通信与网络——网络路由实验

一、 作业要求

1. 利用提供的网络仿真器，巩固课堂所学计算机网络的相关路由算法知识，理解两种动态路由算法的基本思想，掌握二者的相同点和不同点；
2. 通过阅读相关代码实现，结合GUI界面，掌握距离矢量法（DV）的具体实现过程；
3. 完成Dijkstra算法的部分代码实现，结合GUI界面，验证算法实现正确性，掌握链路状态法（LS）的实现过程；
4. 通过阅读并完成相关代码实现，结合GUI界面，分析链路故障时，两种动态路由算法的解决故障过程；
5. 初步掌握在Linux系统环境中，利用python实现网络仿真和算法实现的能力。

二、 作业内容

搭建仿真环境：

1. 安装虚拟机软件VMware Workstation（建议12 Pro版本）；
2. 从清华云链接https://cloud.tsinghua.edu.cn/f/c29abaab4f3744258851/?dl=1下载相关配置文件，在VMware主界面中点击右上角“文件(F)”，在弹出来的下拉框中点击“打开(O)”，选择下载文件的目录，打开.vmx文件，之后根据提示选择.vmdk文件，点击“开启虚拟机”，选择“我已复制该虚拟机”，完成仿真环境搭建；
3. Ubuntu系统用户名：cn，密码：12345678，桌面存放本次作业相关文件，可选择系统自带的Visual Studio Code编写和调试代码。开启VS Code卡慢，可在Terminal输入：

```
code --disable -gpu
```

代码理解和实现：

1. 本次作业的主要文件和功能介绍如下表：

主要文件名称	主要文件功能
0_net.json	构建网络的数据文件
0_net_events.json	网络拓扑同0_net.json，会产生链路断开的异常事件
visualize_network.py	根据.json文件中内容构建网络，为用户client和路由器router开启多个线程，跟踪链路状态，执行相应的路由算法
packet.py	定义网络中client和router发送的数据包的Packet类
link.py	定义router, client之间的链路link类
client.py	定义周期性发送traceroute数据包的用户Client类
router.py	定义router的基类（Base Class）Router类
DVrouter.py	从Router继承的基于DV算法实现的DVrouter类
LSrouter.py	从Router继承的基于LS算法实现的LSrouter类
LSP.py	定义LS算法中链路状态包的LSP类
reference.pdf	参考资料

其中LSrouter.py需要完成部分代码，参见"""TODO:注释处。

注意：只可以修改LSrouter.py中部分代码和DVrouter.py中的debugString()函数。

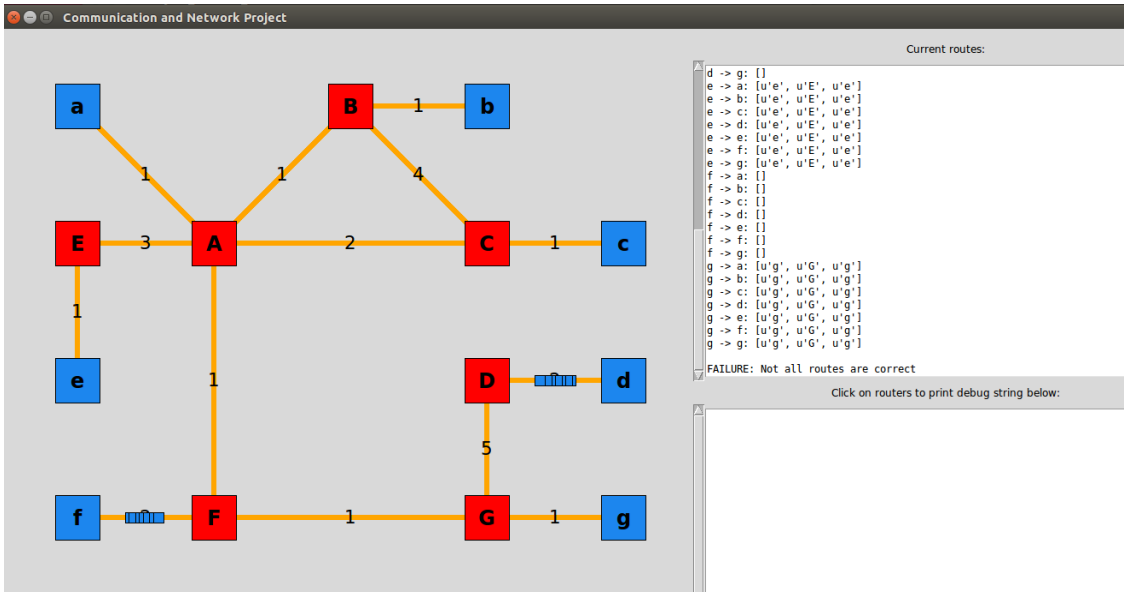
---

2. 熟悉网络仿真器：

- 本次作业基于一个对真实网络高度抽象的网络仿真器进行，包括了路由器（router）、用户（client）、链路（link）和数据包（packet）等基本元素。打开Terminal进入Project目录，可以输入下面代码运行网络仿真器的图形化界面，

```
python visualize_network.py 0_net.json
```

