Version 04/14/2015

Extra Credit Assignment II for ECE 375
Learning Switch

Due: 05/10/2015 at midnight

Note: If you get full credits for this assignment you can improve your final grade by 5%

Preparations:
To be able to carry out this experiment you initially have to carry out the steps listed below.
Note: If you have successfully performed Extra Assignment I you can ignore the information given under "Account" and "Credentials" below.

Account:

1. Go to https://portal.geni.net/ press the "Use GENI" button and log in with your UMass OIT credentials.
2. Click "Join a Project" and select "UMASS-ECE374".
3. I will then get a notification and invite you to join the project.

Credentials:

1. Open browser and go to:
   https://portal.geni.net/
2. Log in using your UMass OIT credentials.
3. When you are logged in, select "Profile" from the top, right menu.
4. Next, click on "Generate SSH keypair" and enter your passphrase. Make sure you remember that passphrase.
5. On the next page click on "Download Private Key" and save key to ~/.ssh locally.
6. In the browser, still on the same page click on the link "SSL".
7. Download the certificate to ~/.ssl locally.
8. Make sure the files that hold your secret keys are adequately protected. To make sure that's the case you can execute the following commands:
   ```
   chmod 0600 .ssh/id_geni_ssh_rsa
   chmod 0600 .ssl/geni-mzink.pem
   ```
   (remember that your .pem file is named differently)

Congratulations. You know have set up everything to get started with carrying out your experiment.

Setting up your slice through GENI Portal:
You can reserve the resources either through the Portal.

The link below shows a video that teaches you how to setup a GENI slice for Extra Assignment I:

http://server.casa.umass.edu/~zink/ECE374/recordings/assign1_topo_setip.mp4

In the case of this assignment the only thing that is different is the name of the RSpec you have to choose. So instead of using "ECE3743Node" (as shown around 1:50 minutes into the video) you now have to use RSpec titled "ECE374_UMass_EG_Assignment2". **This is very important!** If you don't choose the right RSpec you will not succeed in finishing this assignment!

Resource reservation:

1) When you are logged in, select the "Home" tab from the top, right menu.
2) Click on the project name "UMASS-ECE374".
3) Next, click on the "Create Slice" button.
4) On the next page, give a name and description for the slice and click "Create Slice" button.
5) You will then get a message confirming the creation of the slice.
6) Click on the Slice page. Click on "Add Resources"
7) Select "ECE374_Umass_EG_Assignment2" from the list of RSpecs.
8) Click on "Site" in the GUI (Jacks) . You can use any of the **ExoGENI** aggregate managers shown on the list.
9) After a while, you will be able to see the topology. The node switch will be connected to five other nodes as shown in the screen shot below.

Congratulations! You are ready to start the experiment.

