

姓名：滕德淋 班级：111192

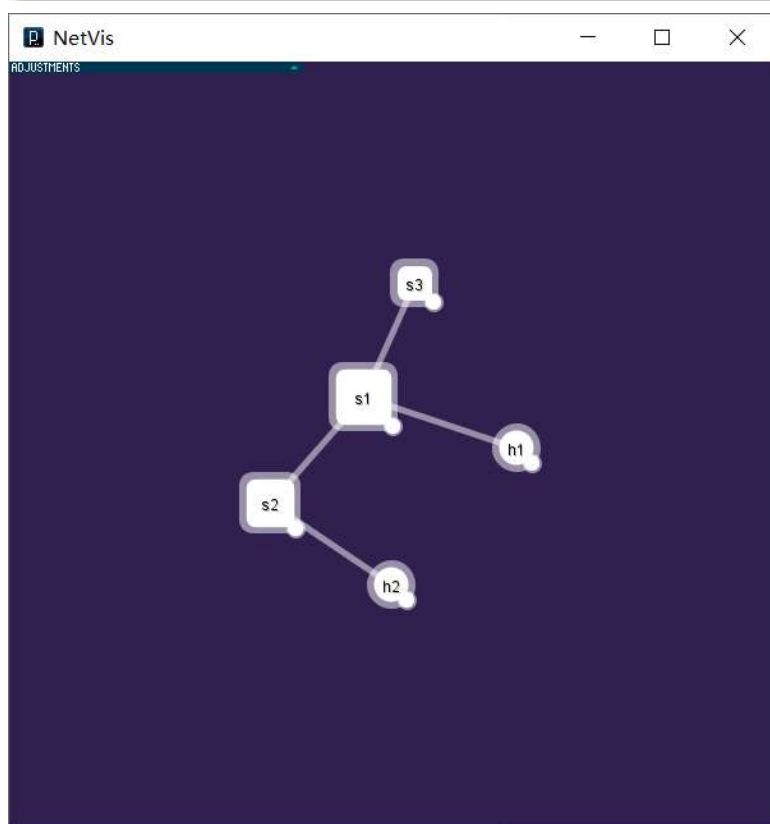
第一步：根据文档操作

根据《先读我.docx》文档的前四步教程，了解到整体框架是什么样的，并且运行了一下：

```
C:\WINDOWS\system32\cmd.exe - run.py
Microsoft Windows [版本 10.0.18363.1256]
(c) 2019 Microsoft Corporation. 保留所有权利。

E:\network stimulator\project3>run.py
INFO:simulator:s1 up!
INFO:simulator:h1 up!
INFO:simulator:s2 up!
INFO:simulator:h2 up!
EE-122 Network Simulator
You can get help on a lot of things.
For example, to see your current scenario, try help(scenario).
If you have a host named h1a, try help(h1a).
If you want to inspect a method of that host, try help(h1a.ping).
For help about the simulator and its API, try help(sim) and help(api).
Type start() to start the simulator.
Good luck!

>>> from hub import Hub
>>> Hub.create('s3')
INFO:simulator:s3 up!
<Hub s3>
>>> topo.link(s3, s1)
(0, 2)
>>>
```



