Programming Assignment 3

CST 311, Introduction to Computer Networks, Fall 2018

READ INSTRUCTIONS CAREFULLY BEFORE YOU START THE ASSIGNMENT.

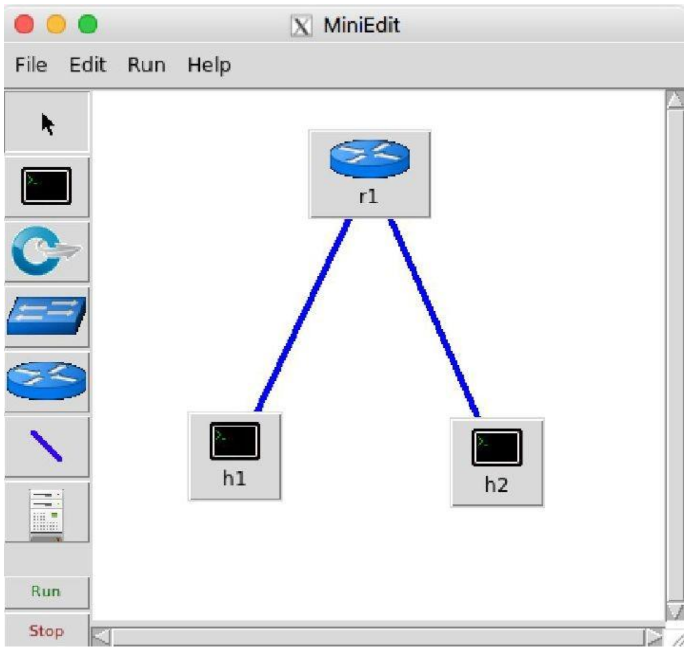
This programming assignment is due on Sunday, November 25, 2018.

Assignment must be submitted electronically to iLearn on https://ilearn.csumb.edu by 11:55 p.m. on the due date. Late assignments will not be accepted.

The assignment requires you to submit modified legacy_router.py program and another file containing items listed in the section **"Grading objectives and What to turn in"** below. You do not need to demo this programming assignment in person. You will be working in teams of two on this assignment. **Put your names in the program as well as the document with answers that you turn in.** Your program must have sufficient comments to clearly explain how your code works. This assignment is worth 100 points.

Subnet addressing in Mininet

In this assignment you will start with Python code that builds a 2 host network connected by a legacy router. You will modify it such that the two hosts can send packets to each other. It requires that you understand subnet addressing and the function of a router.
The network built using Miniedit on a Mininet virtual machine: python mininet/examples/miniedit.py



In order to run miniedit.py, you must install X server (XQuartz for MAC and XMing for Windows), if it is not already installed and setup X11 forwarding on your machine.
https://github.com/mininet/openflow-tutorial/wiki/Set-up-Virtual-Machine

The code generated by exporting it as a Level 2 Script: (unused imports and code added by Miniedit have been removed):

```
#!/usr/bin/python
# File: legacy_router.py
 from mininet.net import Mininet
from mininet.node import Host, Node
from mininet.cli import CLI
from mininet.log import setLogLevel, info
 def
myNetwork():

    net = Mininet( topo=None,
build=False,
ipBase='10.0.0.0/8')
     info( '*** Adding controller\n' )     info(
'*** Add switches\n')     r1 = net.addHost('r1',
cls=Node, ip='0.0.0.0')     r1.cmd('sysctl -w
net.ipv4.ip_forward=1')
     info( '*** Add hosts\n')     h2 = net.addHost('h2', cls=Host,
ip='10.0.0.2', defaultRoute=None)     h1 = net.addHost('h1', cls=Host,
ip='10.0.0.1', defaultRoute=None)
```

```
     info( '*** Add
links\n')
net.addLink(h1, r1)
net.addLink(h2, r1)
     info( '*** Starting
network\n')     net.build()
    CLI(net)
net.stop()
 if __name__ ==
'__main__':
setLogLevel( 'info' )
myNetwork()
```

