CS 4480: Computer Networks - Spring 2020

Programming Assignment: 2
Software Defined Networking
Due dates as indicated in Canvas

1 Overview

The primary goal of this assignment is to become familiar with software defined networking (SDN) concepts by developing a simple OpenFlow application using the Ryu SDN framework [1]. Your SDN application will realize a simple "virtual IP load balancing switch". (I.e., a switch that will map, in round-robin fashion, a virtual IP address to a set of real IP addresses associated with servers "behind" it.) You will make use of two testbed environments for this assignment. First, you will be using a virtual networking environment called Mininet [7]. Mininet allows the emulation of arbitrary network topologies inside a single physical or virtual machine. Of specific interest for our purposes is the fact that Mininet can instantiate OpenFlow capable switches in this emulated environment. Second, to simplify the task of setting up Mininet, we will make use of the Emulab testbed environment [2]. Specifically, we have created a virtual machine (VM) profile in Emulab, which contains both the Mininet environment and the Ryu framework.[1] Each student can use this profile to easily instantiated (their own) VM for the assignment.

2 Setup

Figure 1 depicts the setup you will use for developing your application. (And the setup that will be used for evaluating your application.)

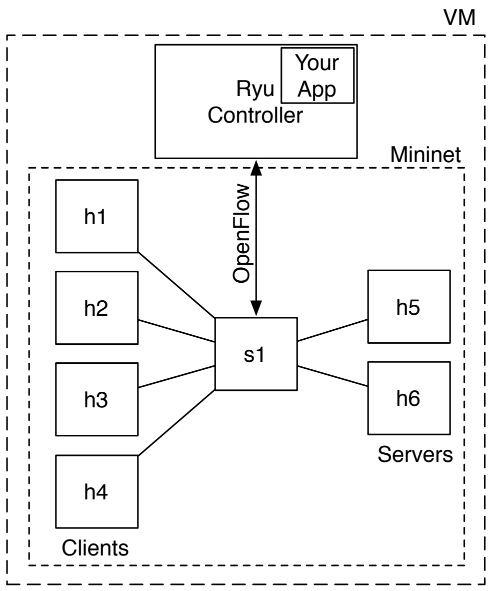
Specifically: You will instantiate a VM containing the Mininet and Ryu frameworks in Emulab. Once your VM is operational you can instantiate a Mininet instance using the mn command shown in the figure. As shown, this command instantiates a single switch ($s1$) and six hosts ($h1$ through $h6$). Next you should use the shown $ovs - vsctl$ command to configure the switch to use OpenFlow version 1.3. Finally, once you have written your SDN application, you will use the shown $ryu - manager$ command to instantiate the Ryu controller and your application.

3 SDN Application

The purpose of your SDN application is to realize a simple "virtual IP load balancing switch". I.e., a switch that will map, in round-robin fashion, a virtual IP address to a set of real IP addresses associated with servers "behind" it.

With reference to Figure 1, for the purposes of this assignment we will assume that switch $s1$ is fronting two servers, $h5$ and $h6$. We will treat the four other hosts, i.e., $h1$ through $h4$ as clients. When one of the client hosts sends traffic to a known virtual IP address, i.e., an address that is not actually associated with any of the host interfaces in the system, your application will map the virtual IP address to one of the server IP addresses in round-robin fashion. Similarly, for traffic returned by the server, the switch will perform the mapping in reverse. I.e., from the perspective of the client hosts, they will all be communicating with a single server (associated with the virtual IP address). In reality, however, each client will be communicating with one of the two "real" servers behind the switch.

[1]The Mininet VM was created by graduate student Binh Nguyen.



