CS 8803NGN: Next-Generation Networking March 9, 2010

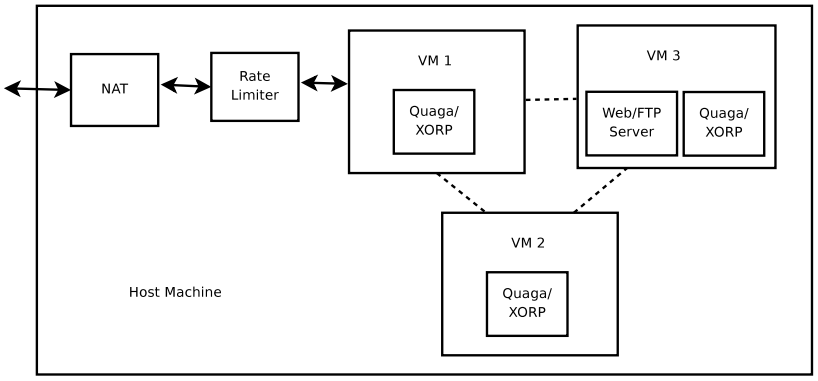
Problem Set 2: Building and Connecting Virtual Networks

Instructor: Prof. Nick Feamster *College of Computing, Georgia Tech*

This problem set has three questions, each with several parts. Answer them as clearly and concisely as possible. You may discuss ideas with others in the class, but your solutions and presentation must be your own. Do not look at anyone else's solutions or copy them from anywhere. (Please refer to the Georgia Tech honor code, posted on the course Web site).

Turn in your solutions in on **April 9, 2010** by 11:59 p.m. -0500, to T-Square.

In this assignment, you will create a network of virtual machines, as shown in the diagram below. You can create your virtual network using virtual machines on your own machine, or using multiple machines on Emulab (more details are below). **You are welcome to work in groups for this assignment.**



1 Virtual Machine Setup

The network as shown in the figure has three virtual machines; all of them are connected to one another. You are welcome to use a virtual machine of your choosing for this part of the assignment. For example, you can use Qemu, OpenVZ, VirtualBox, VMWare, Xen, UML, or the Trellis kernel to create virtual machines.

For the first part of the problem, simply set up three distinct virtual machines, each with two interfaces that you will connect to the other virtual machines in the next problem.

What to submit. Turn in your configuration scripts/written steps for setting up the virtual machines in your environment of choice. We will not test them, but we will ask you to demonstrate them working in a demonstration for us below (see Problem 2 below).

2 Routing and Forwarding Between Virtual Machines

Step #1: Setting up tunnels. Mechanisms to connect virtual machines to each other depend on the choice of the virtualization technology. Some virtualization technologies (OpenVZ, Xen, UML,

VirtualBox) support TUN/TAP devices that can be connected using Linux software bridge. Some like Qemu support VDE devices. We will answer queries about Qemu, OpenVZ, and Trellis kernel. There is also excellent community support for VirtualBox, Xen, and VMware.

If you do not have a powerful machine to run three virtual machines, then you can try allocating a three node topology on Emulab, with each machine running one virtual machine. However, in that case, you will have to connect the virtual machines using some kind of tunneling mechanism such as GRE, EGRE, IPTunnels, OpenVPN, etc.

Step #2: Dynamic routing between virtual machines. In each virtual machine, start Quagga or XORP to establish dynamic connectivity between the nodes; you should do this by running an internal gateway protocol (e.g., OSPF) between the nodes. You can assign static IP addresses to each interface inside the the VM. VM3 will run an FTP or a Web Server. VM1 is connected to the Host interface through a traffic shaper and a NAT module. You can use tc or iptables or Click to implement traffic shaping and/or NAT.

Step #3: Rate Limiter. Connect one of your virtual machines externally through a rate limiter, followed by a NAT (you could use, for example, a Click element for this part of the problem), as shown in the figure.

What to submit. You will demonstrate above setup to us. We will try to fetch a file from the server on VM3 from our machines and run tcpdump on the host and VM interfaces. We will also vary the traffic shaping parameters to change the bandwidth limits. In the middle of the file download, we will take down link between VM1 and VM3 to check that file transfer is not affected.

Some starting papers for this assignment are "In VINI Veritas: Realistic and Controlled Network Experimentation", and "Trellis: A Platform for Building Flexible, Fast Virtual Networks on Commodity Hardware".

There is a lot of software setup involved in this assignment as compared to the first one, so we advise you to start early.

3 Connecting Your Virtual Machine to the Internet

In this part of the problem, you will connect your virtual network to the Internet through the BGP session multiplexer ("BGP Mux"). For more information about the BGP Mux, please see the following technical report:

> V. Valancius, N. Feamster, J. Rexford, A. Nakao, "Wide-Area Route Control for Distributed Services", Georgia Tech Technical Report GT-CS-10-02. (available in T-Square).

To complete this part of the problem, you will need to set up both a control plane (i.e., routes) and a data plane (i.e., packet forwarding) between your virtual network and the BGP Mux. *Note: We are currently investigating how this might be done on a class-wide basis. Depending on our progress, the last part of this problem may be optional.*

3.1 Warm-Up/Background: Reading BGP Routing Tables

1. Download the routing table dump from T-Square.

2. Find the route to `www.stanford.edu` from Georgia Tech. Look up the sequence of ASes that this route traverses. Give the *names and numbers* of these ASes. (Hint: Remember from class, the tool that will help you here is `whois`.)
3. Find the route to `64.250.64.0/20`. Who does this prefix belong to? Give the names and numbers of the ASes along this route.
4. Why is the upstream AS for the route to Stanford different from the upstream AS for the route to the second prefix? (Hint: Go read about Internet2.) How do upstream ASes prevent Georgia Tech from using Internet2 for some destinations?
5. Telnet to `route-views2.routeviews.org`. This routing table shows the routes as seen from many different ASes. Look at the route from 7018 back to Georgia Tech. Notice that, even though from #2, the path goes through AS 7018, the traffic returns *from* 7018 to Georgia Tech via a different set of ASes. Why?

What to submit. Turn in your answers to the questions above.

3.2 Connecting Your Virtual Network to the BGP Mux

In this part of the problem set, you will connect your virtual network to the BGP Mux using OpenVPN. We have set up an OpenVPN server on the BGP Mux. you will conenct to the BGP Mux via an OpenVPN client. Over this tunnel, you will then (1) establish a BGP session; (2) populate a routing table on your local software router; (3) send packets to and from the global Internet via this destination. *Depending on our ability to support this part of the assignment, you may have to schedule working on this one at a time.*

1. Set up a tunnel to the OpenVPN server on BGP Mux from one of your clients via OpenVPN (VM1, as shown in the figure). To set up OpenVPN, run 'openvpn openvpn.conf'. You will need openvpn.conf, client1.key, client1.crt, and ca.crt to do this. We will provide one set of configuration files and keys per group.

2. Establish a BGP session over this tunnel. Your AS is 65000 and the peer AS (BGP-mux router) is 2637. The peer IP address (BGP-mux router tunnel endpoint) is 168.62.21.1.

3. Test the resulting data path. You should see BGP routing table entries on your local VM, and these entries should populate a local forwarding table in your VM. The "next hop" for these routing table entriesshould be the IP address of the OpenVPN tunnel endpoint.

What to submit. Turn in your OpenVPN and BGP routing configurations. We will also ask you to demonstrate the working virtual network for us.