Software-Defined Networking

Lab 3

OpenFlow and Wireshark

University of Colorado Boulder

Department of Computer Science

Professor Levi Perigo, Ph.D.

# Lab Summary

Understanding the OpenFlow protocol is critical to understanding SDN. The purpose of this lab is to capture and analyze the OpenFlow protocol messages using Wireshark. Knowing how OpenFlow works and being able to troubleshoot different scenarios within Wireshark, will prove to be a valuable skillset in industry.

# Objective 1 – X11 Forwarding: Setup the SDN (controller/switch) Environment

1. Run Xming software on your laptop.
2. Start the controllers VM.
3. Verify that your local machine can talk to the controllers VM.
4. Run ODL from the CLI of the SSH session on your local machine (this may take a few min. to initialize (use the Lab 0 document for assistance)).
   1. Provide a screenshot of ODL running on the CLI of the VM. [**5 points**]

A screenshot of a computer program

Description automatically generated

1. Enable the ODL web interface and all the features (use the Lab 0 document for assistance).
2. Login to the ODL web interface.
   1. Provide a screenshot of your local machine at the ODL home page. [**10 points**]

A screenshot of a video

Description automatically generated

1. Start a new PuTTy/SSH session to the OpenDaylight controller.
   1. Before you connect, in PuTTy go to Connection > SSH > X11.
   2. Check the “Enable X11 forwarding” box.
      1. Note: If using MAC/Linux you can simply SSH with the “-X”.
   3. Initiate the PuTTy connection.
   4. Install and initialize Wireshark on the controllers VM (use the Lab 0 document for assistance).
   5. In the new Wireshark window when you go to “Interfaces” it should be the interfaces on the controller (not your local machine).
      1. Provide a screenshot of the Wireshark interfaces that indicates that Wireshark is running on the controller. [**10 points**]

A screenshot of a computer

Description automatically generated

1. Start the Mininet VM.
2. Verify that your local machine can talk to the Mininet VM.
3. Start a new PuTTy/SSH session to the Mininet VM (using the X11 steps above).
   1. Login to Mininet.
   2. Start a new Mininet topology that connects to a remote controller (ODL), uses easy to read MAC addresses, uses OpenFlow v1.3, and uses a network topology that has a single switch and four hosts.
      1. Provide a screenshot of the command used. [**5 points**]

sudo mn --topo=single,4 --mac --controller=remote,ip=10.224.77.108 --switch ovsk,protocols=OpenFlow13

* 1. Within Mininet, issue the command: xterm h1 h2 h3 h4.
     1. Explain what happened, what each of these new windows are, why this would be beneficial; also provide a screenshot of all four windows (proving you got it working). [**10 points**]

These are windows of the actual hosts CLI operating systems. This would be beneficial because you could execute more commands within the actual hosts and not try to troubleshoot by just the mininet commands.

A screenshot of a computer

Description automatically generated

# Objective 2 - Capture OpenFlow Messages

1. Open Wireshark on the controller VM (use steps from Objective 1).
2. Create a display filter in Wireshark to only show OpenFlow version 1.3 messages.
   1. What Wireshark Filter would you use to capture OpenFlow version 1.3 messages? [**3 points**]
      1. Openflow\_v4
3. Start capturing traffic (only showing OpenFlow version 1.3 messages).
4. Start a new Mininet topology that connects to a remote controller (ODL), uses easy to read MAC addresses, uses OpenFlow v1.3, and uses a network topology that has a single switch and two hosts.
   * 1. Provide a screenshot of the command used. [**5 points**]

sudo mn --topo=single,2 --mac --controller=remote,ip=10.224.77.108 --switch ovsk,protocols=OpenFlow13

1. Within Mininet, issue a single ping from Host 1 to Host 2.
   1. After the ping has completed, stop the Wireshark capture.
   2. Provide the command used to ping from Host 1 to Host 2. [**2 points**]
      1. H1 ping h2
   3. Provide a screenshot of Wireshark only displaying OpenFlow messages during the Setup/Ping. [**2 points**]

