Computer Networks

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

(§1.3.4, 6.1.2-6.1.4)

Originally By David Wetherall (djw@), Modified By Rithik Duggempudi(rrd3@)
I’m Rithik!

Senior in CSE

I love distributed systems, network programming and computer security

This is my 3rd quarter as a TA
I’m Daniel!

Final year in BS/MS

I too love security, systems, and network programming

This is my 3rd quarter as a TA
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 attu 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 (§6.1.4)
  - 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>

Computer Networks

https://docs.oracle.com/javase/8/docs/api/java/net/Socket.html
https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html
Using Sockets

Client (host 1)  Time  Server (host 2)
Using Sockets (2)

Client (host 1)  Time  Server (host 2)

1 ← connect 1
2  request 3
3 ← reply
4 ← disconnect 4
Using Sockets (3)

Client (host 1)  \( \rightarrow \)  Server (host 2)

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

connect
request
reply
disconnect

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

Time

request
reply

*= call blocks
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]
...  // wait for request
recv()  // send the reply
send()  // send the reply
close()  // eventually disconnect
Java Examples with Socket & ServerSocket

► Server

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

► Client

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/net/working/datagrams/clientServer.html](https://docs.oracle.com/javase/tutorial/net/working/datagrams/clientServer.html)
- [https://docs.oracle.com/javase/tutorial/net/working/sockets/index.html](https://docs.oracle.com/javase/tutorial/net/working/sockets/index.html)
Questions?
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
Traceroute

- Probes successive hops to find network path
- TTL: time-to-live
Traceroute

Local Host

1 hop

2 hops

3 hops

N-1 hops

N hops

Remote Host
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-GRBTCNES1.red.telefonica-wholesale.net [94.142.103.20]
6  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-grtpapx2.red.telefonica-wholesale.net [94.142.118.178]
9 178 ms 176 ms 176 ms te-4-2.car1.SanJose2.Level13.net [4.59.0.225]
10 190 ms 186 ms 187 ms vian80.csu3.SanJose1.Level13.net [4.69.152.190]
11 185 ms 185 ms 187 ms ae-82-82.ebr2.SanJose1.Level13.net [4.69.153.25]
12 268 ms 205 ms 207 ms ae-2-7.ebr1.Seattle1.Level13.net [4.69.132.50]
13 334 ms 202 ms 195 ms ae-12-51.car2.Seattle1.Level13.net [4.69.147.132]
14 195 ms 196 ms 195 ms PACIFIC-NOR.car2.Seattle1.Level13.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]
16 196 ms 196 ms 195 ms v14000.uwbr-ads-01.infra.washington.edu [209.124.188.133]
17  *   *   * Request timed out.
18 201 ms 194 ms 196 ms ae4--583.uwar-ads-1.infra.washington.edu [128.95.155.131]
19 197 ms 196 ms 195 ms www1.cac washington.edu [128.95.155.134]
Using Traceroute (2)

- ISP names and places are educated guesses

![Diagram showing traceroute paths and delays]

- Home to tde: 1 hop, 3 hops
- tde to Telefonica: 4 hops
- Telefonica to Level3: 6 hops
- Level3 to pnw-gigapop: 1 hop, 3 hops
- pnw-gigapop to UW: 1 hop, 3 hops

- Delays:
  - Home to tde: 100 ms
  - tde to Telefonica: 180 ms
  - Telefonica to Level3: >200 ms
  - Level3 to pnw-gigapop: 1 hop, 3 hops
  - pnw-gigapop to UW: 1 hop, 3 hops

- Destinations:
  - www.uw.edu
  - (www1.cac.washington.edu)
END