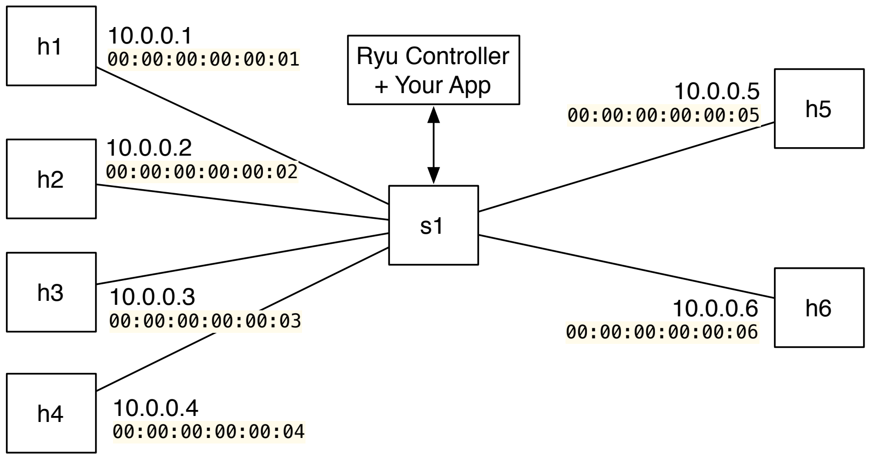
```
// In first shell, start up Mininet:
sudo mn --topo single,6 --mac --controller remote --switch ovsk,protocols=OpenFlow13

// In second shell, configure switch for OpenFlow 1.3:
sudo ovs-vsctl set bridge s1 protocols=OpenFlow13

// In second shell, run Ryu with Your App,
// (assuming your app is your_app.py and copied into ryu/ryu/app):
cd ./ryu && ./bin/ryu-manager --verbose ryu/app/your_app.py
```

Figure 1: VM and Mininet setup

To simplify your SDN application you only need to support ICMP traffic, i.e., *ping*. Further, we will make use of the fact that communication between two hosts in the same subnet is preceded by an ARP request (to map the IP address to a MAC address) to trigger assigning the (new) traffic (which is to follow the ARP request) to the "next" server *and* to set up the appropriate forwarding rules in the switch. I.e., each ARP request from a client node should result in mapping the virtual IP address to the "next" real IP address. (Or in the case with only two real servers, the mapping would alternate between the two servers.)

This process and sequence of events are depicted for an example exchange in Figure 2. Assume host $h1$ issues a ping to the virtual IP address 10.0.0.10 (step 1). The networking stack on host $h1$ will issue an ARP request to map the virtual IP address (10.0.0.10) to a MAC address (step 2). Your SDN application will intercept this ARP request, make a selection of which real server to use and send an ARP response with the appropriate MAC address back to the requesting host (step 3). E.g., this example assumes that host $h5$ was selected and the MAC address associated with it is returned in the response. In addition your application will push the appropriate forwarding rules into the switch to facilitate mapping between the virtual and real IP addresses (step 4). Specifically, as shown in the figure, you will need to set up a rule for traffic from $h1$ to $h5$ as well as in the reverse direction. In the example shown, for the $h1$ to $h5$ direction, you will match on the switch port associated with host $h1$ and the virtual destination IP address, and then set the destination address to be that of $h5$ and forward the packet out on the port associated with $h5$. A similar rule is pushed for traffic in the reverse direction. At this point ICMP echo request and response traffic will be able to flow between hosts $h1$ and $h5$ (steps 5 and 6).

Note that for $h5$ to be able to communicate with $h1$ (i.e., respond to the ping), it will issue a similar



1: h1 ping 10.0.0.10
// user typed command

2: ARP Request who-has 10.0.0.10 tell 10.0.0.1
// h1's network stack

3: ARP Reply 10.0.0.10 is-at 00:00:00:00:00:05
// Your app

4: Push OF rules on s1:
- h1 to h5
match:
inport=h1-port, dst-ip=10.0.0.10
action:
set: dst-ip=10.0.0.5
output: h5-port
- h5 to h1
match:
inport=h5-port, src-ip=10.0.0.5, dst-ip=10.0.0.1
action:
set: src-ip=10.0.0.10
output: h1-port
// Your app

5: ICMP echo request
// h1's network stack

6: ICMP echo response
// h5's network stack

Figure 2: Operation: h1 ping “virtual IP” (10.0.0.10)

