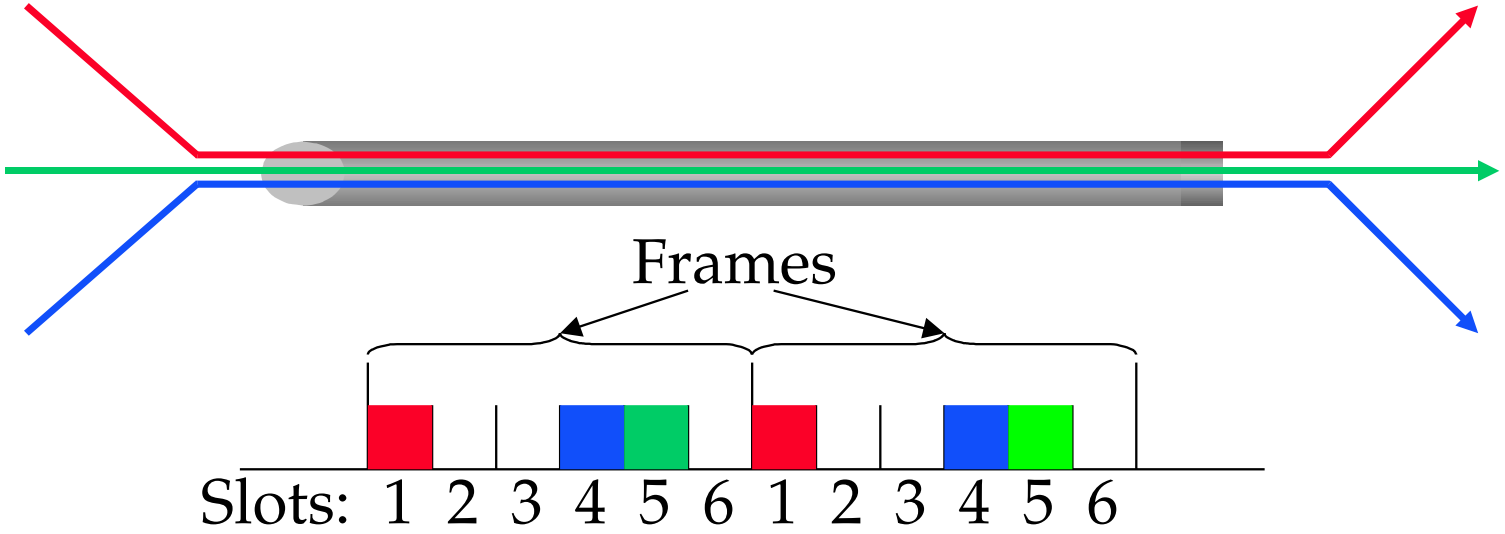
FDM



TDM



STDM

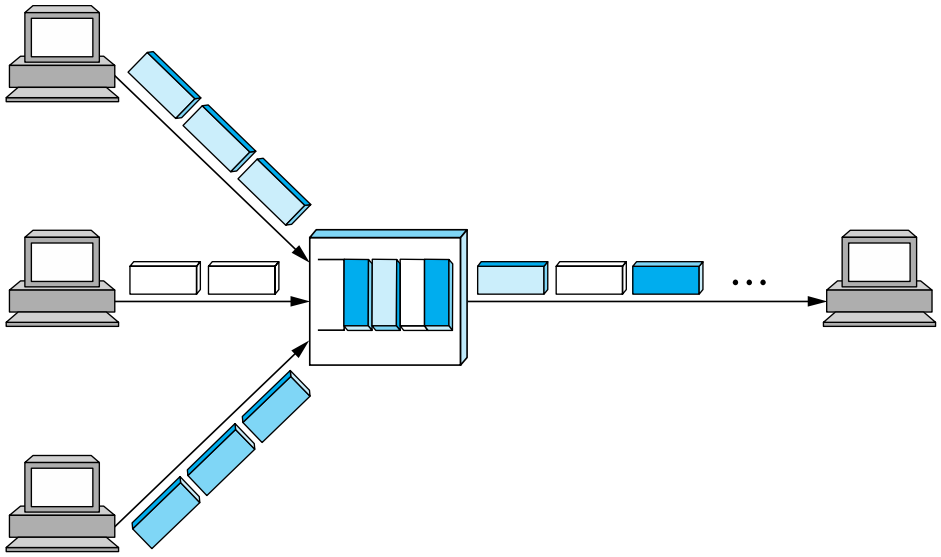


- **Synchronous time-division multiplexing**
  - Divide time into equal-sized quanta, round robin
  - Illusion of direct link for switched circuit net
  - But wastes capacity if not enough flows
  - Also doesn't degrade gracefully when more flows than slots

