Reading

Java
Java Network Programming, Harold 3rd Ed,
Chapter 2 - Network Basics
Chapter 4 Streams
Chapter 9 Sockets for Clients

Ruby
Programming Ruby, Thomas, 2'ed
Chapter 10 Basic Input & Output
Class IO documentation (pp 503-515)
IPSocket & TCPSocket in Appendix A
Wikipedia, various articles, explicit references on individual slides
Network Overview

- Messages divided into packets
- Each packet routed separately
- Routing Issues
- Overhead issues
Send Message To Machine A

This is just a sample message that one might send on a network to another machine.
Sending

This is just a sample message that one might send on a network to another machine.

A:1:This is just a sample message that one might send on a network to another machine.

A:2:le message that one

A:3: might send on a ne

A:4:two to another ma

A:5:chine.

Network Cloud

A

B

Tanenbaum – please forgive the gross oversimplification here.
This is just a sample message that one might send on a network to another machine.

Tanenbaum – please forgive the gross oversimplification here.
Issues

How does the message get to A

How does the message get to the correct program on A

How do packets get lost

How do packets get out of order
Routers

Some Useful Programs

netstat
    Show status of network connections on machine
ls/of
    list open files (& pipes & sockets)
traceroute
    Show the route to remote machine
## netstat

Windows, Unix/Linux

Al pro 14->netstat

### Active Internet connections

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>(state)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp4</td>
<td>17680</td>
<td>0</td>
<td>10.0.1.192.60840</td>
<td>kusc-pc-stream2..irdmi</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.0.1.192.60627</td>
<td>208.43.202.32-st.http</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.0.1.192.60623</td>
<td>adsl-68-20-22-55.28205</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>localhost.26164</td>
<td>localhost.60431</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>localhost.60431</td>
<td>localhost.26164</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.0.1.192.afpovertcp</td>
<td>10.0.1.200.53611</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>localhost.26164</td>
<td>localhost.53896</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>localhost.53896</td>
<td>localhost.26164</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>localhost.26164</td>
<td>localhost.51153</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>localhost.51153</td>
<td>localhost.26164</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>localhost.26164</td>
<td>localhost.49164</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>localhost.49164</td>
<td>localhost.26164</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>37</td>
<td>0</td>
<td>10.0.1.192.49163</td>
<td>174.36.30.66-sta.https</td>
<td>CLOSE_WAIT</td>
</tr>
<tr>
<td>tcp4</td>
<td>37</td>
<td>0</td>
<td>10.0.1.192.49162</td>
<td>174.36.30.67-sta.https</td>
<td>CLOSE_WAIT</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.0.1.192.60862</td>
<td>WAREHOUSE-THREE-.55510</td>
<td>TIME_WAIT</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.0.1.192.60861</td>
<td>dhcp128036163075.55165</td>
<td>TIME_WAIT</td>
</tr>
</tbody>
</table>

etc

Al pro 13->netstat -s

tcp:
   1290678 packets sent
      169 data packets (52330 bytes) retransmitted
      686952 ack-only packets (8670 delayed)
   1530460 packets received
      332351 acks (for 114678930 bytes)
      8686 duplicate acks
      760 completely duplicate packets (611003 bytes)
      27337 out-of-order packets (34890076 bytes)
   11654 connection requests
   104 connection accepts
   48 bad connection attempts
   8 listen queue overflows
   etc.
isof
Unix/Linux

