Network Programming
TDC 561
Lecture # 1

Dr. Ehab S. Al-Shaer
School of Computer Science & Telecommunication
DePaul University
Chicago, IL

 Goals of this Course:
 STUDYING, EVALUATING AND DEVELOPING ALTERNATIVE CLIENT-SERVER NETWORK ARCHITECTURES

 Perquisites
 Text Books
 Assignments and the course Project
 Rules and Policies
 What do you expect?
 Applications of this course

Applications of Network Programming

 Network Tools
 - traditional: telnet, ftp, rsh, rlogin, SMTP,
 Internet Tools
 - gopher, HTTP, NTP, Chat rooms
 Collaborative Tools
 - Application Sharing, Desktop Conferencing, Distance Learning
 Distributed Object Computing
 - SUN RPC, CORBA, JAVA RMI
 Distributed Databases
IRI: Interactive Distance Learning

MASH: Internet-based Video Conferencing

Network Overview

OSI Seven Layer Model

- Layering (service abstraction)
  - Problem decomposition
  - Modular design
  - Interoperability

- OSI Principles:
  - Different level of abstraction
  - Well-defined functions
  - Supporting open system
  - Well-defined interfaces (layers boundaries)
  - Different functions are in different layers
Network Overview

OSI Seven Layer Model

- OSI Seven Layers
  - Physical Layer
  - Data Link Layer
  - Network Layer
  - Transport Layer
  - Session Layer
  - Presentation Layer
  - Application Layer

- Internet (TCP/IP) Model
  - Application Layer (telnet, ftp)
  - Transport Layer (TCP)
  - Network Layer (IP)
  - Host-to-Network: device drivers of various MAC protocols (Ethernet, token-ring)

TCP/IP Overview

(Transmission Control Protocol/Internet Protocol)

- Internet Protocol (IPv4 and IPv6)
  - IP is the network layer: a packet delivery service (host-to-host).
  - Connectionless: each datagram is independent of all others.
  - Unreliable: there is no guaranteed delivery
  - Fragmentation / Reassembly (based on MTU).
  - Routing.
  - Error detection (link).

IP Address Formats

<table>
<thead>
<tr>
<th>Class</th>
<th>NetID</th>
<th>HostID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>HostID</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td>HostID</td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td>Multicast Address</td>
</tr>
</tbody>
</table>
TCP/IP Overview (cont.)

- **IP Addresses**
  - 32 bit, Logical, unique
  - e.g. condor.cs.depaul.edu is 140.192.1.6

- **IP Packet Format:**

<table>
<thead>
<tr>
<th>VERS</th>
<th>HL</th>
<th>TOS</th>
<th>Fragment Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datagram ID</th>
<th>FLAG</th>
<th>Fragment Offset</th>
<th>TTL</th>
<th>Protocol</th>
<th>Header Checksum</th>
<th>Source Address</th>
<th>Destination Address</th>
<th>Options (if any)</th>
<th>Data</th>
</tr>
</thead>
</table>

ARP (Address Resolution Protocols)

- **HEY** - Everyone please listen! Will 128.213.1.5 please send me his/her Ethernet address

- Hi Red! I'm 128.213.1.5, and my Ethernet address is 87:A2:15:35:02:C3

RARP (Reverse Address Resolution Protocols)

- **HEY** - Everyone please listen!
  - My Ethernet address is 22:BC:66:17:01:75. Does anyone know my IP address?

- Hi Red! Your IP address is 128.213.1.17.
IP is the network protocols, means TCP is the transport protocol, right? Not exactly, TCP is part of the transport protocol, and the other part is UDP (User Datagram Protocol).

Ports
- Abstract destination point
- 16 bit positive integer for UDP and TCP
- Reserved/Well-known ports: 1-1023
- Registered ports: 1024-49151
- Dynamic ports: 49151-65535

TCP supports:
- Connection-oriented (establish, terminate, notify)
- Reliable (ordered, no loose, and no duplicates)
- Buffering (configurable, though)
- Bytel stream
- Full-duplex

UDP supports:
- Connectionless
- Unreliable
- Message stream (Datagram)
- Full-duplex
TCP Segment Format

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number</td>
<td>Request Number</td>
</tr>
<tr>
<td>Offset</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent Pointer</td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

TCP/IP Overview

- UDP Datagram Format

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>Checksum</td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

- Hmmmmm, Which one to use: TCP or UDP?

Client-Server Architecture Design

- Terminology
  - A client initiates communication
  - A server waits for incoming requests
  - Integrated model (e.g. server as client)
  - Does TCP/IP care?

- Client-Server Paradigms
  - Message passing (Synchronous and Asynchronous)
  - Remote Procedure Call (Synchronous and Asynchronous)
Client-Server Architecture Design (cont.)

- Client-Server Communication Models
  - Point-to-point Vs. Multi-point/Group
    - A client to multiple servers
    - A server to multiple clients
  - Broadcast Vs. Multicast
  - Connection-oriented Vs. Connectionless

- Servers Design Issues
  - System Ports Vs. User Ports
  - Statefull Vs. Stateless Servers
  - Privileges and Complexity: authentication, authorization, data sharing, privacy, protection
  - Robustness and Reliability
  - Client or Message Independent
  - Client or Message Independent (dispatching)
  - Concurrency
    - Multi-processes
    - Multi-threaded
    - I/O Multiplexing
    - Asynchronous I/O

Client-Server Architecture Design (cont.)

- Client Design Issues
  - Server Independent (dispatching)
  - Stateless Protocol/Messages (e.g. slide tool)
  - Rich and Flexible Parameters (e.g. telnet)
Network Application Programming Interface (API)

- The services provided by the operating system that provide the interface between application and protocol software.

<table>
<thead>
<tr>
<th>Network API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol A</td>
</tr>
</tbody>
</table>

Network API

- Generic Programming Interface.
- Support for message oriented and connection oriented communication.
- Uses the existing I/O services.
- Operating System independence.
- Support multiple communication protocol suites (families): IPv4, IPv6, XNS, UNIX.
- Provide special services for Client and Server?
TCP/IP

- There are a variety of APIs for use with TCP/IP:
  - Sockets
  - TLI
  - Winsock
  - MacTCP

Functions needed

- Specify local and remote communication endpoints
- Initiate a connection
- Wait for incoming connection
- Send and receive data
- Terminate a connection gracefully
- Error handling

Berkeley Sockets

- Generic:
  - support for multiple protocol families.
  - address representation independence
- Uses existing I/O programming interface as much as possible.
**Unix Descriptor Table**

- Data structure for file 0
- Data structure for file 1
- Data structure for file 2

**Socket**

- A socket is a process-level abstract representation of a communication endpoint.
- Sockets work with Unix I/O services just like files, pipes & FIFOs.
- Sockets (obviously) have special needs:
  - establishing a connection
  - specifying communication endpoint addresses