CSE 534 Spring 2014

Project 1

Last updated 10 March, 2014.

Due date Tuesday, 1 April.

Please read carefully

– You are to work on this Project in groups of two.

– Each group should hand in a hard copy of a report in class on 1 April that addresses the issues highlighted in blue font in the Project description below. This report should include copies of the Emulab ns and Click scripts you used.

Make sure that your report is structured along the section numbers (A1., B1., B2., C1., etc.) of the Project description below.

– Also email me soft copies of your report and, especially, of the ns and Click scripts, in case I decide to test them out for myself. Please make sure you do this – you will lose points if you do not. However, I will grade the hard copy you hand in class: in other words, the soft copy does not relieve you of the responsibility of also handing in a hard copy since I do not plan to be running a printing service for you. You will lose points if you do not also hand in the hard copy.

– Please make an early start on this Project. You will find that that the majority of your time will be spent searching for and trawling through documentation trying to figure out how to configure things and make them work. This is a time consuming, slow, painful and frustrating process which will involve many false starts and countless cycles of trying this & that out, and of correction & fine tuning. But it is of the nature of what a self-dependent learning process involves, and is an integral part of what this Project is about. So allow yourselves enough time.

Another reason for starting early is that if all twenty odd groups decide to intensively hammer Emulab simultaneously in the last few days before the deadline, you might find that you have to wait a long time to get nodes assigned to your experiment and not be able to finish the Project in time.

So far as I am concerned, you have been warned (countless times!) – I will not grant any extensions based on excuses of unavailability of Emulab nodes, or any other similarly specious excuses that are a result of your waiting too late to start on your Project. If you have not already made substantial progress with the Project well before the Midterm exam comes around, it is probably too late for you.

– I have set up a Discussion Forum for the Project on Blackboard. Use it to share useful items of documentation, helpful tips, seek advice and guidance from your peers, etc. Also use it to find a partner if you do not already have one.

However, **do not use the Discussion Forum to post your Emulab ns & Click scripts, etc.** It is unfair that lazy, late-starting groups should benefit from the products of your time investment and learning by copy-pasting your scripts and turning in Project reports that will get as good as – if not a better – score than yours, at minimal effort. It would also set up an unhealthy dynamic where these lazier groups hold back on putting any serious effort into their projects in the hope of reaping the fruits from the more responsible and hard-working groups' work, copy-pasting their way to a relatively effortless and spuriously credible Project report without having learned anything substantive themselves.

– Finally, if you and your partner would like to work as a group on Emulab, please email me giving me your name and your partner's name (only one of the two group members should do this, not both). I will give your partner local_root privileges on your Emulab group and email both of you back to let you know this has been done. This should permit the two of you to work with full mutual privileges in your Emulab group.

Preliminaries

- Apply for an Emulab account. Please do this as soon as possible because I will have to approve each account individually through the Emulab system and configure the accounts into individual subgroups.
  - Go to the *Emulab – Network Emulation Testbed* home page https://www.emulab.net/ and click on the *Request Account* button on the left hand side of the page.
  - On the *Request Account* page, click on *Join an Existing Project*.
  - Fill in the form on the *Apply for Project Membership* page.
    - Fill in only the fields marked with an asterisk ***** .
    - The *Institutional Affiliation Abbreviation* is SBU.
    - The *Project Name* is CSE534 (with no space between "CSE" and "534").

A. Emulab

- When your account is up and ready, start familiarizing yourself with Emulab.
  - Go to the *Emulab Tutorial* page https://wiki.emulab.net/wiki/Emulab/wiki/Tutorial and implement the *Getting Started* network given there.

    Begin/run the experiment, find out what IP addresses and "*Qualified Names*" nodes A, B C & D have been assigned.

    Log into these nodes and see if you can get them to ping each other.
    - You can ssh into a node using the *Qualified Name* it has been given in your experiment as its domain name, together with your Emulab *Username* and *Password*.

