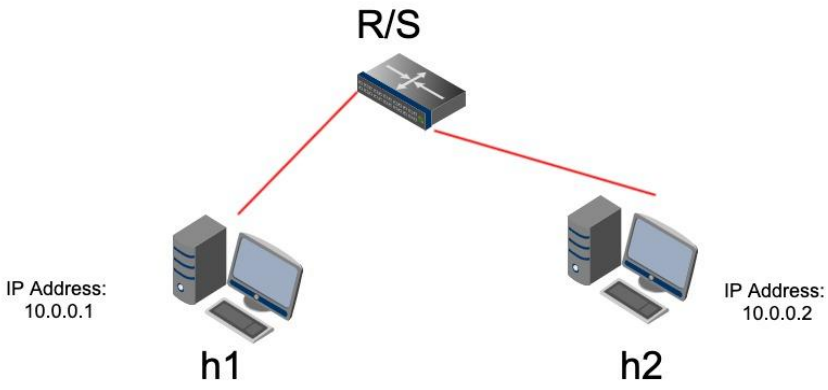
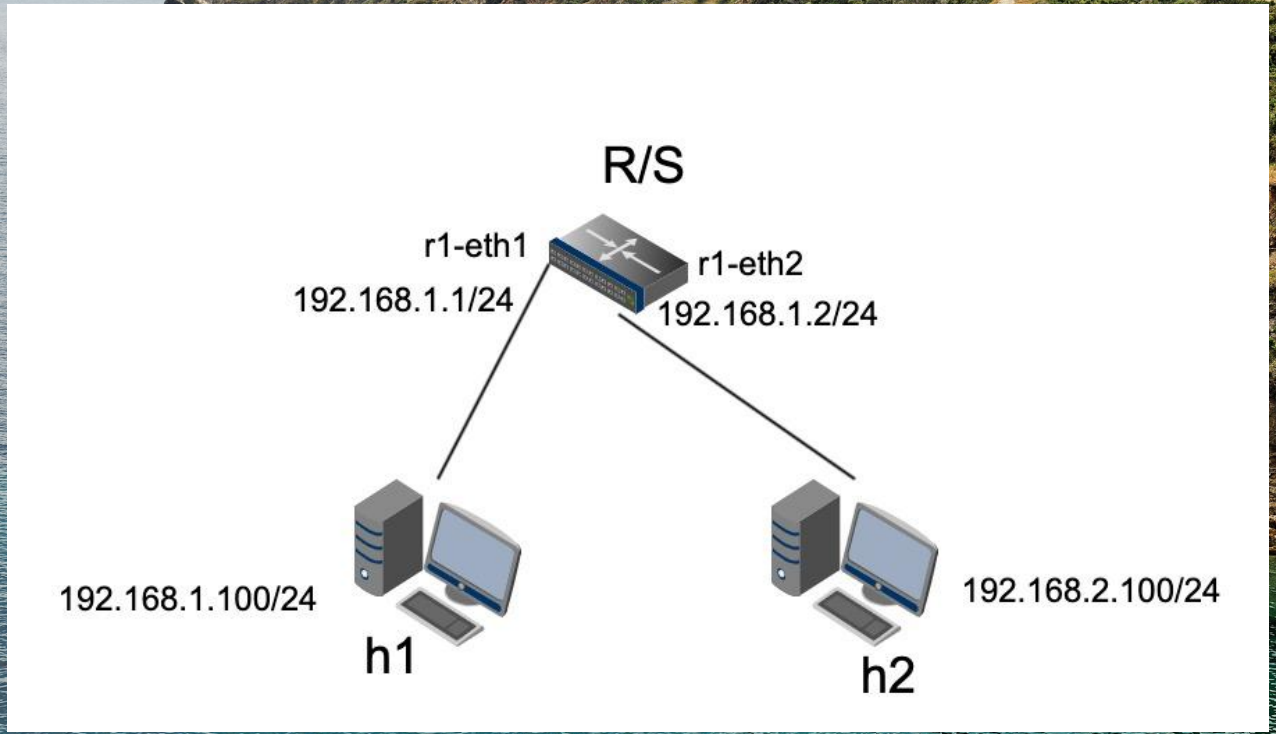

Helping Otters
PA4 Diagrams

Network design of the script given in this document and show in the diagram - what is missing/incorrect?

