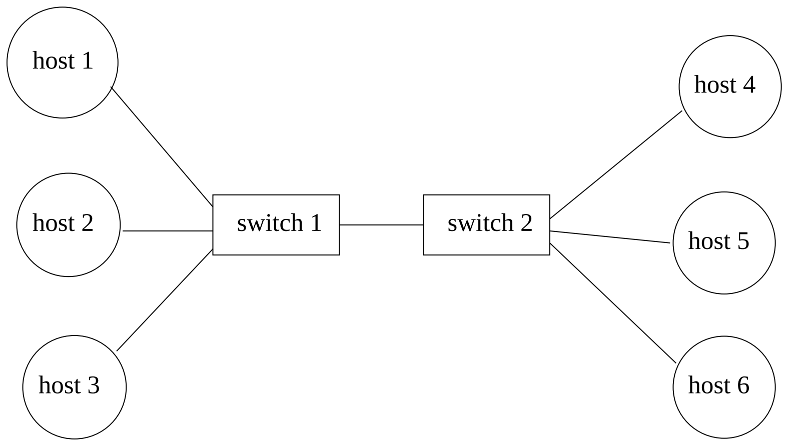
Homework #1: Running Click on Emulab[*]

This homework assignment has three questions, each with several parts. Answer them as clearly and concisely as possible. You are welcome to work with one other partner on this assignment if you want, provided you (1) write up your own answers to the questions and (2) include the name of your partner in your write-up. Your write-up should include *any Tcl scripts* and *Click configurations* you used to set up your experiments. Turn in your solutions by 11:59pm on **September 30, 2008** via e-mail with the subject "CS 561 Assignment #1" and a single PDF attachment.

This problem will give you hand-on experience with Layer 2 and Layer 3 protocols, and with setting up network experiments in Emulab. The experimental configuration language is based on Tcl, which is the same as that which is used for *ns*, a network simulator.

Setting Up Your Experiment

1. **Emulab:** Go to `http://www.emulab.net` and request an account. Join project 'GradNetworks'. *Do this early, as your membership will need to be approved!*
2. **Topology setup:** Write a Tcl procedure that takes an integer n and creates the experiment configuration (i.e., an ns file) for a dumb-bell topology (with two nodes, each connected to n hosts) as shown in the figure below for $n = 3$. You can test your code on Emulab or in *ns*. Make all "edge" links in the topology be duplex links with 100Mbps of bandwidth and a 10ms propagation delay, except for the link between the nodes *switch1* and *switch2* which is a duplex link with 1Mbps of bandwidth and 10ms propagation delay. Name the nodes as in the Figure, as we will be using them in later parts of this problem.



Hint: Reading Emulab and ns documentation will help with this problem.

[*]Based heavily on an assignment created by Professor Nick Feamster at Georgia Tech.

3. **Topology creation:** Create your topology in Emulab. You can use the Emulab interface to get the DNS names and IP addresses of each of the nodes in your topology. By default, Emulab installs FreeBSD on each node; please use Emulab's `tb-set-node-os` to install Linux instead.

We will now experiment on the nodes themselves.

Fun with ARP and Click

1. **Installing your own switch:** In this part of the problem, you'll enable *all* of the hosts to reach each other through the switches. Fortunately for you, Click already has an `EtherSwitch` element, so most of the "hard work" is done. You just have to figure out how to set it up!

   (a) Download, compile, and install the Click (`http://www.read.cs.ucla.edu/click/`) modular router, in user space, on *switch1* and *switch2* in your topology. *Possibly helpful hint:* To save yourself the trouble of compiling Click on the nodes themselves, you can compile on a (possibly faster) local Linux machine, and then copy the binary over to *switch1* and *switch2*.

   (b) Install and configure the Click elements on *switch1* and *switch2*. The `EtherSwitch` element will likely be quite helpful for you in this regard; `ListenEtherSwitch` might also be helpful, particularly for debugging.

   (c) An Ethernet interface card in a conventional end host (e.g., a Linux or Windows machine) discards unicast frames that are not destined to its own MAC address. (Explain why.) What change do you need to make to the Click configuration to allow the switches in your topology (i.e., the Linux machines running the EtherSwitch element) to handle packets destined to other locations?

   (d) Make that change, and then verify that *host1* can ping *host2* and *host4*. Include the outputs of ping in your report.

2. **Monitoring ARP traffic and dumping forwarding tables:**

   (a) Log into *host5* and *host3* and start a tcpdump on each machine that looks for ARP packets.

   (b) Log into *host1* and begin pinging *host5*.

   (c) Show the packet-trace excerpts for any ARP queries or responses. Explain what you see.

   (d) Show a dump of the forwarding tables of *switch1* and *switch2*. (How did you dump the table? Explain how each of the entries got there.)

   (e) What are the privacy implications of ARP flooding? Why do some enterprise networks disallow users from running tcpdump, *even on their own computers*?

3. **Measuring failure and recovery time:**

   (a) Modify your experiment set-up so that the interface on *host5* (connecting *host5* to *switch2*) fails. Explain how you did this, and describe one other way you might have simulated this kind of failure.

   (b) Send repeated ping packets to *host5*. How long does it take before *switch1*'s forwarding-table entry for *host5* expires? (How can you detect this by monitoring the ping packets?)

   (c) Reinstate the link. How long does it take before *host5* begins responding to pings after the link recovers?