Sockets

CPSC 1181 – O.O.

Jeremy Hilliker

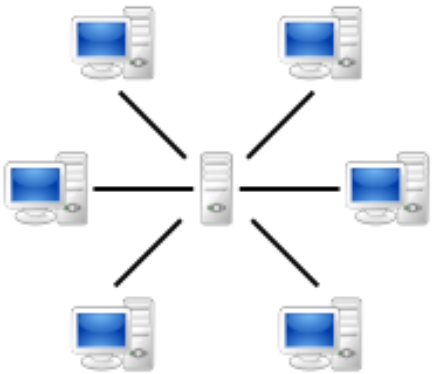
Summer 2017

Overview

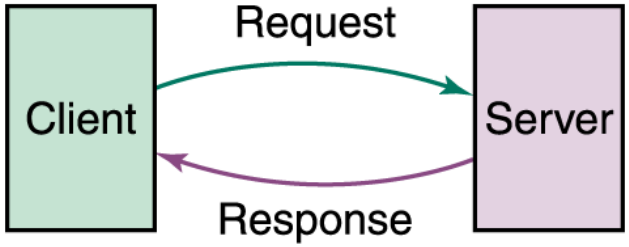
- Models
  - Client/Server
  - Peer-To-Peer
- Examples:
  - TCP Client/Server
  - UDP Client/Server
- Sockets
  - Socket (Regular)
    - w/ 2 Streams
  - ServerSocket
    - Accepts connections
    - Produces Socket
  - DatagramSocket
    - Connectionless
- Marshalling

Client/Server Model

- A centralized application architecture, where:
  - Some central host has some resources
    - It makes them available
  - Some client wants those resources
    - It accesses them
  - Servers are providers
  - Clients are consumers

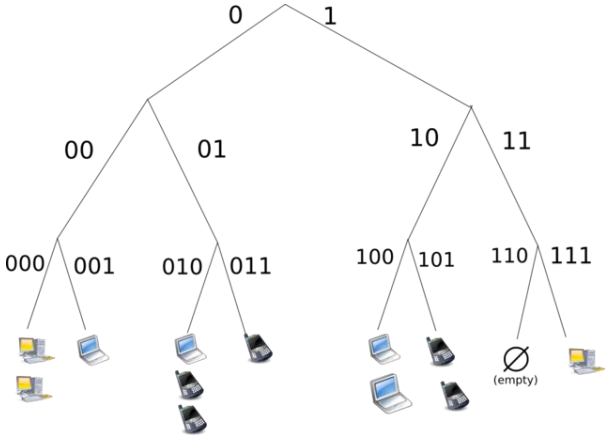


- Sequence:
  - Client ***connects**** to server
  - Repeat:
    - The ***client*** sends a ***request*** to the ***server***
    - The ***server*** processes the request
    - The server sends a ***response***
  - Either side ***disconnects***



Peer-To-Peer (P2P)

- A <u>distributed</u> application architecture, where:
  - Peers are equally privileged
  - No central authority
  - Peers make a portion of their resources available to the others
  - Peers are both providers and consumers of resources
- Most are enabled by a “distributed hash table”
  - Organizes resource sharing



Socket

- Def'n: a ***socket***
  - is a session layer abstraction representing one internal endpoint of a two-way communication link between two programs running on a network
- Sockets have
  - An address
  - Underlying protocol stack
  - Usually (though not necessarily) a connection
- Handled by TCP or UDP
  - (session and transport layers)

