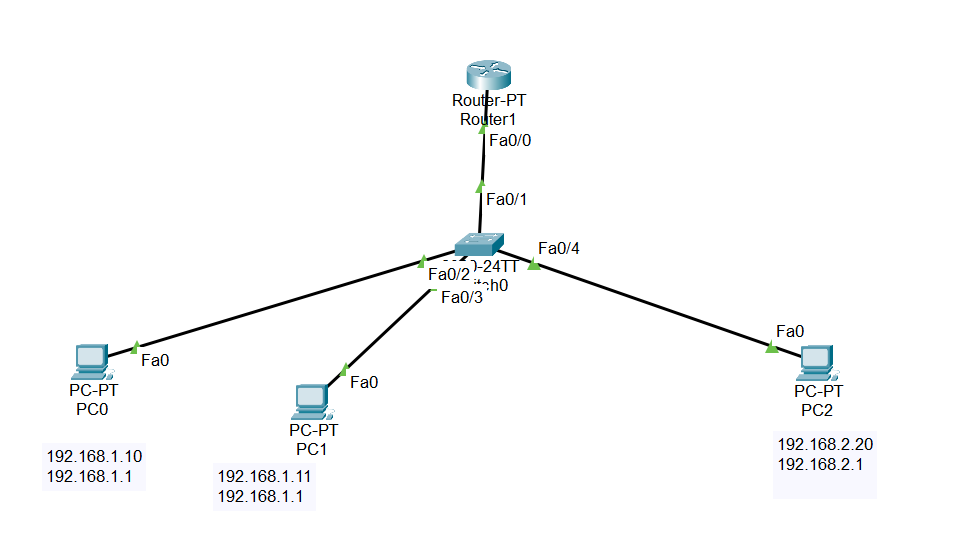
**Index**

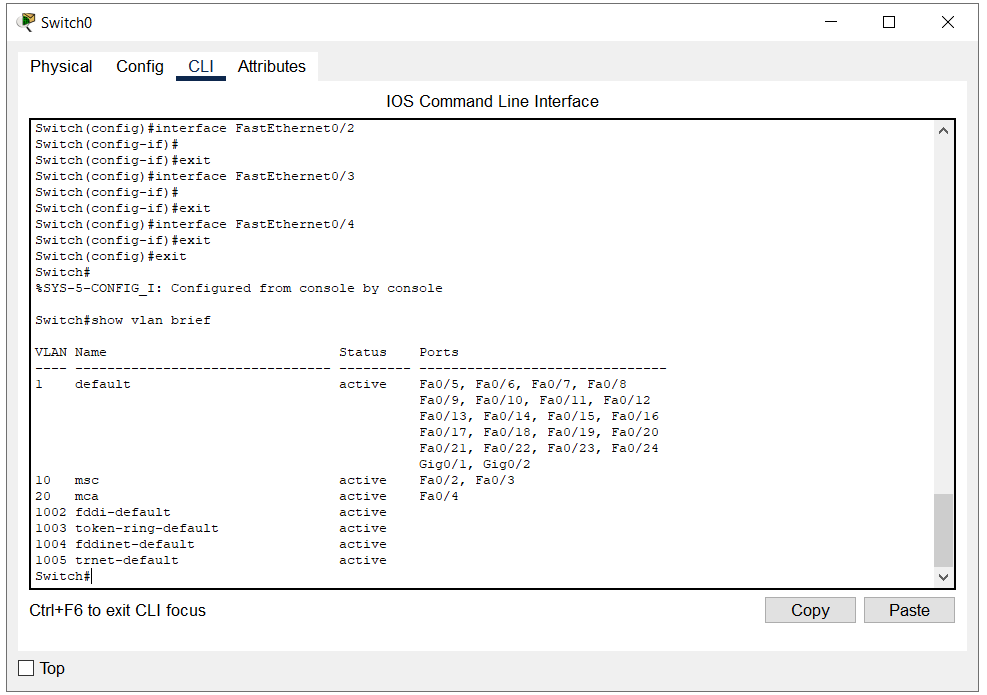
| Sr No | Name | Date | Sign |
| --- | --- | --- | --- |
| 1 | Implementing VLAN routing |  |  |
| 2 | OSPF/BGP/NAT Implementations |  |  |
| 3 | Implementing NAT |  |  |
| 4 | Creating a SDN network using mininet |  |  |
| 5 | To create and save topologies in Mininet |  |  |
| 6 | Implementing Firewall using POX Controller |  |  |
| 7 | Implementing simple router using Mininet |  |  |
| 8 |  |  |  |

Practical No 1

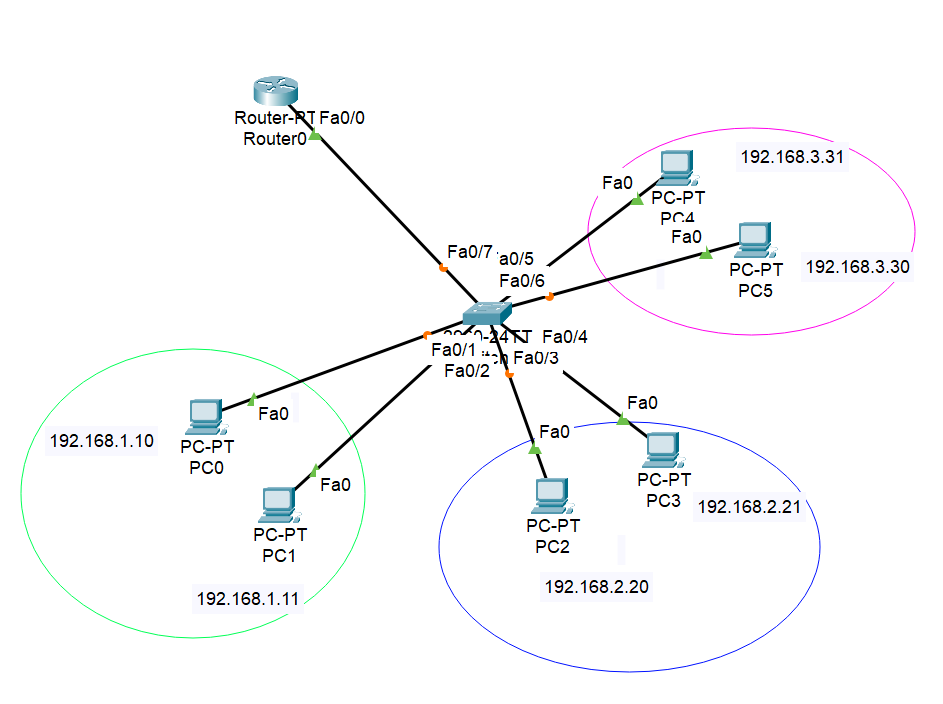
Aim : Implement Inter-VLAN Routing.

