Computer Networks

The Socket API (Project 1) & Traceroute (HW 1)

Originally By David Wetherall (djw@)
Modified By Qian Yan (qiany7@), Jeff Zhao (zhaox27@), Dao Yi (daoyee@)
Outline

- Administrivia
- Project 1 and Socket API
- Traceroute
Administrivia

- Sections will be recorded

- HW1 & Traceroute Experiment due tomorrow

- HW2 due Jan 21st (next Friday)

- Project 1 due on Jan 25th (the Monday after next)
Network-Application Interface

- Defines how apps use the network
  - Application Layer APIs
  - Lets apps talk to each other
  - Hides the other layers of the network
Project 1

- Simple Client
  - Send requests to attached server
  - Wait for a reply
  - Extract the information from the reply
  - Continue...

- Simple Server
  - Server handles the Client requests
  - Multi-threaded
Project 1

- This is the basis for many apps!
  - File transfer: send name, get file
  - Web browsing: send URL, get page
  - Echo: send message, get it back

- Let’s see how to write this app ...
Socket API (Generalized)

- Simple application-layer abstractions (APIs) to use the network
  - The network service API used to write all Internet applications
  - Part of all major OSes and languages; originally Berkeley (Unix) ~1983

- Two kinds of sockets
  - Streams (TCP): reliably send a stream of bytes
  - Datagrams (UDP): unreliably send separate messages
Socket API (2)

- Sockets let apps attach to the local network at different ports
- Ports are used by OS to distinguish services/apps using internet
## Socket API (3)

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKET</td>
<td>Create a new communication endpoint</td>
</tr>
<tr>
<td>BIND</td>
<td>Associate a local address (port) with a socket</td>
</tr>
<tr>
<td>LISTEN</td>
<td>Announce willingness to accept connections; (give queue size)</td>
</tr>
<tr>
<td>ACCEPT</td>
<td>Passively establish an incoming connection</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>SEND</td>
<td>Send some data over the connection</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Receive some data from the connection</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Release the connection</td>
</tr>
</tbody>
</table>

[https://docs.oracle.com/javase/8/docs/api/java/net/Socket.html](https://docs.oracle.com/javase/8/docs/api/java/net/Socket.html)
[https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html](https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html)
Using TCP Sockets (2)

Client (host 1) → Time Server (host 2)

1. Connect
2. Request
3. Reply
4. Disconnect

Computer Networks
Using TCP Sockets (3)

Client (host 1)  Time Server (host 2)

1: socket
5: connect*  
7: send
8: recv*
11: close

connect
request
reply
disconnect

1: socket
2: (bind)
3: (listen)
4: accept*
6: recv*
9: send
10: recv*
12: close

*= call blocks
Using UDP Sockets

Client (host 1)  Time Server (host 2)

1: socket
5: connect*  
7: sendto
8: recvfrom*  
11: close

1: socket
2: (bind)
3: (listen)
4: accept*  
6: recvfrom*  
9: sendto  
10: recvfrom*  
12: close

* = call blocks

Computer Networks
Client Program (outline)

socket()  // make socket
getaddrinfo()  // server and port name
    // www.example.com:80
connect()  // connect to server [block]
...
send()  // send request
recv()  // await reply [block]
...  // do something with data!
close()  // done, disconnect
Server Program (outline)

socket()  // make socket
getaddrinfo() // for port on this host
bind() // associate port with socket
listen() // prepare to accept connections
accept() // wait for a connection [block]
...
recv() // wait for request [block]
...
send() // send the reply
close() // eventually disconnect
Java Examples with Socket & ServerSocket

**Server**

```java
ServerSocket listener = new ServerSocket(9090);
try {
    while (true) {
        Socket socket = listener.accept();
        try {
            socket.getInputStream();
        } finally {
            socket.close();
        }
    }
} finally {
    listener.close();
}
```

**Client**

```java
Socket socket = new Socket(server, 9090);
out = new PrintWriter(socket.getOutputStream(), true);
socket.close();
```