list open files
disk files, pipes and network sockets

Air 18->isof -i

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>PID</th>
<th>USER</th>
<th>FD</th>
<th>TYPE</th>
<th>DEVICE</th>
<th>SIZE/OFF</th>
<th>NODE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemUIS</td>
<td>194</td>
<td>whitney</td>
<td>9u</td>
<td>IPv4</td>
<td>0x3da8f40</td>
<td>0t0</td>
<td>UDP</td>
<td><em>:</em></td>
</tr>
<tr>
<td>adb</td>
<td>1141</td>
<td>whitney</td>
<td>5u</td>
<td>IPv4</td>
<td>0x9f28270</td>
<td>0t0</td>
<td>TCP</td>
<td>localhost:5037 (LISTEN)</td>
</tr>
<tr>
<td>Safari</td>
<td>3234</td>
<td>whitney</td>
<td>56u</td>
<td>IPv6</td>
<td>0x3dac19c</td>
<td>0t0</td>
<td>TCP</td>
<td>localhost:59088-&gt;localhost:59087 (TIME_WAIT)</td>
</tr>
<tr>
<td>Safari</td>
<td>3234</td>
<td>whitney</td>
<td>71u</td>
<td>IPv4</td>
<td>0x623366c</td>
<td>0t0</td>
<td>TCP</td>
<td>146.244.205.67:50097-&gt;a198-189-255-145.deploy.akamaitechnolgies.com:http (CLOSE_WAIT)</td>
</tr>
<tr>
<td>Safari</td>
<td>3234</td>
<td>whitney</td>
<td>74u</td>
<td>IPv4</td>
<td>0x61fb270</td>
<td>0t0</td>
<td>TCP</td>
<td>146.244.205.67:50099-&gt;a198-189-255-145.deploy.akamaitechnolgies.com:http (CLOSE_WAIT)</td>
</tr>
</tbody>
</table>

See http://en.wikipedia.org/wiki/Lsof
traceroute

tracepath - Linux
tracert - Windows

Al pro 15->traceroute www.sdsu.edu
traceroute to www.sdsu.edu (130.191.8.198), 64 hops max, 40 byte packets
  1 10.0.1.1 (10.0.1.1)  0.679 ms  0.192 ms  0.174 ms
  2 ip68-8-224-1.sd.sd.cox.net (68.8.224.1)  8.317 ms  6.879 ms  7.574 ms
  3 fed1sysc01-gex0915.sd.sd.cox.net (68.6.10.106) 15.600 ms  8.736 ms  11.449 ms
  4 fed1sysc10-get0005.sd.sd.cox.net (68.6.8.78) 10.456 ms  10.895 ms  8.740 ms
  5 dt1xaggc01-get0701.sd.sd.cox.net (68.6.8.49) 12.298 ms  23.956 ms  7.625 ms
  6 sdscbcsf01-fex0301.cox-sd.net (209.242.135.150) 7.831 ms  8.904 ms  8.057 ms
  7 sdsccbcsf01-fex0301.cox-sd.net (209.242.135.150) 7.831 ms  8.904 ms  8.057 ms
  8 sdsccbcsf01-fex0301.cox-sd.net (209.242.135.150) 7.831 ms  8.904 ms  8.057 ms
  9 dc-sd-csu-egm--sdg-dc1.cenic.net (137.164.41.138) 13.097 ms  13.603 ms  15.632 ms
10  ***
11  ***
How do packets get out of order

different routes
different wait times in router buffers
Internet Protocol (IP or TCP/IP)

Application Layer
DHCP, DNS, FTP, HTTP, SSH, Telnet, (more)

Transport Layer
TCP, UDP, (more)

Internet Layer
IPv4, IPv6, (more)

Link Layer
Ethernet, DSL, ISDN, FDDI, (more)

This chart shows the IP address space on a plane using a fractal mapping which preserves grouping -- any consecutive string of IPs will translate to a single compact, contiguous region on the map. Each of the 256 numbered blocks represents one /8 subnet (containing all IPs that start with that number). The upper left section shows the blocks sold directly to corporations and governments in the 1990s before the RIRs took over allocation.

http://xkcd.com/195/
TCP

Handles lost packets

Handles packet order

TCP has delays
  Starting of connection
  Closing of connection
  Resending packets

Client & Sever don't have to deal with
  Packet order
  Packet loss
## TCP Header