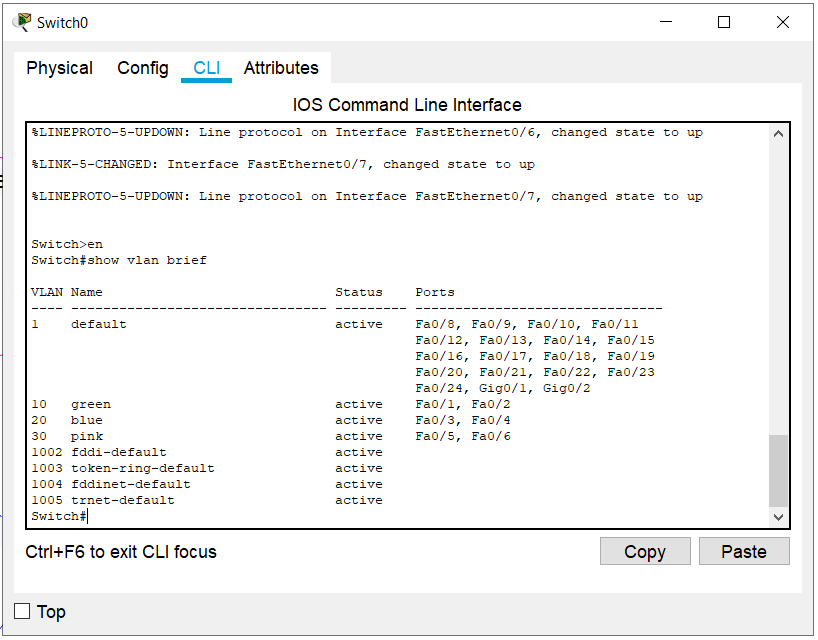
**Two vlans**





**Three vlans**



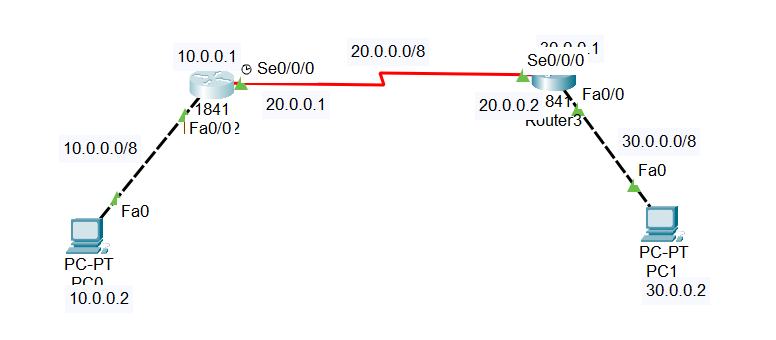


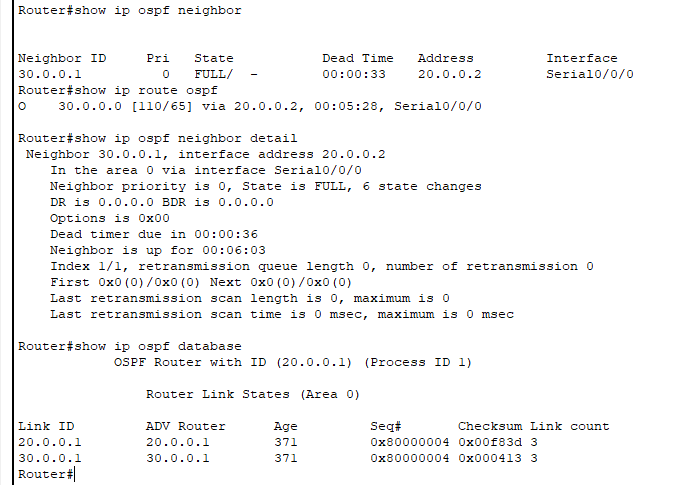
There are 5 main types of VLANs depending on the type of the network they carry:

1. **Default VLAN –**When the switch initially starts up, all switch ports become a member of the default VLAN (generally all switches have default VLAN named as **VLAN 1**), which makes them all part of the same broadcast domain. Using default VLAN allows any network device connected to any of the switch port to connect with other devices on other switch ports. One unique feature of Default VLAN is that it can’t be rename or delete.
2. **Data VLAN –**Data VLAN is used to divide the whole network into 2 groups. One group of users and other group of devices. This VLAN also known as a user VLAN, the data VLAN is used only for user-generated data. This VLAN carrying data only. It is not used for carrying management traffic or voice.
3. **Voice VLAN –**Voice VLAN is configured to carry voice traffic. Voice VLANs are mostly given high transmission priority over other types of network traffic. To ensure voice over IP (VoIP) quality (delay of less than 150 milliseconds (ms) across the network), we must have separate voice VLAN as this will preserve bandwidth for other applications.
4. **Management VLAN –**A management VLAN is configured to access the management capabilities of a switch (traffic like system logging, monitoring). VLAN 1 is the management VLAN by default (VLAN 1 would be a bad choice for the management VLAN). Any of a switch VLAN could be define as the management VLAN if admin as not configured a unique VLAN to serve as the management VLAN. This VLAN ensures that bandwidth for management will be available even when user traffic is high.
5. **Native VLAN –**This VLAN identifies traffic coming from each end of a trunk link. A native VLAN is allocated only to an 802.1Q trunk port. The 802.1Q trunk port places untagged traffic (traffic that does not come from any VLAN) on the native VLAN. It is a best to configure the native VLAN as an unused VLAN.