- [http://cs.lmu.edu/~ray/notes/javanetexamples/](http://cs.lmu.edu/~ray/notes/javanetexamples/)
- [https://docs.oracle.com/javase/tutorial/networking/datagrams/clientServer.html](https://docs.oracle.com/javase/tutorial/networking/datagrams/clientServer.html)
- [https://docs.oracle.com/javase/tutorial/networking/sockets/index.html](https://docs.oracle.com/javase/tutorial/networking/sockets/index.html)
**Traceroute**

- Apps talk to other apps with no real idea of what is inside the network
  - This is good! But you may be curious ...

- Peeking inside the Network with Traceroute
Traceroute

- Widely used command-line tool to let hosts peek inside the network
  - On all OSes (tracert on Windows)
  - Developed by Van Jacobson ~1987
  - Uses a network-network interface (IP) in ways we will explain later

Van Jacobson

Credit: Wikipedia (public domain)
Traceroute

- Probes successive hops to find network path
- Core mechanism: Time-To-Live (TTL)
  - TTL == 0?
    - Discard data, error (ICMP) report to sender
    - Continue with TTL-1
Traceroute

Local Host

Remote Host

1 hop 2 hops 3 hops N-1 hops N hops
Using Traceroute

Tracing route to www.washington.edu [128.95.155.134] over a maximum of 30 hops:

1 1 ms <1 ms 2 ms 192.168.1.1
2 8 ms 8 ms 9 ms 88.Red-80-58-67.staticIP.rima-tde.net [80.58.67.88]
3 16 ms 5 ms 11 ms 169.Red-80-58-78.staticIP.rima-tde.net [80.58.78.169]
4 12 ms 12 ms 13 ms 217.Red-80-58-87.staticIP.rima-tde.net [80.58.87.217]
5 5 ms 11 ms 6 ms et-1-0-0-1-101-GRIBCMES1.red.telefonica-wholesale.net [94.142.182.20]
51 40 ms 38 ms 38 ms 176.52.250.226
7 108 ms 106 ms 136 ms xe-6-0-2-0-grtnycpt2.red.telefonica-wholesale.net [213.140.43.9]
8 180 ms 179 ms 182 ms xe9-2-0-0-grtpacpx2.red.telefonica-wholesale.net [94.142.118.178]
9 178 ms 175 ms 176 ms te-4-2.car1.SanJose2.Level3.net [4.59.0.225]
10 190 ms 186 ms 187 ms vlan80.csu3.SanJose1.Level3.net [4.69.152.190]
11 185 ms 185 ms 187 ms ae-82-82.ebr2.SanJose1.Level3.net [4.69.153.25]
12 268 ms 205 ms 207 ms ae-7-7.ebr1.Seattle1.Level3.net [4.69.132.50]
13 334 ms 202 ms 195 ms ae-12-51.car2.Seattle1.Level3.net [4.69.147.132]
14 195 ms 196 ms 195 ms PACIFIC-NOR.car2.Seattle1.Level3.net [4.53.146.142]
15 197 ms 195 ms 196 ms ae0-=4000.iccr-sttlva01-02.infra.pnw-gigapop.net [209.124.188.132]
15 196 ms 196 ms 195 ms v14000.wubr-ads-01.infra.washington.edu [209.124.188.133]
17 * Request timed out.
18 201 ms 194 ms 196 ms ae4-=583.wuar-ads-1.infra.washington.edu [128.95.155.13]
19 197 ms 196 ms 195 ms www1.cac.washington.edu [128.95.155.134]

Trace complete.
Using Traceroute.