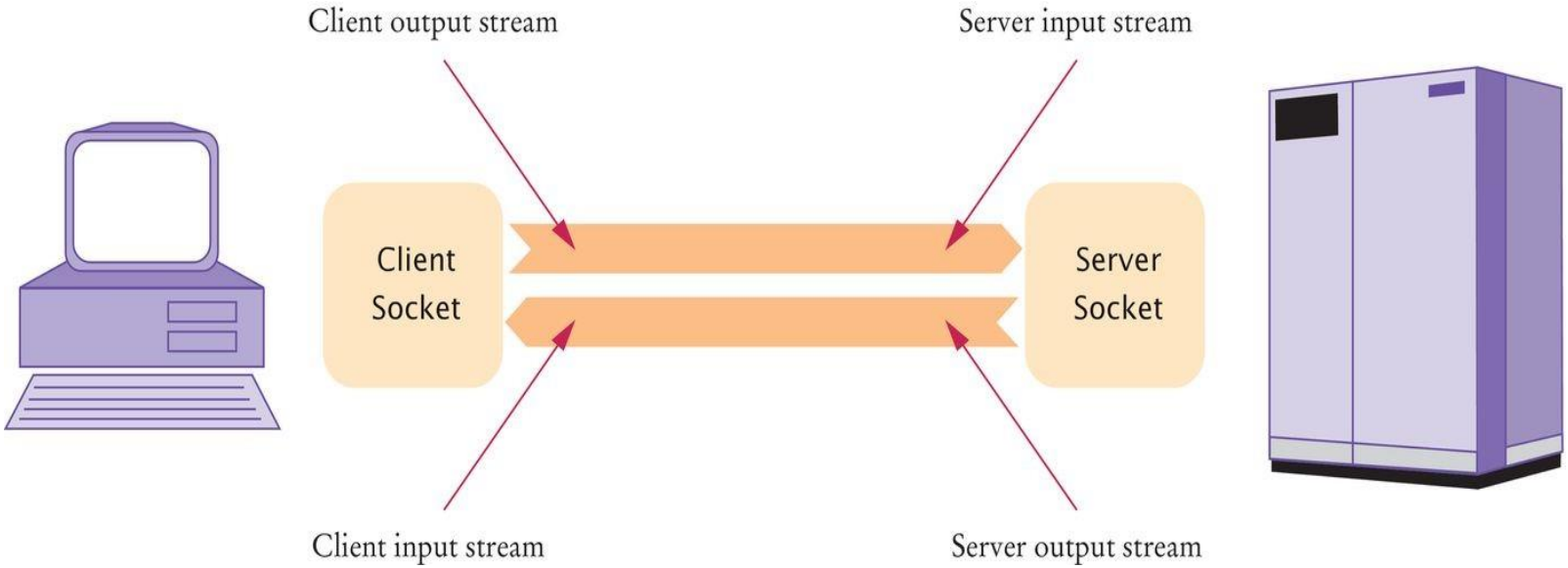
Socket

- “Regular” Socket
  - For sending / receiving data
- Server Socket
  - Waits for and accept incoming connection* requests
  - Produces a regular Socket

Stream

- Def'n: a ***stream***
  - is a <u>sequence</u> of elements (data / bytes) made available over time
- Conceptually:
  - <u>Continuous</u> stream of data (like a river)
  - One way
  - No delineation of when one set of data begins or ends
    - Except for when the entire stream ends
- Regular sockets will have two streams
  - (input and output)

Sockets and Streams



Marshalling

- Def'n: Marshalling is
  - The process of transforming the in-memory representation of data into a common format suitable for transmission or storage
  - Not all computer store things in memory the same way
    - Order of bytes: Big-Endian vs Little-Endian
    - Encoding of characters: ASCII vs UTF-8 vs UTF-16
- Part of the presentation layer
- In the examples, we used "DataOutputStream" and "DataInputStream" to do our Marshalling and unmarshalling

Marshalling

- In general:
  - Use Streams for binary data
  - Wrap them in Reader / Writer for string / character data
- Use higher level abstractions where appropriate
  - REST

