IN2011 Coursework: Testing and Tips

1. Using a web browser

You already have access to some sophisticated HTTP clients which you can use to test how your server responds to GET requests: any web browser will do. Suppose you have your server running and listening on port 1091 and you want to connect and send a GET request with URI `/some-folder/some-file.html`. Then start a browser on the same machine, enter

```
http://localhost:1091/some-folder/some-file.html
```

in the browser's address bar and away you go. The browser will attempt to open a connection to port 1091 and send the HTTP request

```
GET /some-folder/some-file.html HTTP/1.1
```

If all goes well and the file contains valid html, the browser will receive the file content over the connection and display it. However, if your server is not behaving as expected, or if the file does not contain valid html, the result may be hard to interpret; it may even seem as if nothing has happened at all and you won't learn much about what the problem is. An additional complication is that browsers may exhibit some quite sophisticated behaviours (such as trying to open a number of simultaneous connections to your server, anticipating that the first request may be rapidly followed by several follow-up requests to the same server). If your server is only single-threaded, this may result in unexpected and hard to interpret behaviours.

2. Using Telnet (or Putty)

Another very useful tool for testing TCP applications is the **Telnet** client. (You may need to jump through some hoops first to enable telnet if you are a windows user; Google is your friend. Also, see below.) You can connect to your server at the command line (again, this assumes you are working on the same machine that is running the server) like so:

```
telnet localhost 1091
```

Once the connection is established, you can manually enter HTTP requests, one line at a time, to send them to the server; any responses will be echoed back to the same console window in which telnet is running. This gives you much more direct access and control than using a browser.

If you are using Windows and you can't enable telnet (which is probably the case on the university lab machines) then you can use Putty instead, in essentially the same way (when opening a connection, choose type "Raw TCP").

A disadvantage of Telnet is that you cannot close your output stream (client to server) except by quitting Telnet; the problem with that is you never get to see any responses which the server might send after your output stream is closed. An alternative to Telnet which is better in this respect is **netcat** (see below).

3. Networking

The above testing suggestions assume that you are running your server and client on the same machine. Of course, it is more interesting to have them running on separate machines. If you do this you should replace "localhost" in the above examples by the host name or IP address of the machine on which the server is running. To find the IP address of a machine (the one on which you are running your server), use appropriate OS tools. Do *not* rely on external "What is my IP address?" websites, since they will generally report the IP address of the external NIC of your local NAT server, which is not what you need.

4. Using your Linux VM

Your VM has a working Java installation (OpenJDK) so, another possibility is to run your server on the VM and the client on the host machine. (In theory, you could even do all your development work on the VM, but it's not obvious why you would want to.) To make this work you will have to make some changes to the networking setup for your VM, since the default setup does not allow the host to "see" the client on the network. These are the steps I followed (I have only tested this under Mac OS 10.7; you may run into firewall issues, etc, on other platforms):

1. Create and configure a "host only" virtual network in VirtualBox.

in VirtualBox Preferences, Network => Host-only Networks. If the list is empty, click on the green + icon on the right hand side to create a new virtual network (no need if vboxnet0 already exists).

Now, still in the Host-only Networks preferences tab, select vboxnet0 in the list and click on the screwdriver icon on the right to configure the virtual network (you are going to disable its DHCP server). Select the DHCP Server tab and un-tick the "Enable Server" box. Click OK. Click OK again.

2. Configure your Linux VM to use the new host-only network.

First, select NetsOps1516 in the VirtualBox Manager window and then Settings => Network => Adapter 2. Tick the Enable Network box and then choose Host-only Adapter in the drop-down menu. Make sure vboxnet0 is selected in the Name drop-down menu. Click OK.

Now, in your VM, as root, edit the file `/etc/network/interfaces`. Before you edit it, it should look something like this:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet dhcp
```

Don't change any of this. The aim is to configure the *additional* network interface that you just enabled. (Note that VirtualBox calls this interface "Adapter 2" but your Linux VM will see it as "eth1".) At the bottom of the file, add the following

```
# The second (vbox host-only) interface
allow-hotplug eth1
iface eth1 inet static
address 192.168.56.101
netmask 255.255.255.0
network 192.168.56.0
broadcast 192.168.56.255
```

Now reboot your VM. To check if that all worked, run the command `ifconfig` as root. The output should list three interfaces (eth0, eth1, lo) and eth1 should show with " inet addr: 192.168.56.101".

With luck, if you run your http server (or any other server, for that matter) on your VM, you should be able to connect to it from the host machine using IP address 192.168.56.101.

5. Using Netcat

The Netcat tool is already installed on your Linux VM. Similar tools are also available for other flavours of Unix and Windows.

Suppose your http server is listening on port 8081 on host 192.168.1.161. Then you can connect to the server using Netcat as follows:

```
nc -q 60 192.168.1.161 8081
```

Then proceed as for Telnet (see above) except that now you can enter control character **Ctrl-D** at the command line to close the output stream and Netcat will wait up to 60 seconds for a response from your server (that's the purpose of the `-q 60` option). Note that the `-q` option is specific to the Linux version of Netcat; it may not be available (or necessary) for other versions.