Executing this code and trying a ping results in “Destination Host Unreachable”:

```
mininet@mininet-vm:~$ sudo python legacy_router.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
r1 h2 h1 *** Starting
CLI: mininet> h1 ping
h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
```

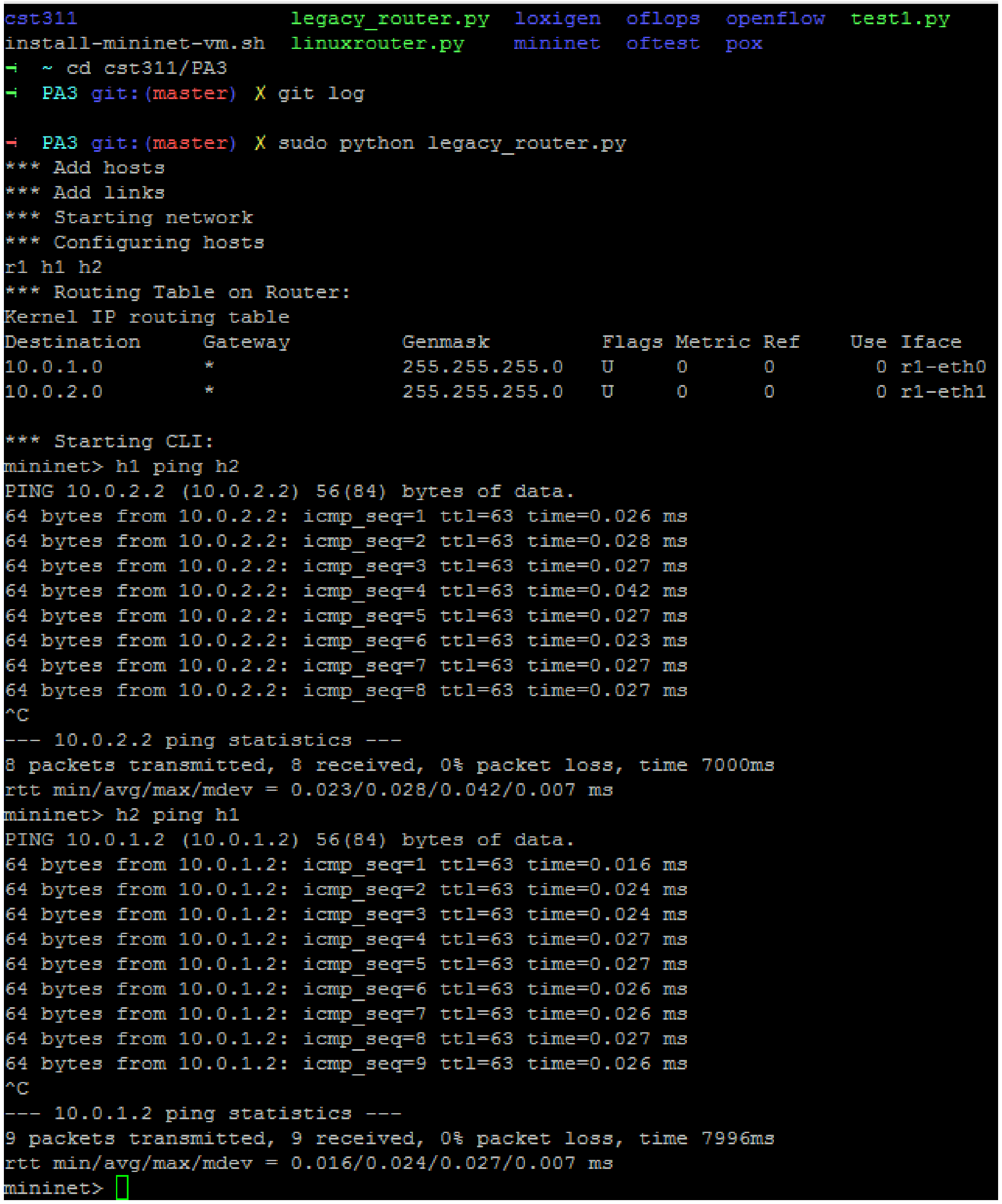
Your task is to modify this program (perhaps using different IP addresses) such that the legacy router is able to forward packets between the two hosts. You will need to understand Internet addressing, subnets, and the function of a router as described in the “IPv4 Addressing” Section of Kurose and Ross (4.3.3 in 7th Edition, 4.4.2 in 6th Edition). You may also find the example Python programs in mininet/examples helpful; in particular linuxrouter.py . I suggest executing that program and studying it to understand how you will need to modify legacy_router.py.