FDM

- **Frequency-division multiplexing: allocates a frequency band for each flow**
  - Same TV channels and radio stations
- **Similar drawbacks to STDM**
  - Wastes bandwidth if someone not sending
  - Can run out of spectrum
  - Scaling, managing complexity

Packet Switching: Statistical Multiplexing



- **Idea: like STDM but with no pre-determined time slots (or order!)**
- **Maximizes link utilization**
  - Link is never idle if there are packets to send

Statistical Multiplexing

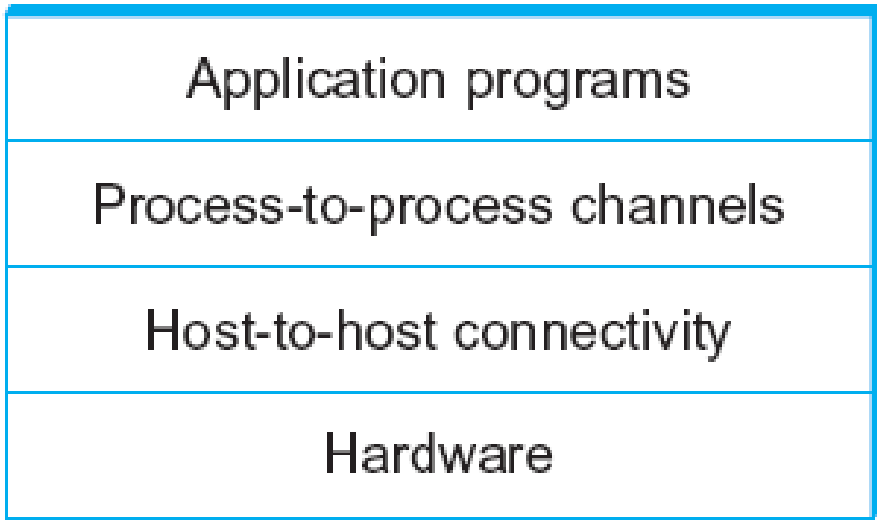
- **Cons:**
  - Hard to guarantee fairness
  - Unpredictable queuing delays
  - Packets may take different paths
- **Yet…**
  - This is the main model used on the Internet

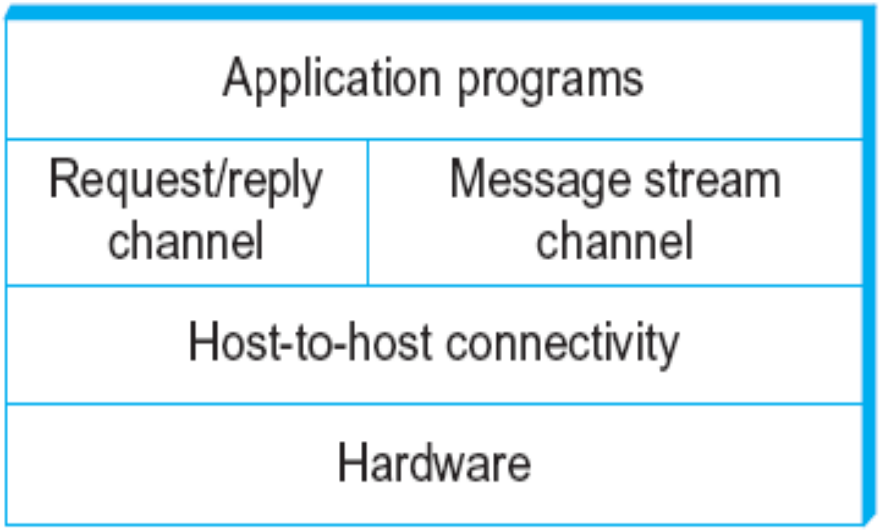
Managing Complexity

- ***Very* large number of computers**
- **Incredible variety of technologies**
  - Each with very different constraints
- **No single administrative entity**
- **Evolving demands, protocols, applications**
  - Each with very different requirements!

