FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Engenharia Electrotécnica

Configuração e Gestão de Redes

2016 / 2017

2º Lab:
Software Defined Networks

pfa@fct.unl.pt **Pedro Amaral**

1. Overview

In this lab you will develop an SDN app using the floodlight controller https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+Documentation that configures the switches in a network to work as learning switches and implements a firewall that only allows hosts to send packets to a limited number of simultaneous destinations. To simulate the Openflow enabled switches and build a test network you will use a simulator called mininet http://mininet.org/walkthrough/ that simulates OpenFlow enabled switches in a computer using Virtual Switches in the linux kernel (Open vSwitch) http://openvswitch.org/.

2. Background concepts

The goal is to make a first contact with the use of the OpenFlow protocol to control the switches of a network. You will use the mininet simulator http://mininet.org/ to simulate the OpenFlow controlled network. You will then connect an external controller to the topology. The controller uses OpenFlow to receive information from the switches (unmatched packets) and instruct the switches how to deal with them.

In this project you will use a VM provided in the course website that contains the latest distribution of the floodlight controller, and the Eclipse IDE where the floodlight controller code is already imported. Under the src/main/java folder in the floodlight project you can find the package net.floodlightcontroller.CGRL2Switch and there the class file CGRL2Switch.java where you should build your code.

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.
In the VM the run button in the eclipse toolbar is already configured to run floodlight only loading the needed set of modules for your L2Switch application to work.
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

A learning switch collects the 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. If the MAC address was not learned yet the mapping in port / source MAC address should be stored and the packet is flooded.

When the destination port for a MAC is already known the controller can handle the packet in two ways:

1. Tell the switch to send the given packet to the destination port using an Openflow PacketOut message (a OFPacketOut object in Floodlight).

2. Ask the switch to forward all future packets for a particular MAC address to the corresponding port by installing an OpenFlow Flow entry in the switch using a FlowModification message (a OFFlowMod object in Floodlight).

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.

The firewall feature implements a connection limitation for all hosts. The controller only allows a host to send packets to a limited number os destinations simultaneously. When a host reaches the MAX_DESTINATION_NUMBER of destinations, the controller should block traffic for other destinations. When a host stops sending traffic to one of the destinations then it can send traffic to a new destination. For example if MAX_DESTINATION_NUMBER = 3, the desired behaviour is the following:

A host can send traffic to up to 3 destinations (say for example a ping) while the pings are being sent to those 3 destinations the host is not able to ping a 4th different destination. When one of the 3 original pings stops sending packets then the host can contact another destination.

Both features should work regardless of the network topology. In other words you should test the application in multiple switch networks (without loops in a loop you might run in to trouble but by bow you already know why). To simplify we assume that the topology never changes and only MAC addresses are used to identify hosts.

3. Instructions and Hints

Download the VM provided in the course website that contains the floodlight controller. Install VM workstation player, and import the VM and power it on. You should then also download a mininet VM directly in the mininet website.

Start by familiarising yourself with Mininet

3.1 Using Mininet

Mininet emulates an OpenFlow network and end-hosts within a single machine. It includes built-in support to create several common topologies, plus it allows for construction of custom topologies using a python script.

Take some time to follow the Mininet walkthrough available at http://mininet.org/walkthrough/ . You can also use it as a reference. Concentrate on understating the options used in the following mininet startup command.

```
sudo mn --topo tree,2 --mac –x –controller=remote,ip=192.168.132.163,port=6633
```

A small description of each of the options is also given here:

- `sudo` runs as root .
- `mn` runs mininet.

- `--topo tree,2` creates a tree topology of depth 2 with the default fanout of 2 (i.e., binary).
- `--mac` makes the mac address of mininet hosts the same as their node number.
- `--x` automatically opens a xterm for each host and switch in the topology.
- `--switch ovsk` uses Open vSwitch in kernel mode for each of the switches.
- `--controller remote,ip=<ip>, port=<port>` specifies the IP and port of the controller

Once Mininet is running, you can obtain information about the network, generate traffic, and run commands on individual hosts. You can also use the XTerms for each network element and run the necessary commands on them.

3.2 Developing the Application

A new module to implement the project called CGRL2Switch is already created in the provided virtual machine. It can be found in the eclipse project under the package net.floodlightcontroller.CGRL2Switch. Its implementation is on the file CGRL2Switch.java that contains the code for the java class that implements the module.

The file as it is works instructs the switches to flood the packets they receive and therefore works like a hub. You should change the class code to provide the specified functionality.

You can run floodlight in two different ways:

The first one is using the Eclipse IDE itself by choosing the main.java file under the net.floodlight.core package and choosing run as java application or directly in the run button of the IDE (choose the configuration Floodlight-CGRL2Switch). The log messages will be visible in the console tab in Eclipse.

The second way is to build floodlight using ant. For this you should issue the following commands in a terminal windows inside the floodlight directory:

```
pfa@dev-virtual-machine:~/floodlight$ ant clean
```
(to clean up previous builds)

```
pfa@dev-virtual-machine:~/floodlight$ ant
```
(to build)

You can then run the controller with java using the .jar file with the following command:

```
pfa@dev-virtual-machine:~/floodlight$ java -jar target/floodlight.jar -cf src/main/
resources/CGRL2Switch.properties
```

(the -cf option is to load only the floodlight modules needed for the project and some configuration options like the port where the switch listens for switch connections)

You should search for the following two lines of output that signal that our module project 2 was loaded:

```
12:06:51.279 INFO [n.f.p.CGRL2Switch] CGR L2 Switch module initialized
```

and that the controller is listening for connections from switches:

```
12:06:51.644 INFO [n.f.c.i.OFSwitchManager:main] Listening for OpenFlow switches on
[0.0.0.0]:6633
```

This second is particularly important since it indicates that the controller is listening on all interfaces of the machine (hence the 0.0.0.0/0.0.0.0 IP and mask) on the port 6633. This means that when you start mininet you should define in the remote controller IP address the IP address of the VM running floodlight and the port should be the one you see in the above line.

