Project1 Distance Vector Routing

项目介绍

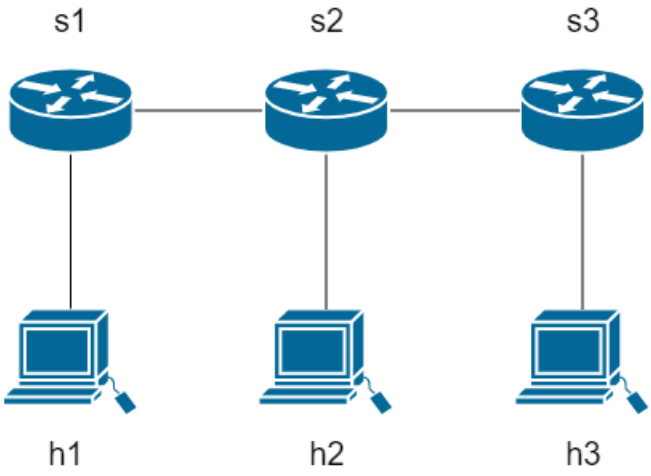
本项目来自伯克利大学 **cs168**课程(网址为https://github.com/NetSys/cs168_proj_routing_student)，该项目的目标是让您学习实现分布式路由算法，其中所有路由器都运行一种算法，允许它们将数据包传输到目的地，但没有中央机构确定转发路径。 您将实现在路由器上运行的代码，我们将提供一个路由模拟器，该模拟器构建一个将您的路由器相互连接并连接到网络上模拟主机的图形。 在本项目结束时，您将实现一个距离矢量协议版本，该协议可计算网络中的有效路径。(详细的介绍可以参照文件夹中的`project1_writeup.pdf`文件，里面有更加详细的说明与指导)

Quick Start

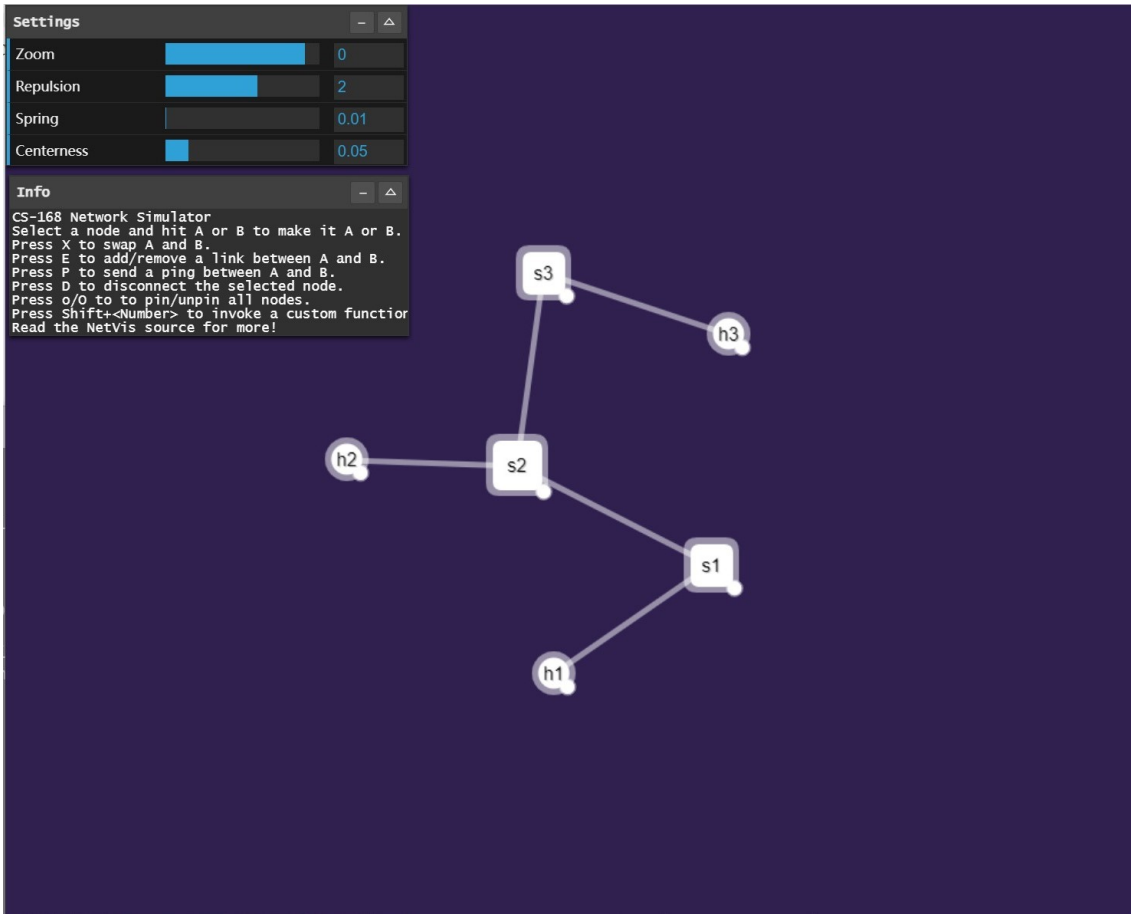
1. 首先确保你的运行环境为`python3`，请打开命令行输入`python --version`来确定你的版本为`python3`版本。(对操作系统无要求，可以使用windows和linux)
2. 解压文件，进入`simulator`文件夹，在该文件夹下打开命令行。
3. 我们的目标是编写一个具有RIP路由协议的路由设备，那么我们作为演示我们提供一个`hub`的实现，`hub`是一种网络设备，它可以将它收到的任何数据包泛洪到它所有的端口(除了数据包来的端口)。该集线器我们已经实现，可以直接使用。