As you modify the program commit those changes using the ‘ git ’ version control software – use meaningful commit messages. Before committing and modifying the above program initialize your local repository with your name, e.g.:

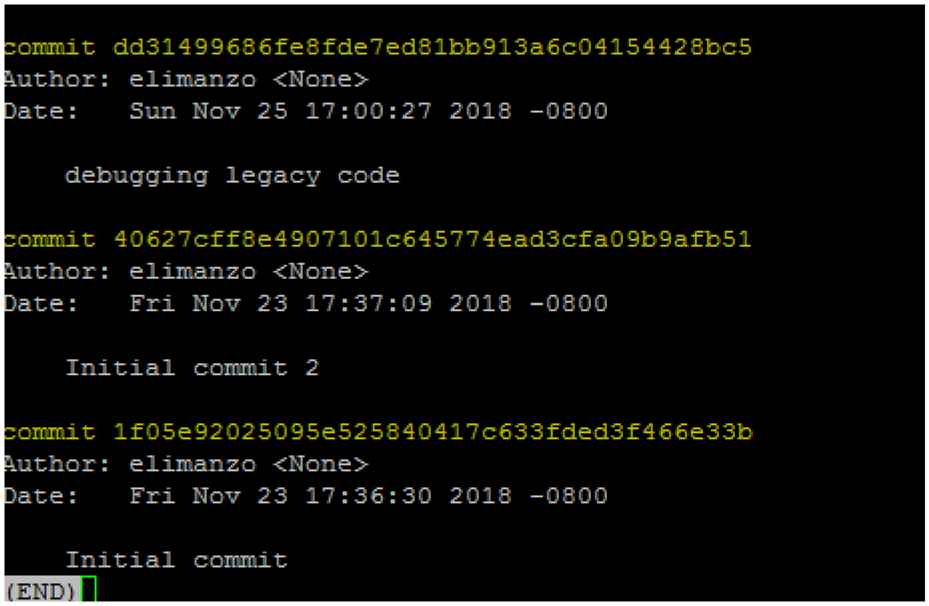
```
git init git config user.name
"Jane Student"
```

Grading Objectives and What to turn in:

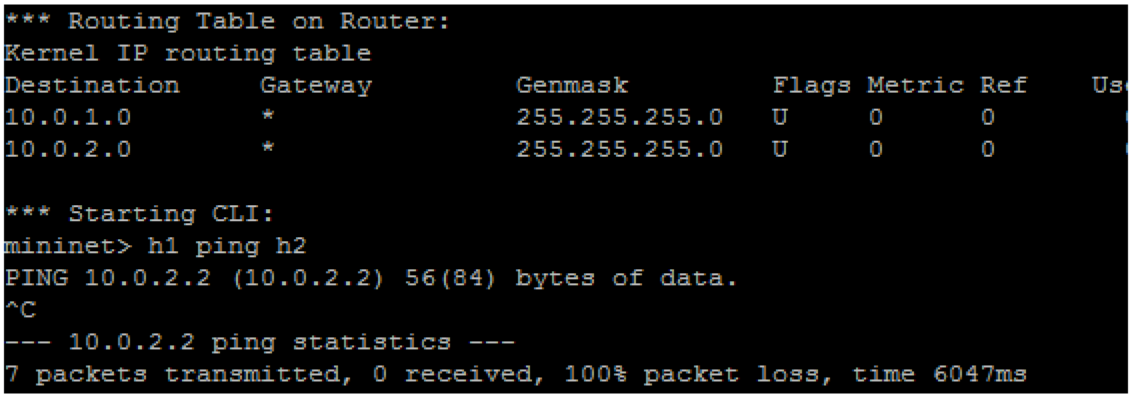
1. (25 points) Screen capture of successful ‘ h1 ping h2 ’ and ‘ h2 ping h1 ’ commands at the minnet> prompt.

   a. 

```
cst311                 legacy_router.py  loxigen  oflops  openflow  test1.py
install-mininet-vm.sh  linuxrouter.py    mininet  oftest  pox
→  ~ cd cst311/PA3
→  PA3 git:(master) ✗ git log

→  PA3 git:(master) ✗ sudo python legacy_router.py
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
r1 h1 h2
*** Routing Table on Router:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.1.0        *               255.255.255.0   U     0      0        0 r1-eth0
10.0.2.0        *               255.255.255.0   U     0      0        0 r1-eth1

*** Starting CLI:
mininet> h1 ping h2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=63 time=0.026 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=63 time=0.028 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=63 time=0.027 ms
64 bytes from 10.0.2.2: icmp_seq=4 ttl=63 time=0.042 ms
64 bytes from 10.0.2.2: icmp_seq=5 ttl=63 time=0.027 ms
64 bytes from 10.0.2.2: icmp_seq=6 ttl=63 time=0.023 ms
64 bytes from 10.0.2.2: icmp_seq=7 ttl=63 time=0.027 ms
64 bytes from 10.0.2.2: icmp_seq=8 ttl=63 time=0.027 ms
^C
--- 10.0.2.2 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7000ms
rtt min/avg/max/mdev = 0.023/0.028/0.042/0.007 ms
mininet> h2 ping h1
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=63 time=0.016 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=63 time=0.024 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=63 time=0.024 ms
64 bytes from 10.0.1.2: icmp_seq=4 ttl=63 time=0.027 ms
64 bytes from 10.0.1.2: icmp_seq=5 ttl=63 time=0.027 ms
64 bytes from 10.0.1.2: icmp_seq=6 ttl=63 time=0.026 ms
64 bytes from 10.0.1.2: icmp_seq=7 ttl=63 time=0.026 ms
64 bytes from 10.0.1.2: icmp_seq=8 ttl=63 time=0.027 ms
64 bytes from 10.0.1.2: icmp_seq=9 ttl=63 time=0.026 ms
^C
--- 10.0.1.2 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 7996ms
rtt min/avg/max/mdev = 0.016/0.024/0.027/0.007 ms
mininet> 
```

2. (25 points) Your modified legacy_router.py program as a separate test file.
3. (20 points) Output of ‘ git log ’ command, showing changes you made to the program above.
   a. 

```
commit dd31499686fe8fde7ed81bb913a6c04154428bc5
Author: elimanzo <None>
Date:   Sun Nov 25 17:00:27 2018 -0800

    debugging legacy code

commit 40627cff8e4907101c645774ead3cfa09b9afb51
Author: elimanzo <None>
Date:   Fri Nov 23 17:37:09 2018 -0800

    Initial commit 2

commit 1f05e92025095e525840417c633fded3f466e33b
Author: elimanzo <None>
Date:   Fri Nov 23 17:36:30 2018 -0800

    Initial commit
(END)
```

4. (30 points) Answers to these questions:
   a. Why didn’t the original program forward packets between the hosts?
      i The original program did not work because there was no default route set up for the connections to be made between the router and the hosts.
   b. Is the line ‘ r1.cmd('sysctl -w net.ipv4.ip_forward=1') ’ required?
      i It is required because that line of code enables port forwarding and without it, there is a 100% packet lost.
      ii 

```
*** Routing Table on Router:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Us
10.0.1.0        *               255.255.255.0   U     0      0
10.0.2.0        *               255.255.255.0   U     0      0

*** Starting CLI:
mininet> h1 ping h2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
^C
--- 10.0.2.2 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6047ms
```

   c. Intentionally break your working program, e.g.: change a subnet length, IP address, or default route for a host. Explain why your change caused the network to break.
      i By changing the default route for a host we end up breaking a connection in that link. Resulting in 100% packet loss for that connection.