- Now change the topology so that instead of nodes B, C & D being joined together by means of the *lan0* ethernet hub shown in the topology figure on the tutorial page, they are joined together by a new physical node E that you introduce into the topology.
  Log into the nodes (including the new node E) and see if you can get them to ping each other.

A1. Report on how you managed to change the original topology, replacing the hub *lan0* by the new physical node E. What were the IP addresses assigned by Emulab to the interfaces on Nodes A, B, C, D & E? Was each node able to successfully ping all the other nodes? Hand in the ns script that defines this new topology.

In the next step we shall use Click to make this new node E work as an Ethernet switch, so make sure that this new node E is booted with an Operating System that Click can run on (preferably, Linux).

B. The Click Modular Router

- The *Click Modular Router Project* home page is at http://www.read.cs.ucla.edu/click/ .
  - You can download the gzip tarball of sources, *click-2.0.1.tar.gz*, from the *Download Click releases* link (http://www.read.cs.ucla.edu/click/download) on the *Click Project* home page. (You should not need to also download the *click-packages-2.0.tar.gz* tarball.)

    Download *click-2.0.1.tar.gz* to your own machine first and from there you can ftp it over to node E by ssh'ing in to that node.
  - The following tutorial on Installing Click should prove useful: http://www.pats.ua.ac.be/software/click/click-2.0/installation.pdf .
  - There is also a tutorial available from the same source that gives an overview of the Click Router concepts: http://www.pats.ua.ac.be/software/click/click-2.0/concepts.pdf .

- We want to configure the node E that you introduced into the Emulab Tutorial topology so that it works as an Ethernet switch. Fortunately, Click already has an *EtherSwitch* element defined, so in this respect most of the hard work is already done. We "just" have to figure out how to set it up. (There is also a *ListenEtherSwitch* which might also be helpful, particularly for debugging.)

  Documentation on Click elements can be found at the *Elements* link (http://www.read.cs.ucla.edu/click/elements) on the *Click Modular Router Project* home page.

- There is quite a bit of documentation that comes with the Click package once you have installed it, as well as links to various *Online documentation* at the *Click Modular Router Project* home page.

B1. Report on what you had to do to install and correctly configure Click as an Ethernet switch on node E. Include the Click configuration scripts you had to develop.

- Note that when you ran the original Emulab Tutorial topology with the *lan0* hub, nodes B (more specifically, the node B interface connected to *lan0*), C & D were assigned IP addresses that showed them as belonging to the same IP subnetwork. Node A (and the node B interface connected to node A) were assigned IP addresses showing them to belong to a second, distinct IP subnetwork.

  When you introduced node E in place of the *lan0* hub, the Emulab experiment manager will have by default assigned the node B interface connected to node E, and the interfaces at nodes C, D & E, with IP addresses from distinct IP subnetworks; it will also have populated the forwarding tables of nodes B, C, D & E with static routing entries accordingly.

Since node E is now an Ethernet switch and not an IP-capable node, we will want to make sure that: (i) the interfaces of nodes B, C & D connected to this Ethernet switch are assigned IP addresses that show them as belonging to the same IP subnetwork; (ii) the entries in the forwarding tables of nodes B, C & D reflect the new IP address assignments; (iii) all IP address assignments are removed from the interfaces of node E; and (iv) node E's forwarding table is removed/purged/disabled entirely.

- The instructions at https://wiki.emulab.net/wiki/nscommands (see the section entitled *IP Address Commands* on that page) explain how to automate IP address assignment in the ns experiment script itself.
- The Unix/Linux command **`netstat -r -n`** will permit you to view you the forwarding table at a node, and the command **`route`** will permit you to edit its entries.

- Log into nodes A, B, C & D and make sure they are able to ping each other.
  - You might find at this stage that you are still unable to ping nodes across the Ethernet switch (using **`traceroute`** might be marginally helpful here). Why might that be? How does an Ethernet interface of the switch have to be configured so that it is able to listen to and read **all** frames on the wire? How do you achieve such a configuration?