进行了创建点、连接线、删除线、删除点的操作；
体验了主机节点的模拟 ping 的过程，观察到包的发送的动画演示；

第二步：写学习型交换机

根据老师给的伪代码可以理解到学习型交换机工作的原理：

```
class LearningSwitch (Entity):
    def __init__(self):
        # Add your code here!
        self.ports= {}

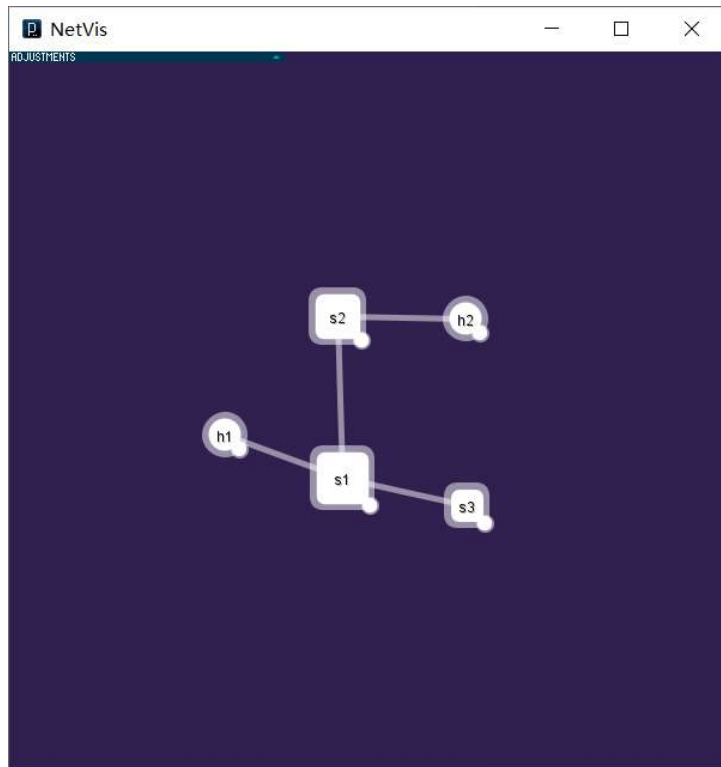
    def handle_rx (self, packet, port):
        # Add your code here!
        packetDest = packet.dst;
        if not self.ports.has_key(packet.src):
            self.ports[packet.src] = port;

        if self.ports.has_key(packetDest):
            self.send(packet, self.ports[packetDest])
        else:
            self.send(packet, port, flood=True)
        # raise NotImplementedError
```

编写好交换机之后，可以将 switch 从集线器 hub 换成 learningswitch;

```
C:\WINDOWS\system32\cmd.exe - run.py
Routing update is {<BasicHost h2>: 1, <BasicHost h1>: 100}
premin = {<BasicHost h2>: (<RIPRouter s2>, 2), <BasicHost h1>: (<BasicHost h1>, 1), <RIP
Router s2>: (<RIPRouter s2>, 1)}
mincosts is {<BasicHost h2>: (<RIPRouter s2>, 2), <BasicHost h1>: (<BasicHost h1>, 1),
<RIPRouter s2>: (<RIPRouter s2>, 1)} changed = False
routing table is {<BasicHost h1>: {<BasicHost h1>: 1}, <RIPRouter s2>: {<BasicHost h2>:
2, <BasicHost h1>: 100, <RIPRouter s2>: 1}}
mincost table is {<BasicHost h2>: (<RIPRouter s2>, 2), <BasicHost h1>: (<BasicHost h1>,
1), <RIPRouter s2>: (<RIPRouter s2>, 1)}

>>> from learning_switch import LearningSwitch
Traceback (most recent call last):
  File "<console>", line 1, in <module>
ImportError: No module named learning_switch
>>> from learning_switch import LearningSwitch
>>> LearningSwitch.create('s3')
INFO:simulator:s3 up!
<LearningSwitch s3>
>>> topo.link(s3, s1)
(0, 2)
>>> <RIPRouter s1> got <DiscoveryPacket from s3> NullAddress, 1, True> at 2
premin = {<BasicHost h2>: (<RIPRouter s2>, 2), <BasicHost h1>: (<BasicHost h1>, 1), <RIP
Router s2>: (<RIPRouter s2>, 1)}
mincosts is {<BasicHost h2>: (<RIPRouter s2>, 2), <LearningSwitch s3>: (<LearningSwitch
s3>, 1), <BasicHost h1>: (<BasicHost h1>, 1), <RIPRouter s2>: (<RIPRouter s2>, 1)} chang
ed = True
Sending {<BasicHost h2>: 2, <BasicHost h1>: 1, <RIPRouter s2>: 1} to <LearningSwitch s
3>
Sending {<BasicHost h2>: 2, <LearningSwitch s3>: 1, <RIPRouter s2>: 1} to <BasicHost h
```