3.3 Testing the Application

Start by checking the flooding behaviour that the CGRL2Switch module produces in the switches.

Start a mininet simulation as described above and start the floodlight controller with the CGRL2Switch module included.

You will see on the floodlight output the indication of the negotiation between the various modules of floodlight and the switches with at some point the following messages indicating that the switches have connected with the controller.

```
15:32:09.993 WARN [n.f.c.i.C.s.notification:main] Switch 00:00:00:00:00:00:00:03
connected.
15:32:10.016 WARN [n.f.c.i.C.s.notification:main] Switch 00:00:00:00:00:00:00:02
connected.
15:32:10.017 WARN [n.f.c.i.C.s.notification:main] Switch 00:00:00:00:00:00:00:01
connected.
```

You can verify that hosts can ping each other, and that all hosts see the exact same traffic (the behaviour of a hub). To do this you can use the Xterms of the hosts.

Arrange each Xterm so that they're all on the screen at once. This may require reducing the height of them to fit a cramped laptop screen.
In the Xterms for each host, run tcpdump, a utility to print the packets seen by a host:
For example for hosts h2 and h3:

```
# tcpdump -XX -n -i h2-eth0
```

and respectively:

```
# tcpdump -XX -n -i h3-eth0
```

In the xterm for h1 for example, send a ping to host 2:

```
# ping -c1 10.0.0.2
```

The ping packets are now going up to the controller (floodlight installs a flow entry with 0 priority, a match any and an action forward controller causing all unmatched packets in other flows entries to be sent to the controller in a PacketIn message) the controller resends them via a PacketOut message to the switches with

the action set to flood the packet out all interfaces except the sending one. You should see identical ARP and ICMP packets corresponding to the ping in all xterms running tcpdump. This is how a hub works; it sends all packets to every port on the network. So the ping packet is seen in all hosts.

If you start mininet with the -x option an Xterm also opens for one of the switches. The reason for only one terminal opening is that mininet uses linux name spaces to create a different network name spaces. By default only hosts are put in a different namespace. Switches are all in the linux space so they all share the same environment so you can issue commands in any of them in the same window. If you wish to see the OpenFlow flow entries installed in a switch you can use the following OpenVswitch command in the Xterm of the switch opened by mininet or in a terminal of the host system (since switches run in the host namespace):

```
ovs-ofctl dump-flows s1
```

This for switch 1 you can do the same for switch 2 replacing `s1` by `s2` and so on.

NOTE: Since the network namespace of the switches in mininet is the same as the host system where it is installed if you use the mininet in the Floodlight VM you will receive multicast packets that are sent by any applications you have installed in the switches of mininet. Switches in mininet appear as network interfaces of the host system therefore if an application sends a multicast packet that packet will be received in all interfaces. If the mininet switches have a rule to send unmatched packets to the controller this means that you will receive these packets in the controller. If you use the mininet VM these packets are less frequent since the OS is tailored for mininet only. Either way this is not a problem for the implementation you should only be aware that such packets might appear if you use mininet in the floodlight VM:

In mininet you can use other commands like iperf:

```
mininet> iperf
```

This runs an iperf TCP server on one virtual host, then runs an iperf client on a second virtual host. Once connected, they blast packets between each other and report the results.

3.6 Implementing the application

You should change the CGRL2Switch class in file CGRL2Switch.java to perform according to the specifications. Some of the needed code for the implementation of the learning switch feature is already commented in the file. You can also use the methods in class SwitchCommands.java in the package net.floodlightcontroller.CGRutil already imported in class CGRL2Switch.

To understand the general structure of the class read the floodlight tutorial on creating a module https://floodlight.atlassian.net/wiki/display/floodlightcontroller/How+to+Write+a+Module

Besides the methods the IFloodlightModule and IOFMessageListener interfaces, the class has methods already defined to handle PacketIn Messages, write PacketOut Messages from Packet In messages as well as a series of suggested methods to handle the data structure to store the MAC address / Port mappings for the learning switch feature.

The tutorials on how to process a Packet_In message https://floodlight.atlassian.net/wiki/display/floodlightcontroller/How+to+Process+a+Packet-In+Message and on how to create a Packet_out message https://floodlight.atlassian.net/wiki/display/floodlightcontroller/How+to+Create+a+Packet+Out+Message are also a great source of examples on how to use the several needed OpenFlow structures as implemented by Floodlight.

Floodlight uses the OpenFlowJ-Loxi Java generated API to implement Openflow concepts in Java objects. For reference on how to create Matches, Flow entries etc, you can check the OpenFlowJ-Loxi tutorial in floodlight https://floodlight.atlassian.net/wiki/display/floodlightcontroller/How+to+use+OpenFlowJ-Loxigen

You can also use the base java documentation for OpenFlowJ and floodlight that can be found in floodlight.github.io/floodlight/javadoc/openflowj-loxi/index.html (OpenFlowJ) floodlight.github.io/floodlight/javadoc/floodlight/index.html (Floodlight base classes), these links are also added in the firefox browser bookmarks tab in the floodlight VM.

HINT

You can take a look in the original floodlight modules for code examples.