<table>
<thead>
<tr>
<th>Hop</th>
<th>RTT 1</th>
<th>RTT 2</th>
<th>RTT 3</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 ms</td>
<td>&lt;1 ms</td>
<td>2 ms</td>
<td>192.168.1.1</td>
</tr>
<tr>
<td>2</td>
<td>8 ms</td>
<td>8 ms</td>
<td>9 ms</td>
<td>88.Red-80-58-67.staticIP.rima-tde.net [80.58.67.88]</td>
</tr>
<tr>
<td>3</td>
<td>16 ms</td>
<td>5 ms</td>
<td>11 ms</td>
<td>169.Red-80-58-78.staticIP.rima-tde.net [80.58.78.169]</td>
</tr>
<tr>
<td>4</td>
<td>12 ms</td>
<td>12 ms</td>
<td>13 ms</td>
<td>217.Red-80-58-87.staticIP.rima-tde.net [80.58.87.217]</td>
</tr>
<tr>
<td>5</td>
<td>5 ms</td>
<td>11 ms</td>
<td>6 ms</td>
<td>et-1-0-0-1-101-GRIBCMES1.red.telefonica-wholesale.net [94.142.103.20]</td>
</tr>
<tr>
<td>6</td>
<td>40 ms</td>
<td>38 ms</td>
<td>38 ms</td>
<td>176.52.250.226</td>
</tr>
<tr>
<td>7</td>
<td>108 ms</td>
<td>106 ms</td>
<td>136 ms</td>
<td>xe-6-0-2-0-grtnycpt2.red.telefonica-wholesale.net [213.140.43.9]</td>
</tr>
<tr>
<td>8</td>
<td>180 ms</td>
<td>179 ms</td>
<td>182 ms</td>
<td>xe9-2-0-0-grtpaopx2.red.telefonica-wholesale.net [94.142.118.178]</td>
</tr>
<tr>
<td>9</td>
<td>178 ms</td>
<td>175 ms</td>
<td>176 ms</td>
<td>te-4-2.car1.SanJose2.Level3.net [4.59.0.225]</td>
</tr>
<tr>
<td>10</td>
<td>180 ms</td>
<td>186 ms</td>
<td>187 ms</td>
<td>vlan80.cs3.SanJose1.Level3.net [4.69.152.190]</td>
</tr>
<tr>
<td>11</td>
<td>185 ms</td>
<td>185 ms</td>
<td>187 ms</td>
<td>ae-82-82.ebr2.SanJose1.Level3.net [4.69.153.25]</td>
</tr>
<tr>
<td>12</td>
<td>268 ms</td>
<td>205 ms</td>
<td>207 ms</td>
<td>ae-7-7.ebr1.Seattle1.Level3.net [4.69.132.50]</td>
</tr>
<tr>
<td>13</td>
<td>334 ms</td>
<td>202 ms</td>
<td>195 ms</td>
<td>ae-12-51.car2.Seattle1.Level3.net [4.69.147.132]</td>
</tr>
<tr>
<td>14</td>
<td>195 ms</td>
<td>195 ms</td>
<td>195 ms</td>
<td>PACIFIC-NOR.car2.Seattle1.Level3.net [4.53.146.142]</td>
</tr>
<tr>
<td>15</td>
<td>197 ms</td>
<td>195 ms</td>
<td>196 ms</td>
<td>ae0--4000.iccr-sttlva01-02.infra.pwn-gigapop.net [209.124.188.132]</td>
</tr>
<tr>
<td>16</td>
<td>196 ms</td>
<td>196 ms</td>
<td>195 ms</td>
<td>v14000.uwbr-ads-01.infra.washington.edu [209.124.188.133]</td>
</tr>
<tr>
<td>17</td>
<td>Request timed out.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>201 ms</td>
<td>194 ms</td>
<td>196 ms</td>
<td>ae4--583.uwar-ads-1.infra.washington.edu [128.95.155.131]</td>
</tr>
<tr>
<td>19</td>
<td>197 ms</td>
<td>196 ms</td>
<td>195 ms</td>
<td>www1.cac.washington.edu [128.95.155.134]</td>
</tr>
</tbody>
</table>
Using Traceroute

Router settings affect results
Using Traceroute (2)

- ISP names and places are educated guesses

![Diagram showing network route with times and hops]
END

© 2013 D. Wetherall