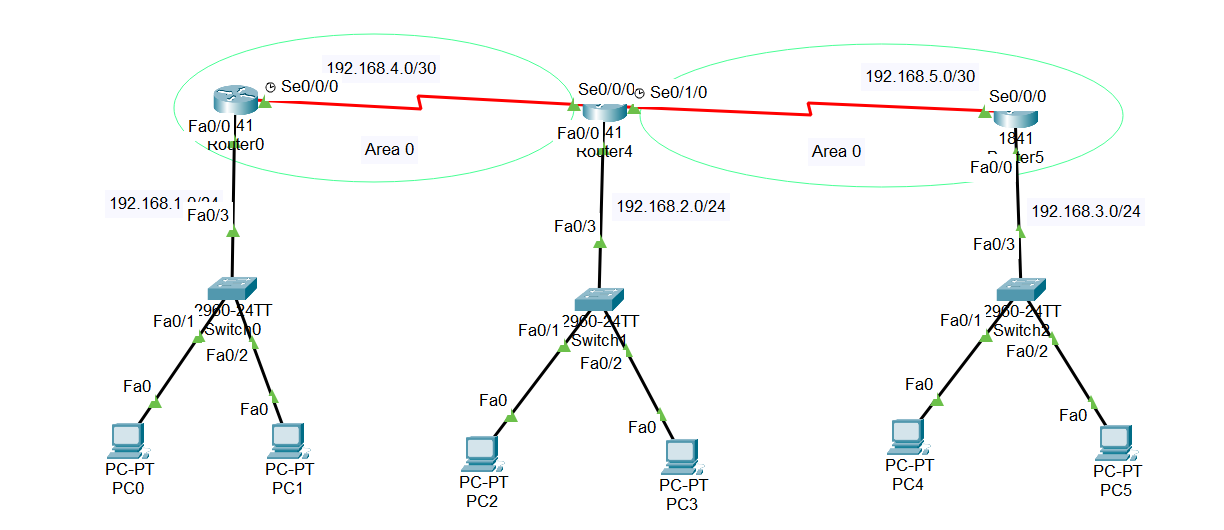
Practical No 2

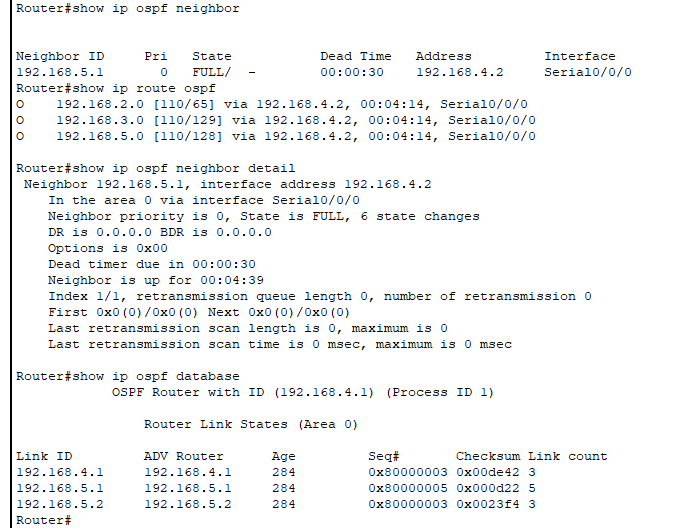
Aim : OSPF/BGP/NAT Implementations



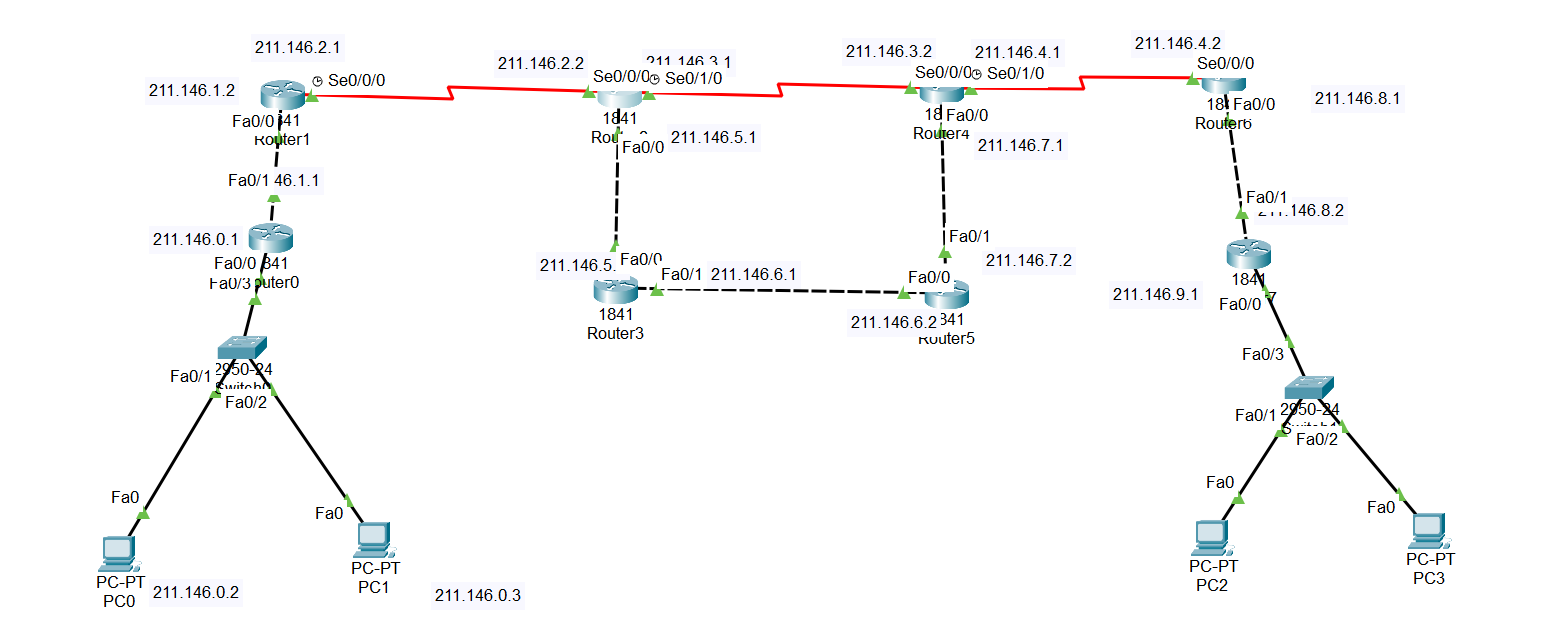


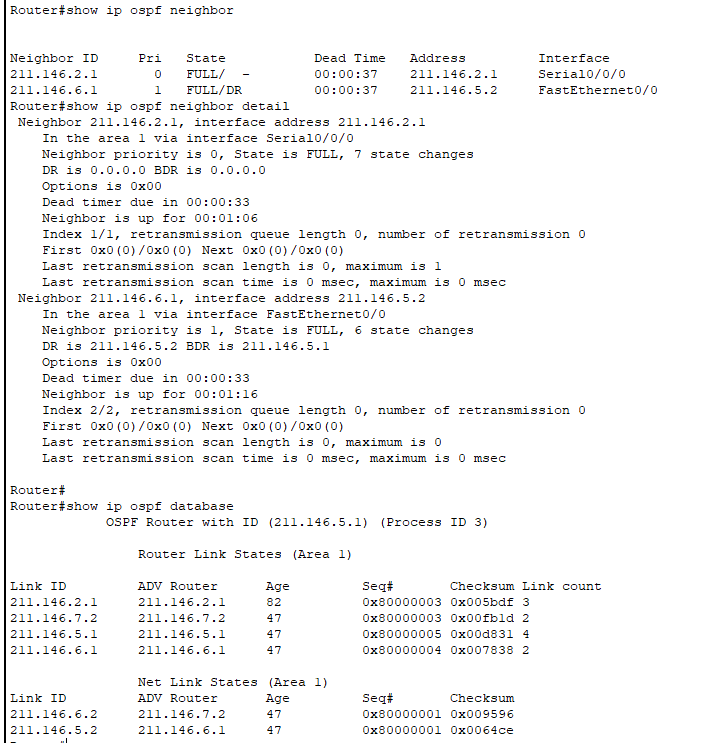
1)