<table>
<thead>
<tr>
<th>Bit offset</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Source port</td>
<td>Destination port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Sequence number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Acknowledgment number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>Data offset</td>
<td>Reserved</td>
<td>CW</td>
<td>EC</td>
<td>URG</td>
<td>ACK</td>
<td>PSH</td>
<td>RST</td>
<td>SYN</td>
<td>FIN</td>
<td>Window Size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>Checksum</td>
<td>Urgent pointer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>Options (if Data Offset &gt; 5)</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## IP Header

<table>
<thead>
<tr>
<th>bit offset</th>
<th>0–3</th>
<th>4–7</th>
<th>8–15</th>
<th>16–18</th>
<th>19–31</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Version</td>
<td>Header length</td>
<td>Differentiated Services</td>
<td>Total Length</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Identification</td>
<td>Flags</td>
<td>Fragment Offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Time to Live</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>Source Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>Destination Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>Options (if Header Length &gt; 5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160 or 192+</td>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Ethernet Frame

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Start-of-Frame-Delimiter</th>
<th>MAC destination</th>
<th>MAC source</th>
<th>802.1Q header (optional)</th>
<th>Ethertype/Length</th>
<th>Payload (Data and padding)</th>
<th>CRC32</th>
<th>Interframe gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 octets of 10101010</td>
<td>1 octet of 10101011</td>
<td>6 octets</td>
<td>6 octets</td>
<td>(4 octets)</td>
<td>2 octets</td>
<td>46–1500 octets</td>
<td>4 octets</td>
<td>12 octets</td>
</tr>
<tr>
<td>64–1522 octets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72–1530 octets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84–1542 octets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://en.wikipedia.org/wiki/Ethernet
UDP

- Fast
- Packets are treated individually
- Packets may arrive out of order
- Packets may be lost
- Client & Server must handle resulting problems

Used by:

- Games
- NFS
## UDP Header

<table>
<thead>
<tr>
<th>bits</th>
<th>0 - 15</th>
<th>16 - 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Source Port</td>
<td>Destination Port</td>
</tr>
<tr>
<td>32</td>
<td>Length</td>
<td>Checksum</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>Data</td>
</tr>
</tbody>
</table>

TCP States

Flow Control

Receiver
   Tells sender how much data it will buffer (Receive window)

Sender
   Sends one buffer full of data
   Waits for acknowledgement & window update
Congestion Control

Slow start
Congestion avoidance
Fast retransmit
Fast recovery

IP Addresses

IP address is currently a 32-bit number

130.191.3.100 (Four 8 bit numbers)

IPv6 uses 128 bit numbers for addresses

105.220.136.100.0.0.0.0.0.0.0.18.128.140.10.255.255

69DC:8864:0:0:1280:8C0A:FFFF

69DC:8864::1280:8C0A:FFFF

Machines on a network need a unique IP address
What is the difference between
MAC address
IP address
Domain Name System (DNS)

Maps machine names to IP addresses

Internet Corporation for Assigned Names and Numbers (ICANN http://www.icann.org/) oversees assigning TLDs

Graphic is from http://en.wikipedia.org/wiki/Domain_name_system
Unix "host" command

Shows mapping between machine names and IP address

->host rohan.sdsu.edu
rohan.sdsu.edu has address 130.191.3.100

->host 130.191.3.100
100.3.191.130.IN-ADDR.ARPA domain name pointer rohan.sdsu.edu
Ports

TCP/IP supports multiple logical communication channels called ports

Ports are numbered from 0 - 65535

A connection between two machines is uniquely defined by:

- Protocol (TCP or UDP)
- IP address of local machine
- Port number used on the local machine
- IP address of remote machine
- Port number used on the remote machine
When a client connects to a server, it has to specify a machine and a port. The OS on the server keeps a table of port numbers and applications (sockets from the program) associated with each port number. When a client request comes in, the OS will forward the request to the socket associated with the port number if one is associated (connected) with the port. A similar thing happens on the client side. When you open a socket on the client to connect to the server, the client socket is assigned a port on the client machine. When the server responds to the client, it sends the response to that port on the client machine.
## Some Port Numbers

