Computer Networks

The Socket API, DNS Lookup & more

Originally By David Wetherall (djw@), Modified By Qian Yan (qiany7@)
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
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/ServerSocket.html
Using Sockets

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

Client (host 1)  Time  Server (host 2)

\[
\begin{array}{c}
1 \leftarrow \text{connect} \rightarrow 1 \\
2 \rightarrow \text{request} \\
\text{reply} \leftarrow 3 \\
4 \leftarrow \text{disconnect} \rightarrow 4
\end{array}
\]
Using Sockets (3)

Client (host 1)  Time  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

*= call blocks
Client Program (outline)

```c
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
...
send() // send the reply
close() // eventually disconnect
Blocking, Non-blocking calls

Socket calls `recv()` and `send()` can be blocking/nonblocking.

Default: blocking, can be changed with `fcntl()` modifies the file descriptor.

- [https://www.scottklement.com/rpg/socktut/nonblocking.html](https://www.scottklement.com/rpg/socktut/nonblocking.html)
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)
Dig & DNS

- Recursively query local/ISP DNS
- Local DNS non-recursively query from top-down
- Use dig to trace the process
- Reverse DNS lookup
- IP -> server domain

- [https://www.golinuxhub.com/2014/01/how-does-dns-query-works-when-you-type.html](https://www.golinuxhub.com/2014/01/how-does-dns-query-works-when-you-type.html)
ping & nmap

- Ping uses ICMP protocol which is on top of Network layer.
- nmap
  - Send TCP/UDP packet to specific host and port and examine the response

- https://en.wikibooks.org/wiki/Communication_Networks/Ping
- https://resources.infosecinstitute.com/nmap/
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-00-58-67.staticIP.rima-tde.net [80.58.67.80]
3  16 ms  5 ms  11 ms  169.Red-00-58-78.staticIP.rima-tde.net [80.58.78.169]
4  12 ms  12 ms  13 ms  217.Red-00-58-87.staticIP.rima-tde.net [80.58.87.217]
5  5 ms  11 ms  6 ms  et-1-0-1-101-GRIBONES1.red.telefonica-wholesale.net [94.142.100.20]
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  xe0-2-0-0-grtpaopx2.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.csw3.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  195 ms  195 ms  PACIFIC-NOR.car2.Seattle1.Level3.net [4.53.146.142]
15  197 ms  195 ms  196 ms  ae0--4000.iccr-sttlwa01-02.infra.pn-w-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]
Trace complete.
```
Using Traceroute (2)

ISP names and places are educated guesses

```
<table>
<thead>
<tr>
<th>Location</th>
<th>Hop Count</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>My computer</td>
<td>1 hop</td>
<td>100 ms</td>
</tr>
<tr>
<td>NYC</td>
<td>3 hops</td>
<td>180 ms</td>
</tr>
<tr>
<td>San Jose</td>
<td>4 hops</td>
<td></td>
</tr>
<tr>
<td>Level3</td>
<td>6 hops</td>
<td></td>
</tr>
<tr>
<td>pnw-gigapop</td>
<td>1 hop</td>
<td></td>
</tr>
<tr>
<td>UW</td>
<td>3 hops</td>
<td>&gt;200 ms</td>
</tr>
<tr>
<td><a href="http://www.uw.edu">www.uw.edu</a> (www1.cac.washington.edu)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```