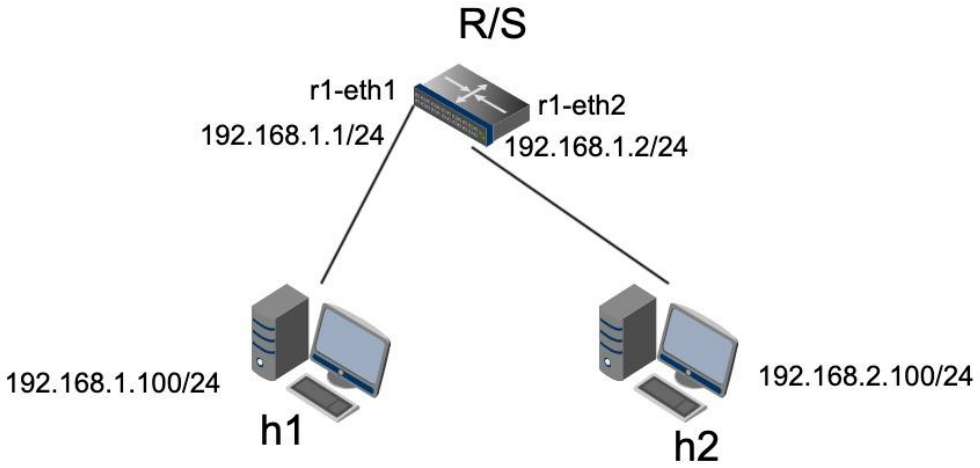


- The links to router are broken, because no default route is specified.
- Host network subnet masks are not specified.
- Router link Interfaces are not defined.
- The IP base for the Mininet topology is defined as 10.0.0.0 / 8 but it should be 0.0.0.0, because the topology is supposed to start empty.

Correct Network Design which allows h1 to ping h2 and for h2 to be able to ping h1.