Experiment

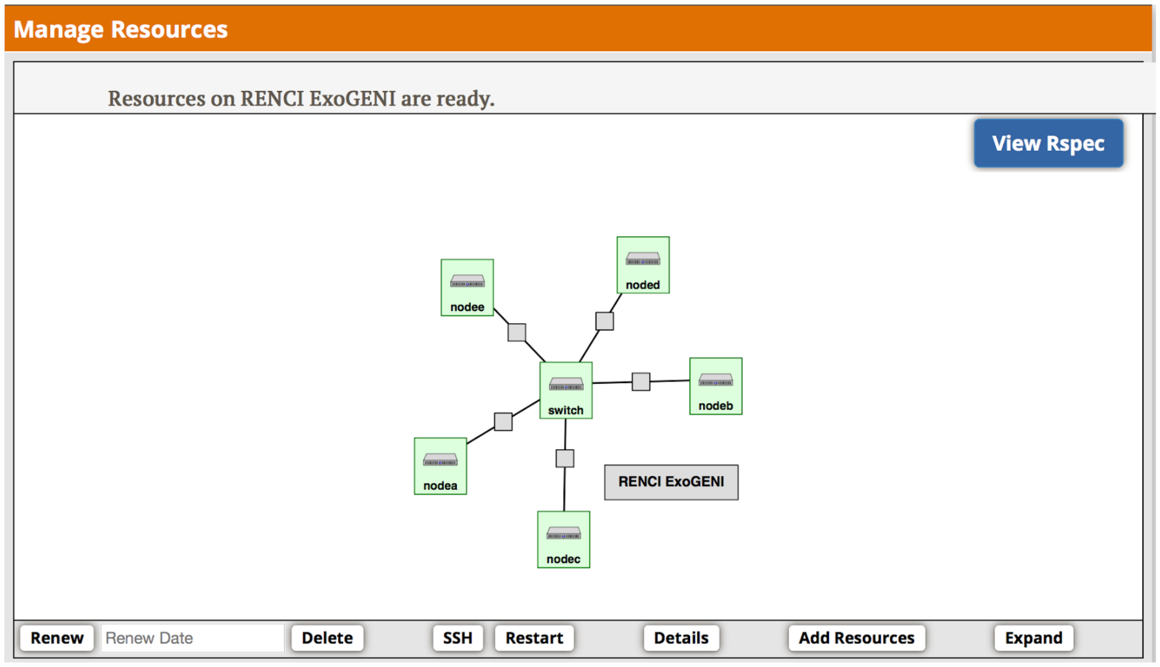


Figure A

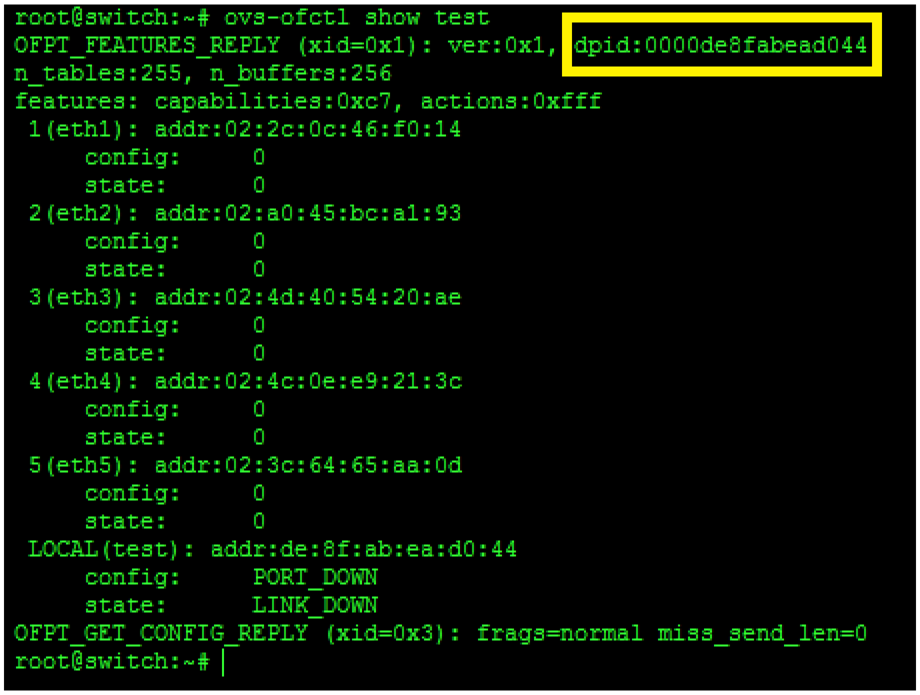
The goal of this assignment is to implement the learning switch capability that is used by Ethernet switches (see Chapter 5 slides 63-66) by using a software-based OpenFlow switch. In the case of the topology shown in Figure A, this software switch is to be implemented in node "switch". All the other nodes represent regular hosts. To realize this implementation of a learning switch it is your task to implement a trema-based OpenFlow controller. You will have to verify the correct functionality of the learning switch by creating an experiment script in which node A pings all other nodes (B – E) in LabWiki.

Working:

In the topology shown in figure A, the switch node will act as the learning switch, which connect nodes A-E with each other. Whenever a regular node pings any regular node for the first time, the switch does not know the destination address of the node. At this point, the switch node floods the packet to all the nodes it is connected to. When it receives a reply to the flood, this information will be used to populate its switching table. Hereafter, whenever any node wants to ping any other nodes, the switch node will fetch the information from the table. For this purpose, two scripts are already preloaded in the switch node, namely learning-switch.rb and fdb.rb. The second script offers rudimentary database functionality that is used to maintain the switching table.