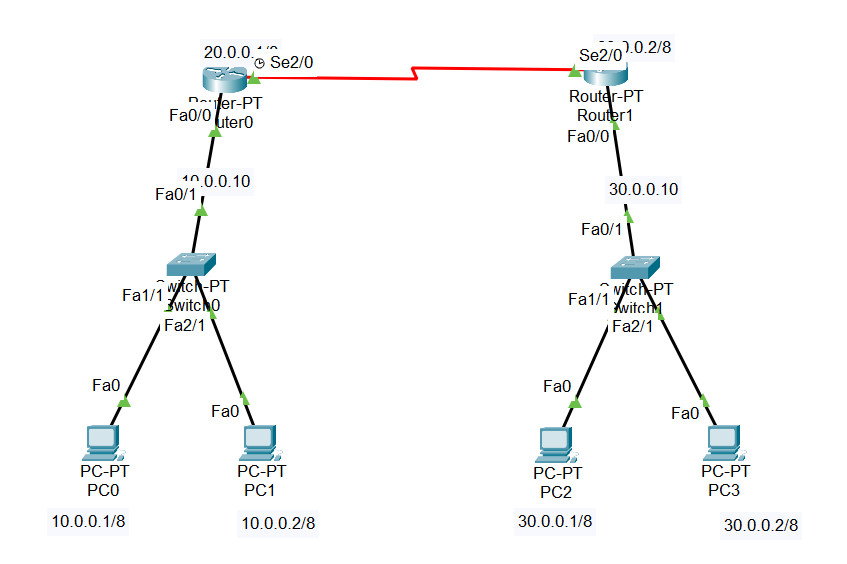


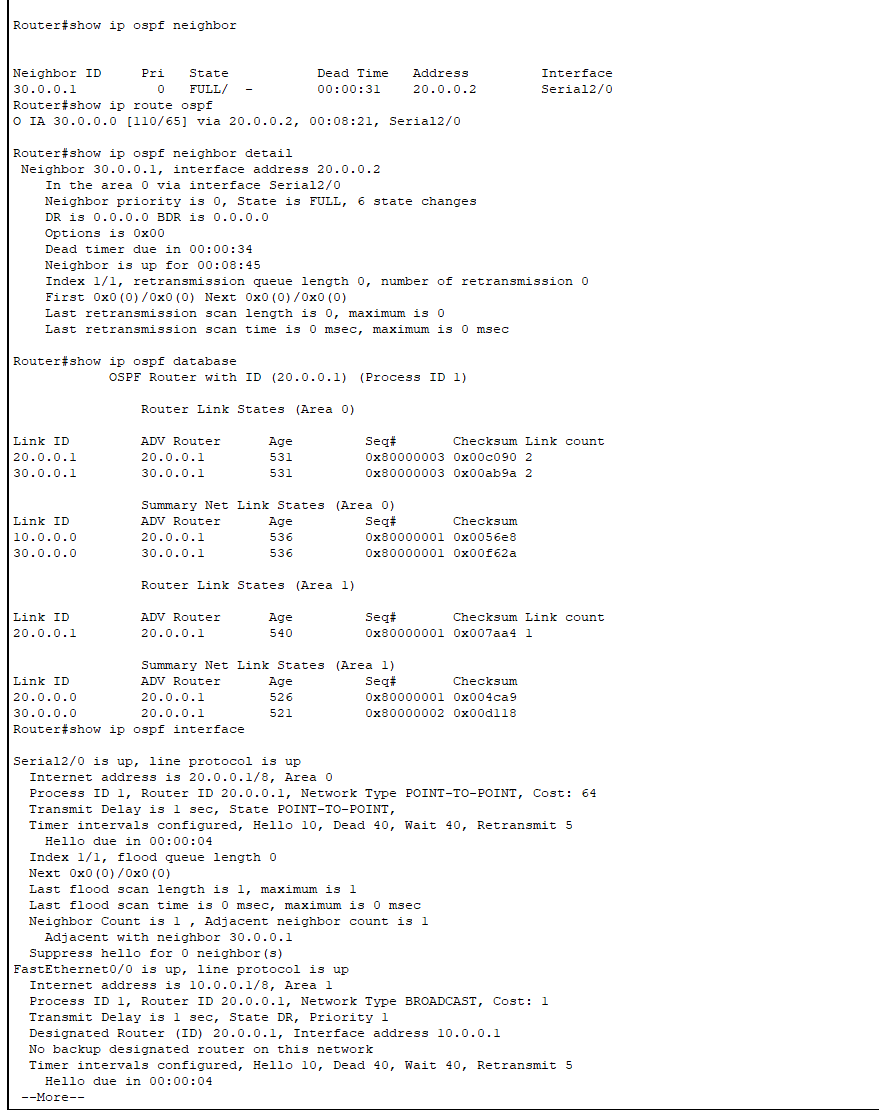
2)





3)





**Routing** is the process of establishing the routes that data packets must follow to reach the destination. In this process, a routing table is created which contains information regarding routes that data packets follow. Various routing algorithms are used for the purpose of deciding which route an incoming data packet needs to be transmitted on to reach the destination efficiently.

**1. Adaptive Algorithms –**

These are the algorithms that change their routing decisions whenever network topology or traffic load changes. The changes in routing decisions are reflected in the topology as well as the traffic of the network. Also known as dynamic routing, these make use of dynamic information such as current topology, load, delay, etc. to select routes. Optimization parameters are distance, number of hops, and estimated transit time.

Further, these are classified as follows:

* **(a) Isolated –** In this method each, node makes its routing decisions using the information it has without seeking information from other nodes. The sending nodes don’t have information about the status of a particular link. The disadvantage is that packets may be sent through a congested network which may result in delay. Examples: Hot potato routing, backward learning.
* **(b) Centralized –** In this method, a centralized node has entire information about the network and makes all the routing decisions. The advantage of this is only one node is required to keep the information of the entire network and the disadvantage is that if the central node goes down the entire network is done. The link state algorithm is referred to as a centralized algorithm since it is aware of the cost of each link in the network
* **(c) Distributed –** In this method, the node receives information from its neighbors and then takes the decision about routing the packets. A disadvantage is that the packet may be delayed if there is a change in between intervals in which it receives information and sends packets. It is also known as a decentralized algorithm as it computes the least-cost path between source and destination

**2. Non-Adaptive Algorithms –**

These are the algorithms that do not change their routing decisions once they have been selected. This is also known as static routing as a route to be taken is computed in advance and downloaded to routers when a router is booted.

Further, these are classified as follows:

* **(a) Flooding –** This adapts the technique in which every incoming packet is sent on every outgoing line except from which it arrived. One problem with this is that packets may go in a loop and as a result of which a node may receive duplicate packets. These problems can be overcome with the help of sequence numbers, hop count, and spanning trees.
* **(b) Random walk –** In this method, packets are sent host by host or node by node to one of its neighbors randomly. This is a highly robust method that is usually implemented by sending packets onto the link which is least queued.

The OSPF (Open Shortest Path First) protocol is one of a family of IP Routing protocols, and is an Interior Gateway Protocol (IGP) for the Internet, used to distribute IP routing information throughout a single Autonomous System (AS) in an IP network.

The OSPF protocol is a link-state routing protocol, which means that the routers exchange topology information with their nearest neighbors. The topology information is flooded throughout the AS, so that every router within the AS has a complete picture of the topology of the AS. This picture is then used to calculate end-to-end paths through the AS, normally using a variant of the Dijkstra algorithm. Therefore, in a link-state routing protocol, the next hop address to which data is forwarded is determined by choosing the best end-to-end path to the eventual destination.

The main advantage of a link state routing protocol like OSPF is that the complete knowledge of topology allows routers to calculate routes that satisfy particular criteria. This can be useful for traffic engineering purposes, where routes can be constrained to meet particular quality of service requirements. The main disadvantage of a link state routing protocol is that it does not scale well as more routers are added to the routing domain. Increasing the number of routers increases the size and frequency of the topology updates, and also the length of time it takes to calculate end-to-end routes. This lack of scalability means that a link state routing protocol is unsuitable for routing across the Internet at large, which is the reason why IGPs only route traffic within a single AS.

Each OSPF router distributes information about its local state (usable interfaces and reachable neighbors, and the cost of using each interface) to other routers using a Link State Advertisement (LSA) message. Each router uses the received messages to build up an identical database that describes the topology of the AS.