进行 ping 操作可以实现，学习型交换机完成；

第三步：距离向量路由算法编写

通过 flood 进行建表，然后查表进行包的转发。RIPRouter 需要处理三个类型的包：DiscoverPacket, RoutingUpdate, others。

当收到 DiscoverPacket 类型的包，表示此节点和邻居是相连的，需要将包的 src 存放到路由表中。DiscoverPacket 只是相邻节点进行发包。

当收到 RoutingUpdate 类型的包时，如果这个包的 src，以前没有收到，这时需要把包的 src 存放到路由表中，如果路由表中有该 src 的记录，需要比较路径的长度，如果长度比路由表中存的小，需要更新路由表，然后已有的路由表里面的信息 flood 出去。

当收到 other 类型的包时，数据包直接根据目的地址进行转发。

```
if hasattr(packet, 'is_link_up'): # DISCOVERY PACKET!
    self.ports[packet.src] = port #how to get to my neighbours
    if packet.is_link_up:
        self.routingTable[packet.src] = {packet.src: 1}
    else:
        self.routingTable.pop(packet.src)
```

```

elif hasattr(packet, 'paths'): # UPDATE PACKET!

    print "Routing update is ", packet.str_routing_table();
    if not self.routingTable.has_key(packet.src): #if its not a neighbour, fuck it
        pass
    else: #its a neighbour, deal with it
        for dest in packet.all_dests():
            if packet.get_distance(dest) == 100:
                self.routingTable[packet.src][dest] = 100
            else:
                throughSrc = self.minCosts[packet.src][1] + packet.get_distance(dest)
                #if not self.routingTable[packet.src].has_key(dest) or throughSrc < self.routingT
                self.routingTable[packet.src][dest] = throughSrc

        self.lastPackets[packet.src] = packet

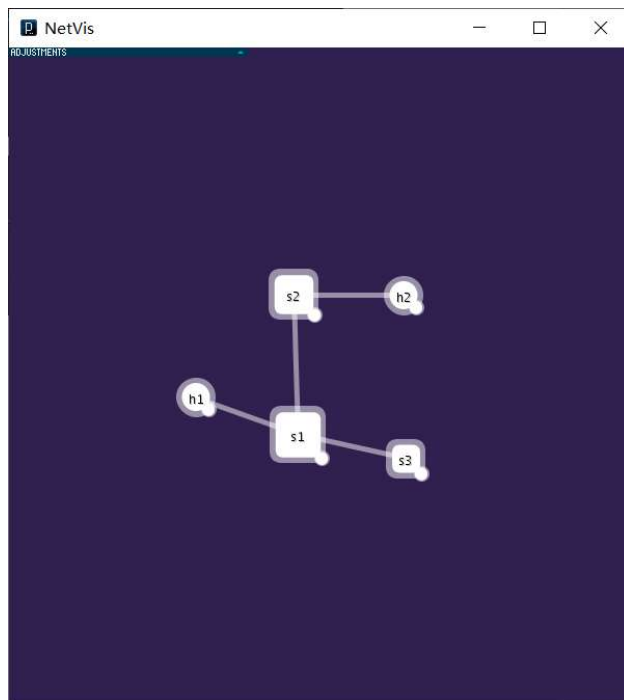
```

```

else: #DATA PACKET!
    if self.minCosts.has_key(packet.dst):
        print "Trying to send to ", packet.dst, " through ", self.minCosts[packet.dst]
        if self.minCosts[packet.dst][1] != 100:
            self.send(packet, self.ports[self.minCosts[packet.dst][0]]) # data packet, send it thr