- **How do we make sense of all this?**

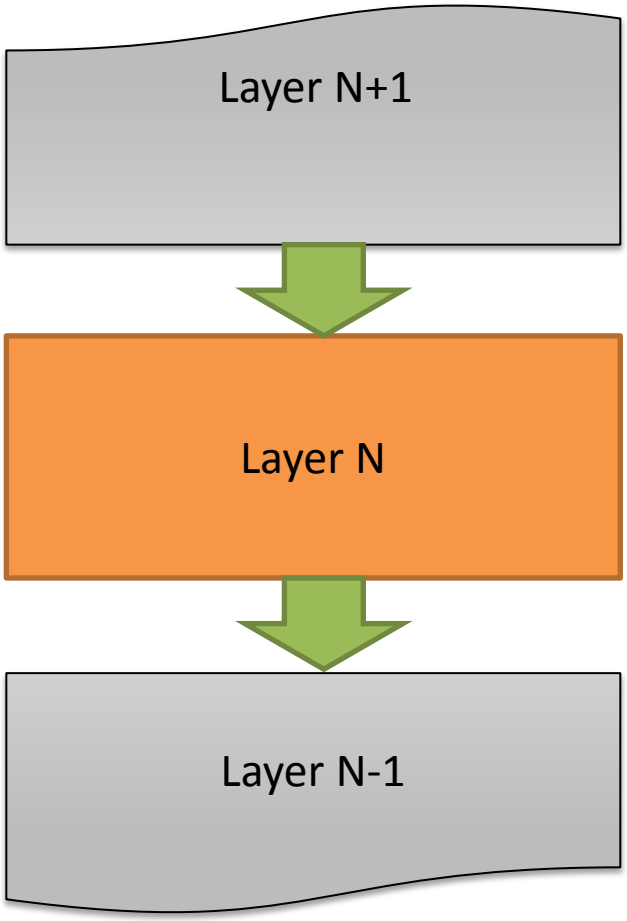
Layering: Network Architecture





- **Separation of concerns**
  - Break problem into separate parts
  - Solve each one independently
  - Tie together through common interfaces: abstraction
  - Encapsulate data from the layer above inside data from the layer below
  - Allow independent evolution

Layers, Services, Protocols



Service: abstraction provided to layer above
API: concrete way of using the service

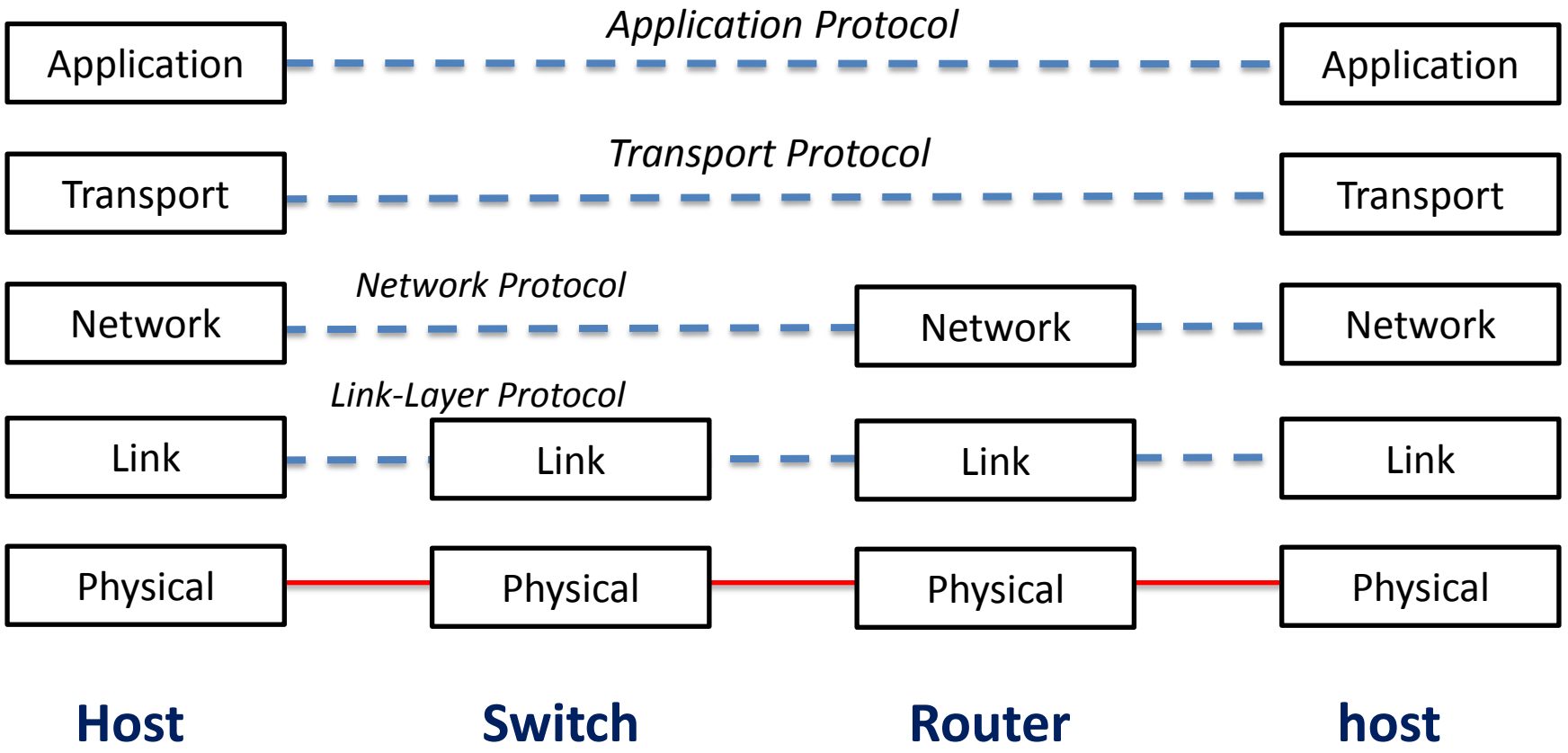
Protocol: rules for communication within same layer

