CNT4713 – Project 4

Overview:

Practice Dijkstra's algorithm by implementing a shortest path routing module in Floodlight. Develop the skills to read and modify open source software. This is a group project with up to 2 students per group. Each group should submit only a single copy. The programming language used by Floodlight is Java.

Instructions:

You will need Mininet and Floodlight for this project, which you should have been familiar with in the previous project. Mininet will set up a simulated network, in which each switch will communicate with the Floodlight controller by the OpenFlow protocol. Whenever a switch receives a new packet, it will forward the packet to Floodlight, and Floodlight will tell the switch how to process the packet also by the OpenFlow protocol. In this project, your module will process only IP packets, and leave other packets such as ARP to existing Floodlight modules. When a new IP packet is received, the receive() function of your module will be called, in which you should calculate the shortest path using the hop count metric, and install the routing rules on switches of the calculated path. The detailed steps are as follows.

1. Set up Mininet as in the previous project.

2. Download from Moodle the pre-compiled Floodlight and import it into Eclipse.

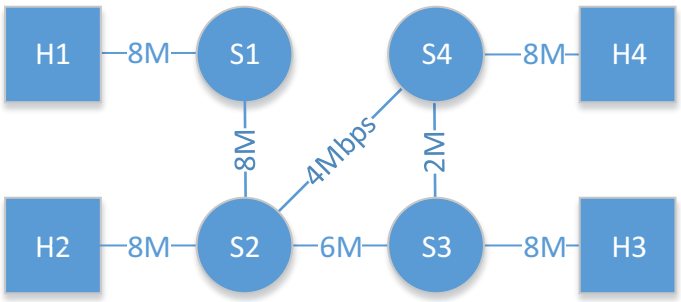
3. In Eclipse Package Explorer, navigate to floodlight -> src/main/java -> net.floodlightconroller.myrouting -> MyRouting.java, where you will find a module skeleton for your reference.

4. As the first step of the Dijkstra's algorithm, collect the topology of the network. Print all the switches and links. Sample output is attached.

5. When an IP packet is received, print the source and destination IP addresses.

6. Implement the Dijkstra's algorithm to calculate and print the shortest path between the source and the destination. The cost of each link is a unit.

7. You may use the test.py script posted in Moodle to test your program. Type "sudo python test.py" in Mininet to run the script. It will create the following topology.



With proper implementation, the Mininet output should be similar as follows.

```
mininet@mininet-vm:~/mininet/custom$ sudo python test.py
*** Add controller
*** Add switches
*** Add hosts
*** Add links
(8.00Mbit)  (8.00Mbit)  (8.00Mbit)  (8.00Mbit)  (8.00Mbit)  (8.00Mbit)
(8.00Mbit)  (8.00Mbit)  (8.00Mbit)  (8.00Mbit)  (6.00Mbit)  (6.00Mbit)
(4.00Mbit) (4.00Mbit) (2.00Mbit) (2.00Mbit) *** Start network
*** Configuring hosts
h1 h2 h3 h4
*** Start controller
*** Start switches
(8.00Mbit)  (8.00Mbit)  (8.00Mbit)  (8.00Mbit)  (6.00Mbit)  (4.00Mbit)
(8.00Mbit)  (6.00Mbit)  (2.00Mbit)  (8.00Mbit)  (4.00Mbit)  (2.00Mbit)  ***
Configuring switches
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['7.64 Mbits/sec', '8.77 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['5.73 Mbits/sec', '7.73 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['3.83 Mbits/sec', '5.33 Mbits/sec']
*** Starting CLI:
```

(Due to sending buffers, the client rate may be much higher than the actual transmission rate. Please see the explanation at: https://github.com/mininet/mininet/blob/125e66972354253c8ee45b4e1a1ced64b0c1fd4f/mininet/net.py)

The floodlight output should be similar as follows.

```
*** Print topology
switch 1 neighbors: 2
switch 2 neighbors: 1, 4, 3
switch 3 neighbors: 2, 4
switch 4 neighbors: 2, 3
*** New packet
srcIP: 10.0.1.1
dstIP: 10.0.2.2
route: 1 2
*** New packet
srcIP: 10.0.1.1
dstIP: 10.0.3.3
route: 1 2 3
*** New packet
```

```
srcIP: 10.0.1.1
dstIP: 10.0.4.4
route: 1 2 4
```

8. If you see the following error in Floodlight, it does not affect the operation of Floodlight.

```
Exception in thread "debugserver" Traceback (most recent call last):
  File "<string>", line 1, in <module>
ImportError: No module named debugserver
```

References:

Dijkstra's algorithm, in textbook or http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
OpenFlow protocol, http://archive.openflow.org/documents/openflow-wp-latest.pdf

Submission Guide:

Each group should submit a single copy of the finished MyRouting.java file. Please write the group member names and IDs as comments at the beginning of the MyRouting.java file.

The code must be well-documented.

Grading Criteria:

Item	Percentage
Print all switches	15%
Print all links	15%
Print source IP address	15%
Print Destination IP address	15%
Print the correct path	20%
Successfully install the path (i.e. packets are routed in the correct path as seen by iperf output)	20%

Plagiarism will be reported to the university for academic dishonesty.