Networking Overview

Amin Vahdat
CSE 123B
April 11, 2006
Announcements

• First assignment available
  • HTTP Server: due Apr 21, 5 pm

• Late policy
  • 72 hours of flex time that you can allocate to any programming assignment
Sockets API

- Creating a socket
  
  ```c
  int socket(int domain, int type, int protocol)
  
  domain = AF_INET, AF_UNIX
  
  type = SOCK_STREAM, SOCK_DGRAM
  ```

- Passive Open (on server)
  
  ```c
  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)
  
  int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeeval *timeout);
  ```
Sockets API

- **Active Open (on client)**
  
  ```c
  int connect(int socket, struct sockaddr *addr, int addr_len)
  ```

- **Sending/Receiving Messages**
  
  ```c
  int send(int socket, char *msg, int mlen, int flags)
  int recv(int socket, char *buf, int blen, int flags)
  ```
Server/HTTP Protocol

• HTTP Server
  • Creates a socket (socket)
  • Binds to an address
  • Listens to setup accept backlog
  • Can call accept to block waiting for connections
  • Can call select to check for data on multiple socks

• Requests (hand off to separate thread? separate process?)
  • GET /index.html HTTP/1.0
  <optional body, multiple lines>
  \n  \n
Layering
Layering in Network Design

- Computer System design is often about defining the right set of \textit{abstractions}
  - What services do apps require from the bare hardware? Don’t force each app to re-implement the same functionality
  - Operating systems are all about abstractions
    What set of services do you put on top of the hardware?
  - Network links hand a frame to the operating system?
  - But what abstraction does the application desire?
  - Do all applications need the same abstraction?
  - What abstractions do intermediate hosts (routers) in the network need?
**Layering in Network Design**

- Encapsulate each layer in lower-layer format
- Ethernet’s view of the world

<table>
<thead>
<tr>
<th>Header</th>
<th>Payload</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Eth</th>
<th>IP</th>
<th>TCP</th>
<th>HTTP</th>
<th>Payload</th>
</tr>
</thead>
</table>

- IP View

<table>
<thead>
<tr>
<th>Header</th>
<th>Payload</th>
</tr>
</thead>
</table>

- TCP View

<table>
<thead>
<tr>
<th>Header</th>
<th>Payload</th>
</tr>
</thead>
</table>
Protocols

• Protocol: agreement between two parties as to how information is to be transmitted
  • More valuable with more users
  • How to introduce new protocols if everyone must agree?

• Network protocols typically implemented in software
  • Adds overhead to communication
  • Network bandwidth versus Application throughput
  • Small matter of code to deploy new protocol

• Examples
  • Internet Protocol (IP): global packet transmission/addressing
  • Transmission Control Protocol (TCP): reliable byte stream
OSI Model

**Function**
- Ultimate data destination
- Format conversion
- Interaction across presentation
- Reliable, ordered delivery
- Routing/Internetworking
- Data framing over links
- Bits on the wire

**Example**
- Web browser
- ASCII/XDR
- Restartable file transfer
- TCP
- IP
- Ethernet, ATM
- SONET, 100BT

**Application**

**Presentation**

**Session**

**Transport**

**Network**

**Data link**

**Physical**
## OSI Model

<table>
<thead>
<tr>
<th>Function</th>
<th>Application</th>
<th>Presentation</th>
<th>Session</th>
<th>Transport</th>
<th>Network</th>
<th>Data link</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ultimate data destination</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format conversion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interaction across presentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reliable, ordered delivery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routing/Internetworking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data framing over links</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits on the wire</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Where does security go?**

**What about reliability?**
OSI Model Discussion

• OSI standardized before implemented
  • IETF philosophy: “We reject kings, presidents and voting. We believe in rough consensus and working code”
  • IETF requires two working/interoperable versions before considering a standard

• Modular design, but some boundaries are arbitrary
  • Why seven layers?
  • What exactly is the session layer?
  • Much basic network functionality at multiple layers
    Reliability, flow control, security
Internet Architecture

- IP Hourglass:
  - Telecollaboration
  - NFS
  - HTTP
  - email
  - rlogin
  - RSVP
  - RPC
  - UDP
  - TCP
  - IP
  - Ethernet
  - modem
  - ATM
  - PPP
  - packet radio
  - SONET
  - air
  - 100BT

- Layering not strict
  - Can define new abstractions on any existing protocol
Layering in Internet Applications

- Bottlenecks
  - Boundary crossings
  - Copies
  - Context switches

- Layering nice way to logically consider protocols
  - May not lead to fastest implementation
  - But! Processors are getting faster… people are getting more expensive