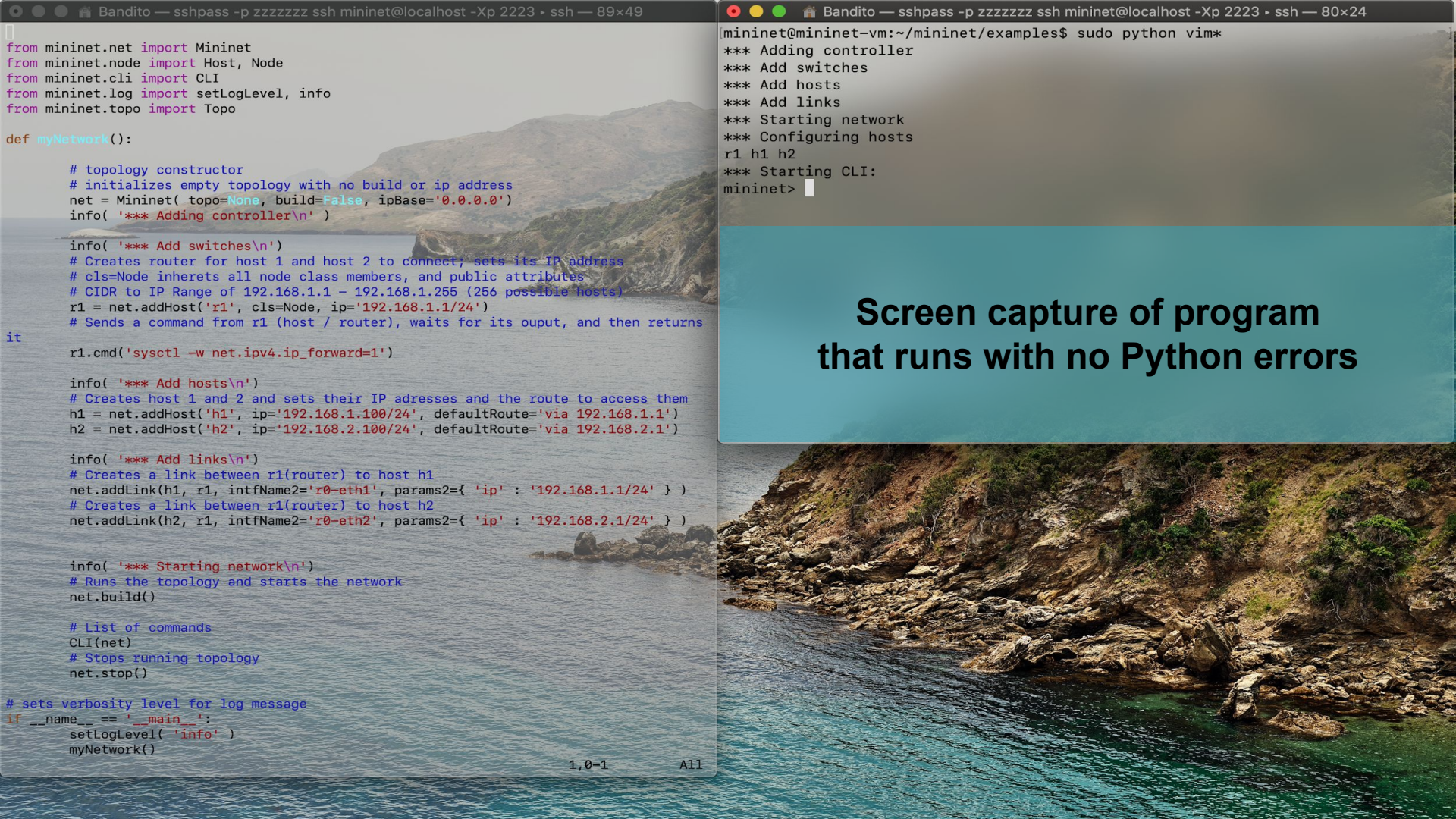


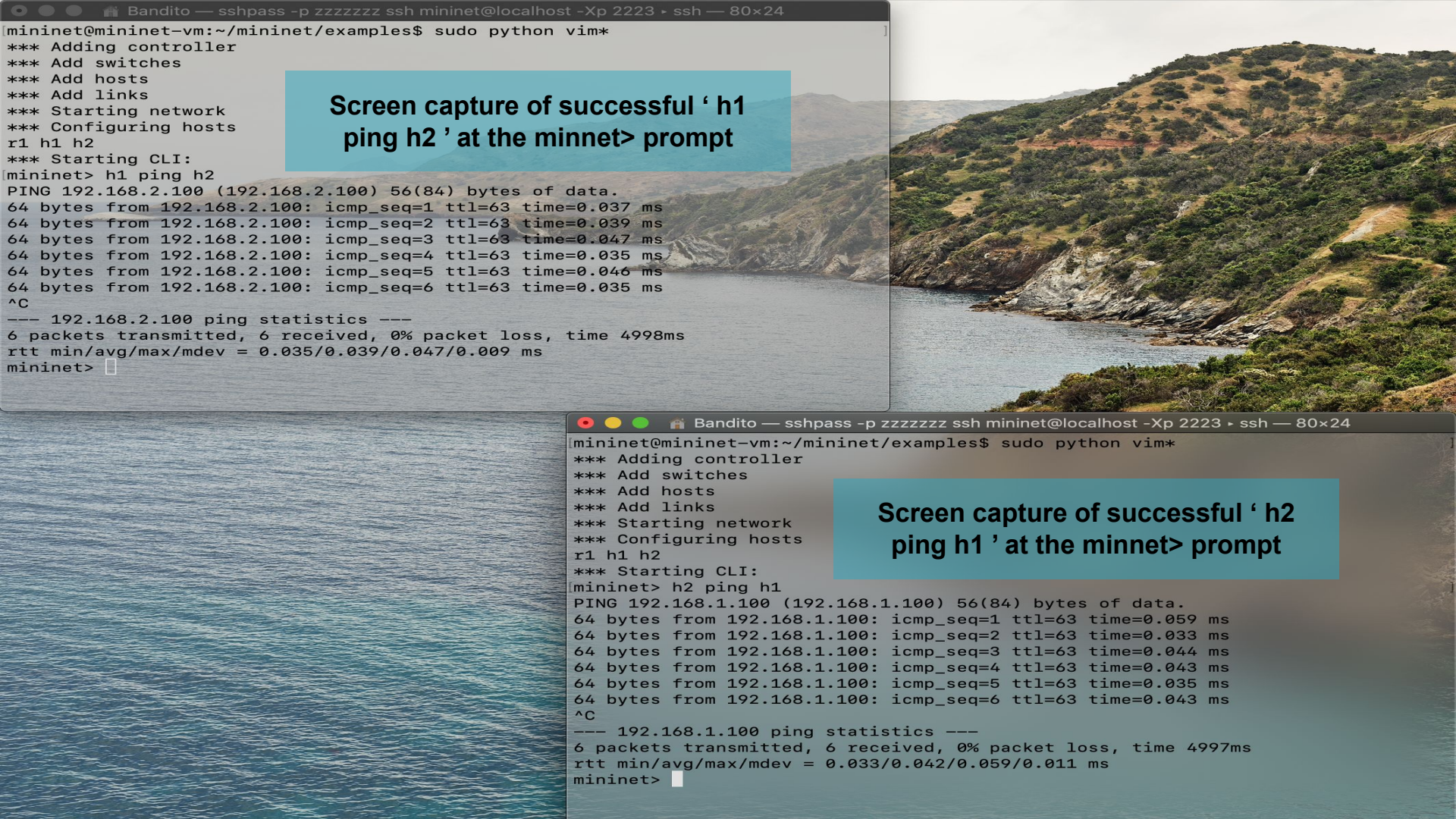
In a separate network diagram, show what lines were changed and why.