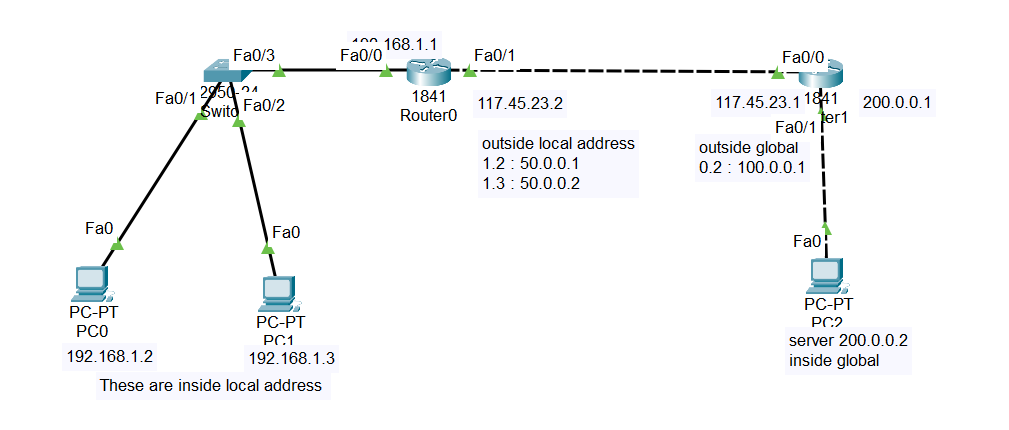
From this database, each router calculates its own routing table using a Shortest Path First (SPF) or Dijkstra algorithm. This routing table contains all the destinations the routing protocol knows about, associated with a next hop IP address and outgoing interface.

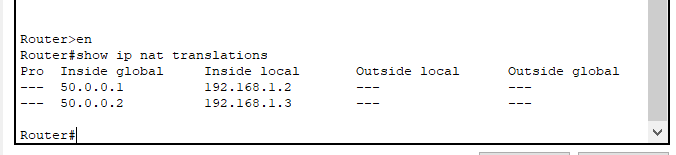
* The protocol recalculates routes when network topology changes, using the Dijkstra algorithm, and minimises the routing protocol traffic that it generates.
* It provides support for multiple paths of equal cost.
* It provides a multi-level hierarchy (two-level for OSPF) called "area routing," so that information about the topology within a defined area of the AS is hidden from routers outside this area. This enables an additional level of routing protection and a reduction in routing protocol traffic.
* All protocol exchanges can be authenticated so that only trusted routers can join in the routing exchanges for the AS.

Practical No 3

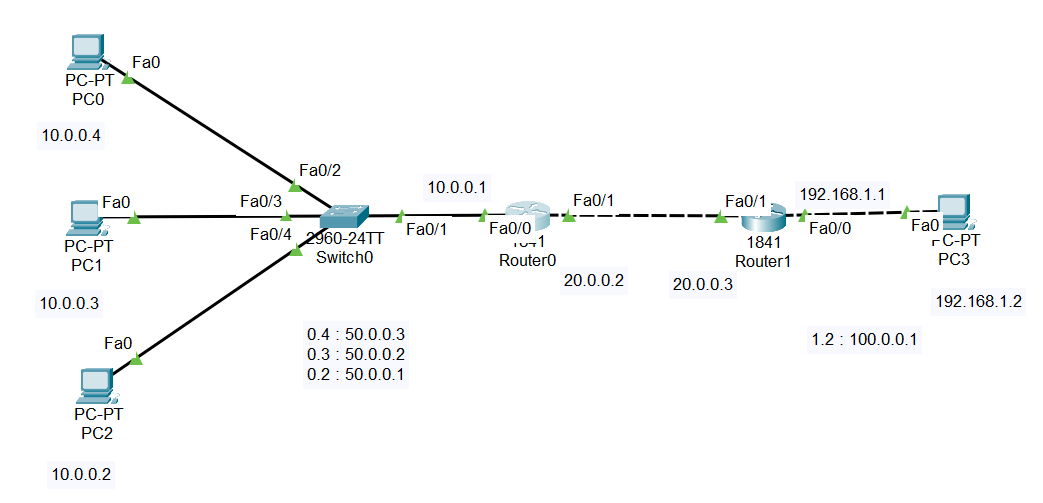
Aim : Implementing NAT

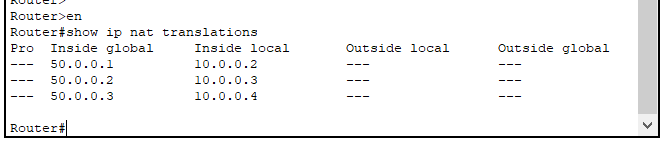
Q1)



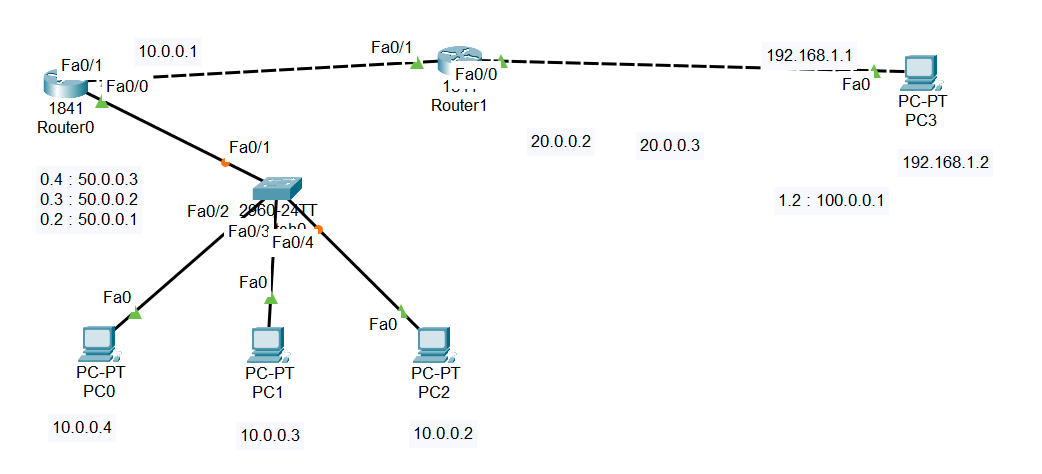


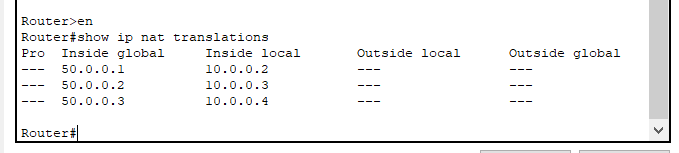
Q2)





Q3)





**Network Address Translation (NAT)** is used to convert a private IP address into a public IP address and also a public IP address into private IP address. We use NAT due to the shortage of IP addresses.

IP Version 4 is a 32- bit address and it has almost 4.3 billion IP addresses, but the population of the world is much higher and it is approximately 7.8 billion as of the latest reports of June 2020.

Most of them use multiple gadgets and devices like smartphones, laptops, tablets, and many more for accessing the internet for various needs. All these devices need an IP address.

So, 4.3 billion of IP addresses is not sufficient for more than 7.8 billion of people using , that is the reason why we are using NAT. NAT is used to convert a private IP address onto public and public IP address into private.

## **Types of NAT**

The types of Network Address Translation (NAT) are explained below −

### **Static NAT**

It is otherwise called balanced NAT. In this sort of NAT, just the IP addresses and the header checksum are changed among the general organization addresses.

These are actualized for interconnection of two particular IP networks having contrary tending to.

STATIC NAT is ONE to ONE NAT. This means in the local area network side and outside network side we need an equal number of IP addresses to translate. Due to that reason this is a bit expensive because we need more Public IP addresses.

Keep in mind Public IP addresses are always expensive. In this example 192.168.4.1 always translates to 225.20.120.11.

### **Dynamic NAT**

In this kind of NAT, planning of IP from an unregistered private organization is finished with the single IP address of the enrolled network from the class of enlisted IP addresses.

