《通信与网络》动态路由实验指导书

一、 实验目的

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、 实验内容

1. 搭建仿真环境：
   a) 安装虚拟机软件 VMware Workstation（建议 12 Pro 版本）；
   b) 从清华云链接 https://cloud.tsinghua.edu.cn/d/1d5ec2a7292e43d997e0/下载相关配置文件（由于文件较大，可能需要安装客户端，具体请参看清华云相关说明），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；
   c) Ubuntu 系统用户名：cn，密码：12345678，桌面存放本次实验相关文件，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
```

2. 代码理解和实现：
   a) 本次实验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。

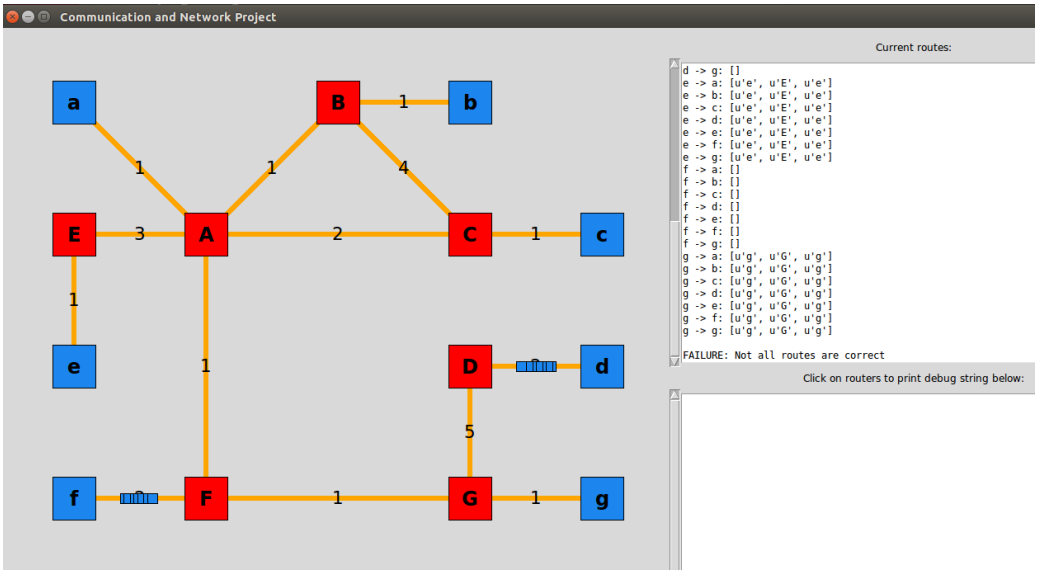
注：只可以修改 LSrouter.py 中部分代码和 DVrouter.py 中的 debugString()函数。

b) 熟悉网络仿真器：本次实验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
```

c) 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。

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

e) 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。

f) 链路故障分析 1：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。

g) 链路故障分析 2：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。

3. 思考题：
   a) 请给出 LSrouter.py 初始化函数中定义的 dict 含义（key-value），参加下面代码###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）

   b) 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
```

   c) 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。

```
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)
```

三、 实验验收

1. 提交补充代码后的 LSrouter.py；
2. 完成实验内容第 2 部分第(c)-(g)小题，并逐题在实验报告中进行分析回答，必要时附上对应结果的截图；
3. 提交实验报告。