- 实现router.py中的路由算法，打开的图形化界面如下图，



- 其中，红色方块代表router，蓝色方块代表client，链路上的数字代表相应的cost，蓝色小方块和红色小方块分别代表client发出的数据包（traceroute packet）和router发出的路由包（routing packet）。
- client周期性发送traceroute packet到每个client，右上角的文本框显示最新的用户之间的路由，路由算法正确时将提示"SUCCESS: All routes are correct!"
- 点击某个client，GUI将只显示目的地地址为该client的packet，再次点击client将恢复显示所有packet。
- 点击某个router，右下角的文本框将根据对应算法中的debugString()函数，显示相关内容，用于调试路由算法的实现和分析链路故障的解决过程。
- 在Terminal分别输入下面代码可显示执行DV和LS算法的网络仿真，

```
python visualize_network.py 0_net.json DV
python visualize_network.py 0_net.json LS
```

---

3. 理解代码实现：

- 阅读DVrouter.py，结合具体的注释，理解DV算法的实现过程和解决链路故障、新增链路等问题时的方法。针对0_net.json文件，结合GUI界面，重点分析**init**()、handlePacket()、updateNode()、debugString()函数的功能和实现，明白**init**()函数中定义的dict含义。

---

4. 阅读LSrouter.py和LSP.py，理解LS算法的具体实现流程，回答思考题。

---

5. Dijkstra算法实现：

- 结合Dijkstra算法，补全LSrouter.py中的calPath()函数。针对0_net.json文件，结合GUI界面，验证算法实现正确性。

6.链路故障分析：

- 针对0_net_events.json文件，首先观察DVrouter.py解决链路故障的过程，如每个router中存储的距离矢量和路由表的变化，可以修改DVrouter.py中的debugString()函数进行分析。
- 针对0_net_events.json文件，通过补全debugString()函数，观察LSrouter.py解决链路故障的过程。

思考题：

- 请给出LSrouter.py初始化函数中定义的dict含义（key-value），参见下面代码###部分。

```
19      def __init__(self, addr, heartbeatTime):
20          """class fields and initialization code here"""
21          Router.__init__(self, addr)  # initialize superclass - don't remove
22          self.routersLSP = {}  ###
23          self.routersAddr = {} ###
24          self.routersPort = {} ###
25          self.routersNext = {} ###
26          self.routersCost = {} ###
27          self.seqnum = 0 ###
28          self.routersLSP[self.addr] = LSP(self.addr, 0, {})
```

  提示：实际网络中用端口号（port）表示链路（link）

- 请给出你对LSP.py中更新函数的执行过程的理解，参见下面代码。

```
8       def updateLSP(self, packetIn):
9           if self.seqnum >= packetIn["seqnum"]:
10              return False
11          self.seqnum = packetIn["seqnum"]
12          if self.nbcost == packetIn["nbcost"]:
13              return False
14          if self.nbcost != packetIn["nbcost"]:
15              self.nbcost = packetIn["nbcost"]
16              return True
17
```

- 请给出你对LSrouter.py的handlePacket()函数中Routing packet处理过程的理解，参见下面代码。

```
32
33      def handlePacket(self, port, packet):
34          """process incoming packet"""
35          # deal with traceroute packet
36          if packet.isTraceroute():
37              if packet.dstAddr in self.routersNext:
38                  next_nb = self.routersNext[packet.dstAddr]
39                  self.send(self.routersPort[next_nb], packet)
40          # deal with routing packet
41          transfer = False
42          if packet.isRouting():
43              if packet.dstAddr == packet.srcAddr:
44                  return
45
46              packetIn = loads(packet.content)
47              addr = packetIn["addr"]
48              seqnum = packetIn["seqnum"]
49              nbcost = packetIn["nbcost"]
50              if addr not in self.routersLSP:
51                  self.routersLSP[addr] = LSP(addr, seqnum, nbcost)
52                  transfer = True
53              if self.routersLSP[addr].updateLSP(packetIn):
54                  transfer = True
55
56              if transfer:
57                  for portNext in self.routersAddr:
58                      if portNext != port:
59                          packet.srcAddr = self.addr
60                          packet.dstAddr = self.routersAddr[portNext]
61                          self.send(portNext, packet)
62
```

三、 作业提交

- 提交补充代码后的LSrouter.py
- 完成作业内容第3~6题，并逐题进行分析回答，必要时附上对应结果的截图