Layer N uses the services provided by N-1 to implement its protocol and provide its own services

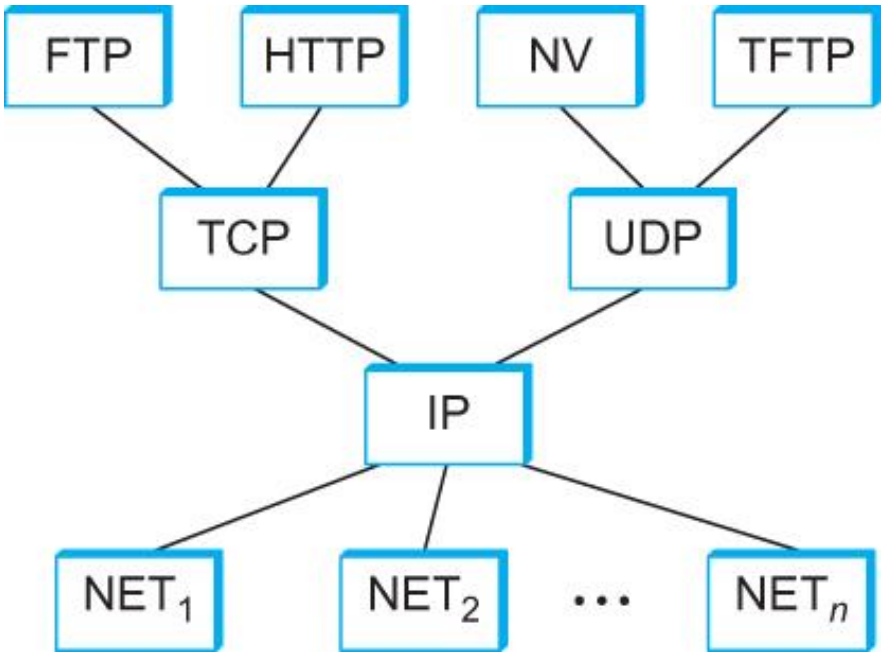
Internet: Layers, Services, Protocols

Application	Service: user-facing application. Application-defined messages
Transport	Service: multiplexing applications Reliable byte stream to other node (TCP), Unreliable datagram (UDP)
Network	Service: move packets to any other node in the network IP: Unreliable, best-effort service model
Link	Service: move frames to other node across link. May add reliability, medium access control
Physical	Service: move bits to other node across link