Dynamic NAT means we have a Pool of IP addresses in the outside list. So which IP addresses are Free in the outside list which will associate to LAN side IP first.

This example 192.168.4.1 from Local will check which Public IP address is from the pool.

If First IP free then it translates to First IP. If the first IP is already associated with any other local side IP, it will check the next available IP.

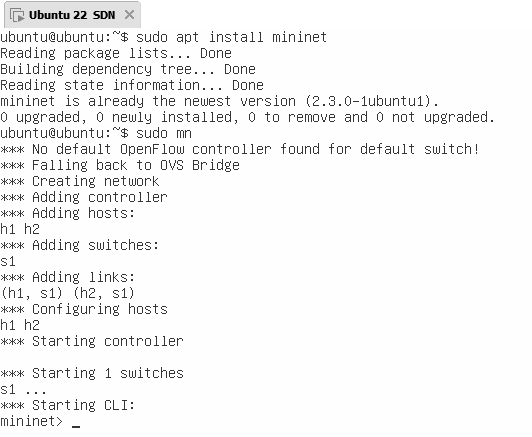
Suppose there is no IP available from the Public IP pool for translating Private IP addresses. Then, that traffic will drop by Router or Firewall.

Practical no 4

Aim : Creating SDN networks using Mininet

Q1 ) Install and create the environment to execute the following

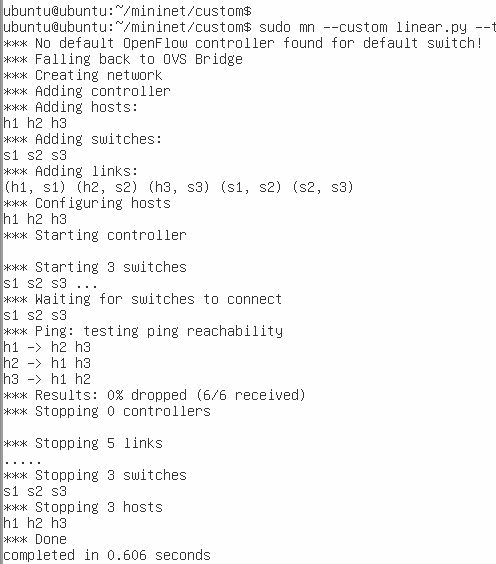
1. Display node



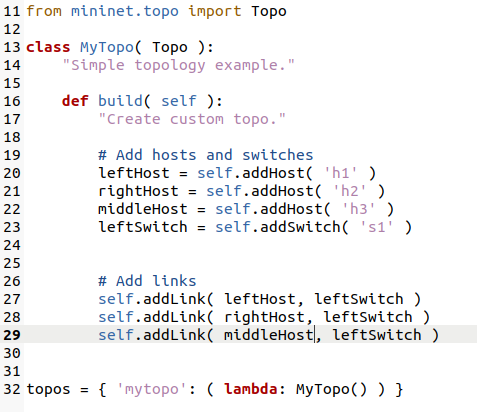
1. Display topology (all 3 type-simple linear and tree)

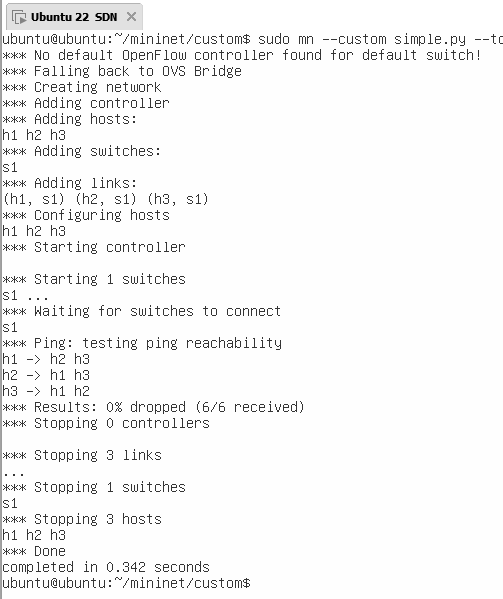
**Linear**





**Simple**



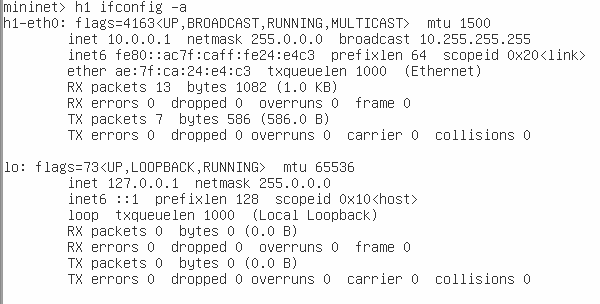


**Tree**

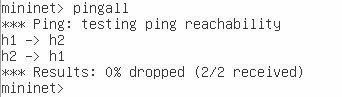




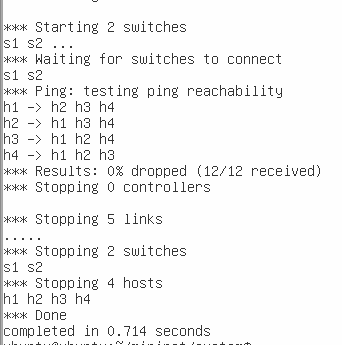
1. Show the network interface



1. Test the connectivity

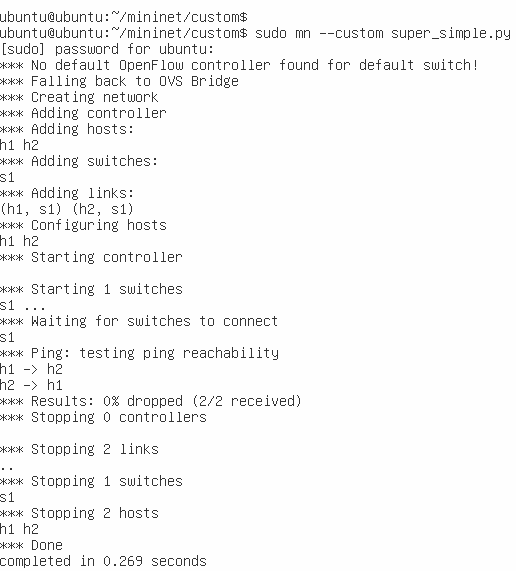


1. Execute the default custom topology

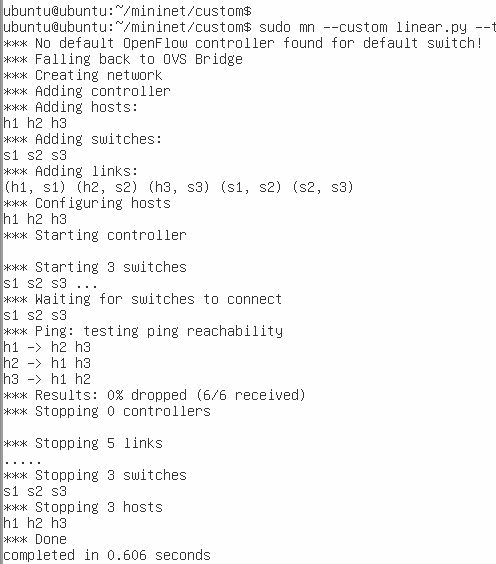


Q2) Implement the following custom topologies using mininet.

