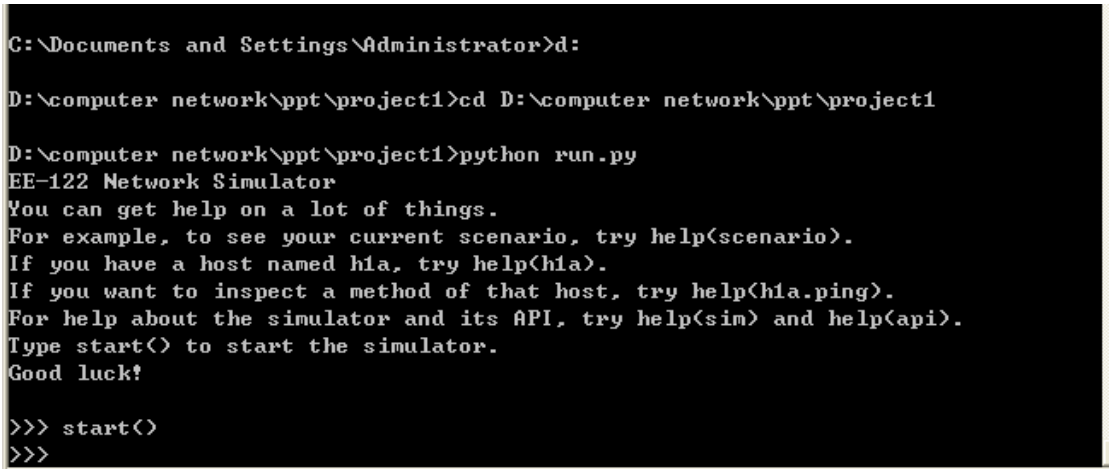
计算机网络 路由仿真指南

注意：强烈建议先不看代码，先玩起来再说。具体按照下面操作来玩。事实上，遵循下面步骤，就可以完成此次作业。

第一步：让框架代码能够运行起来。

将 project3 解压到一个路径。运行 cmd，进入该路径，运行如下代码：python run.py
注意 1：如果以上方法不行，可以仅输入 run.py 试一下；如果还不行，可以使用 IDLE 打开 run.py 文件，然后运行它。
注意 2：解压到的文件路径上，最好不要含有任何中文。
会出现如下界面：



输入 start()，框架代码运行开始。
注：运行成功后，你可以跳到第三步先玩，然后再回到第二步。

第二步：加载你的场景

文件 run.py 中的第 23、24 行是加载场景的代码：

```
# import scenarios.linear as scenario
# import scenarios.candy as scenario
```

可任选其一加载，而将另外的注释掉。
注：此处的场景是指虚拟的网络，即有几台主机，路由器，以及它们是如何相连接的。
创建新场景的方法有两种：

方法 1：编写场景文件

在 scenarios 文件夹中，添加新的文件，如 newscernarios.py, 参照 linear 的写法来构造场景：

首先，你需要引入（import） 模块：

```
import sim
from sim.core import CreateEntity, topoOf
from sim.basics import BasicHost
from hub import Hub
from sim import topo as topo
```

接着，你需要定义一个 create 函数：

```
def create (switch_type = Hub, host_type = BasicHost):
    """
    create my own topo.
    """
```

在这个函数里，调用 host_type.create(主机名)创建主机(host)对象，调用 switch_type.create(交换机名)创建交换机（switch）对象，然后通过 topo.link()函数，将他们（即主机、交换机等）连接起来：

如，

```
h1 = host_type.create('h1')   #新建一台主机 h1
s1 = switch_type.create('s1') #新建一台交换机 s1
topo.link(h1, s1)             #用网线连接主机 h1 和交换机 s1
#通过构造对象实体，并且用 link 将他们连接起来，就可以构造你自己的场景了。
```

Switch_type 默认是 Hub 类型的实体，如果想构建 RIPRouter 或者 LearningSwitch，更改 switch_type, 让 switch_type = RIPRouter

方法 2：在已加载的场景中添加新的结点和连线

run.py 文件会默认的加载一个场景，可以通过命令来修改此场景。如，添加一台 RIP 路由器（RIPRouter 对象）s3，并且用线将 s3 和 s1 连起来的指令如下：

```
>>> start()
>>> from rip_router import RIPRouter
>>> from sim import topo as topo
>>> s3 = RIPRouter.create('s3')
>>> topo.link(s3, s1)
(0, 2)
>>>
```

注意：上面的过程会出错的(因为 RIPRouter 代码你还没有编写，我们只写了 Hub)，正确的做法是将所有的 RIPRouter 换成 Hub,而将 rip_router 换成 hub。

通过调用 create（）函数来构建对象，

如果想构造一台集线器（Hub），可通过调用 Hub.create(对象名),

如果想构造一台主机（Host），可通过调用 BasicHost.create(对象名)

如果想删除某个节点，可调用 remove()函数。例如 s3.remove()

