COMS E6998-10 Homework 1 Prof. Li Erran Li
Software Defined Networking September 16, 2014

A custom Floodlight-based OpenFlow controller development.
Due Date: 10/1/2014 18:00:00 EST

1 Overview

In this homework, you are asked to write a Floodlight module that works as a learning switch and a super firewall. In your implementation, you will have to coordinate the two features, so that both features work together smoothly. This is because Floodlight, like other OpenFlow implementations, provides no complete module reconciliation.

2 Background Concepts

Floodlight is an open source SDN controller written in Java. Floodlight's module-based loading system offers a flexible and simple management of various service components and applications. A L2 learning switch is a simple, a bit smarter switch than a repeater (hub). Unlike a hub which floods every incoming packet to every port (except the incoming port), a learning switch sends the packet to only one port if the switch already knows the port which the destination node is connected to.

3 Specifications

1. A learning switch collects packets' incoming port and source MAC address and uses this information to find the port for the future packets destined to the associated MAC address.
If the destination MAC address is found in the collected data, it sends the packet directly to the port, instead of flooding which is however inevitable when the port is unknown for the destination MAC address. There are two ways for the controller to handle when the destination port in a switch is known.
(a) Tell the switch to send the given packet to the destination port (use OFType.PACKET_OUT).
(b) Ask the switch to forward future packets for the particular MAC address to the port (use OFType.FLOW_MOD).
The latter is preferred because it reduces the amount of communication between the switches and the controller while the switch can handle packets using the installed rules.

2. The super firewall feature required in this homework is to block users who abuse the network for 10 seconds. The super firewall contains 2 parts:
(a) Connection Limitation: For any host, the controller only allows it send packets to a limited number of destinations. Once it talked to more than MAX_DESTINATION_NUMBER destinations, the controller should block it.

(b) Elephant Flow Limitation: In the learning switch implementation, we can use OFType.FLOW_MOD to generate a flow entry for future packet forwarding. By doing this, any future packet matches the flow entry will be forwarded to port without asking controller. However, some hosts may abuse the network and consume large bandwidth. We call a flow Elephant Flow when its bandwidth exceeds ELEPHANT_FLOW_BAND_WIDTH. And for a switch, if the number of Elephant Flows is more than MAX_ELEPHANT_FLOW_NUMBER, the controller should block all the sources (not destinations) of these Elephant Flows.

3. The two features should work regardless of the network topology. In other words, the controller will be tested both under single-switch networks and multiple-switch networks.

4. For test purpose, we assume the network topology will never be changed once we configured it. Also, we always use MAC address to identify hosts.

4 Preparation Steps

1. Download a VM image from http://128.59.14.24/SDN/hw1vm.zip and unzip.
2. Install VM Player, if not already installed, from http://www.vmware.com/go/downloadplayer.
3. Execute VM Player, import the image, and power on the image.
4. After the image finishes booting, login with mininet / mininet.
5. Change password ASAP.
6. Follow Openflow Tutorial [3] Section 4.1 to 4.4.
7. Clone hw1 git repo by typing “git clone https://github.com/sz2425/SDN_hw1.git.

5 Work Steps

1. Change working directory by “cd SDN hw1”.
2. Modify “src/main/edu/columbia/cs6998/sdn/hw1/Hw1Switch.java”.
3. Compile the code by typing “ant”.
4. Launch Floodlight with the modified module by typing “./foodlight.sh”.
5. Test your code by launching mininet, pinging, tcpdumping, and so on.
(e.g. “sudo mn --topo single,6 --mac --switch ovsk --controller remote”)
6. Create an achieve by typing “./make_archive.sh”. You should be able to see an archive file, named “sdn-hw1-UNI.bz2” (Note that it will include src directory only. Thus if you have modified Floodlight or used 3rd party libraries, they have to be uploaded in a separate archive)
7. Upload the archive file to the courseworks' hw1 page.

6 Hints

1. Floodlight's document [5] is a helpful resource for module creation, APIs reference, and use of other modules in Floodlight.

2. OpenFlow protocol Java APIs (OpenFlowJ) including OFFlowMod, used by Floodlight, is not well documented. You may want to look at OpenFlow Switch Specification [2] and OpenFlow Java API [4] to better understand how to install rules onto switches using Java.
3. Taking a look at Floodlight's default modules is allowed. (LearningSwitch.java and Firewall.java)
4. You can force coding style check by typing “ant checkstyle”.
5. Implementation of Elephant Flow Limitation can be based on OFType.FLOW_MOD. Set HARD_TIMEOUT_DEFAULT carefully may help you calculate the bandwidth.

7 Bonus

Implement a load balancer. The controller knows a list of servers (MAC addresses) that serve the Columbia web pages. Clients use a single IP address to fetch Columbia web pages. The controller tries to load balancing elephant flows among the web servers. Once the controller detects an elephant flow, if the current web server is not the one with the minimal number of active elephant flows, the controller will redirect the elephant flow to the least loaded webserver in terms of the number of active elephant flows. This can be done by rewriting the destination MAC and destination IP to the least loaded web server.

8 Grading Rules

Maximum Grade: 100
Minimum Grade: 0
Learning switch +20
Connection Limitation +40
Elephant Flow Limitation +40
Bonus (max +20)
Coding style -1 per case
Use case failure -5 per case
Late submission $-\sum_{i=1}^{d} 50/i$, where d is the round-up number of days beyond the deadline.

9 Late Policy

Follows the late submission guide in Grading Rules Section.

References

[1] Nate Foster et al. Frenetic: a network programming language. SIGPLAN Not., 46(9):279-291, September 2011.
[2] Open Networking Foundation. Openflow switch specification. June 2012.
[3] http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial. Openflow tutorial. 2012.

[4] https://openflow.stanford.edu/static/openflowj/releases/1.0.2/apidocs. Openflow java 1.0.2 api. 2012.
[5] http://www.openflowhub.org/display/floodlightcontroller. Floodlight documentation. 2012.
[6] http://www.projectfloodlight.org/floodlight. Floodlight.