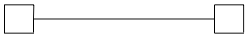
Introduction

Outline

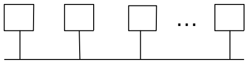
Statistical Multiplexing
Inter-Process Communication
Network Architecture
Performance Metrics
Implementation Issues

Building Blocks

- Nodes: PC, special-purpose hardware…
  - hosts
  - switches
- Links: coax cable, optical fiber…
  - point-to-point

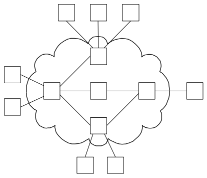


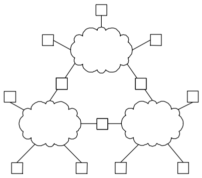
  - multiple access

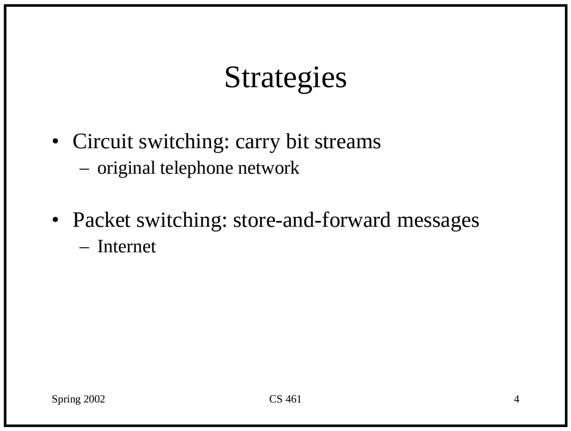


Switched Networks

- A network can be defined recursively as...
  - two or more nodes connected by a link, or
  - two or more networks connected by a node

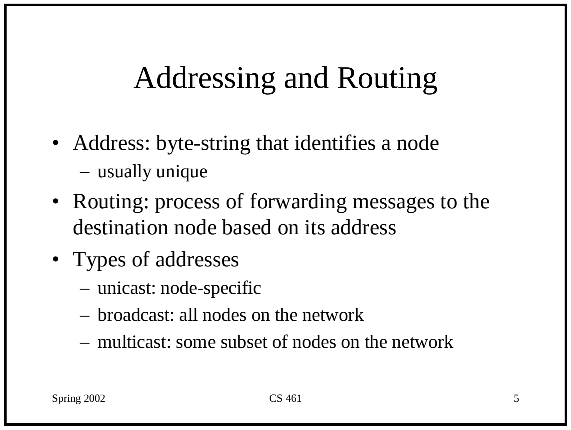






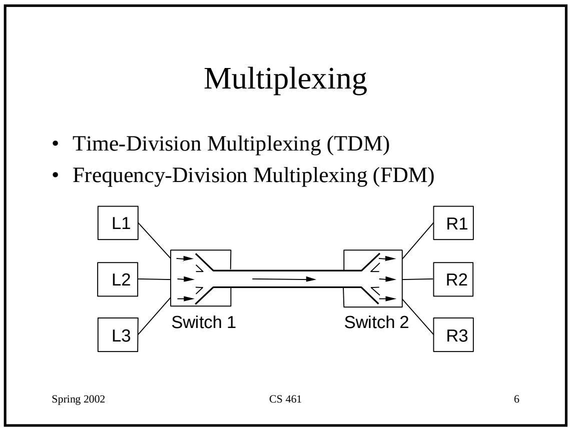
Strategies

- Circuit switching: carry bit streams
  - original telephone network
- Packet switching: store-and-forward messages
  - Internet



Addressing and Routing

- Address: byte-string that identifies a node
  - usually unique
- Routing: process of forwarding messages to the destination node based on its address
- Types of addresses
  - unicast: node-specific
  - broadcast: all nodes on the network
  - multicast: some subset of nodes on the network

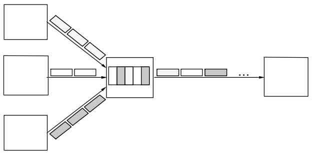


Multiplexing

- Time-Division Multiplexing (TDM)
- Frequency-Division Multiplexing (FDM)

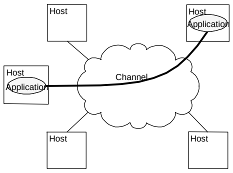
Statistical Multiplexing

- On-demand time-division
- Schedule link on a per-*packet* basis
- Packets from different sources interleaved on link
- Buffer packets that are *contending* for the link
- Buffer (queue) overflow is called *congestion*



Inter-Process Communication

- Turn host-to-host connectivity into process-to-process communication.
- Fill gap between what applications expect and what the underlying technology provides.



IPC Abstractions

- Request/Reply
  - distributed file systems
  - digital libraries (web)
- Stream-Based
  - video: sequence of frames
    - 1/4 NTSC = 352x240 pixels
    - (352 x 240 x 24)/8=247.5KB
    - 30 fps = 7500KBps = 60Mbps
  - video applications
    - on-demand video
    - video conferencing

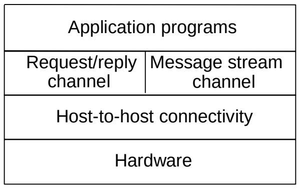
What Goes Wrong in the Network?