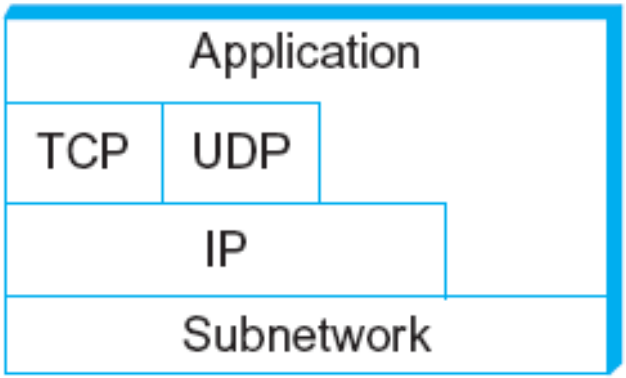
Internet Layering



Internet Architecture

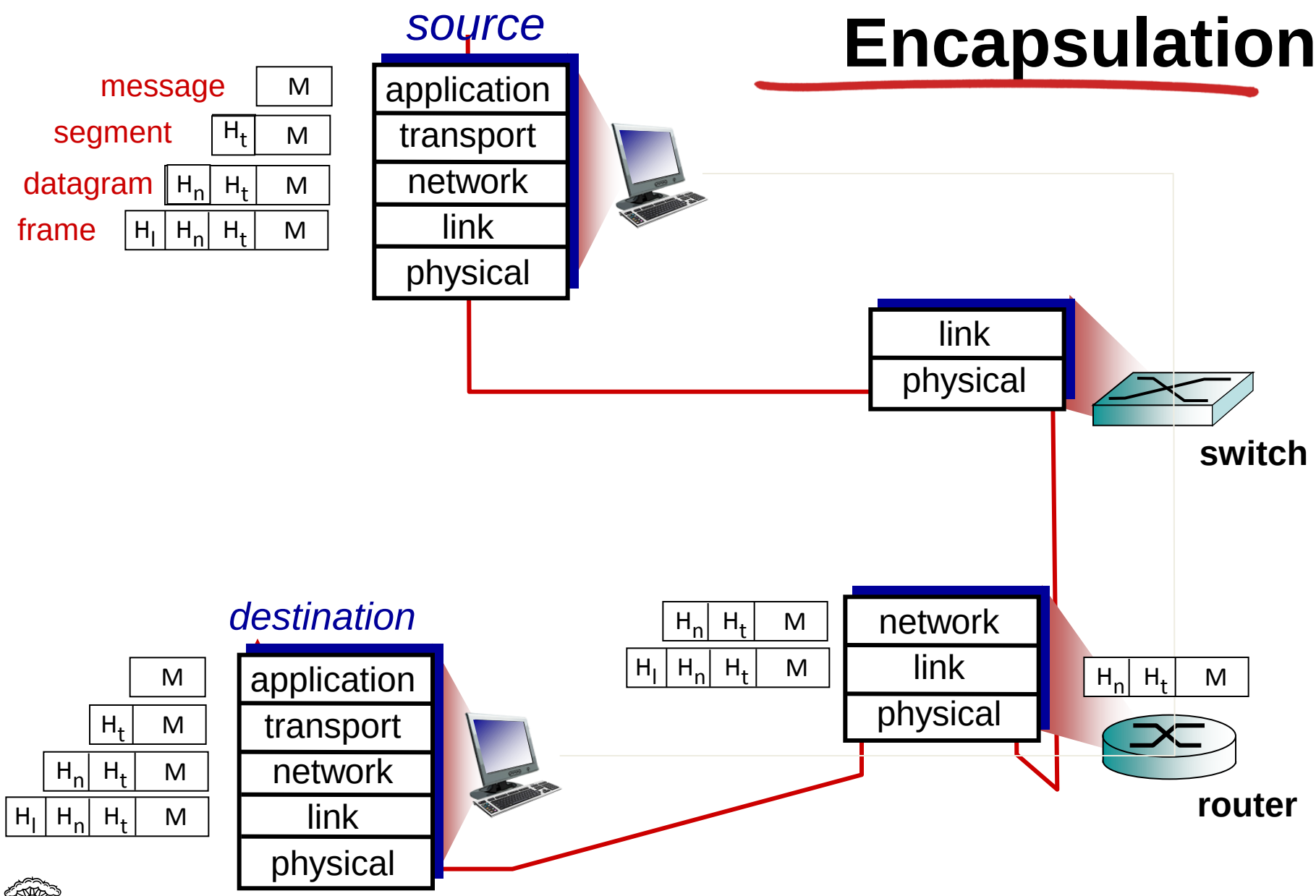


Internet Protocol Graph