A screenshot of a computer

Description automatically generated

* 1. What is the logical interface that connects each OpenFlow switch to a controller? [**5 points**]
     1. It is called the OpenFlow channel

1. Which device (switch or controller) initiated the OpenFlow connection?
   1. Provide a screenshot of the capture that indicates this. [**2 points**]

Switch

A screenshot of a computer

Description automatically generated

1. Within Wireshark indicate the messages from the switch and the controller that indicate which version of OpenFlow they support.
   1. Provide a screenshot of each. [**10 points**]

Controller:

A screenshot of a computer

Description automatically generated

Switch:

A screenshot of a computer

Description automatically generated

1. Within Wireshark indicate the message that shows what features the switch supports.
   1. Provide a screenshot of the Wireshark message, indicating the features. [**5 points**]

A screenshot of a computer

Description automatically generated

1. Within Wireshark indicate how many flow tables the switch supports in the OpenFlow pipeline?
   1. Provide a screenshot highlighting the answer from the capture. [**5 points**]

A screenshot of a computer

Description automatically generated

1. The controller sent a OFPT\_MULTIPART\_REQUEST, and the switch replied with a “REPLY.” Explain what a multipart OpenFlow message is, and why/when they are used, also indicate, by showing in the capture, one switch reply to a multipart request. [**5 points**]
   1. A multipart openflow message is a series of shorter messages with content that are used to process large data responses. This could be used for port or switch descriptions

A screenshot of a computer

Description automatically generated

1. Within Wireshark select the OpenFlow message that includes the Ping request message.
   1. Provide a screenshot of this message highlighting the Ping request OpenFlow message. [**5 points**]

A screenshot of a computer

Description automatically generated

* 1. What type of OpenFlow message is it? [**3 points**]
     1. OFPT\_PACKET\_IN
  2. What is the Buffer\_ID field, and what does it mean? [**3 points**]
     1. The buffer ID field identifies and tracks the IP packer if it is in a buffer
  3. Indicate the src MAC and dst MAC, and src IP and dst IP in this OpenFlow packet. Are these the same or different from the actual src/dst of the packet in Wireshark? Why? [**3 points**]
     1. They are different because the actual packet is going to the controller, but the open flow IP packet contains that h1 wants to ping h2.

A screenshot of a computer program

Description automatically generated

1. After the Ping (request/reply) is successful, what does the controller do? What message does it use to do this? Provide a screenshot of the message. [**15 points**]
   1. After the ping is successful, the controller does a request to add the flow into the switch flow table.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Within the ODL web interface, provide a screenshot of the network topology showing all hosts and switches. [**5 points**]

A computer screen shot of a computer

Description automatically generated

# Objective 3 - Capture OpenFlow Failure Messages

1. Clean up and clear Mininet
   1. What command do you use in Mininet to clean it up/clear it? [**2 points**]
      1. Sudo mn -c
2. After you clear the Mininet process, refresh the ODL topology. What happened? Why? [**3 points**]
   1. The topology cleared, there are no devices in the topology. I stopped mininet and cleared all the configs for the topology, and therefore the controller won’t see the devices anymore.
3. Use the same steps above to start a new Wireshark capture
4. Use the same steps above to initiate a new Mininet topology. Use the same topology used before in step 2.4, but this time instead of using OpenFlow 1.3 use version 1.2.
   1. Did the switch connect to the controller? Why or why not? How do you know? Provide screenshots of the capture to support your answer. [**15 points**]
      1. The switch did not connect to the controller because they support different versions of OpenFlow. I know because the below screenshot shows the switch doing a hello message, and the controller message saying we don’t support your version.