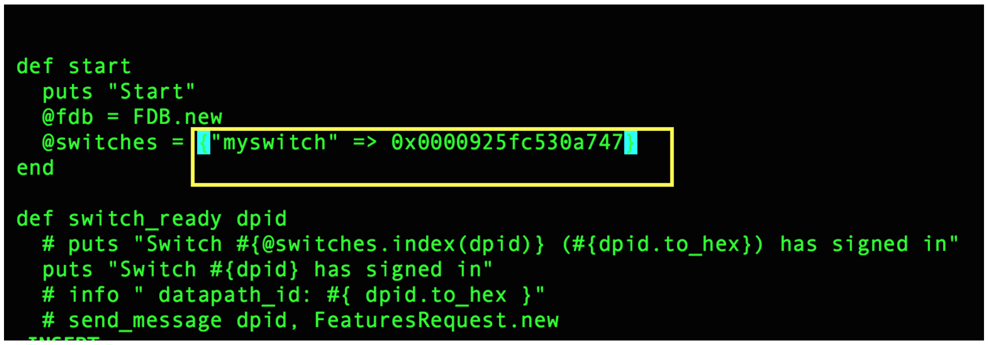
Step-by-Step Instructions

1. The first step is to add the openflow switch and configure the bridge with interfaces. To do so, SSH into the "switch" node and execute the following commands:
   - **a) sudo bash**
     source /etc/bash/bash.bashrc
     ifconfig
     Check if all the five interfaces from eth1-eth5 are up in the switch node.
   - **b) ovs-vsctl add-br test**
     This command creates an ovs bridge named test. Any name can be given for the bridge.
   - **c) ovs-vsctl add-port test eth1**
     ovs-vsctl add-port test eth2
     ovs-vsctl add-port test eth3
     ovs-vsctl add-port test eth4
     ovs-vsctl add-port test eth5
     These commands configure the bridge with the interfaces. eth1-eth5 specifies the interfaces on node "switch"
   - **d) ovs-vsctl set-controller test tcp:<IP.OF.CONTROLLER>:6653**
     In this case, the IP.OF.CONTROLLER should be set to 127.0.0.1
   - **e) ovs-ofctl show test**
     dpid is "data path id" that uniquely identifies an OpenFlow instance on the device. This command gives the 16 digit dpid. Make note of this dpid.



```
root@switch:~# ovs-ofctl show test
OFPT_FEATURES_REPLY (xid=0x1): ver:0x1, dpid:0000de8fabead044
n_tables:255, n_buffers:256
features: capabilities:0xc7, actions:0xfff
 1(eth1): addr:02:2c:0c:46:f0:14
     config:     0
     state:      0
 2(eth2): addr:02:a0:45:bc:a1:93
     config:     0
     state:      0
 3(eth3): addr:02:4d:40:54:20:ae
     config:     0
     state:      0
 4(eth4): addr:02:4c:0e:e9:21:3c
     config:     0
     state:      0
 5(eth5): addr:02:3c:64:65:aa:0d
     config:     0
     state:      0
 LOCAL(test): addr:de:8f:ab:ea:d0:44
     config:     PORT_DOWN
     state:      LINK_DOWN
OFPT_GET_CONFIG_REPLY (xid=0x3): frags=normal miss_send_len=0
root@switch:~#
```

f) We have already installed a template for the OpenFlow controller in /root. The name of the template is “learning-switch_copy.rb”. Open this file using any text editor and replace the copied dpid into this file.



```
def start
  puts "Start"
  @fdb = FDB.new
  @switches = {"myswitch" => 0x0000925fc530a747}
end

def switch_ready dpid
  # puts "Switch #{@switches.index(dpid)} (#{dpid.to_hex}) has signed in"
  puts "Switch #{dpid} has signed in"
  # info " datapath_id: #{ dpid.to_hex }"
  # send_message dpid, FeaturesRequest.new
```

2. This script is used to implement the learning switch functionality. To achieve that you have to add additional functionality to the “def packet_in datapath_id, message” routine. That is the only routine that you need to work on. The others are all set. This routine implements the functionality required when a packet enters the switch.
3. The controller makes use of a simple database implementation, which is installed in the same directory as the controller. The name of the script is “fdb.rb”. This is where the information is populated on flooding to all the nodes.
4. You can start the controller with the following commands:
   trema run /root/learning-switch_copy.rb
5. After you have started the controller switch over to LabWiki. To prove the functionality of your learning switch controller re-use the “ECE374_assignment2.oedl” experiment script. This script does automatic pinging between nodes.
6. Modify that script such that node A pings all other hosts (nodes B – E).

Further information:

- Trema: http://trema.github.io/trema/
- OpenFlow: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf