实验的第一部分旨在修改代码，第二部分就是实际要操作的了。

（1）首先，可以直接把Git上的克隆到Ubuntu的桌面上，在桌面打开终端；

注：如果桌面找不到Git下来的tutorial文件夹，可以打开文件资源管理器，就可以看到tutorial文件夹。我们需要的文件就是tutorial/exercises/basic\_tunnel文件夹下的myTunnel\_header.py

receive.py

send.py

以及和以上文件在同一个文件夹下的solution中的basic\_tunnel.p4文件，这个是已经改好的p4代码。把这几个文件都移到桌面上。

指令：gti clone <https://github.com/p4lang/tutorials.git>

（2）打开终端，cd到桌面，命令：cd ~/Desktop

注：其中~的含义我猜是回到根目录，类似于home。

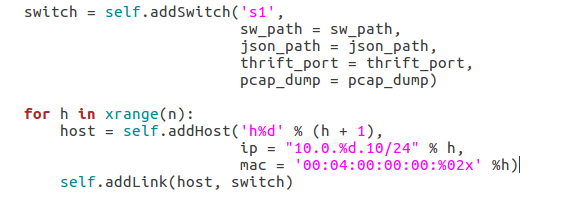
由于之前的p4文件basic\_tunnel.p4还没编译，就需要用命令

p4c-bm2-ss basic\_tunnel.p4 -o mytest.json

编译我们的p4文件。如果成功编译会在桌面生成一个mytest.json文件。

（3）可以去写拓扑了：

可以先去把这个topo文件p4/behavioral-model/mininet/1sw\_demo.p复制到桌面，再删去里面关于交换机和主机的定义，就中间那块

用以下代码代替。将文件重命名为mytopo.py。

拓扑的定义部分：

switch1 = self.addSwitch('s1',

sw\_path = sw\_path,

json\_path = json\_path,

thrift\_port = 9090,

pcap\_dump = pcap\_dump)

switch2 = self.addSwitch('s2',

sw\_path = sw\_path,

json\_path = json\_path,

thrift\_port = 9091,

pcap\_dump = pcap\_dump)

switch3 = self.addSwitch('s3',

sw\_path = sw\_path,

json\_path = json\_path,

thrift\_port = 9092,

pcap\_dump = pcap\_dump)

host1 = self.addHost('h1',

ip = "10.0.0.10/24",

mac = '00:04:00:00:00:01')

host2 = self.addHost('h2',

ip = "10.0.1.10/24",

mac = '00:04:00:00:00:02')

host3 = self.addHost('h3',

ip = "10.0.2.10/24",

mac = '00:04:00:00:00:03')

self.addLink(host1, switch1)

self.addLink(host2, switch2)

self.addLink(host3, switch3)

self.addLink(switch1,switch2)

self.addLink(switch2, switch3)

self.addLink(switch3, switch1)

注：交换机有三个，s1，s2，s3；分别为他们三个分了三个端口9090,9091,9092；由于一个交换机只能绑定一个CLI端口，CLI用于给交换机添加流表。所以在登录CLI的时候参数只有端口号，就指定了是为哪个交换机在添加流表。

（4）将p4/behavioral-model/mininet/下的p4\_mininet.py拷贝到桌面上，因为topo的前面有一句**from p4\_mininet import P4Switch, P4Host，**不复制的话运行topo会报错。