- Bit-level errors (electrical interference)
- Packet-level errors (congestion)
- Link and node failures

- Packets are delayed
- Packets are deliver out-of-order
- Third parties eavesdrop

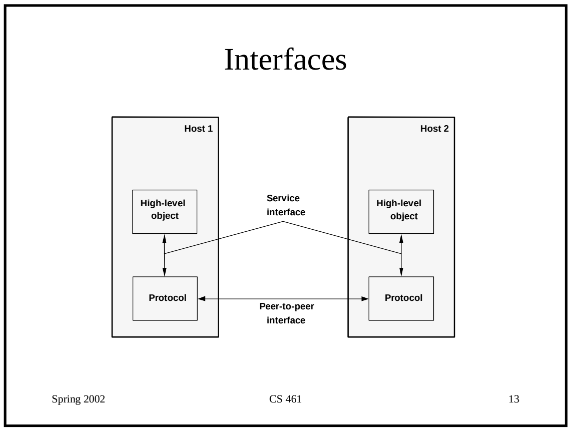
Layering

- Use abstractions to hide complexity
- Abstraction naturally lead to layering
- Alternative abstractions at each layer

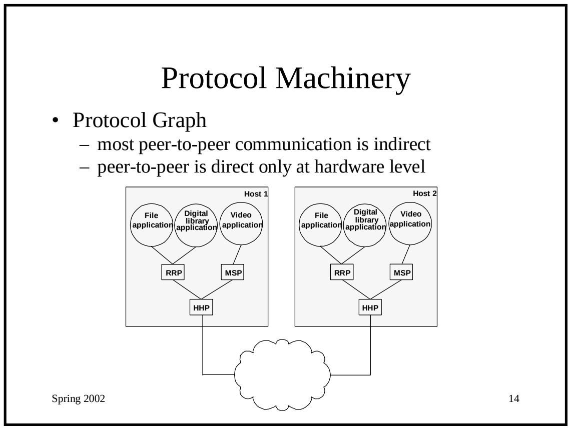


Protocols

- 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

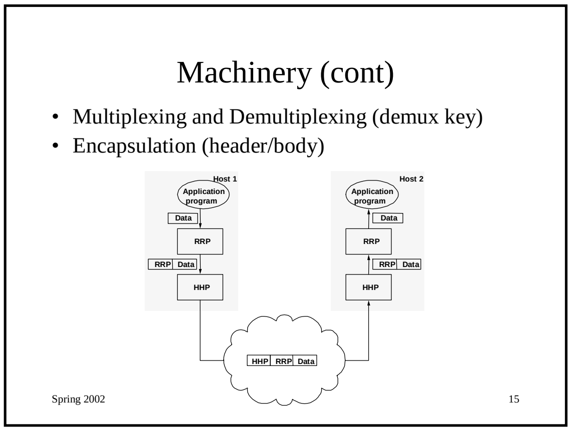


Interfaces



Protocol Machinery

- Protocol Graph
  - most peer-to-peer communication is indirect
  - peer-to-peer is direct only at hardware level

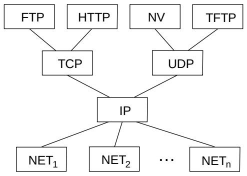


Machinery (cont)

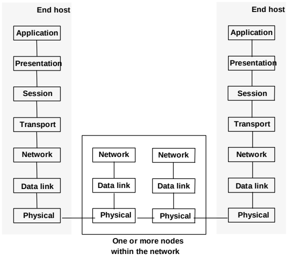
- Multiplexing and Demultiplexing (demux key)
- Encapsulation (header/body)

Internet Architecture

- Defined by Internet Engineering Task Force (IETF)
- Hourglass Design
- Application vs Application Protocol (FTP, HTTP)



