Practical Session Two – Simulation with ns

In this practical class you will learn a little bit about the network simulation program ns. More about ns can be found at the web site:

`www.isi.edu/nsnam/ns/`

Log into the unix machines and bring up a command shell. (If things don't work then try logging into tower instead). Note that the simulator creates some quite large temporary files so you may want to free up some disk space. The first thing you need to do is to include the directory with the simulator into your searches. Then type:

`set path= ($path ∼/rgc2/bin)`

This should enable you to run the **ns** program. Note that if, for some reason, you close down the shell you are using then you will need to retype this line which includes my directory in the list of directories from which you can run programs. Next you need to make a directory to keep the files you will use in this practical:

`mkdir` *nsfiles* (or whatever other name you wish to call it).

`cd` *nsfiles* (now change into the directory).

`cp ∼/rgc2/nsegs/*` . (remember the final dot — copy the files over to your new directory).

You are nearly ready to run your first example with ns. You are not expected to learn how to write simulations in this example but it will help your understanding to take a look at how this is done. Let us start with the first exercise file. Open up `ex1.tcl` with an editor you like to use. If you don't like any particular editor then use:

`nedit ex1.tcl &`

You are *not* expected to learn tcl in this practical class. However, spend five minutes looking through the file. Comments are lines beginning with #. Work out roughly what you expect it to do. The tcl file is giving instructions to the simulator which will tell it about the system we are simulating. You should have a good idea of what the system being simulated is by the time you run ns. Do not change the file. When you think you have an idea of what is going on then try:

`ns ex1.tcl`

If all goes well then the simulation will run and finally pop up a viewing window for you. This first simulation isn't very interesting but it will get you used to the viewing interface. The **nam** viewing interface should be very straight forward. It works much like a video recorder. Experiment with the playback until you are comfortable with how it works. Try the following:

1. Play the simulation forward.

2. Alter the speed with which it plays.

3. Drag the TIME bar to a particular time in the simulation.

4. Zoom in and out.

5. Click the EDIT button and move the nodes about.

6. Stop the simulation and right click on a packet to find out more about it and monitor it.

7. Right click on a packet to find information about it.

8. Right click on a link to graph data on it.

The first simulation is fairly boring. Two streams of traffic in opposite directions sending things with a constant bit rate (an equally spaced stream of traffic). When you've finished playing with the visualisation, look again at the source code and see if you can see which lines have what effect.

If you are feeling adventurous, try to add in a third node (called `$n2`) to the simulation and connect one of the data streams to that instead of `$n0`. You will need three lines to set up a node — one to set up the node, one to tell it which other nodes it is connected to and one to tell the visualisation tool `nam` which direction it links in. You could also try playing with the bandwidth and length of the links to see what that affects.

There are a few more parameters you might alter to see the effects in the model. Try changing bandwidths and packet sizes. You will notice that the size of a packet on screen is a function of the amount of a line it is taking up. That is, it is a function of the size of the packet, the length of the connection and the bandwidth of the connection. (Don't worry if you accidentally break things, you can always take another copy of the file).

Our next example is a bit more interesting. When you have finished with the first example, take a look at `ex2.tcl`. Again, spend five or ten minutes working out what you think the simulation should do. When you are ready, run this simulation as before:

```
ns ex2.tcl
```

This should be a bit more interesting. The simulation is of two nodes using the file transfer protocol (FTP) to send files to a third node. In this case we are using the TCP protocol. There is also a bottleneck where queuing can occur at the middle point where the streams join. Pay particular attention to what happens when each ftp stream starts up. Eventually you will notice packets being *dropped* (rather comically in my opinion) from the queue. Consider the following questions:

1. Why is the first packet sent by the blue stream very small compared with the packets it gets back?

2. Why are the packets returned from the end node smaller than the packets which are sent?

3. What happens to the streams when packets are dropped?

This example should demonstrate many of the important features of TCP – window size increase and decrease and a response to packets dropped to control bandwidth (unfortunately, not FIN packets to close a session in this example). Once you are sure that you have seen all of these features either move on to the next example or, if you are feeling adventurous add a third FTP session – you will have to add another node and add another sink to the final node. You will probably have to increase the bandwidth of the middle link to reasonably accommodate another ftp session down that link.

Our next example `ex3.tcl` is concerned with dynamic routing — that is how networks deal with problems of finding the way between places, particularly when links fail. This is done with the exchange of routing packets.

Look at the file in your editor. This one is a little more complex than the first two so take a while to work out what you think will happen. Remember, it is not important that you know how to program in tcl but you should try to get a feel for what is going to happen and how you might alter the program. Take five or ten minutes to work out what you think might happen in the simulation before running it. When you are ready, type:

```
ns ex3.tcl
```

1. Note the small packets which are periodically sent between nodes even before all the constant bit rate streams are turned on. These are routing packets.

2. Are all packets taking efficient routes around the ring?

3. What happens immediately after the link fails?

4. How does the recovery happen?

5. What happens when the link comes back up?

Try inserting more link failures into the network. What happens if the network becomes disconnected?

`ex4.tcl` is another example showing some more features of ns. Play with it and see what it does. ns is an extremely complex simulation and there is certainly not time in this course to give you a full feeling for everything it is capable of. I hope that this worksheet has given you some insight into how the TCP and UDP protocols work and how rerouting works in practice. Feel free to play with ns and try to write your own scripts. See what types of simulation you can generate.