（5）接下来在桌面新建三个文本文件，用于存储需要添加的流表命令：文件命名为command1.txt

command2.txt

command3.txt

其中：

**Command1.txt**内容如下：

table\_set\_default ipv4\_lpm drop

table\_set\_default myTunnel\_exact drop

table\_add ipv4\_lpm ipv4\_forward 10.0.0.10/32 => 00:04:00:00:00:01 1

table\_add ipv4\_lpm ipv4\_forward 10.0.1.10/32 => 00:04:00:00:00:02 2

table\_add ipv4\_lpm ipv4\_forward 10.0.2.10/32 => 00:04:00:00:00:03 3

table\_add myTunnel\_exact myTunnel\_forward 1 => 1

table\_add myTunnel\_exact myTunnel\_forward 2 => 2

table\_add myTunnel\_exact myTunnel\_forward 3 => 3

**command2.txt内容如下：**

table\_set\_default ipv4\_lpm drop

table\_set\_default myTunnel\_exact drop

table\_add ipv4\_lpm ipv4\_forward 10.0.0.10/32 => 00:04:00:00:00:01 2

table\_add ipv4\_lpm ipv4\_forward 10.0.1.10/32 => 00:04:00:00:00:02 1

table\_add ipv4\_lpm ipv4\_forward 10.0.2.10/32 => 00:04:00:00:00:03 3

table\_add myTunnel\_exact myTunnel\_forward 1 => 2

table\_add myTunnel\_exact myTunnel\_forward 2 => 1

table\_add myTunnel\_exact myTunnel\_forward 3 => 3

**command3.txt的内容如下：**

table\_set\_default ipv4\_lpm drop

table\_set\_default myTunnel\_exact drop

table\_add ipv4\_lpm ipv4\_forward 10.0.0.10/32 => 00:04:00:00:00:01 3

table\_add ipv4\_lpm ipv4\_forward 10.0.1.10/32 => 00:04:00:00:00:02 2

table\_add ipv4\_lpm ipv4\_forward 10.0.2.10/32 => 00:04:00:00:00:03 1

table\_add myTunnel\_exact myTunnel\_forward 1 => 3

table\_add myTunnel\_exact myTunnel\_forward 2 => 2

table\_add myTunnel\_exact myTunnel\_forward 3 => 1

以上指令有2种：

table\_set\_default 表名 动作名

表示：当table中的key不匹配的时候默认的action，就是要丢掉包

table\_add 表名 如果key匹配了的动作名 key中的匹配字 => 对应的action的参数

举个例子：table\_add myTunnel\_exact myTunnel\_forward 1 => 3

就是说当myTunnel\_exact表中的 dst\_id匹配的时候，就采用myTunnel\_forward从交换机的3端口发出。这里的1指的是主机ID。在此处默认h1为1，h2为2，以此类推。再举个栗子：

table\_add ipv4\_lpm ipv4\_forward 10.0.0.10/32 => 00:04:00:00:00:01 3

分析一下:

Ipv4\_lpm这个表的key是32位的IP地址，而ipv4\_forward的函数参数是mac地址和端口号。此处的mac地址就是目的主机的mac地址。IP也是目的主机的IP。端口号表示的是从交换机哪个端口发出。

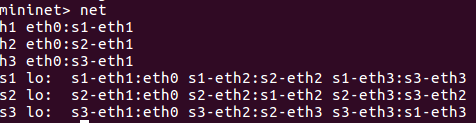
现在就可以开始操作了：

1. 在桌面打开终端，运行topo：

cd ~/p4/behavioral-model/mininet



(2可以输入在mininet输入net查看当前拓扑：



具体含义：如：

**H1 eth0:s1-eth1就表示h1的0端口连着s1的1端口。所以如果要通过交换机s1给h1发包的话就要从s1的1端口发出。**

1. **在mininet终端输入xterm h1 h2 s1，表示为这三个分别打开终端。**
2. **接下来再打开一个terminal终端，输入：**



**这两条指令。**

**第二行中的9090端口们再拓扑定义中给了s1交换机，所以现在是在给s1添加表项。**

1. **再打开一个终端为s2添加表项：**



1. **再开一个终端为s3添加：**



**此时在为s1开的终端输入sudo wireshark打开wireshark，准备抓包。**

**此时可以在mininet想让谁ping谁都行。可以抓包看自定义的mytunnel字段。**

**接下来需要的步骤：**

1. **在为h2打开的终端cd到桌面，运行receive.py程序作为接收端**

**指令就是一般的Python程序运行指令**

1. **在为h1打开的终端cd到桌面，运行send.py**

**此处的指令为两种：**

**python send.py 10.0.1.10 “hello”默认为用ipv4给IP地址为10.0.1.10的主机发hello**

**Python send.py 10.0.1.10 “hello --dst\_id=2 用没有tunnel字段解析发送。**

**可以在wireshark选择要抓包的端口。在报文中查看值为0X1212的字段，因为定义的mytunnel字段值为0x1212.**