- The links were updated to connect hosts to the router via the correct default route
- The IP addresses of all interfaces were updated
- Added subnet masks to all interfaces


Helping Otters
PA4 Screenshots


Bandito — sshpass -p zzzzzzz ssh mininet@localhost -Xp 2223 ▸ ssh — 89×49
from mininet.net import Mininet
from mininet.node import Host, Node
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.topo import Topo
def myNetwork():
# topology constructor
# initializes empty topology with no build or ip address
net = Mininet(topo=None, build=False, ipBase='0.0.0.0')
info('*** Adding controller\n')
info('*** Add switches\n')
# Creates router for host 1 and host 2 to connect; sets its IP address
# cls=Node inherets all node class members, and public attributes
# CIDR to IP Range of 192.168.1.1 - 192.168.1.255 (256 possible hosts)
r1 = net.addHost('r1', cls=Node, ip='192.168.1.1/24')
# Sends a command from r1 (host / router), waits for its ouput, and then returns
it
r1.cmd('sysctl -w net.ipv4.ip_forward=1')
info('*** Add hosts\n')
# Creates host 1 and 2 and sets their IP adresses and the route to access them
h1 = net.addHost('h1', ip='192.168.1.100/24', defaultRoute='via 192.168.1.1')
h2 = net.addHost('h2', ip='192.168.2.100/24', defaultRoute='via 192.168.2.1')
info('*** Add links\n')
# Creates a link between r1(router) to host h1
net.addLink(h1, r1, intfName2='r0-eth1', params2={ 'ip' : '192.168.1.1/24' })
# Creates a link between r1(router) to host h2
net.addLink(h2, r1, intfName2='r0-eth2', params2={ 'ip' : '192.168.2.1/24' })
info('*** Starting network\n')
# Runs the topology and starts the network
net.build()
# List of commands
CLI(net)
# Stops running topology
net.stop()
# sets verbosity level for log message
if __name__ == '__main__':
setLogLevel('info')
myNetwork()
1,0-1 All
Bandito — sshpass -p zzzzzzz ssh mininet@localhost -Xp 2223 ▸ ssh — 80×24
mininet@mininet-vm:~/mininet/examples$ sudo python vim*
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
r1 h1 h2
*** Starting CLI:
mininet>
Screen capture of program
that runs with no Python errors


