Project2 of Computer Network

Routing Algorithms via Mininet

Qiong Yang

1 Problem Description

We need to implement two routing algorithms(Link-State Routing and Distance-Vector Routing)via Mininet. The input and output can be described as follows:

- Input: Shanghai Metro Map and arbitrary two stations(for example:from Zhangjiang Hi-tech Park to Zhaojiangbang Rd)
- Output:the least path or the least switching times with delay as low as possible

2 Problem Analysis

1. Target:Find the shortest path between two arbitrary station.
2. Algorithm:Link-State Routing Algorithm & Distance-Vector Algorithm
3. Tools:Mininet & Pox

According to the above analysis,we need to complete the following steps.
Step1: Look for the Shanghai Metro Map and transform it into topology via mininet.
Step2: Then we need to test the topology whether it can be ping successfully.
Step3: Write the Controller python Script with two kinds of algorithms under pox.
Step4: Run command,start pox and mininet.

3 Process Realization

3.1 Construct Topology

Firstly, I consider the running condition of the metro, and choose the switches to represent these stations and write the topology script via python language.The meaning of Class,functions and parameters is shown as follows.

1. Mininetthe main class and it is used to construct topology
2. Toopo:the basic topo class
3. addSwitch():add a switch into topology and return the name of switch
4. addLink():add the bidirectional links into topology
5. self.addLink(node1, node2, bw=10, delay='1ms', max_queue_size=1000, loss=10, use_htb=True):
    - bw:the value of bandwidth
    - delay:the delay of link transmission
    - loss:the percentage of losing package(between 1 and 100)
    - max_queue_size:the data package

Then we run the ShanghaiMetrotopo.py file by the command"sudo mn –custom ∼/mininet/custom/ShanghaiMetrotopo.py –topo mytopo –link tc controller remote".Then I discovery it can receive the package successfully. The result is shown as follow.

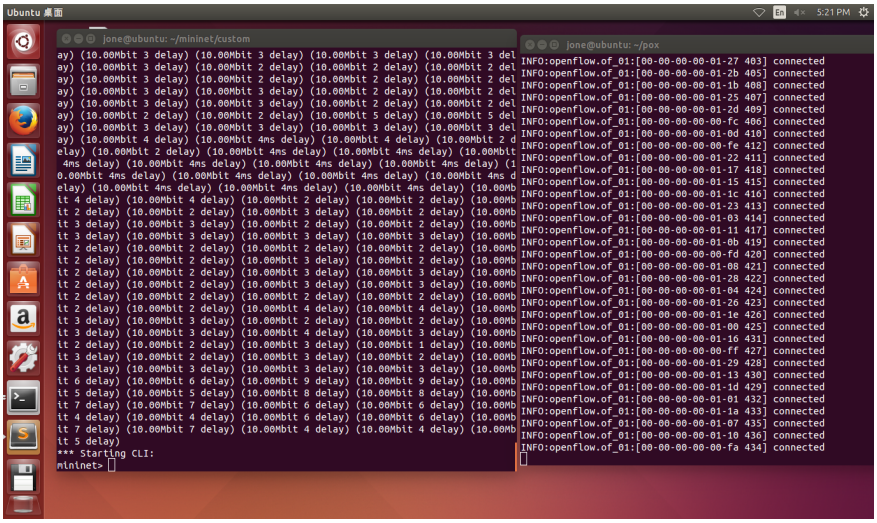
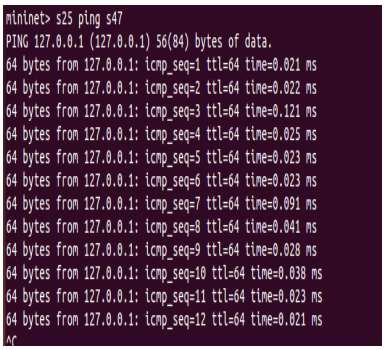
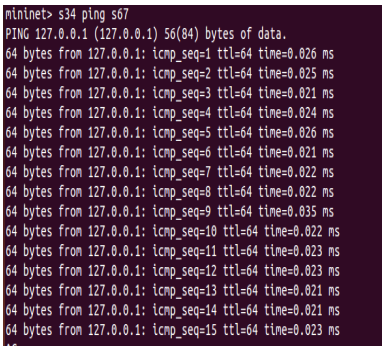


Figure 1: The topology running under mininet and pox



(a) s25 ping s48



(b) s34 ping s67

Figure 2: Test the connectivity between the switches

This show that topology has been constructed successfully.

3.2 Algorithm Realizing

After constructing the topology, we need add the two kinds of algorithm into pox controller so that we can achieve the target of finding the shortest path.

Firstly, I will give the algorithm process.

3.2.1 Link-State Algorithm(Dijkstra)

The Dijkstra algorithm solves the single-source shortest path problem for a graph G with non-negative edge weights, producing a shortest path tree. It is a greedy algorithm that starts at the source node, then it grows T and spans all nodes reachable from the source. Nodes are added to T in order of distance. The relaxation process is performed on outgoing edges of minimum-weight nodes. The total time is $O(ElogV)$. The Algorithm steps are below.
Step1: Choose source node,S=v,U=other nodes.If node v connect with node u(in U),then ¡u,v¿ has weight.Otherwise the weight of ¡u,v¿ is inf.
Step2: Choose node k from U and the path between k and v is the shortest. Add the node k into S.
Step3: Take the node k as the updated middle node, update the distance between source node and nodes in U.If distance between v and u(contain k) is shorter than origin distance (not contain k), then update the distance between source and node u.
Step4: Repeat the step2 and step3 until all nodes in S.
Because I don't konw the pox very well,so I can't complete the controller script. I only modify the **l2_multi.py** file and add dijkstra algorithm into it. But I don't transmit the delay data into the controller, so, I can't get the result.
However, I get the shortest path via python. The result is shown below.

```
...
============= RESTART: C:\Users\IBM\Desktop\project2\dijkstra.py =============
Dijkstra's shortest path
The shortest path : ['s22', 's21', 's20', 's19', 's38', 's39', 's14', 's13']
>>>
```

Figure 3: The shortest path from s22 to s13

```
============= RESTART: C:\Users\IBM\Desktop\project2\dijkstra.py =============
Dijkstra's shortest path
The shortest path : ['s37', 's246', 's192', 's173', 's90', 's89']
>>>
```

Figure 4: The shortest path from s37 to s89

```
...
============= RESTART: C:\Users\IBM\Desktop\project2\dijkstra.py =============
Dijkstra's shortest path
The shortest path : ['s95', 's43', 's42', 's41', 's40', 's16', 's39', 's38']
>>>
```

Figure 5: The shortest path from s95 to s38

I compare the result with Shanghai Metro Map and I find that there are between two ways. Because I don't consider the switch time.

3.2.2 Distance-Vector Algorithm(Bellman-Ford)

The Bellman-Ford algorithm computes shortest paths from a single source to all of the other nodes in a weighted graph G in which some of the edge weights are negative. The algorithm relaxes all the edges and it does this $|V| - 1$ times. At the last stage, negative cycles detection is performed and their existence is reported. The algorithm runs in $O(V * E)$time.

4 Conclusion

Because the time of my learning mininet and pox is not long, and is not familiar with mininet, so ,firstly, I spended much time to consider that how to set up the complex topology and how the pox find the topology. Now, I know that how the mininet cooperate with pox to complete the task.

Even so, I still can't complete the project. Maybe I is't familiar with the pox command and the deep principle.