A screenshot of a computer

Description automatically generated

# Objective 4 - OpenFlow Failure Messages (Hardware) [Team]

1. Each team will select one Dell/ABMX server. Label your server with your team number/name.
2. For this objective you have to choose a version of Ubuntu image, and do a fresh install of it onto the server.
3. Once Ubuntu is installed, install Open vSwitch on the server.
4. Paste screenshots of the Ubuntu version and OvS version on your server. [**5 points**]
5. Physically connect your server to an SDN controller. The controller IP is 10.20.30.2/24 and it connected to the ‘Adv. NGN – VPN Mgmt. Switch’ on FastEthernet1/48 and the Mgmt-2 Switch on FastEthernet1/0/1.
6. Connect two laptops to your server. Create an OvS bridge and add the two server interfaces connected to the laptops to the bridge. Before establishing an OpenFlow connection to the controller, ping between two hosts connected to the OVS. Are the pings successful? Show the OpenFlow table of the OVS and explain why the pings are/aren’t successful. [**5 points**]
7. Write a Python script that –
   1. Automates the steps required to connect your OvS to the controller.
   2. Verifies and displays if the OpenFlow connection is successful.
   3. Indicates the OpenFlow failover mode of the switch.

Provide screenshots of the output of your script and attach it with your submission. [**15 points**]

1. Issue a continuous ping between hosts.
   1. Show the flow table entry from OVS that indicates the ICMP traffic between hosts. [**3 points]**
   2. Drop the connection between the OVS/controller.
      1. Do the pings fail or continue? Are there flow table entries in the OVS or not? Explain why? [**10 points**]
2. Establish the connection between OVS and the controller.
   1. Issue a continuous ping between hosts.
   2. Configure OVS to be in failover secure mode.
   3. Drop the connection between the OVS/controller.
      1. Do the pings continue? Why/why not? Show the flow table to help with your explanation [**5 points**]

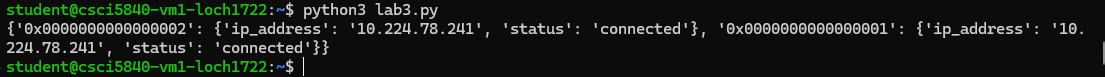
# Objective 5 – Python script to capture, display and save

1. For this objective you have to create a Python script that captures packets using tcpdump between the Mininet switches and the ODL controller.
2. Initialize a linear Mininet topology with two switches and connect it to the ODL controller.
3. Your script should capture OpenFlow packets and identify when a switch has made a successful OpenFlow connection to the controller.
   1. How will you identify a switch has been successfully connected to ODL from the tcpdump captures? [**5 points**]
      1. I will identify whether a switch has been successfully connected to ODL from the captures by whether the capture has a OFPT\_FEATURES\_REPLY and a DPID
4. The script should print out the dpid of the connected switch and also save this information in a file connected.txt in JSON format.

Format –

{ “<dpid>” : { “ip\_address” : “<IP address of switch>”, “status” : “<connected/not connected>” } }

1. Paste screenshots of the script output and the text file saved. [**20 points**]



A screen shot of a computer code

Description automatically generated

1. Connect another OvS in the Mininet VM to the controller (in addition to the two switches connected before). Provide screenshot of the commands used for this purpose. [**5 points**]
   1. py net.addSwitch('s3')
   2. py net.addLink('s3','s2')
2. Paste screenshots of the script output and the updated text file after the new switch connects to the controller. [**5 points**]



A computer screen shot of text

Description automatically generated

1. Describe a use case wherein a file like connected.txt would be useful for network management. [**2 points**]

This file would be useful for net-man when a network engineer wants to identify and confirm new switches added to a SDN at a large scale.

# Objective 6 – Version control using Git and GitHub

Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files. GitHub provides hosting service for software development version control using Git.

1. Create a free account on GitHub if you don’t already have one.
2. Set up a **private** repository with the name sdn-lab3. Provide a screenshot of the repository created as well as its private setting. [**5 points**]

A screenshot of a computer

Description automatically generated

1. Now push the scripts you created in Objectives 4 and 5 from your local system to your private repo on GitHub using git commands. Provide screenshots of the git commands used and of the scripts present in your GitHub repo. [**5 points**]

Commands used:

git init

git remote add origin https://github.com/logan-chayet/sdn-lab3

git add .

git commit -m “Committing Lab3”

git push

A screenshot of a computer

Description automatically generated

1. Describe a use case of version control/Git from the perspective of a network engineer. [**2 points**]

When we update a config on a device and it fails, we can go to a different version of the config to retain working state.

# Total Score = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ 230