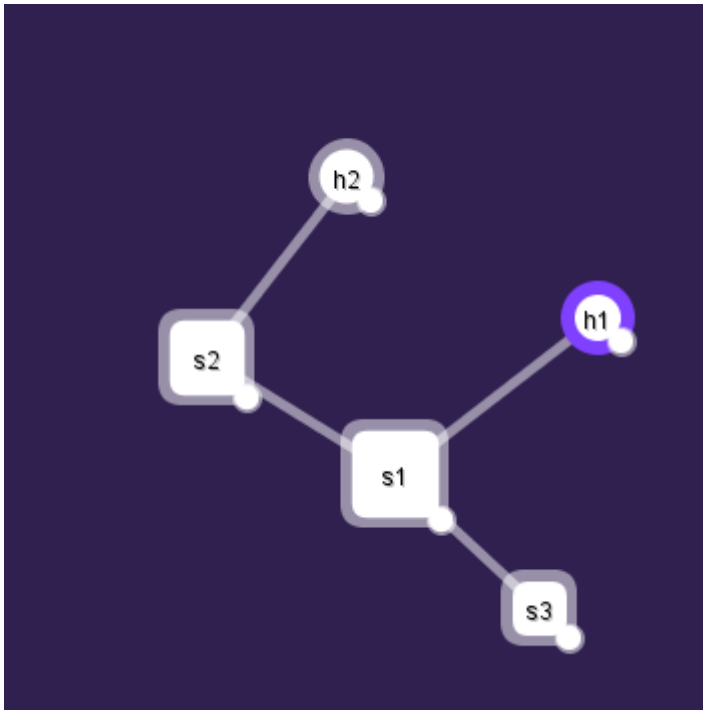
如果想删除某条连线，可调用 disconnect()函数。

第三步：让可视化界面动起来

在 netvis/bin 里面，解压你所用操作系统的压缩包。例如，如果你使用 windows 操作系统，可解压 windows.zip 文件，会得到 application.windows 文件夹，进入该文件夹，双击 NetVis.exe 即可。（其他系统自己玩吧，支持 APPLE MAC 和 LINUX）

如果在此之前你已经启动了第一步，即运行了 run.py，并且输入了 start（），那么，你的场景就会在窗口中显示出来。

如：



注意：Netvis 是 Java 编写的，所以一定要确保你的电脑里面有 Java 的运行环境，如何配置，自行百度”配置 java 的运行环境，设置环境变量“等关键字。

第四步:ping 操作

可在主机（Host）上模拟进行 ping 操作。如：你可以输入 h1.ping(h2)，意即在主机 h1 上运行“ping 主机 2”的指令。那么，h1 就会向 h2 发送一个包，你可以在可视化界面中看到包的运动轨迹。

注意：只能在主机（Host 对象）上运行 ping 操作，交换机上（switch 对象）不能够进行 ping 操作。

概念解释

PING 是什么：PING 是一个应用层的协议，其实质应该是 PING-PANG 更合适一些，即发送的主机发送 PING 消息，而接收的主机会发送 PANG 消息。应用层协议仅在主机上实现，路由器交换机等网络设备仅实现网络层以下的协议。因此在主机上有 PING 函数，在交换机上是没有 PING 函数的。即在交换机上无法使用 PING，而如果 PING 的目标是交换机，交换机也不会有 PANG 的动作。

第五步：编写学习型交换机 LearningSwitch

Run.py 文件第 14 行，默认选择 Hub 作为 switch。
查看 hub.py 文件，你可以看到，里面只有一个处理包的函数 hand_rx(packet，port，flood).
第一个参数，包对象
第二个参数：发出包的端口
第三个对象，是否 flood
在 hub 中，没有路由转发表，对于任意包的处理都是 flood。
你需要编写一个 LearningSwitch 类，构造一个学习型交换机。
在学习型交换机里，你需要有一个路由转发表。
LearningSwitch和Hub路由很像，只需要重写handle_rx(self, packet, port)函数即可。
LearningSwtich 需要有一个路由转发表，首先你需要使用 flood 的方法，构建一个路由转发表，即当你收到一个包，你需要记录该包的来源，存储在路由转发表中，然后通过 flood 的方法，把该包发送出去。如果是其他类型的包，你只需要在路由表中查询对应的端口号，然后 send 出去即可，不需要 flood。（具体参加 ppt 里面的学习型交换机算法）。
构造好 LearningSwitch 以后，你需要更改 run.py 里面 switch，添加下面代码：
import LearningSwitch as switch，注释其他的 switch

LearningSwithch 算法，参考 ppt 里面的：

```
当交换机收到一个包:
使用目标 ID 检索交换机中的路由表
if 在路由表中找到目标 ID 的入口项 {
     如果 目标对应的端口就是包到达的端口
       那么 drop packet
     否则在对应（指定）的端口转发包
 }
    else flood
```

上面代码是一个比较完整的代码，如果你自己觉得自己的水平不是很高，可以试着完成下面这个稍简单的代码：

```
使用目标 ID 检索交换机中的路由表
if  在路由表中找到目标 ID 的入口项 {
     则在对应（指定）的端口转发包
 }
    else flood
```

问题：上面二个代码都实现了，很显然前一个代码更好一些，你能否设计一种网络，用实验证明上面的代码更好。

第六步：编写 RIPRouter

在 rip_router.py 文件中，完成 RIPRouter
现在，你需要实现RIPRouter。RIPRouter和LearningSwitch很像，都需要通过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，你需要将包的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，你以前没有收到，这时你需要把包的src存放到路由表中，如果你的路由表中有该src的记录，你需要比较路径的长度，如果长度比你的路由表中存的小，你需要更新你的路由表,然后将自己的路由表里面的信息flood出去。

```
你需要用到isinstance这个函数判断包的类型：
if isinstance(packet, DiscoverPacket):
    将packet.src存放在路由表中
elif isinstance(packet, RoutingUpdate):
    更新自己的路由表
    如果有更新，将路由表flood出去。
    将自己的路由表信息flood出去的时候，你需要构建自己的RoutingUpdate包，包里面存放的就是路由表的信息。
else:
    查找路由表，进行发包
```

第七步：查看 log 信息

框架提供了 log 信息的打印。
你需要另外开启一个 cmd，然后进入 project1 的路径，输入命令：
python logviewer.py
这样，你有三个窗口，一个是 run.py, 一个 logview，还有一个可视化窗口。