COMS E6998-8 **Homework 1** Prof. Li Erran Li
Software Defined Networking September 10, 2013

A custom Floodlight-based OpenFlow controller development.
Due Date: 9/24/2013 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 firewall. In your implementation, you will have to coordinate the two features, so that both features work together smoothly. This is because Floodlight[1], like other OpenFlow implementations, provides no complete module reconciliation [1].

2 Background Concepts

Floodlight [6] 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 Specification

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 firewall** feature required in this homework is to block users who abuse the network. The controller should maintain a data structure that stores accumulated packet lengths for each user based on their MAC address. Once the sum of the packets' lengths from a particular user exceeds 15MB, the controller should block the user for 10 seconds. This blocking should be maintained across all the switches (mobility support), as the user may move to another switch to make a connection. Note that this will eventually make each node blocked periodically.

[1] Floodlight provides the IRoutingDecision interface to allow modules pass routing decisions across modules but you do not need to use it within one module.

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 the homework, the both features do not necessarily need VLan or broadcasting support.

5. For test purpose, set each idle timeout value to 10 seconds and no hard timeout value for the switch rules to be installed.

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/jcybha/SDN_hw1.git'.

5 Work Steps

1. Change working directory by 'cd SDN_hw1'.

2. Modify "src/edu/columbia/cs6998/sdn/hw1/Hw1Switch.java".

3. Compile the code by typing 'ant'.

4. Launch Floodlight with the modified module by typing './floodlight.sh &'.

5. Test your code by launching mininet, pinging, tcpdumping, and so on. (e.g. 'sudo mn –topo linear –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 Optional Work

1. Rest API support for the new module by implementing ILearningSwitchService.

2. Expanding the module for a general rule-based firewall and implementing IFirewallService.

7 Grading Rules

Maximum Grade: 100
Minimum Grade: 0

Learning switch +40
Rule installation (using OFType.FLOW_MOD) +20
Firewall +40

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.

8 Late Policy

Follows the late submission guide in Grading Rules Section.

9 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'

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.