TCP Client / Server

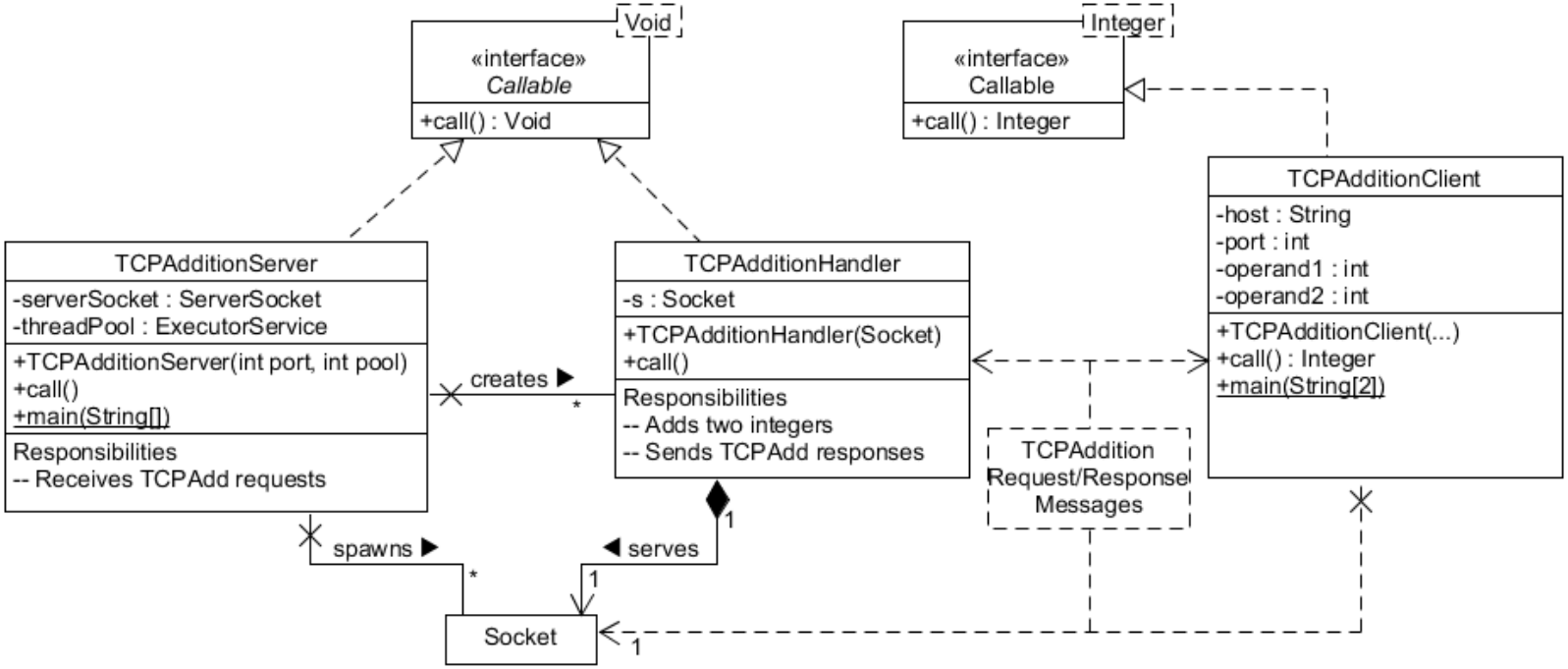
- “TCPAdditionServer”
  - A server that reports the sum of adding two integers

- Protocol Design:
  - Application layer
  - Presentation layer
  - Session Layer
  - Transport Layer
  - Network Layer

TCPAdditionServer Protocol Design

- Application Layer:
  - Request: an int immediately followed by an int
    - <int><int>
  - Response: a single int
    - <int>
- Presentation Layer:
  - Marshalling: however DataOutputStream marshalls
  - <int> := (4 bytes, high byte first)
    - [Big-endian, a/k/a Network Byte Order]
- Session Layer: TCP (unencrypted)
- Transport Layer: TCP
- Network: IP

Program Design



```java
public class TCPAdditionServer implements Callable<Void> {

    public static final int DEFAULT_PORT = 12345;
    private static final int DEFAULT_POOL = 5;
    private static final Logger LOG =
        Logger.getLogger(TCPAdditionServer.class.getName());

    private final ServerSocket serverSocket;
    private final ExecutorService pool;

    public TCPAdditionServer(int port, int pool) throws IOException {
        this.serverSocket = new ServerSocket(port);
        this.pool = Executors.newFixedThreadPool(pool);
    }
```

```java
    public static void main(final String[] array) throws IOException {
        int port = DEFAULT_PORT;
        int pool = DEFAULT_POOL;
        if (array.length >= 1) {
            port = Integer.parseInt(array[0]);
        }
        if (array.length >= 2) {
            pool = Integer.parseInt(array[1]);
        }
        new TCPAdditionServer(port, pool).call();
        LOG.exiting(LOG.getName(), "Main");
    }
}
```

```java
    @Override
    public Void call() throws IOException {
        LOG.info("Listening at: " + this.serverSocket.getLocalSocketAddress());
        try {
            while (!Thread.currentThread().isInterrupted()) {
                LOG.info("Awaiting request.");
                final Socket accept = this.serverSocket.accept();
                LOG.info(
                    "Received connection from: " + accept.getRemoteSocketAddress()
                    + " Handling on " + accept.getLocalSocketAddress());
                pool.submit(new TCPAdditionHandler(accept));
            }
        }
        finally {
            LOG.info("Shutting down.");
            pool.shutdown();
            serverSocket.close();
        }
        return null;
    }
```

MINGW64:/c/Users/J/Desktop/code

```
J@e7240 MINGW64 ~/Desktop/code
$ java TCPAdditionServer
Jul 12, 2017 12:32:49 AM TCPAdditionServer call
INFO: Listening at: 0.0.0.0/0.0.0.0:12345
Jul 12, 2017 12:32:49 AM TCPAdditionServer call
INFO: Awaiting request.
```

MINGW64:/c/Users/

```
J@e7240 MIN
$
```

```java
public class TCPAdditionHandler implements Callable<Void> {
    private static final Logger LOG =
        Logger.getLogger(TCPAdditionHandler.class.getName());

    private final Socket s;

    public TCPAdditionHandler(Socket s) {
        this.s = s;
    }
    @Override
    public Void call() throws IOException {
        LOG.info("Handling request from: " + s.getRemoteSocketAddress());
        try (final Socket s = this.s;
             final DataInputStream dataInputStream =
                new DataInputStream(s.getInputStream());
             final DataOutputStream dataOutputStream =
                new DataOutputStream(s.getOutputStream())) {

            LOG.info("Reading from: " + s.getRemoteSocketAddress());
            final int sum = dataInputStream.readInt() + dataInputStream.readInt();

            LOG.info("Writing \"" + sum + "\"to: " + s.getRemoteSocketAddress());
            dataOutputStream.writeInt(sum);
            LOG.info("Done with: " + s.getRemoteSocketAddress() + " ******** ");
            return null;
        }
    }
}
```

```java
9  public class TCPAdditionClient implements Callable<Integer> {
10     private static final Logger LOG =
11         Logger.getLogger(TCPAdditionClient.class.getName());
12
13     private final String host;
14     private final int port;
15     private final int op1;
16     private final int op2;
17
18     public TCPAdditionClient(String host, int port, int op1, int op2) {
19         this.host = host;
20         this.port = port;
21         this.op1 = op1;
22         this.op2 = op2;
23     }
24
25     public TCPAdditionClient(int op1, int op2) {
26         this("localhost", TCPAdditionServer.DEFAULT_PORT, op1, op2);
27     }
```

```java
61     public static void main(final String[] array)
62         throws IOException, UnknownHostException {
63
64         if (array.length != 2) {
65             throw new IllegalArgumentException("Need two operands");
66         }
67         final int int1 = Integer.parseInt(array[0]);
68         final int int2 = Integer.parseInt(array[1]);
69         int result = new TCPAdditionClient(int1, int2).call();
70         System.out.println("" + int1 + " + " + int2 + " = " + result + " ! ");
71     }
```

```java
    public Integer call() throws IOException, UnknownHostException {
        LOG.info("Connecting to: " + this.host + ":" + this.port);
        try (final Socket socket = new Socket(this.host, this.port);
             final DataInputStream dataInputStream =
                new DataInputStream(socket.getInputStream());
             final DataOutputStream out =
                new DataOutputStream(socket.getOutputStream())) {

            LOG.info("Connected to: " + socket.getRemoteSocketAddress()
                + " via " + socket.getLocalSocketAddress());

            out.writeInt(this.op1);
            out.writeInt(this.op2);
            out.flush();

            LOG.info("Sent " + this.op1 + "," + this.op2
                + " as " + out.size() + " bytes to "
                + socket.getRemoteSocketAddress() + " via "
                + socket.getLocalSocketAddress());
            LOG.info("Reading response from " + socket.getRemoteSocketAddress()
                + " via " + socket.getLocalSocketAddress());

            final int result = dataInputStream.readInt();

            LOG.info("Received \"" + result + "\"");
            LOG.info("Done with " + socket.getRemoteSocketAddress() + " via "
                + socket.getLocalSocketAddress());
            return result;
        }
```

```
J@e7240 MINGW64 ~/Desktop/code
$ java TCPAdditionServer
Jul 12, 2017 12:35:55 AM TCPAdditionServer call
INFO: Listening at: 0.0.0.0/0.0.0.0:12345
Jul 12, 2017 12:35:55 AM TCPAdditionServer call
INFO: Awaiting request.
Jul 12, 2017 12:36:04 AM TCPAdditionServer call
INFO: Received connection from: /127.0.0.1:51436 Handling on /127.0.0.1:12345
Jul 12, 2017 12:36:04 AM TCPAdditionServer call
INFO: Awaiting request.
Jul 12, 2017 12:36:04 AM TCPAdditionHandler call
INFO: Handling request from: /127.0.0.1:51436
Jul 12, 2017 12:36:04 AM TCPAdditionHandler call
INFO: Reading from: /127.0.0.1:51436
Jul 12, 2017 12:36:04 AM TCPAdditionHandler call
INFO: Writing "69"to: /127.0.0.1:51436
Jul 12, 2017 12:36:04 AM TCPAdditionHandler call
INFO: Done with: /127.0.0.1:51436 ********
```

```
J@e7240 MINGW64 ~/Desktop/code
$ java TCPAdditionClient 42 27
Jul 12, 2017 12:36:04 AM TCPAdditionClient call
INFO: Connecting to: localhost:12345
Jul 12, 2017 12:36:04 AM TCPAdditionClient call
INFO: Connected to: localhost/127.0.0.1:12345 via /127.0.0.1:51436
Jul 12, 2017 12:36:04 AM TCPAdditionClient call
INFO: Sent 42,27 as 8 bytes to localhost/127.0.0.1:12345 via /127.0.0.1:51436
Jul 12, 2017 12:36:04 AM TCPAdditionClient call
INFO: Reading response from localhost/127.0.0.1:12345 via /127.0.0.1:51436
Jul 12, 2017 12:36:04 AM TCPAdditionClient call
INFO: Received "69"
Jul 12, 2017 12:36:04 AM TCPAdditionClient call
INFO: Done with localhost/127.0.0.1:12345 via /127.0.0.1:51436
42 + 27 = 69 !
```

UDP Client/Server

- Same protocol, except:
  - Session Layer: none
  - Transport Layer: UDP

- Connectionless
- Datagrams

- “UDPAdditionServer.java”
  - Client

```java
15 public class UDPAdditionServer implements Callable<Void> {
16
17     public static final int DEFAULT_PORT = 23456;
18     private static final Logger LOG =
19         Logger.getLogger(UDPAdditionServer.class.getName());
20
21     private final DatagramSocket dgSocket;
22
23     public UDPAdditionServer(int port) throws SocketException {
24         this.dgSocket = new DatagramSocket(port);
25     }
```

```java
73     public static void main(final String[] array) throws Exception {
74         int port = UDPAdditionServer.DEFAULT_PORT;
75         if (array.length >= 1) {
76             port = Integer.parseInt(array[0]);
77         }
78         new UDPAdditionServer(port).call();
79     }
```

```java
    @Override
    public Void call() throws IOException {
        final byte[] buf = new byte[8];
        final DatagramPacket packet = new DatagramPacket(buf, buf.length);
        LOG.info("Listening at: " + dgSocket.getLocalSocketAddress());
        try {
            while (!Thread.currentThread().isInterrupted()) {
                LOG.info("Awaiting request.");
                dgSocket.receive(packet);
                LOG.info("Received datagram from: " + packet.getSocketAddress());
                DatagramPacket resp = handle(packet);
                LOG.info("Sending response to: " + resp.getSocketAddress());
                dgSocket.send(resp);
            }
        }
        finally {
            LOG.info("Shutting down.");
            dgSocket.close();
        }
        return null;
    }
```

```java
    private DatagramPacket handle(DatagramPacket packet) throws IOException {
        try (DataInputStream dataInputStream = new DataInputStream(
                new ByteArrayInputStream(packet.getData()));
            ByteArrayOutputStream byteArrayOutputStream =
                new ByteArrayOutputStream(8);
            DataOutputStream dataOutputStream =
                new DataOutputStream(byteArrayOutputStream)) {

            SocketAddress socketAddress = packet.getSocketAddress();

            LOG.info("Reading from: " + socketAddress);
            final int op1 = dataInputStream.readInt();
            final int op2 = dataInputStream.readInt();
            LOG.info("Done reading from: " + socketAddress);

            dataOutputStream.writeInt(op1 + op2);
            dataOutputStream.close();

            final byte[] buf = byteArrayOutputStream.toByteArray();
            packet = new DatagramPacket(buf, buf.length, socketAddress);
            return packet;
        }
    }
```

```
J@e7240 MINGW64 ~/Desktop/code
$ java UDPAdditionServer
Jul 12, 2017 12:39:21 AM UDPAdditionServer call
INFO: Listening at: 0.0.0.0/0.0.0.0:23456
Jul 12, 2017 12:39:21 AM UDPAdditionServer call
INFO: Awaiting request.
```

```
J@e7240 MINGW64 ~/Desktop/code
$ netstat -a -o -n

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  UDP    0.0.0.0:23456          *:*                                    3116
```

```java
15 public class UDPAdditionClient implements Callable<Integer> {
16     private static final Logger LOG =
17         Logger.getLogger(UDPAdditionClient.class.getName());
18
19     private final String host;