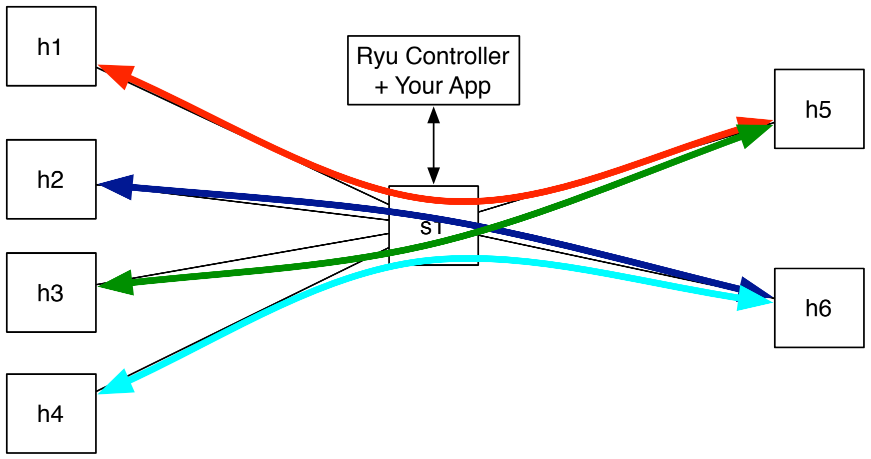


Figure 3: Operation: h1 through h4 ping "virtual IP" (10.0.0.10)

ARP request in the opposite direction, (i.e., to find a MAC address associated with $h1's$ IP address). Your solution will have to accomodate this for the communication to succeed. The best solution to this would be for your application to also intercept ARP requests in this direction, and respond with the appropriate MAC address.

Figure 3 shows the resulting interaction assuming that hosts $h1$ to $h4$ issued ping commands to the virtual IP address in sequence. I.e., $h1$ is mapped to $h5$, $h2$ is mapped to $h6$, $h3$ is again mapped to $h5$ etc. Such a sequence of ping commands will be used to verify the functionality of your application during evaluation.

4 Approach

This is a non-trivial assignment which will acquaint you with a variety of concepts, several of which might be new. It is strongly recommended that you start early and follow a staged approach to work through all the needed materials. Suggested/required approach is outlined in the following sections.

4.1 Become familiar with Mininet, OpenFlow and Ryu

- Read through the complete assignment so that you know what is expected, e.g., in terms of things to capture for your progress report.
- Follow the steps described in Section 6 to sign up for an Emulab account and instantiate the Mininet VM profile.
- Work through (not just read through) the "Mininet Walkthrough" [8] and the "Introduction to Mininet" [6]. Main objectives are to get a practical understanding of Mininet, its namespace isolation, how it fits into the SDN VM, how to use Mininet commandline options, how to script Mininet commands, how to run an external controller (e.g., one running on the VM itself), how to clean things up when something goes wrong, how to monitor both data plane and control plane interaction using Wireshark etc.

  Note:

  - Mininet is already installed on your VM.
  - To run wireshark and xterms on Mininet hosts you will have to access your VM via an xterm capable ssh terminal, *not* via a browser shell. See Section 6.

- Work through the SDN Hub OpenFlow Tutorial [3].
- Work through the SDN Ryu Controller Tutorial [4].
- Work through the "Welcome to RYU the Network Operating System (NOS)" document [5].
- Work through some of the example controller applications provided as part of the tutorial and the Ryu documentation.
- **Submit first progress report**. See details of what to submit in Section 5.

4.2 Core assignment execution

- A reasonable approach might be to first figure out how to set the necessary flows without using a virtual IP address and using the OpenFlow commandline tool (ovs-ofctl). Then do the same with a Ryu application. Then add the ARP intercept functionality. Then update the rules to make use of a virtual IP address and perform the necessary IP address mapping.
- **Submit application code**. See details of what to submit in Section 5.

5 Grading and Evaluation

5.1 What to hand in

Note: Reports should be properly typed and submitted as pdf files. (Submitting a hand written scanned pdf is not acceptable.)