B2. Report on the new IP address assignments in your topology, giving the IP address for each interface (an annotated topology diagram would be a nice, succinct way of doing this). How did you implement these new IP address assignments? Show the forwarding tables at nodes A, B, C & D and explain what you did to configure them correctly with respect to the IP address assignments. Explain what you had to do to the interfaces of Ethernet switch node E, and how, in order to be able to ping across the switch. Explain what you did about the forwarding table of node E. Demonstrate that each of nodes A, B, C & D was able to ping the other nodes. Include the Emulab ns script that sets up the topology with the new IP address assignments.

C. Some 'Rough & Ready' Performance Evaluation

- We shall use the venerable tool ***ttcp*** to do some rough and ready TCP and UDP performance evaluation across the span of our topology (e.g., between nodes A & C, for example, or between nodes A & D).
  - Do a Google search for "ttcp" and read whatever looks interesting. In particular, look for the Wikipedia article on ttcp and The Story of the TTCP Program. I am not sure if ttcp is available on the Emulab nodes; if not you will have to figure out how to download and install it on node A, and on either node C or node D.
  - You will also, of course, have to figure out how to use ttcp for your TCP and UDP experiments below. Note that this might entail having to mess about with the buffer sizes (see, for example, Computing the TCP Buffer Size; also read further on beyond just that section in the document for some potentially useful further guidance).

- You should carry out four sets of experiments using ttcp across the span of the topology (i.e., between either nodes A & C, or nodes A & D; you need not do both). Each set should include testing with both TCP and UDP traffic. The experiments need be neither extensive nor exhaustive. They should, however be carefully thought out (a couple of well structured experiments of good quality is preferable to a gazillion ill thought-out, random ones), and should involve similar ttcp configurations for the four sets so that we can draw reasonable comparisons between TCP and UDP on the one hand, and between the results across the four sets, on the other.
  - **Set 1:** using the loopback interface at node A. This will give us throughput results on the same computer and serve as a kind of baseline for the results.

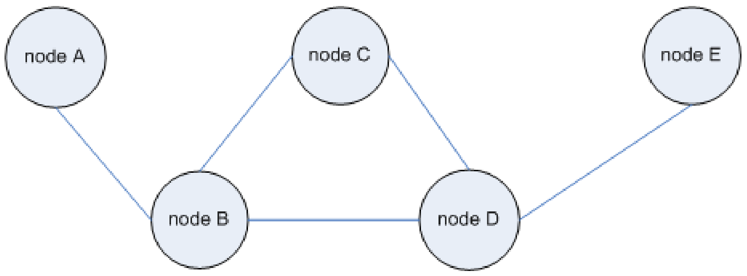
    Start one copy of ttcp in receive mode and another in transmit mode on node A. I suggest you use the -s option so that ttcp will automatically source (that is, create) and sink (discard) its own data. You can use the -n option to tell ttcp how many buffers of data to send; this should be sufficiently large that the transfer takes tens of seconds. For example, for the TCP experiment subset:

    ```
    > ttcp -r -s &
    [1] 2121
    > ttcp -t -s -n 100000 localhost
    ```
  - **Set 2:** using the original topology of the Emulab Tutorial with the *lan0* hub. Start the ttcp receiver on either node C or D and the ttcp sender on node A.
  - **Set 3:** using the topology with node E that you implemented just before step **A1.** above (i.e., before configuring node E as an Ethernet switch using Click). Use the same send node and the same receive node as for Set 2.
  - **Set 4:** using the topology with node E configured as a Click Ethernet switch. Use the same send node and the same receive node as for Set 2.