Bandito — sshpass -p zzzzzzz ssh mininet@localhost -Xp 2223 ▸ ssh — 80×24
mininet@mininet-vm:~/mininet/examples$ sudo python vim*
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
r1 h1 h2
*** Starting CLI:
mininet> h1 ping h2
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data.
64 bytes from 192.168.2.100: icmp_seq=1 ttl=63 time=0.037 ms
64 bytes from 192.168.2.100: icmp_seq=2 ttl=63 time=0.039 ms
64 bytes from 192.168.2.100: icmp_seq=3 ttl=63 time=0.047 ms
64 bytes from 192.168.2.100: icmp_seq=4 ttl=63 time=0.035 ms
64 bytes from 192.168.2.100: icmp_seq=5 ttl=63 time=0.046 ms
64 bytes from 192.168.2.100: icmp_seq=6 ttl=63 time=0.035 ms
^C
--- 192.168.2.100 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4998ms
rtt min/avg/max/mdev = 0.035/0.039/0.047/0.009 ms
mininet>
Screen capture of successful ‘ h1 ping h2 ’ at the minnet> prompt
Bandito — sshpass -p zzzzzzz ssh mininet@localhost -Xp 2223 ▸ ssh — 80×24
mininet@mininet-vm:~/mininet/examples$ sudo python vim*
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
r1 h1 h2
*** Starting CLI:
mininet> h2 ping h1
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_seq=1 ttl=63 time=0.059 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=63 time=0.033 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=63 time=0.044 ms
64 bytes from 192.168.1.100: icmp_seq=4 ttl=63 time=0.043 ms
64 bytes from 192.168.1.100: icmp_seq=5 ttl=63 time=0.035 ms
64 bytes from 192.168.1.100: icmp_seq=6 ttl=63 time=0.043 ms
^C
--- 192.168.1.100 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4997ms
rtt min/avg/max/mdev = 0.033/0.042/0.059/0.011 ms
mininet>
Screen capture of successful ‘ h2 ping h1 ’ at the minnet> prompt

Barbosa, Halbert, Reynolds, Sedano
CST 311 PA4 Questions
June 16, 2020

1. What were any interesting findings and lessons learned?
    - When the Mininet topology is initialized, ipBase is set to 10.0.0.0/8 when it should be set to 0.0.0.0 to indicate a blank topology.
    - The router is initialized with an IP address of 0.0.0.0, but it should be hard coded with an actual IP address and subnet mask.
    - The defaultRoute needs to be specified when adding a host and it needs to be in the same subnet as the interface it is attached to on the router.
    - The hosts IP addresses were missing subnet masks.
    - The router did not have any interfaces specified, so the hosts could not connect to it.
    - When links are added between the hosts and router, the corresponding router interface needs to be specified. The router interface must also be named and passed in as an argument to the addLink function.
2. Why didn't the original program forward packets between the hosts?
    - Links between the hosts and the router were not fully implemented, which meant there was no way for an ICMP packet to travel from one host to the other host.
    - IP addresses and subnet masks needed to be defined for each host and for each interface of the router.
    - These defined interfaces could then be connected with a link.
3. Is the line ' r1.cmd('sysctl -w net.ipv4.ip_forward=1') ' required?
    - Yes, this line enables IP forwarding on the router/switch. If this line were deleted, the program would break when you try to have h1 ping h2. Packets are sent to the router, but are never forwarded.
4. 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.
    - Change subnet length:
        - What happens - If you arbitrarily change the subnet length on a host, you will receive this error: "Destination Host Unreachable"
        - Why - The host's IP address does not align with the topology of the rest of the network when it's subnet is changed. The host cannot be reached because its IP address is pointing to a host in a different subnet, one with a different length network address.
    - Change IP address:
        - What happens - If the IP address of a host is changed to an address outside of the subnet, you will receive this error: "Network is unreachable"
        - Why - If the host's IP address is changed, then in terms of its naming scheme, it is no longer in the same subnet as the router interface it is connected to. Each individual interface on the router and the host it is

connected to, must be in the same subnet, so their IP addresses will reflect that.

- Change default route:
    - What happens - If you change a hosts default route to an IP address outside of the specified subnet, you will get this error: “Network is unreachable”
    - Why - This causes the same problem as the previous bullet point. The router interface and the host connected to it must be within the same subnet.
        - Note - If you change a host’s default route to a different IP address, but one that is still within the subnet and change the router’s interface as well, this will not break the program.