<table>
<thead>
<tr>
<th>Well known Ports</th>
<th>1-1023</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registered Ports</td>
<td>1024-49151</td>
</tr>
<tr>
<td>Dynamic/Private Ports</td>
<td>49152-65535</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>echo</td>
<td>7</td>
</tr>
<tr>
<td>discard</td>
<td>9</td>
</tr>
<tr>
<td>ftp</td>
<td>21</td>
</tr>
<tr>
<td>ssh</td>
<td>22</td>
</tr>
<tr>
<td>telnet</td>
<td>23</td>
</tr>
<tr>
<td>smtp</td>
<td>25</td>
</tr>
<tr>
<td>time</td>
<td>37</td>
</tr>
<tr>
<td>http</td>
<td>80</td>
</tr>
<tr>
<td>pop</td>
<td>110</td>
</tr>
<tr>
<td>https</td>
<td>443</td>
</tr>
<tr>
<td>doom</td>
<td>666</td>
</tr>
<tr>
<td>mysql</td>
<td>3306</td>
</tr>
<tr>
<td>postgresql</td>
<td>5432</td>
</tr>
<tr>
<td>gnutella</td>
<td>6346 6347</td>
</tr>
</tbody>
</table>

For a local list of services
file://rohan.sdsu.edu/etc/services

For a complete list see:
http://www.iana.org/assignments/port-numbers

See IANA numbers page http://www.iana.org/numbers.html for more information about protocol numbers and assignment of services
What is Telnet?

Protocol
Send text between client & server

Server
Requests login
Sends text to shell to be executed
Returns result of commands

Client
Transfers text between user and server
Telnet & Other Text-based Protocols

rohan 37 -> telnet www.eli.sdsu.edu 80
GET /courses/spring06/cs580/index.html HTTP/1.0 <CR><CR>

Note <CR> indicates were you need to hit return

rohan 38->telnet cs.sdsu.edu 110
Trying 130.191.226.116...
Connected to cs.sdsu.edu.
Escape character is '^]'.
+OK QPOP (version 3.1.2) at sciences.sdsu.edu starting.

USER whitney
+OK Password required for whitney.
PASS typeYourPasswordHere
+OK whitney has 116 visible messages (0 hidden) in 640516 octets.
**Simple Date Example - Protocol**

<table>
<thead>
<tr>
<th>Client Commands</th>
<th>Server Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;date&quot; ended by line feed &quot;date\n&quot;</td>
<td>current date ended by line feed &quot;January 30, 2007\n&quot;</td>
</tr>
<tr>
<td>&quot;time&quot; ended by line feed &quot;time\n&quot;</td>
<td>Current time ended by line feed &quot;6:58 pm\n&quot;</td>
</tr>
</tbody>
</table>

Server listens for an incoming request

On request
  - reads command
  - returns response
  - closes connection

On client errors - action not specified
Beware

Can only send bytes across network

Client & server maybe different hardware platforms

What is a newline?

End-of-file indicates connection is closed
import java.io.*;
import java.net.Socket;

class DateClient {
    String server;
    int port;

    public DateClient(String serverAddress, int port) {
        server = serverAddress;
        this.port = port;
    }

    public String date() {
        return send("date\n");
    }

    public String time() {
        return send("time\n");
    }
}
private String send(String text) {
    try {
        Socket connection = new Socket(server, port);
        OutputStream rawOut = connection.getOutputStream();
        PrintStream out = new PrintStream(new BufferedOutputStream(rawOut));
        InputStream rawIn = connection.getInputStream();
        BufferedReader in = new BufferedReader(new InputStreamReader(rawIn));

        out.print(text);
        out.flush();
        String answer = in.readLine();
        out.close();
        in.close();
        return answer;
    }
    catch (IOException e) {
        return "Error in connecting to server";
    }
}

