Project Proposal: Deep Routing Simulation

Principal Investigator: Mr. A. Herbert

February 2012

1 Statement of the Problem

Currently the large scale network simulators lacks the ability to correctly simulate delay and implementing such functionality may result in some other features needing to be reviewed within the network simulator.

2 Object of Research

To re-implement/improve on existing large scale traffic routing simulators and features such as delay, packet loss, data corruption and other such common network errors. These features have to implemented all while making sure not to cause unexpected behaviour within the overall features of the network simulator systems which the traffic routing simulator will be running alongside. This simulator will run at the core of the simulated network and will simulate edge to edge traffic by routing traffic through itself by acting as a Super Node that handles all responses in such a way that the client machine believes it is actually receiving replies from multiple nodes. This will lead to a more realistic traffic routing simulator which will yield more accurate results.

3 History and Background

A network simulator can best be described as a piece of hardware or software that can be used to simulate network activity without the presence of an actual network. These network simulators usually support protocols such as TCP and UDP and even allow for these protocols to be simulated on WLAN (Wireless Local Area Network) [9].

There is already quite a lot of research that has been put into this area of knowledge and it is still growing, this is owing to the fact that there is a need to know what strains are put on large networks during peak times. These strains on networks that lead to peak periods are mainly due to major network congestion, heavy bandwidth use through FTP and HTTP protocols and even real time traffic such as UDP and VoIP cause major time constraint demands on networks as the need for timely delivery of data is required. This comes with problems though as traffic can vary greatly from each of these peaks and getting exact figures is more difficult than it may seem [2].

Existing simulators include PacketStorm made by Communications Inc [5], Linktropy by Apposite Technologies [1] and ISEAGE [7] among many more. The Linktropy emulator allows for simulation speeds between 300bps and 1Gbps [1]. This with functionality to capture and replay network conditions [1] makes it a powerful tool in network simulation.

The use of such simulators are relevant in todays day and age as one can simulate traffic within a network and use such data to find bottlenecks and possible problems that may arise in the future. From this one can create solutions and as a result increase throughput and reduce delays within a network leading to a more efficient network.

Also simulations can be done on denial of service. This can be achieved by removing nodes out of the simulated network and then preforming tests to see how the simulated network would function after such changes have been applied.

Other applications that can be done are data transfer times to test bandwidth and through put, delay between hosts, infection simulation of viruses and time taken to cure a network of them. There are also other possibilities that are available for testing on such systems.

There are also simulators such as NS-2, NS-3 and BPGVista. NS-2 is designed to simulate TCP (Transport Control Protocol), routing and multicasting over not only wired but also wireless networks and even satellites [6]. NS-3, the newer release of NS-2, is primarily designed for education purposes and is driven by discrete-event based system to simulate its network [8].

simBGP was designed and implemented to simulate BGP (Border Gateway Protocol) traffic [3]. As BGP is possible to simulate it may be used as a future feature in the traffic routing simulator that I am to design and implement.

4 Approach

The first step is to create a working proof of concept. The idea behind this is to read in standard trace routes as done by a Unix Terminal and graph the nodes. This will be done by reading in a file which has all the routes dumped into it, the program will have to discern from what is usable and what is garbage, such as unresponsive connections and information headers generated by the traceroute command.

Initially each node simulated will store the real nodes IP and average latency however this will be expanded on later to include other data fields such as MTU (Maximum Transfer Unit) and connection speed. Connection speed may be specified or may need to be calculated before it is inputted into a node. As the IP is unique to nodes in network, they should not be duplicated. This will be handled by a hash table as it provides quick access to unique values, which is needed considering the number of nodes that will be simulated.

A node will contain all details about the real node it is simulating as well as which nodes it is connected to. As traffic can flow both ways the algorithm will have to include not only linking the newly added node to its adjacent nodes, but also connecting them back again. For the proof of concept it will only be a one way graph as there should be no down streamed data. Later on the when the nodes require it for communication they will be doubly linked as to allow for an upstream and downstream of data flow.