Progress Report The progress report needs to be submitted via Canvas (on the due date stated in Canvas). The report should briefly summarize what you have done up to that point to familiarize yourself with Mininet, Openflow and Ryu. The progress report submission should be in pdf. In addition to the progress summary your report should contain a screenshot for each of the four "work through" activities listed in Section 4.1, showing and describing some output of that activity. Note that the screenshot should clearly show the output in *your experiment (i.e., your VM) on Emulab*. For example, if you tried out the example Mininet script in "Introduction to Mininet", you might include a screenshot and description as shown in Figure 4.

Completed Assignment Your completed assignment will consist of:

- A single python file which implements your SDN application, **submitted via handin on Cade**.

  You should use the following naming convention for your python file:

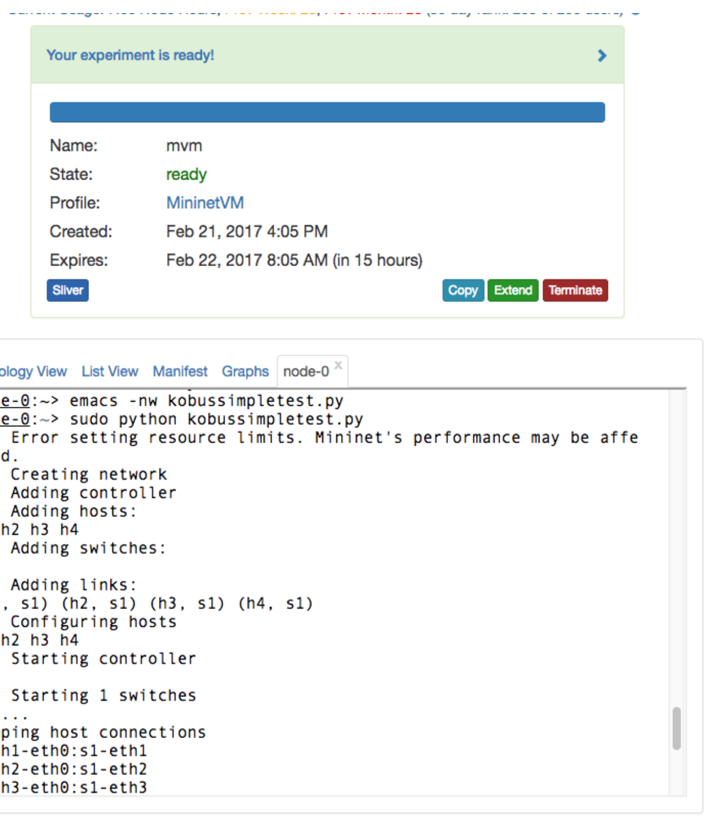
  ```
  Firstname_Lastname_UID.py
  e.g.,
  Joe_Doe_u0000000.py
  ```

  For evaluation purposes the TAs will copy your python file into the *ryu/ryu/app* directory and execute it with the Mininet setup shown in Figure 1.

  To electronically submit files while logged in to a CADE machine, use:

  ```
  % handin cs4480 assignment_name name_of_file
  ```

  where `cs4480` is the name of the class account and `assignment_name` (pa1_a, pa1_final etc.) is the name of the appropriate subdirectory in the handin directory. Use pa2 for this assignment.



Name: mvm
State: ready
Profile: MininetVM
Created: Feb 21, 2017 4:05 PM
Expires: Feb 22, 2017 8:05 AM (in 15 hours)

```
node-0:~> emacs -nw kobussimpletest.py
node-0:~> sudo python kobussimpletest.py
*** Error setting resource limits. Mininet's performance may be affe
cted.
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
```

Figure 4: Execution of example Mininet script from “Introduction to Mininet”

5.2 Grading

Criteria	Points
Progress report	10
Code structure and inline documentation	10
Code executes as described	80
Total	100