20     private final int port;
21     private final int op1;
22     private final int op2;
23
24     public UDPAdditionClient(String host, int port, int op1, int op2) {
25         this.host = host;
26         this.port = port;
27         this.op1 = op1;
28         this.op2 = op2;
29     }
30
31     public UDPAdditionClient(final int op1, final int op2) {
32         this("localhost", UDPAdditionServer.DEFAULT_PORT, op1, op2);
33     }
```

```java
    public static void main(final String[] args) throws IOException {
        if (args.length != 2) {
            throw new IllegalArgumentException("Need two operands");
        }
        final int int1 = Integer.parseInt(args[0]);
        final int int2 = Integer.parseInt(args[1]);
        int res = new UDPAdditionClient(int1, int2).call();
        System.out.println("" + int1 + " + " + int2 + " = " + res + " ! ");
    }
```

```java
    @Override
    public Integer call() throws IOException, UnknownHostException {
        try (final DatagramSocket datagramSocket = new DatagramSocket()) {
            this.sendReqeust(datagramSocket);
            return this.getResponse(datagramSocket);
        }
    }
```

```java
    private void sendReqeust(final DatagramSocket datagramSocket)
        throws IOException {

        final ByteArrayOutputStream byteArrayOutputStream =
            new ByteArrayOutputStream(8);
        try (final DataOutputStream dataOutputStream =
            new DataOutputStream(byteArrayOutputStream)) {

            dataOutputStream.writeInt(op1);
            dataOutputStream.writeInt(op2);
        }
        final byte[] buf = byteArrayOutputStream.toByteArray();
        datagramSocket.send(
            new DatagramPacket(buf, buf.length,
                InetAddress.getByName(this.host), this.port));
    }
```

```java
    private int getResponse(final DatagramSocket datagramSocket)
        throws IOException {

        final byte[] buf = new byte[4];
        final DatagramPacket datagramPacket = new DatagramPacket(buf, buf.length);
        datagramSocket.receive(datagramPacket);
        try (final DataInputStream dataInputStream = new DataInputStream(
                new ByteArrayInputStream(datagramPacket.getData()))) {
            return dataInputStream.readInt();
        }
    }
```

```
J@e7240 MINGW64 ~/Desktop/code
$ java UDPAdditionClient 1181 1280
1181 + 1280 = 2461 !
```

```
J@e7240 MINGW64 ~/Desktop/code
$ java UDPAdditionServer
Jul 12, 2017 12:39:21 AM UDPAdditionServer call
INFO: Listening at: 0.0.0.0/0.0.0.0:23456
Jul 12, 2017 12:39:21 AM UDPAdditionServer call
INFO: Awaiting request.
Jul 12, 2017 12:44:29 AM UDPAdditionServer call
INFO: Received datagram from: /127.0.0.1:65144
Jul 12, 2017 12:44:29 AM UDPAdditionServer handle
INFO: Reading from: /127.0.0.1:65144
Jul 12, 2017 12:44:29 AM UDPAdditionServer handle
INFO: Done reading from: /127.0.0.1:65144
Jul 12, 2017 12:44:29 AM UDPAdditionServer call
INFO: Sending response to: /127.0.0.1:65144
Jul 12, 2017 12:44:29 AM UDPAdditionServer call
INFO: Awaiting request.
```

Recap

- Models
  - Client/Server
  - Peer-To-Peer
- Examples:
  - TCP Client/Server
  - UDP Client/Server
- Sockets
  - Socket (Regular)
    - w/ 2 Streams
  - ServerSocket
    - Accepts connections
    - Produces Socket
  - DatagramSocket
    - Connectionless
  - Marshalling