Outline

- Quiz #3
- Network architecture
- Protocols
- Sockets
- Server Sockets
- Multi-threaded Servers
- Homework #4
Quiz 3

Quiz #3 scores

<table>
<thead>
<tr>
<th>Points</th>
<th># of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=9</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
</tbody>
</table>
Network layers

<table>
<thead>
<tr>
<th>Protocol Layers</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application Protocols and Services</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP</td>
</tr>
<tr>
<td>Internet</td>
<td>RARP IP ARP ICMP Routing Protocols</td>
</tr>
<tr>
<td>Network Interface</td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td>Network Driver and Network Interface Card (NIC)</td>
</tr>
</tbody>
</table>
1. Datagrams enter the packet-switched network in proper sequence order.

2. Datagrams arrive at destination PAD in wrong sequence, which gives new meaning to intended message, and are reordered properly by destination PAD.
Application-layer protocols

- SMTP
- HTTP
- FTP
- many others
  - all rely on services of TCP/IP network
Client/Server Web Interaction
HTTP Protocol

- Request
  - “I want something”

- Response
  - “Here it is”
  - or “Not found”

- Headers

- Body
HTTP Response Example

HTTP/1.1 200 OK
Date: Fri, 25 May 2001 22:51:05 GMT
Server: Apache/1.3.4 (Unix)
Set-Cookie: visit=1102002283923049566;
path=/;
expires=Sat, 26 May 2001 22:51:05 GMT
Last-Modified: Mon, 21 May 2001 14:17:23 GMT
Accept-Ranges:bytes
Content-Length: 2000
Content-Type: text/html

<HTML>
<HEAD>
<TITLE>All About Servers</TITLE>
</HEAD>
<BODY>
...
</BODY>
</HTML>
Need for Protocol

- Client programs must know how to state requests
- Server must know how to interpret
- The protocol is the agreement

Important
- A weak or inflexible protocol limits what can be done
- A complex protocol may be difficult to implement
Example

- Voting application
- Design
  - client applications – allow users to vote
  - server – receives and counts votes
Need a Protocol

- Example
  - Establish credentials
  - Transmit votes
  - Verify
Implementing a protocol

- States
- Messages
  - state transition
  - data transmission
Example

- States
  - unauthenticated
  - authenticated
  - votes complete
  - votes verified
Messages

- **State transition**
  - server: “credentials OK”
  - client: “done with votes”

- **Data transmission**
  - client: “vote for candidate64 in race111”
Sockets

- Bi-directional stream-oriented communication
- Directed towards a host and port
Communicating with sockets

- Server waits at a port
- Client connects to a machine + port combination
- Port number must be agreed upon
  - often conventional (port 80 = http)
- Bi-directional stream established
Java Sockets

- **Socket**
  - abstraction of TCP layer
  - turns network into a pair of data streams
  - address = machine + port

- **Two types**
  - ServerSocket
  - Socket
Client Socket Details

- Creating a socket
  - `s_socket = new Socket (host, 4445);`

- Once the socket exists
  - `s_socket.getOutputStream();`
  - `s_socket.getInputStream();`

- Use
  - when connecting to a host that supplies a service
Server Socket

- Create a listening socket
  - `serverSock = new ServerSocket (port);`

- Listen for clients
  - `Socket sock = serverSock.accept ();`

- Wait for a client connection
  - A new socket is created

- Use to offer a service
Demultiplexing

- Port number
  - Server sockets are created with a port number
  - Connection sockets are assigned port numbers automatically

- TCP “demultiplexes”
  - Based on IP addresses and port numbers at both ends

- Multiple connections to same “port” possible
  - Each result of an “accept()” call is unique
Accept

- Creates a new Socket
  - associated with a different port
- Result
  - server can accept many connections
- How to service them all?
Multi-threaded Servers

- Typical network application
  - > 80% waiting (network, file system)
  - < 20% computation

- Theoretically
  - 5 simultaneous requests
  - No change in response time
Multi-threaded Server

while (listening)
    new KKMultiServerThread(serverSocket.accept()).start();

Or

while (listening)
{
    Socket socket = serverSocket.accept();
    Thread worker = new KKMultiServerThread(socket);
    worker.start();
}
Problem

- Creating and destroying Thread objects
- Better solution
  - Thread pooling
Homework #4

- Networked version of HotelGUIMode
Objectives

- Add server thread to HotelGUIMode to listen for connections
- Add worker threads to execute commands
- Add a Server menu to interface
- Synchronize database access