6 Signing up with Emulab

- Note: *The Chrome browser seems to work best.*
- Go to: `https://www.emulab.net/portal/signup.php`
- Select **Join Existing Project** and enter **CS4480-2020** for the Project Name.
- Enter your personal information on the left.
  - Use a valid email address. It is used during the signup process.
  - (Optional) Generate and load a valid SSH Public Key file. This makes interaction with your resources in Emulab a lot easier. For example, you can run xterms and other X11-based applications. (Note: If you are working on a Windows system you would need to download and install terminal software, e.g., `http://mobaxterm.mobatek.net`, to run X11-based applications (and generate ssh keys).)
- Hit the **Submit Request** button.
- When your account is approved you will be able to log in: `https://www.emulab.net/portal/login.php`
- Once logged in follow these steps to instantiate a VM with Mininet and Ryu pre-installed:
  - Click on the **Experiments** button at the top and select **Start Experiment**.
  - On the **1. Select a Profile** page, click the **Change Profile** button (bottom right).
  - In the **Select a Profile** window, enter **MininetVM1** in the search box (top left) and hit enter.
  - Verify that the MininetVM1 profile is selected (should display the fact that this is a VM with Ryu, Mininet and OVS installed) and hit the **Select Profile** button (bottom right).
  - Back on the **1. Select a Profile** page, select **Next**.
  - On the **3. Finalize** page, provide: (i) a name for your experiment (optional), (ii) Select CS4480-2020 as the project (if you are associated with only one project it will be selected automatically).
  - On the **4. Schedule** page, stay with the default options and select the **Finish** button (bottom right).
  - Your experiment will be instantiated by Emulab. This will take a while (approximately 5 minutes). Wait till the status page says that **Your experiment is ready!** before trying to interact with it.
- Once your experiment is ready, you can log into your VM:
  - If you have uploaded a public SSH key you should be able to click on the **List View** tab, then click on the black SSH command associated with **node-0** and have a terminal window pop up and ssh into your VM.

    Note: If a terminal window doesn't start automatically, (e.g., might be the case if you use terminal software on Windows), you can manually start a terminal window and then copy, past and execute the SSH command shown in black to ssh into your VM.

  - Alternatively, you can click on the blue **Actions** button on the **node-0** line and select **Shell**. This will open a shell directly in the browser.

- At this point you should be ready to start exploring MiniNet, OpenFlow and Ryu!

7 Additional notes

Additional notes provided by graduate student Josh Kunz who tried out the assignment:

- Note that you can't use the arp target or source address in a 'set-field' action. When I tried doing it the switch kept sending back a BAD_ARGUMENT error.
- One thing that tripped me up was trying to do the OFP_PACKET_OUT messages to send the arp responses back from the controller. If you want to send the packets out the same port they came in on you have to specify the special OFPP_IN_PORT as the target port of the OFPActionOutput action and set the "in_port" properly in the OFPPacketOut message.
- In the OFPPacketOut message you should always set buffer_id=OFP_NO_BUFFER
- There's a useful-but-hackey way to copy a packet in ryu by doing:

```
p.serialize()
p_copy = packet.Packet(p.data[:])
```

- Here is a list of reference material I found useful:
  - `https://ryu.readthedocs.io/en/latest/library_packet.html#introduction`
  - `https://ryu.readthedocs.io/en/latest/library_packet_ref.html`
  - `https://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html`
  - `https://github.com/osrg/ryu/blob/master/ryu/app/simple_switch.py`
  - `https://github.com/osrg/ryu/tree/master/ryu/lib/packet`
  - `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.5.pdf`

References

[1] Build SDN Agilely. `https://osrg.github.io/ryu/`.

[2] Emulab. `https://www.emulab.net/portal`.

[3] OpenFlow version 1.3 tutorial,. `http://sdnhub.org/tutorials/openflow-1-3/`.

[4] RYU Controller Tutorial,. `http://sdnhub.org/tutorials/ryu/`.

[5] Welcome to RYU the Network Operating System(NOS). `http://ryu.readthedocs.io/en/latest/index.html`.

[6] Lantz, B., Handigol, N., Heller, B., and Jeyakumar, V. Introduction to Mininet. `https://github.com/mininet/mininet/wiki/Introduction-to-Mininet`.

[7] Mininet Team. Mininet - An Instant Virtual Network on your Laptop (or other PC). `http://mininet.org`.

[8] Mininet Team. Mininet Walkthrough. `http://mininet.org/walkthrough/`.