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*
IPC Abstractions

- Request/Reply
  - distributed file systems
  - digital libraries (web)
  - Based on TCP

- Stream-Based
  - video: sequence of frames
    - $1/4$ NTSC = $352 \times 240$ pixels
    - $(352 \times 240 \times 24)/8=247.5$KB
    - $30$ fps = $7500$KBps = $60$Mbps
  - video applications
    - on-demand video
    - video conferencing
  - Based on UDP

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 (extensible)

<table>
<thead>
<tr>
<th>Application programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request/reply channel</td>
</tr>
<tr>
<td>Host-to-host connectivity</td>
</tr>
<tr>
<td>Hardware</td>
</tr>
</tbody>
</table>

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

• Latency-Bound
  – 1-byte request / reply with 100ms RTT
  – 1Mbps Channel: transmit time 8 µs.
  – 100Mbps Channel: transmit time 0.08 µs.

• Bandwidth-Bound
  – 25MB transfer
  – 10Mbps Channel: transmit time 20 seconds
  – The effect of RTT is negligible.

• Throughput = \( \frac{\text{TransferSize}}{\text{TransferTime}} \)
• \( \text{TransferTime} = \text{RTT} + \frac{1}{\text{Bandwidth}} \times \text{TransferSize} \)

Delay x Bandwidth Product

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

- Creating a socket
  \[\text{int socket}(\text{int domain, int type, int protocol})\]
  \[\begin{align*}
  &\bullet\ \text{domain} = \text{PF_INET, PF_UNIX} \\
  &\bullet\ \text{type} = \text{SOCK_STREAM, SOCK_DGRAM, SOCK_RAW}
\end{align*}\]

- Passive Open (on server)
  \[\begin{align*}
  &\text{int bind}(\text{int socket, struct sockaddr *addr, int addr_len}) \\
  &\text{int listen}(\text{int socket, int backlog}) \\
  &\text{int accept}(\text{int socket, struct sockaddr *addr, int addr_len})
\end{align*}\]

Sockets (cont)

- Active Open (on client)
  \[\text{int connect}(\text{int socket, struct sockaddr *addr, int addr_len})\]

- Sending/Receiving Messages
  \[\begin{align*}
  &\text{int send}(\text{int socket, char *msg, int mlen, int flags}) \\
  &\text{int recv}(\text{int socket, char *buf, int blen, int flags})
\end{align*}\]