1. Simple network with 2 hosts and 1 switch.



1. Design a simple linear network with 3 switches and 3 hosts.



1. Design the following



SDN

Software-Defined Networking (SDN) is an approach to networking that uses software-based controllers or application programming interfaces (APIs) to communicate with underlying hardware infrastructure and direct traffic on a network.

This model differs from that of traditional networks, which use dedicated hardware devices (i.e., routers and switches) to control network traffic. SDN can create and control a virtual network – or control a traditional hardware – via software.

Why SDN

Typically, networks have been defined and governed by their tangible properties—the hardware and wiring that make the physical connections between points. Software-Defined Networking (SDN)—a burgeoning network architecture—is turning that approach on its head. SDN separates the control of the network from the hardware. It uses software applications to program your network intelligently through centralized control. This means the underlying hardware and associated technologies are still there, but they are programmed centrally. As a result, you can consistently and holistically manage your entire network with utmost flexibility and speed.

Mininet is a software emulator for [prototyping](https://www.techtarget.com/searcherp/definition/prototype) a large network on a single machine.

Mininet can be used to quickly create a realistic virtual network running actual [kernel](https://www.techtarget.com/searchdatacenter/definition/kernel), [switch](https://www.techtarget.com/searchnetworking/definition/switch) and software application [code](https://www.techtarget.com/whatis/definition/code) on a personal computer. Mininet allows the user to quickly create,

interact with, customize and share a [software-defined network](https://www.techtarget.com/searchnetworking/definition/software-defined-networking-SDN) (SDN) prototype to simulate a [network topology](https://www.techtarget.com/searchnetworking/definition/network-topology) that uses [Openﬂow](https://www.techtarget.com/whatis/definition/OpenFlow) switches.

Commands:

* Display Mininet CLI commands:
  + mininet> help
* Display nodes:
  + mininet> nodes
* Display links:
  + mininet> net
* Dump information about all nodes:
  + mininet> dump
* Show network information
  + h1 ifconfig -a

Installation:

sudo mn -h

git clone https://github.com/mininet/mininet

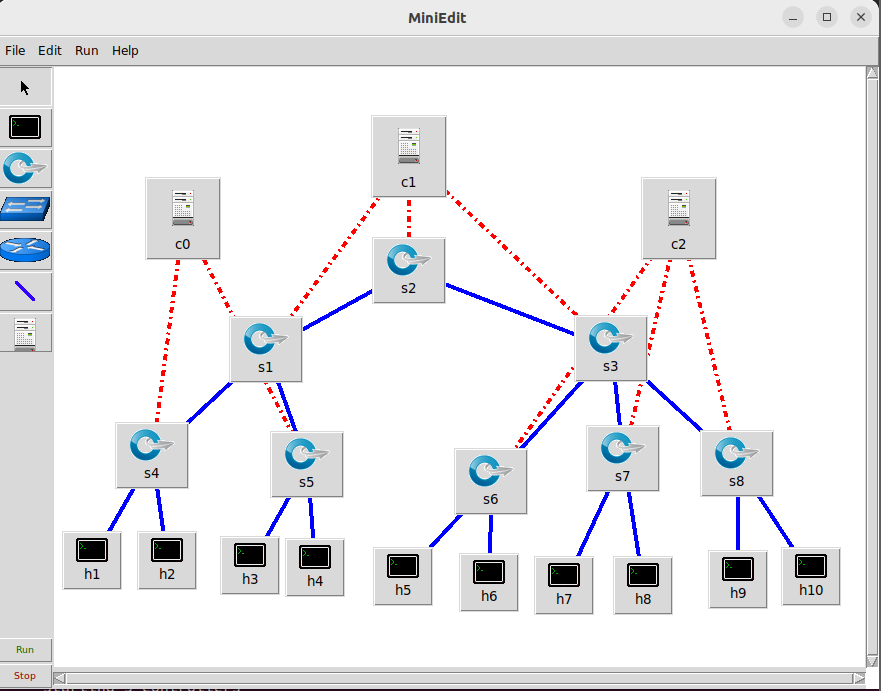
Three topologies :

1. Simple
2. Linear
3. Tree

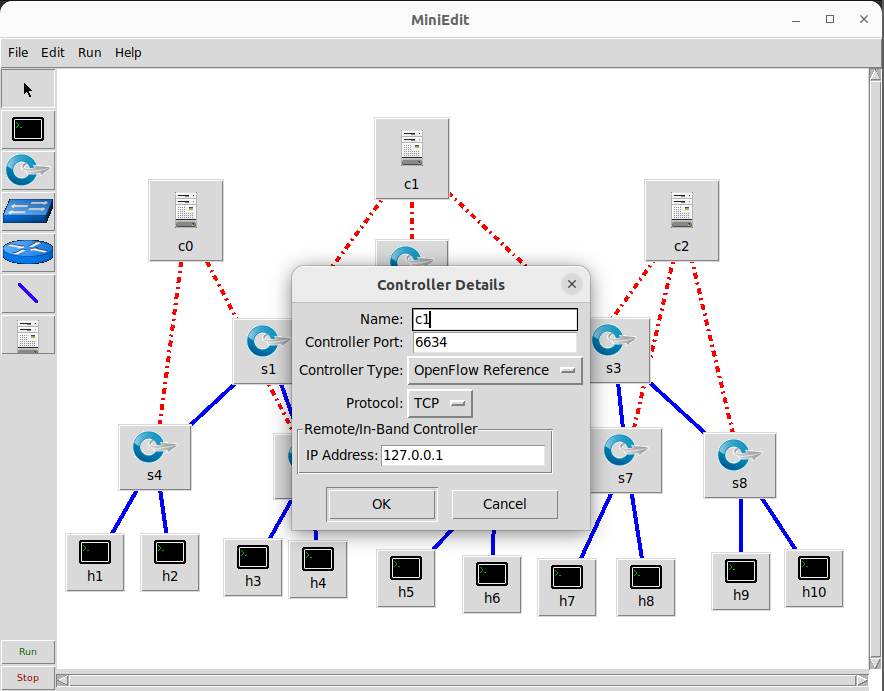
Practical No 5

Aim : To create and save topologies in Miniedit

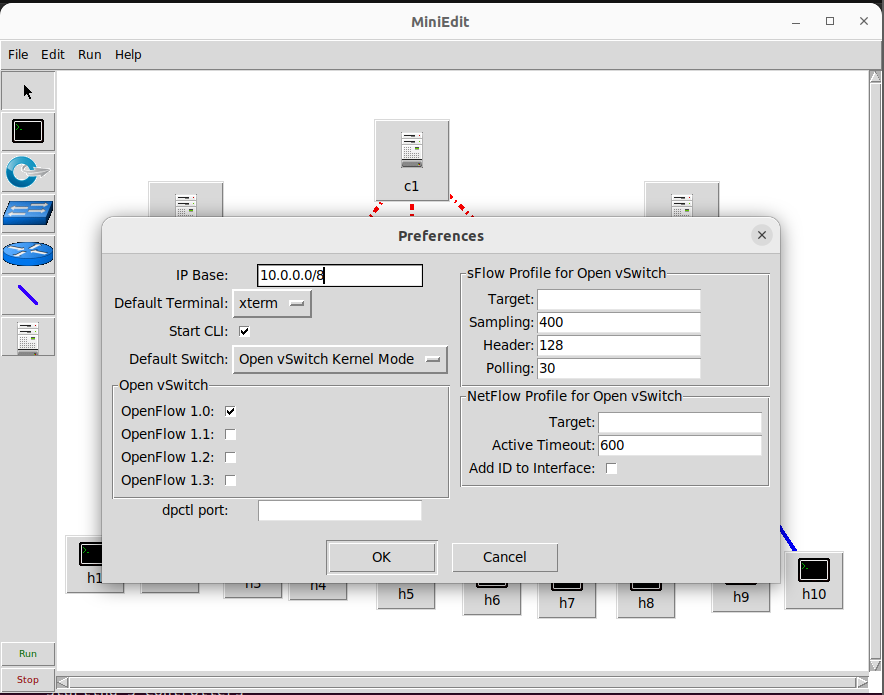
Step 1 : Open miniedit GUI with sudo privileges, and create the following topologies