C1. Report on the ttcp configurations you used for your experiments, how you determined and set the buffer sizes, the size of traffic you sent, etc. Report on the TCP and UDP throughput rates for your traffic – use tables, graphs and/or bar charts to do this so that the comparative results between TCP and UDP on the one hand, and between the four sets, on the other, are presented in a clear, succinct, easy-to-absorb way. What do you conclude about TCP vs. UDP in general from your results? Comparing the results of the four sets with each other, can you reach any conclusions about the nature of extra processing overhead delays involved as node E changes from being an emulated *lan0* hub (Set 2) to a statically-configured router node (Set 3) to a Click configured Ethernet switch (Set 3)? Why do you think it is that I asked you to use node A as the sender and one of nodes C or D as the receiver, rather than vice-versa? In what way might this help contribute more clarity to your results?

D. Experiments with RIP

- We shall now experiment with the RIP intradomain routing protocol using the topology shown in the figure below in which all links are {100Mbps, 0 propagation delay, 0 loss}, and for which you have to develop the Emulab ns script.



- In this topology the Emulab experiment manager will by default assign the nodes with IP addresses from distinct IP subnetworks. Effectively, nodes A & E will be acting as "gateway" routers that belong to LAN subnetworks whose other nodes are not represented in the topology, and nodes B, C & D will be acting as routers that provide connectivity between these two LANs.
- Note however that, unlike the Emulab ns scripts you have been using thus far, you will have include a line that says `$ns rtproto Manual` rather than `$ns rtproto Static`, so that the Emulab experiment manager does not set up static routing for the topology but rather leaves it up to you to define your own routing.

- Run the RIP software on all the nodes.
  - See Configure and Run RIP Software on an Emulab for how to do this, and note with respect to this documentation that:
    - *routed* is the Unix/Linux name for the RIP process;
    - we are using the Emulab at the University of Utah.
  - Use `netstat -r -n`, `ping`, and `traceroute` to examine your forwarding tables and to check out your network. Make sure that each node can reach all the other nodes. In particular, check that the path between nodes A & E is the three-hop path A-B-D-E and not the four-hop path A-B-C-D-E.

D1. Report on the IP address assignment in your topology (again, an annotated topology diagram would be a nice, succinct way of doing this), and how you activated and configured RIP on your nodes. Show the forwarding tables. Was each node able to successfully ping all the other nodes? Provide the traceroute output that gives the path between nodes A & E. Hand in the ns script that defines the topology.

- Now, bring down the link B-D and observe how quickly connectivity is re-established between nodes A & E via the path A-B-C-D-E.
  - Make sure that you do **not** start the experiment with the link B-D down. The idea is to start with all links up, wait until RIP has discovered all routes (including the A-B-D-E path between nodes A & E) and has stabilized, then bring down the link B-D and have RIP readjust to the new situation.
  - Refer to the Emulab Tutorial - A More Advance Example on how you can go about doing this. In particular, see the sections:
    - *Dynamic Scheduling of Events*, and *Supported Events*.

  - Also, the section *Link Tracing and Monitoring* shows, as it name says, how you can trace and monitor links.

D2. Report on how you made the link B-D go down, and at what time. How did you determine that time? If you made the link B-D go down by means of your ns script, then hand in this modified script. What was the evolution of RIP readjusting to the new situation and how quickly was connectivity re-established to node E via the B-C-D path? Show the forwarding tables. Provide the traceroute output that gives the new path between nodes A & E.

- Restart your experiment again, with all links up. Wait until RIP has discovered all routes and stabilized, as before. Now bring down the link D-E and observe the count-to-infinity "dance" that takes place between nodes B, C & D with respect to routing to the now unreachable node E. Note that RIP can be slow to converge, especially for this kind of scenario. It may take several minutes, so be patient!

D3. Report on how you made the link D-E go down, and at what time. How did you determine that time? What was the evolution of RIP readjusting to the new situation, and how long did it take nodes A, B, C & D to finally realize that node E was unreachable? Show this count-to-infinity evolution by taking one node of your choice as an example and giving the evolution of its forwarding table.