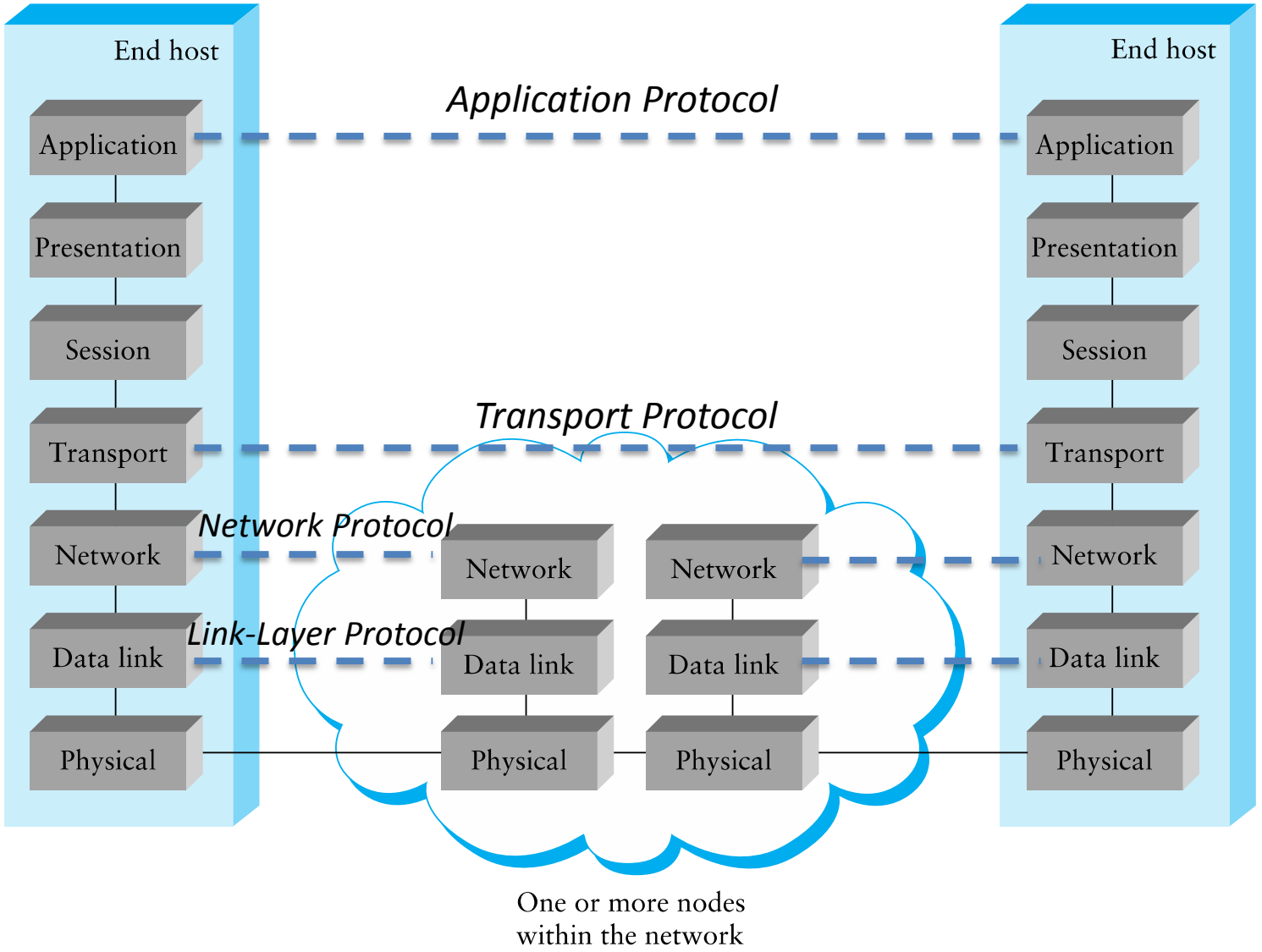


Alternative view of the Internet architecture. The "Network" layer shown here is sometimes referred to as the "sub-network" or "link" layer.