At this point all that'll be left to do in the proof of concept will be to sniff a packet from a connected client machine and simulate a basic traceroute by replying with the appropriate responses that the client would expect to receive if it were waiting for a response from a real network. After this task is completed the proof of concept will be complete.

How the routing simulator will pick up on network requests and respond to it will be implemented by using the libnet and pcap libraries to sniff packets on the network and send packets back onto the network. Once a packet has been sniffed it will be broken down and the network simulator will act as the host and respond to the client with appropriate messages to the requests made by that client.

Next routing data will have to be parsed in much larger quantity. The data will not be collected by myself but rather by requesting the data from CAIDA who deals in collecting such information [4]. This will lead to changes in the systems route parsing algorithm as the routes read in will most likely not be in the same format as a simple UNIX traceroute.

During the simulation process, features such as adding latency to the responses of packets will be implemented by looking up the average delay of the link between nodes and sleeping the thread handling the request for the average latency time. This will make the simulated network more realistic, however this feature may be implemented in real time where one would have to wait for the reply, but one may have the ability to turn off real time simulation and just have packets respond with actual latency values attached to them except without having to physically wait for them.

Throughput is also a possible feature to be added where simulated file transfer times will be calculated by the maximum bandwidth available on the routed connection. A maximum flow algorithm may need to be used here to ensure maximum bandwidth utilization.

Other features will be added as deemed needed and currently cannot be commented on until a later stage when the need arises for their addition.

Once the simulator is stable, threading will be introduced as to allow for more effective use of resources available on the host machine. The threads will be pooled and as the network simulator receives a request it will assign a thread to handle the request. In this way the multi-threaded host machine will be able to handle higher data throughput. No problems should arise from using multiple threads in this system as once the graph has been laid out, there is no longer any writing of data into the system and only data read requests will occur, thus resources don't have to be shared.

Also a graphical representation of the network's nodes will be advantageous to further the ease of understanding of the links made between nodes. This will most likely be done by dumping all nodes in the simulator to file or other memory source and then reading them in using python and one of its graphical libraries. The reason for deviating from C/C++ architecture is because the graphical representation is not essential to the running of the simulator and also python is an easy to use language with many libraries that handle such graphical data representation and allow for easy implementation all round.

5 Requirements/Resources

For proof of concept a standard desktop computer will be required but when this system is implemented on a large scale it would require much more resources to work with.

Memory usage and CPU usage is fairly linear and so standard calculations can be done to determine a good estimate to the memory and CPU needs of the machine set to run this simulator.

So to start off development and initial testing will be run a on single system and once deemed stable, it will be moved to a larger system to allow for a larger scale of traffic routing simulation.

References

[1] Apposite Technologies. WAN Emulation Made Easy. Online, 2011. Site for packet storm. Available from: `http://www.apposite-tech.com/index.html`.

[2] Asmussen, S., and Glynn, P. W. *Stochastic Simulation: Algorithms and Analysis*, vol. 57. Springer Science++ Business Media, LLC, 2007.

[3] BGPVista. bgpVista. Online, 2006 - 2009. Available from: `http://www.bgpvista.com/simbgp.php`.

[4] CAIDA. CAIDA Data - Overview of Datasets, Monitors, and Reports. Online, 2002 - 2012. Available from: `http://www.caida.org/data/overview/`.

[5] Communications, Inc. Network emulation with data rates up to 10 Gbps. Online, 2004-2012. site of packetstorm. Available from: `http://packetstorm.com/psc/psc.nsf/site/index`.

[6] DARPA, SAMAN, NSF, CONSER and ACIRI. The Network Simulator NS-2. Online, 1995. Available from: `http://www.isi.edu/nsnam/ns/`.

[7] Iowa State University. What is ISEAGE? Online, 2012. site of ISEAGE. Available from: `http://www.iac.iastate.edu/iseage/`.

[8] National Science Foundation. NS-3. Online, 2011 - 2012. Available from: `http://www.nsnam.org/`.

[9] Salam A. Najim, Ibrahiem M. M. El Emary, and Samir M. Saied. Performance evaluation of wireless ieee 802.11b used for e-learning classroom network. *IAENG International Journal of Computer Science 34* (2007), 1.