   下面我们尝试使用一个有着三个hosts的线性拓扑图(如下图所示)，我们使用之前提供的默认的hub来充当网络中的路由器，在命令行中输入如下指令：

   ```
   python simulator.py --start --default-switch-type=examples.hub
   topos.linear --n=3
   ```



之后就可以使用可视化工具，请使用浏览器访问`http://127.0.0.1:4444`，此时可以观察到整个路由结构并且可以拖动各个部件，如下图所示：



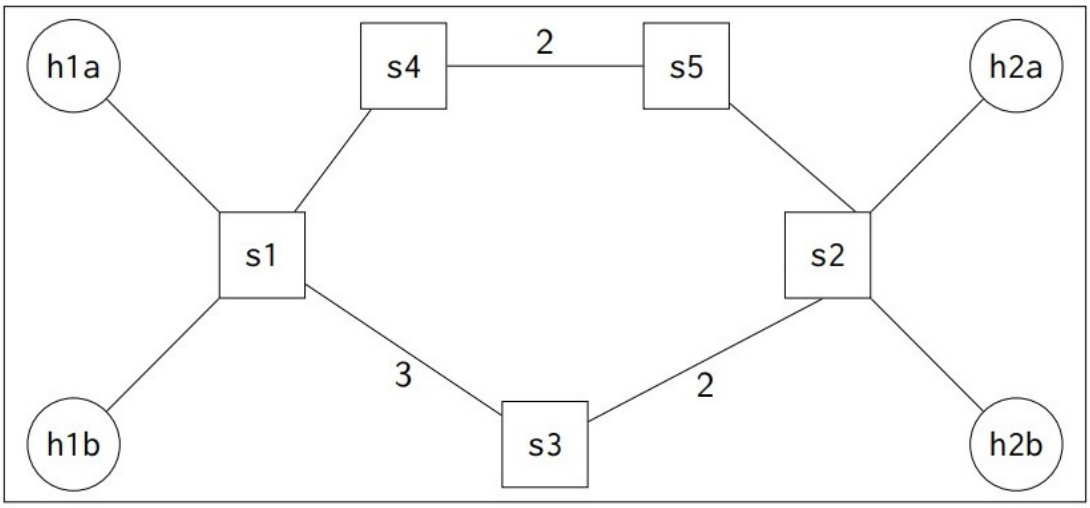
此时在原来的命令行中输入如下参数：

```
>>> h1.ping(h3)
```

