Sockets in Java

Short introduction

Computer Networks 2015

vrije Universiteit amsterdam

Client side sockets

We only consider client side sockets here

- Enough for the assignments
- Server side sockets are very similar
  - For more information, see link at end of slides

TCP client side sockets

TCP is connection oriented; we create a socket object which we connect to a certain address:

```
Socket s =
   new Socket("host.domain.com",80);
```

TCP client side sockets

From a Socket object, we obtain a BufferedReader to read from:

```
BufferedReader in =
   new BufferedReader(
      new InputStreamReader(
         s.getInputStream()
      )
   );
```

TCP client side sockets

Similarly, we obtain a PrintWriter to write to:

```
PrintWriter out =
   new  PrintWriter(
     s.getOutputStream(),true
   );
```

TCP client side sockets

To send a message to the server, we can use:

```
out.println(“Hello!”);
```

TCP client side sockets

Messages from the server can be obtained by reading from the BufferedReader:

```
String message = in.readLine();
```

When the server has closed the connection, `message` will become `null` if you read from `in`

TCP client side sockets

When we are done talking to the server, we neatly close the connection:

```
s.close();
```

UDP sockets

UDP sockets are not connected to anything; they are simply created and can then exchange packets with any host:

```
DatagramSocket s =
   new DatagramSocket();
```

UDP sockets

To send a message, we first encapsulate it in a DatagramPacket object:

```
String m = “Hello!”;
DatagramPacket out =
   new DatagramPacket(
      m.getBytes(), m.length(),
      InetAddress.getByName(“host”),
      80
   );
```

UDP sockets

Now to send the DatagramPacket, we can use:

```
s.send(out);
```

UDP sockets

Receiving messages is also done using a DatagramPacket:

```
DatagramPacket in =
   new DatagramPacket(
      new byte[1024], 1024
   );
s.receive(in);
String message =
   new String(in.getData());
```

UDP sockets

`s.receive(in)` will block until there is a message; you can prevent this by setting a timeout before you call receive:

```
s.setSoTimeout(1000); /* 1 second */
```

UDP sockets

When we are done, we close the socket as usual:

```
s.close();
```

More information

Java Socket javadoc (client side TCP):
http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html
Java ServerSocket javadoc (server side TCP):
http://docs.oracle.com/javase/7/docs/api/java/net/ServerSocket.html
Java DatagramSocket javadoc (UDP):
http://docs.oracle.com/javase/7/docs/api/java/net/DatagramSocket.html