Bad very bad – using PrintStream for network code.
Running the Client

System.out.println("hi");
DateClient client = new DateClient("127.0.0.1", 4444);
System.out.println( client.date());
System.out.println( client.time());
Issue - Avoid Small Packets

```java
OutputStream rawOut = connection.getOutputStream();
PrintStream out = new PrintStream(new BufferedOutputStream(rawOut));
```
Issue - Actually Send the request

out.flush();
Issue - Client will not work on all platforms

String answer = in.readLine();
Don't Do this

String answer = in.readLine();

I did it to keep the example small. One can not get much code on a slide using 24 point font. Plus the Ruby example is sorter than this.
Issue - Close the connection when done

    out.close();
    in.close();
Issue - Testing

How does one test the client?
Issue - Background material

Java
Streams
Read Chapter 4

Sockets
Read Chapter 10

Java Network Programming, Harold 3rd Ed
require 'socket'

class DateClient
  def initialize(serverAddress, port)
    @server = serverAddress
    @port = port
  end

  def date()
    send("date\n")
  end

  def time()
    send("time\n")
  end

  private
  def send(text)
    connection = TCPSocket.new(@server, @port)
    connection.send(text, 0)
    answer = connection.gets("\n")
    connection.close
    answer
  end
end

Running the client

client = DateClient.new("127.0.0.1", 4444)
puts client.date
puts client.time
def send(text)
    connection = TCPSocket.new(@server, @port)
    connection.print(text)
    connection.flush
    answer = connection.gets("\n")
    connection.close
    answer
end
Ruby Background

Sockets

Read IPSocket & TCPSocket in Appendix A

IO

Chapter 10 Basic Input & Output
Class IO documentation (pp 503-515)

Programming Ruby, Thomas, 2'ed
Server

Basic Algorithm

while (true) {
    Wait for an incoming request;
    Perform whatever actions are requested;
}
Basic Server Issues

How to wait for an incoming request?
How to know when there is a request?
What happens when there are multiple requests?
How do clients know how to contact server?
How to parse client request?
How do we know when the server has the entire request?
public class DateServer {
    private static Logger log = Logger.getLogger("dateLogger");

    public static void main (String args[]) throws IOException {
        ProgramProperties flags = new ProgramProperties( args);
        int port = flags.getInt( "port" , 8765);
        new DateServer().run(port);
    }

    public void run(int port) throws IOException {
        ServerSocket input = new ServerSocket( port);
        log.info("Server running on port " + input.getLocalPort());

        while (true) {
            Socket client = input.accept();
            log.info("Request from " + client.getInetAddress());
            processRequest( client.getInputStream(),
                            client.getOutputStream());
            client.close();
        }
    }
}
Java Date Server Continued

```java
void processRequest(InputStream in, OutputStream out)
    throws IOException {

    BufferedReader parsedInput =
        new BufferedReader(new InputStreamReader(in));

    boolean autoflushOn = true;
    PrintWriter parsedOutput = new PrintWriter(out, autoflushOn);

    String inputLine = parsedInput.readLine();

    if (inputLine.startsWith("date")) {
        Date now = new Date();
        parsedOutput.println(now.toString());
    }
}
```

This server needs work
rohan 16-> java -jar DateServer.jar
Feb 19, 2004 10:56:59 AM DateServer run
INFO: Server running on port 8765
require 'socket'

class DateServer
  def initialize(port)
    @port = port
  end

  def run()
    server = TCPServer.new( @port )
    puts("start " + @port.to_s )
    while (session = server.accept) 
      Thread.new(session) do |connection|
        process_request_on(connection)
        connection.close
      end
    end
  end

  private
  def process_request_on(socket)
    request = canonical_form( socket.gets("\n") )
    now = Time.now
    answer = case request
      when 'time'
        now.strftime("%X")
      when 'date'
        now.strftime("%x")
      else
        "Invalid request"
    end
    socket.send(answer + "\n",0)
  end

  def canonical_form(string)
    string.lstrip.rstrip.downcase
  end
end
Issue - Date Format

What format does the server use for time and date?

Clients need to know so can parse them