此时应该可以看到ping和pong数据包在网络中出现，但是可以注意到h2端此时也接收到了来自h1的ping包和h2的pong包(WARNING信息如下图所示)，这种行为是可以预料到的，因为hub只是接收数据包后向除过接收数据包的端口之外的所有端口泛洪数据包。

```
WARNING:user:h2:NOT FOR ME: <Ping h1->h3 ttl:17> s1,s2,h2
DEBUG:user:h3:rx: <Ping h1->h3 ttl:16> s1,s2,s3,h3
WARNING:user:h2:NOT FOR ME: <Pong <Ping h1->h3 ttl:16>> s3,s2,h2
DEBUG:user:h1:rx: <Pong <Ping h1->h3 ttl:16>> s3,s2,s1,h1
```

如果网络中出现环路，那么可能使用hub这种方式连接网络会出现比较大的问题，此时我们使用另一个网络，该网络为环形网络，拓扑结构如下图所示：



此时退出原命令(输入`exit()`或者直接按`ctrl`+`z`),再次在命令行中输入：

```
python simulator.py --start --default-switch-type=examples.hub
topos.candy
```

此时从主机h1a向主机h2b发送ping命令。此时可以看到已经出现泛洪问题，我们观察到有过多的冗余数据包在网络中流动，这也就是我们为什么要编写新的路由器，编写新的路由协议的原因。

ATTENTION

> 在这里命令行中输入什么样的规则请参照文件夹中的`simulator_guide.pdf`文件，里面有如何进行调试的详细讲解(不需要过多了解，因为后面的测试都会给出测试的命令行参数)

Test

最后验收分为必选项和可选项(可选项**不会对分数造成太大影响**，主要目标是让大家体验下伯克利大学的作业形式)