Encapsulation



OSI Reference Model



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

Protocols

- **What do you need to communicate?**
  - Definition of message formats
  - Definition of the semantics of messages
  - Definition of valid sequences of messages
    - Including valid timings

Addressing

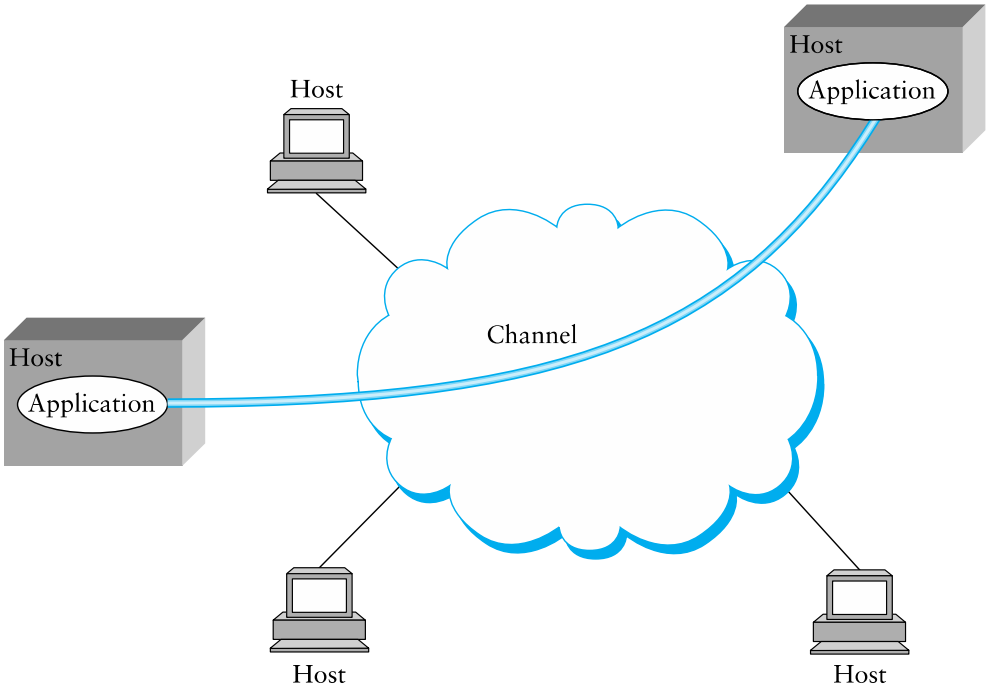
- **Each node typically has a unique* name**
  - When that name also tells you how to get to the node, it is called an *address*
- **Each layer can have its own naming/addressing**
- ***Routing* is the process of finding a path to the destination**
  - In packet switched networks, each packet must have a destination address
  - For circuit switched, use address to set up circuit
- **Special addresses can exist for broadcast/multicast/anycast**

** or thinks it does, in case there is a shortage*

Network Layer: Internet Protocol (IP)

- **Used by most computer networks today**
  - Runs *over* a variety of physical networks, can connect Ethernet, wireless, modem lines, etc.
- **Every host has a unique 4-byte IP address (IPv4)**
  - *E.g.*, www.cs.brown.edu →128.148.32.110
  - The *network* knows how to route a packet to any address
- **Need more to build something like the Web**
  - Need naming (DNS)
  - Interface for browser and server software (next lecture)
  - Need demultiplexing within a host: which packets are for web browser, Skype, or the mail program?

Inter-process Communication



- **Talking from host to host is great, but we want abstraction of inter-process communication**
- **Solution: *encapsulate* another protocol within IP**

Transport: UDP and TCP

- **UDP and TCP most popular protocols on IP**
  - Both use 16-bit *port* number & 32-bit IP address
  - Applications *bind* a port & receive traffic on that port
- **UDP – User (unreliable) Datagram Protocol**
  - Exposes packet-switched nature of Internet
  - Sent packets may be dropped, reordered, even duplicated (but there is corruption protection)
- **TCP – Transmission Control Protocol**
  - Provides illusion of reliable ‘pipe’ or ‘stream’ between two processes anywhere on the network
  - Handles congestion and flow control

Uses of TCP

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

Using TCP/IP

- **How can applications use the network?**
- ***Sockets* API.**
  - Originally from BSD, widely implemented (*BSD, Linux, Mac OS X, Windows, …)
  - Important do know and do once
  - Higher-level APIs build on them
- **After basic setup, much like files**

Sockets: Communication Between Machines

- **Network sockets are file descriptors too**
- **Datagram sockets: unreliable message delivery**
  - With IP, gives you UDP
  - Send atomic messages, which may be reordered or lost
  - Special system calls to read/write: send/recv
- **Stream sockets: bi-directional pipes**
  - With IP, gives you TCP
  - Bytes written on one end read on another
  - Reads may not return full amount requested, must re-read

Coming Up

- **Next class: Physical Layer**
- **Thu 13th: Snowcast milestones**