```

编写好路由器之后，将路由器作为 switch，进行调用：

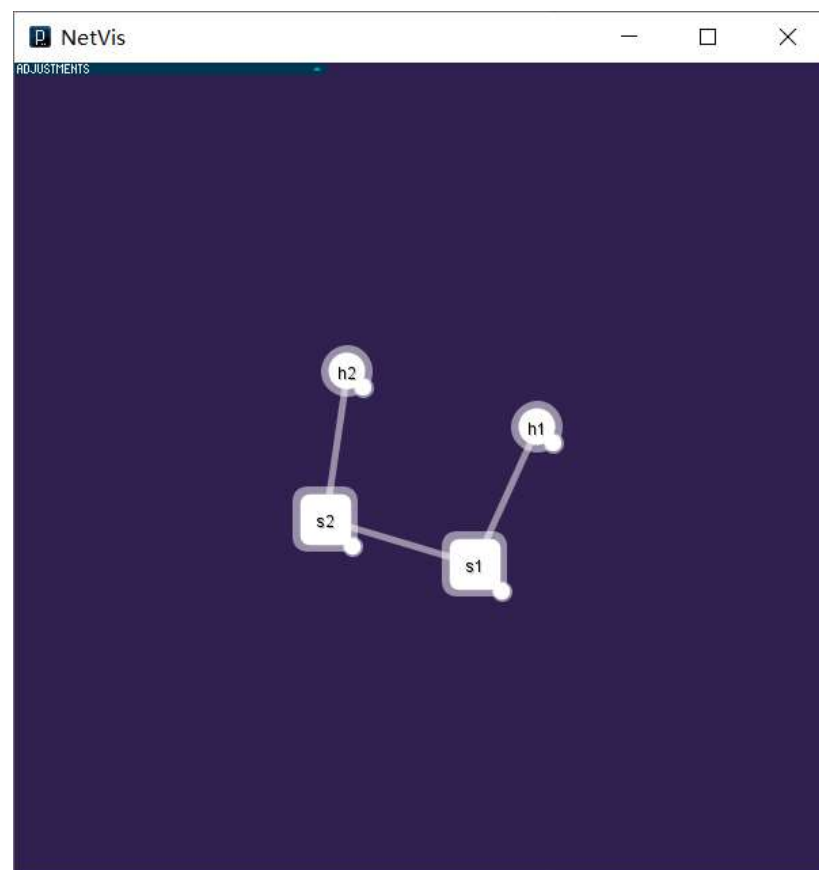


第四步：进行 log 信息查看

```
C:\WINDOWS\system32\cmd.exe - run.py
Microsoft Windows [版本 10.0.18363.1256]
(c) 2019 Microsoft Corporation. 保留所有权利。

E:\network stimulator\project3>run.py
INFO:simulator:s1 up!
INFO:simulator:h1 up!
INFO:simulator:s2 up!
INFO:simulator:h2 up!
EE-122 Network Simulator
You can get help on a lot of things.
For example, to see your current scenario, try help(scenario).
If you have a host named h1a, try help(h1a).
If you want to inspect a method of that host, try help(h1a.ping).
For help about the simulator and its API, try help(sim) and help(api).
Type start() to start the simulator.
Good luck!

>>> start()
>>> <RIPRouter s1> got <DiscoveryPacket from h1->NullAddress, 1, True> at 0
<RIPRouter s2> got <DiscoveryPacket from h2->NullAddress, 1, True> at 0
<RIPRouter s2> got <DiscoveryPacket from s1->NullAddress, 1, True> at 1
<RIPRouter s1> got <DiscoveryPacket from s2->NullAddress, 1, True> at 1
<RIPRouter s1> got <RoutingUpdate from s2->NullAddress> at 1
<RIPRouter s2> got <RoutingUpdate from s1->NullAddress> at 1
<RIPRouter s2> got <RoutingUpdate from s1->NullAddress> at 1
<RIPRouter s1> got <RoutingUpdate from s2->NullAddress> at 1
>>>
```



```
74 Network Simulator Log
--- Connected -----
S 22:01:23 INFO      s1 up!
S 22:01:23 INFO      h1 up!
S 22:01:23 INFO      s2 up!
S 22:01:23 INFO      h2 up!
```