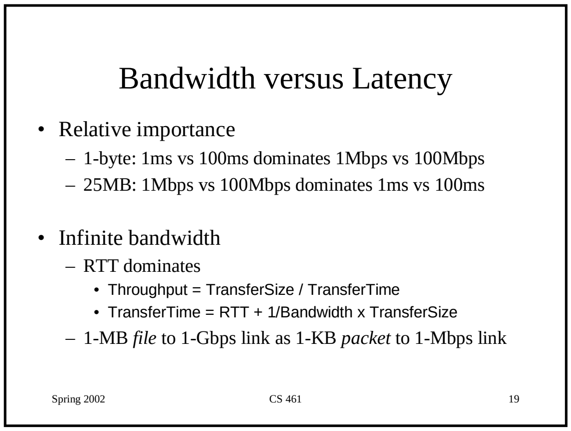
ISO Architecture



Performance Metrics

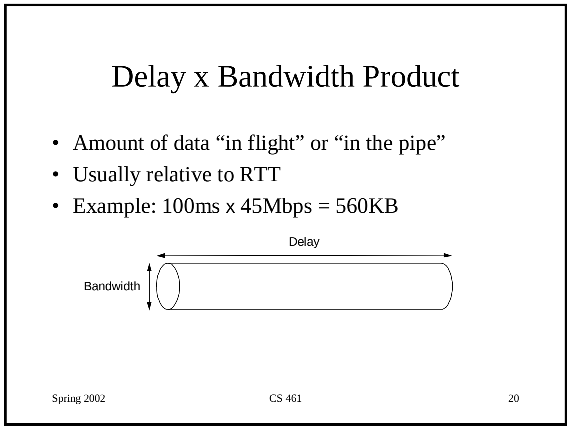
- Bandwidth (throughput)
  - data transmitted per time unit
  - link versus end-to-end
  - notation
    - KB = 2^{10} bytes
    - Mbps = 10^6 bits per second
- Latency (delay)
  - time to send message from point A to point B
  - one-way versus round-trip time (RTT)
  - components

Latency = Propagation + Transmit + Queue
Propagation = Distance / c
Transmit = Size / Bandwidth



Bandwidth versus Latency

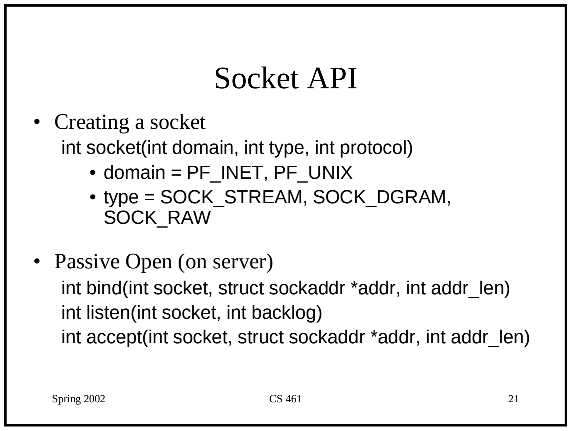
- Relative importance
  - 1-byte: 1ms vs 100ms dominates 1Mbps vs 100Mbps
  - 25MB: 1Mbps vs 100Mbps dominates 1ms vs 100ms
- Infinite bandwidth
  - RTT dominates
    - Throughput = TransferSize / TransferTime
    - TransferTime = RTT + 1/Bandwidth x TransferSize
  - 1-MB *file* to 1-Gbps link as 1-KB *packet* to 1-Mbps link



Delay x Bandwidth Product

- Amount of data “in flight” or “in the pipe”
- Usually relative to RTT
- Example: 100ms x 45Mbps = 560KB

Delay
Bandwidth

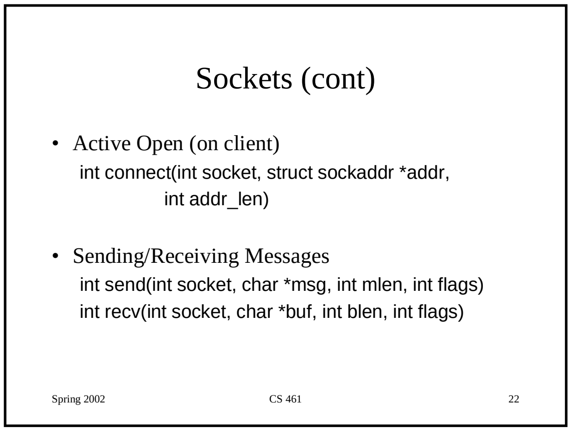


Socket API

- Creating a socket

  int socket(int domain, int type, int protocol)
  - domain = PF_INET, PF_UNIX
  - type = SOCK_STREAM, SOCK_DGRAM, SOCK_RAW
- Passive Open (on server)

  int bind(int socket, struct sockaddr *addr, int addr_len)
  int listen(int socket, int backlog)
  int accept(int socket, struct sockaddr *addr, int addr_len)



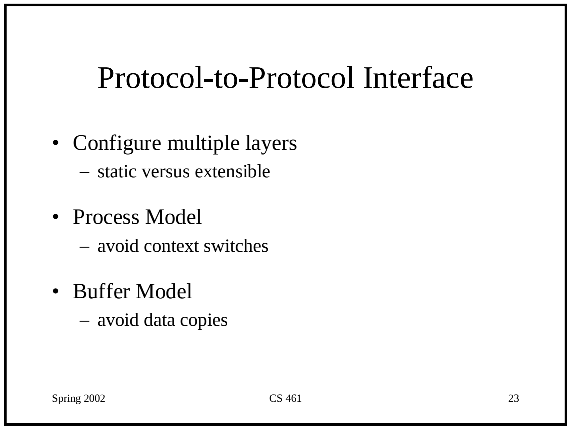
Sockets (cont)

- Active Open (on client)

```
int connect(int socket, struct sockaddr *addr,
            int addr_len)
```

- Sending/Receiving Messages

```
int send(int socket, char *msg, int mlen, int flags)
int recv(int socket, char *buf, int blen, int flags)
```



Protocol-to-Protocol Interface

- Configure multiple layers
  - static versus extensible
- Process Model
  - avoid context switches
- Buffer Model
  - avoid data copies