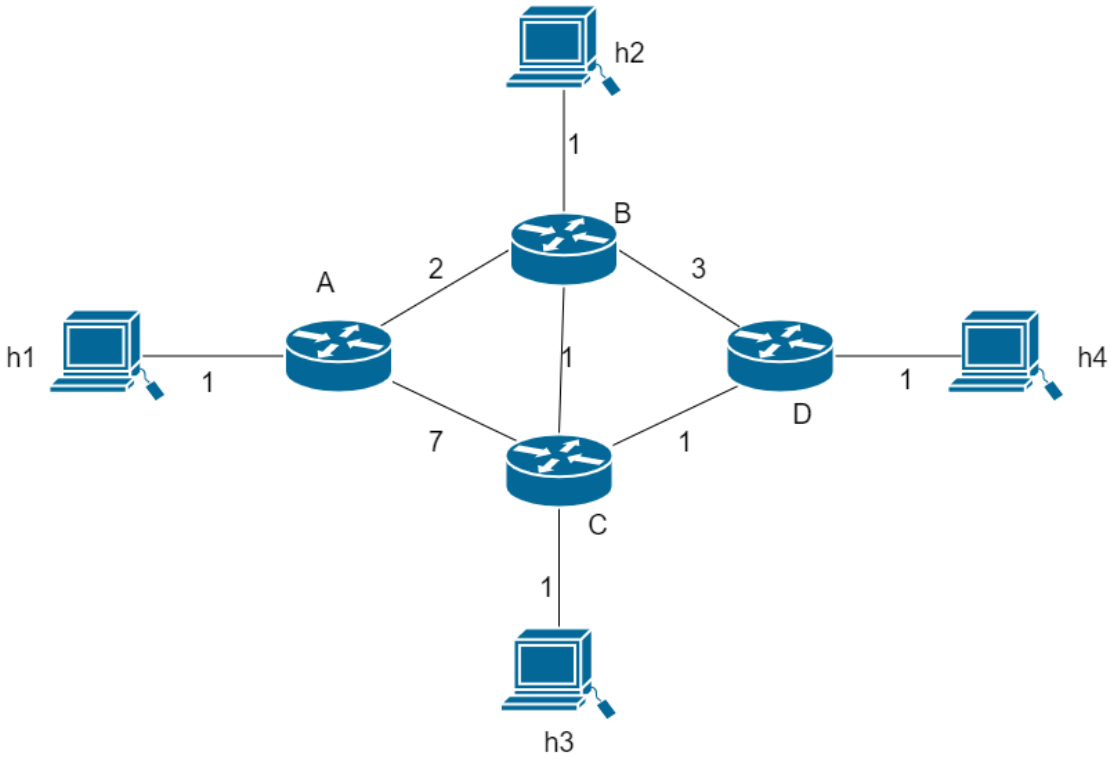
必选项

过程如下所示：

1. 熟悉所给资料：平台代码，仿真器(可以对编写的部分进行测试)。

   > 一些重要文件(需要阅读)如下：
   >
   > **simulator.py** Starts up the simulator
   >
   > **dv_router.py** Starting point for your distance vector router
   >
   > **dv_unit_tests.py** Stage-by-stage unit tests for your distance vector router.
   >
   > **dv_comprehensive_test.py** Comprehensive test for your distance vector router.
   >
   > **sim/api.py** Parts of the simulator that you'll need to use (such as the Entity class). See help(api).
   >
   > **sim/basics.py** Basic simulator pieces built with the API. See help(basics).
   >
   > **sim/core.py** Inner workings of the simulator.
   >
   > **topos/** Test topologies and topology generators that you can use and modify for your own testing.
   >
   > **examples/** Examples for Entities, interacting with NetVis, automating your testing, and so forth.
   >
   > **cs168/** Additional topologies.

2. 编写网络拓扑代码， 可以参考`topos`文件夹中的其他拓扑结构，然后在该文件夹中编写自己的拓扑文件(注意添加权重，也就是latency，可以参考别的topo文件进行编写)。要实现的拓扑结构如下图所示：



3. 编写路由器代码(实际上是补全`dv_router.py`)，实现`project1_writeup.pdf`文件中的***state 1 ~ 5***(在每一个state处均有详细的介绍与测试方法，这里不再赘述)，此时应该已经实现了RIP路由器的基本功能，使用编写的网络拓扑代码和路由器代码进行下面的测试：

```
python simulator.py --start --default-switch-type=dv_router topos.myTopo
```

此时的myTopo是因为`topos`文件夹下有自己编写的拓扑文件`myTopo.py`（也就是上一步我们编写的网络拓扑文件），使用时替换为自己定义的名字即可。

```
h1.ping(h4)
```

最后网络中没有出现泛洪现象，同时包走的是最短路径(A -> B -> C -> D)，说明我们的RIP算法设计成功。

ATTENTION

对于state 1 ~ 5，每一个state都有相应的测试，只有测试结果正确才能说明我们的路由器代码编写正确(最终验收时可以展示每一个state的测试也可以不展示，只要**最终的测试——即上面的测试通过即可**)

可选项

完成***state 6~10***，通过每一个state的相关测试即可(具体要求请参照`project1_writeup.pdf`中相关要求)

Attention

1. 编写的路由器代码时，只需要修改`dv_router.py`文件，不需要添加其他文件。
2. 在`dv_router.py`中提供的构造函数(`__init__`)不要修改和删除，在后面的测试中你会使用到他们。
3. 在编写路由器代码时严禁使用全局变量，类变量调用其他实例的方法，每个DVrouter应该完全独立。
4. 可以讨论代码的思路，**严禁抄袭代码**（包含上一届学长的代码），希望保持高度的学术诚信，并且为自己的工作感到自豪。涉嫌作弊或伪造的作业将根据课堂指定的相关规则予以处理。

5. 如遇到代码框架的相关疑问或者问题(不包括DVrouter的实现部分)，请在群里联系`计算机实验班81王烨`(qq：`983754578`)，欢迎对本次实验提出建议和意见，以助于更好的改进课程，提升课程质量。