当程序运行起来之后就会自动建立两个主机 h1、h2；两个路由节点 s1、s2；Log 消息中可以看到；

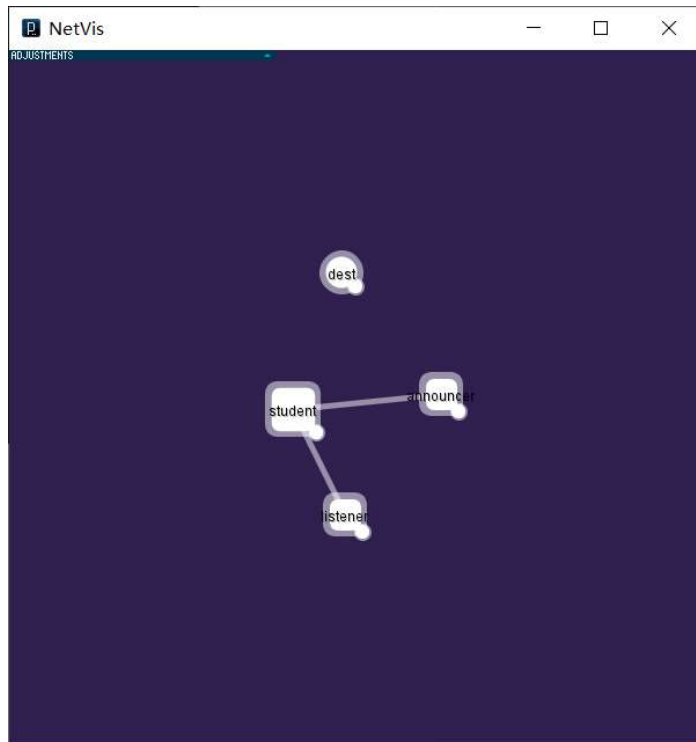
第五步：测试

在命令行中调用测试用例：tests*.py
compatibility_test 测试：

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.18363.1256]
(c) 2019 Microsoft Corporation。保留所有权利。

E:\network stimulator\project3>tests\compatibility_test.py
INFO:simulator:student up!
INFO:simulator:dest up!
INFO:simulator:announcer up!
INFO:simulator:listener up!
<RIPRouter student> got <DiscoveryPacket from announcer->NullAddress, 1, True> at 0
<RIPRouter student> got <DiscoveryPacket from listener->NullAddress, 1, True> at 1
Received Packet: <RoutingUpdate from student->NullAddress>
<FakeEntity announcer> 1
Announcing from <FakeEntity announcer>
<BasicHost dest> 7
<RIPRouter student> got <RoutingUpdate from announcer->NullAddress> at 0
Received Packet: <RoutingUpdate from student->NullAddress>
<FakeEntity announcer> 1
<BasicHost dest> 8
You Passed Compatibility Test!

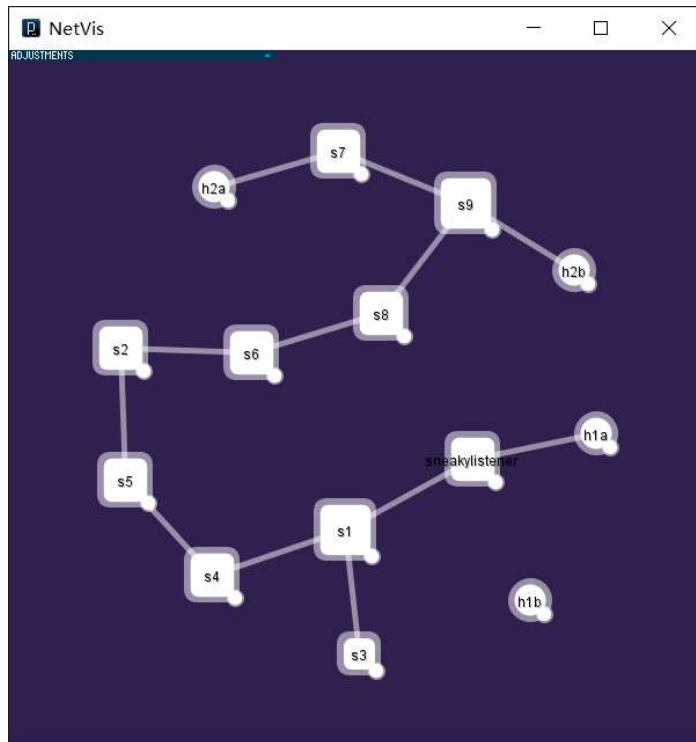
E:\network stimulator\project3>_
```



测试通过;

really_big_network_multiple_failures 测试:

```
C:\WINDOWS\system32\cmd.exe
E:\network>stimulator project3\tests\really_big_network_multiple_failures.py
INFO:simulator:s1 up!
INFO:simulator:s2 up!
INFO:simulator:s3 up!
INFO:simulator:s4 up!
INFO:simulator:s5 up!
INFO:simulator:s6 up!
INFO:simulator:s7 up!
INFO:simulator:s8 up!
INFO:simulator:s9 up!
INFO:simulator:h1a up!
INFO:simulator:h1b up!
INFO:simulator:h2a up!
INFO:simulator:h2b up!
INFO:simulator:sneakylistener up!
<RIPRouter s1> got <DiscoveryPacket from sneakylistener->NullAddress, 1, True> at 0
<RIPRouter s1> got <DiscoveryPacket from h1b->NullAddress, 1, True> at 1
<RIPRouter s1> got <DiscoveryPacket from s2->NullAddress, 1, True> at 0
<RIPRouter s2> got <DiscoveryPacket from s6->NullAddress, 1, True> at 0
<RIPRouter s8> got <DiscoveryPacket from s6->NullAddress, 1, True> at 0
<RIPRouter s8> got <DiscoveryPacket from s8->NullAddress, 1, True> at 1
<RIPRouter s9> got <DiscoveryPacket from s8->NullAddress, 1, True> at 0
<RIPRouter s8> got <DiscoveryPacket from s9->NullAddress, 1, True> at 1
<RIPRouter s9> got <DiscoveryPacket from h2a->NullAddress, 1, True> at 1
<RIPRouter s7> got <DiscoveryPacket from s6->NullAddress, 1, True> at 0
<RIPRouter s6> got <DiscoveryPacket from s7->NullAddress, 1, True> at 2
<RIPRouter s9> got <DiscoveryPacket from s7->NullAddress, 1, True> at 2
<RIPRouter s7> got <DiscoveryPacket from s9->NullAddress, 1, True> at 1
<RIPRouter s7> got <DiscoveryPacket from h2a->NullAddress, 1, True> at 2
<RIPRouter s3> got <DiscoveryPacket from s1->NullAddress, 1, True> at 0
<RIPRouter s1> got <DiscoveryPacket from s3->NullAddress, 1, True> at 2
<RIPRouter s2> got <DiscoveryPacket from s3->NullAddress, 1, True> at 1
<RIPRouter s3> got <DiscoveryPacket from s2->NullAddress, 1, True> at 1
<RIPRouter s4> got <DiscoveryPacket from s1->NullAddress, 1, True> at 0
<RIPRouter s1> got <DiscoveryPacket from s4->NullAddress, 1, True> at 3
<RIPRouter s5> got <DiscoveryPacket from s4->NullAddress, 1, True> at 0
<RIPRouter s4> got <DiscoveryPacket from s5->NullAddress, 1, True> at 1
<RIPRouter s2> got <DiscoveryPacket from s5->NullAddress, 1, True> at 2
<RIPRouter s5> got <DiscoveryPacket from s2->NullAddress, 1, True> at 1
<RIPRouter s2> got <RoutingUpdate from s6->NullAddress> at 0
<RIPRouter s9> got <RoutingUpdate from s8->NullAddress> at 0
<RIPRouter s8> got <RoutingUpdate from s8->NullAddress> at 1
<RIPRouter s8> got <RoutingUpdate from s9->NullAddress> at 1
<RIPRouter s2> got <RoutingUpdate from s6->NullAddress> at 0
<RIPRouter s7> got <RoutingUpdate from s6->NullAddress> at 0
<RIPRouter s8> got <RoutingUpdate from s6->NullAddress> at 0
<RIPRouter s7> got <RoutingUpdate from s9->NullAddress> at 1
<RIPRouter s8> got <RoutingUpdate from s9->NullAddress> at 1
```



完成测试;