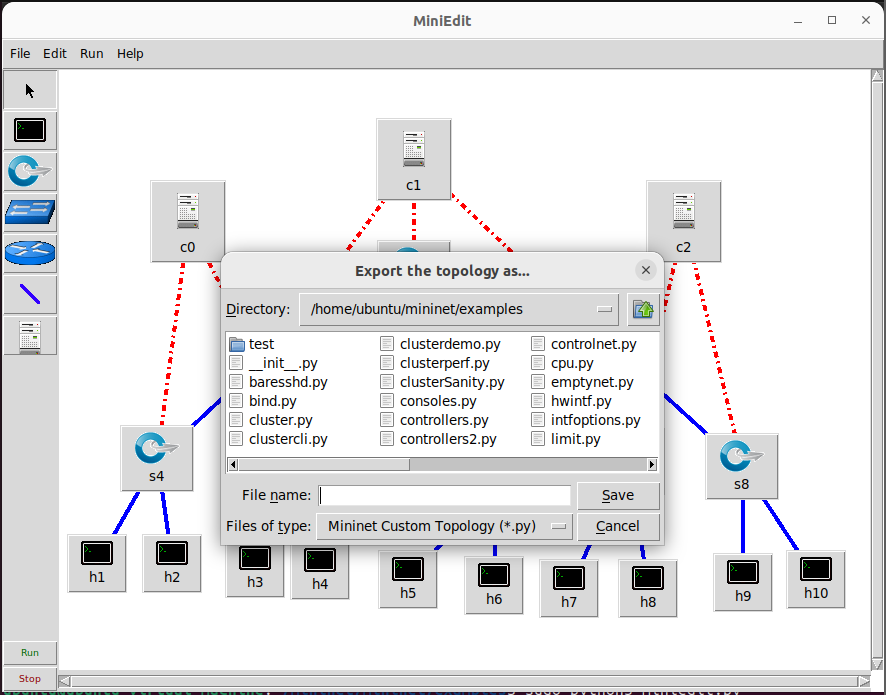
Step 2 : Configure the following controllers placed in the topology



Step 3 : Under Edit > Preferences select the checkbox start cli



Step 4 : Save the following configuration under File as .mn file



**Write Up:**

Mininet is a virtual testbed enabling the development and testing of network tools and protocols. With a single command, Mininet can create a realistic virtual network on any type of machine (Virtual Machine (VM), cloud-hosted, or native). Therefore, it provides an inexpensive solution and streamlined development running in line with production networks

Mininet offers the following features:

• Fast prototyping for new networking protocols.

• Simplified testing for complex topologies without the need of buying expensive hardware.

• Realistic execution as it runs real code on the Unix and Linux kernels.

• Open source environment backed by a large community contributing extensive documentation.

Mininet is useful for development, teaching, and research as it is easy to customize and interact with it through the CLI or the GUI. Mininet was originally designed to experiment with OpenFlow and Software-Defined Networking (SDN). This lab, however, only focuses on emulating a simple network environment without SDN-based devices.

Mininet’s logical nodes can be connected into networks. These nodes are sometimes called containers, or more accurately, network namespaces. Containers consume sufficiently fewer resources that networks of over a thousand nodes have created, running on a single laptop. A Mininet container is a process (or group of processes) that no longer has access to all the host system’s native network interfaces. Containers are then assigned virtual Ethernet interfaces, which are connected to other containers through a virtual switch4. Mininet connects a host and a switch using a virtual Ethernet (veth) link. The veth link is analogous to a wire connecting two virtual interfaces, as illustrated below.

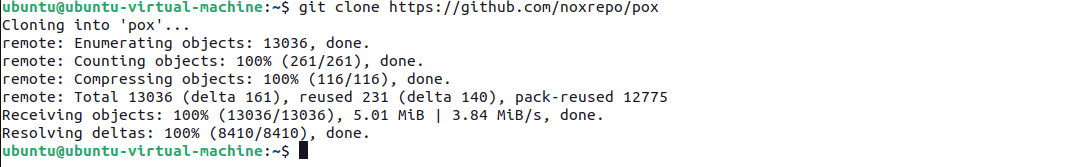
MiniEdit is an experimental tool created to demonstrate how Mininet can be extended. To show how to use MiniEdit to create and run network simulations, we will work through a tutorial that demonstrates how to use MiniEdit to build a network, configure network elements, save the topology, and run the simulation.

Practical No 6

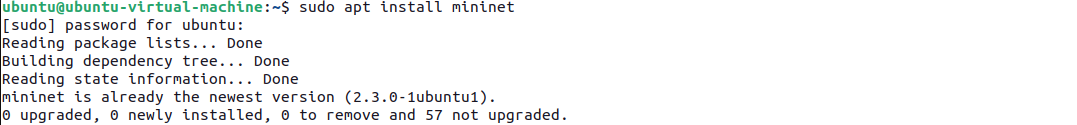
Aim : Implementing Firewall using pox controller

Steps :

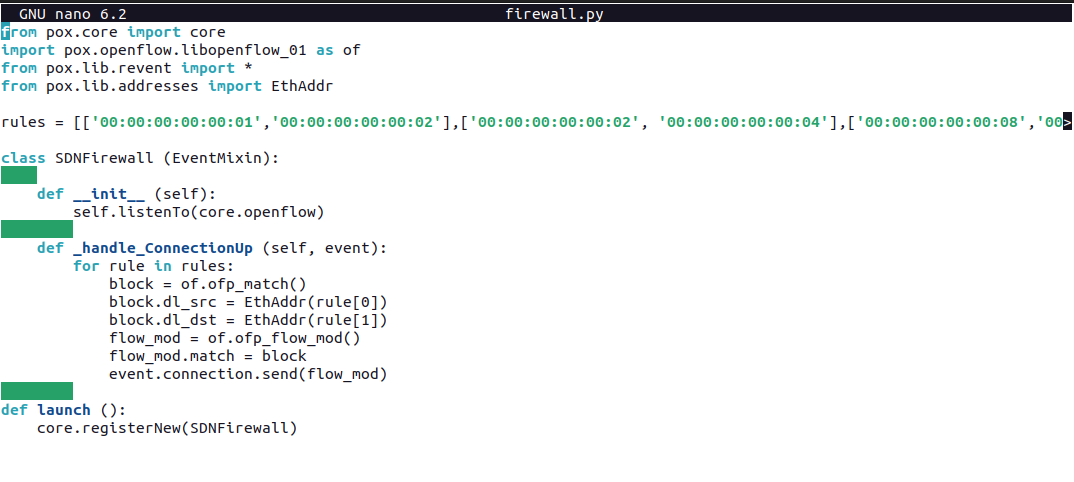
**Clone the repo**



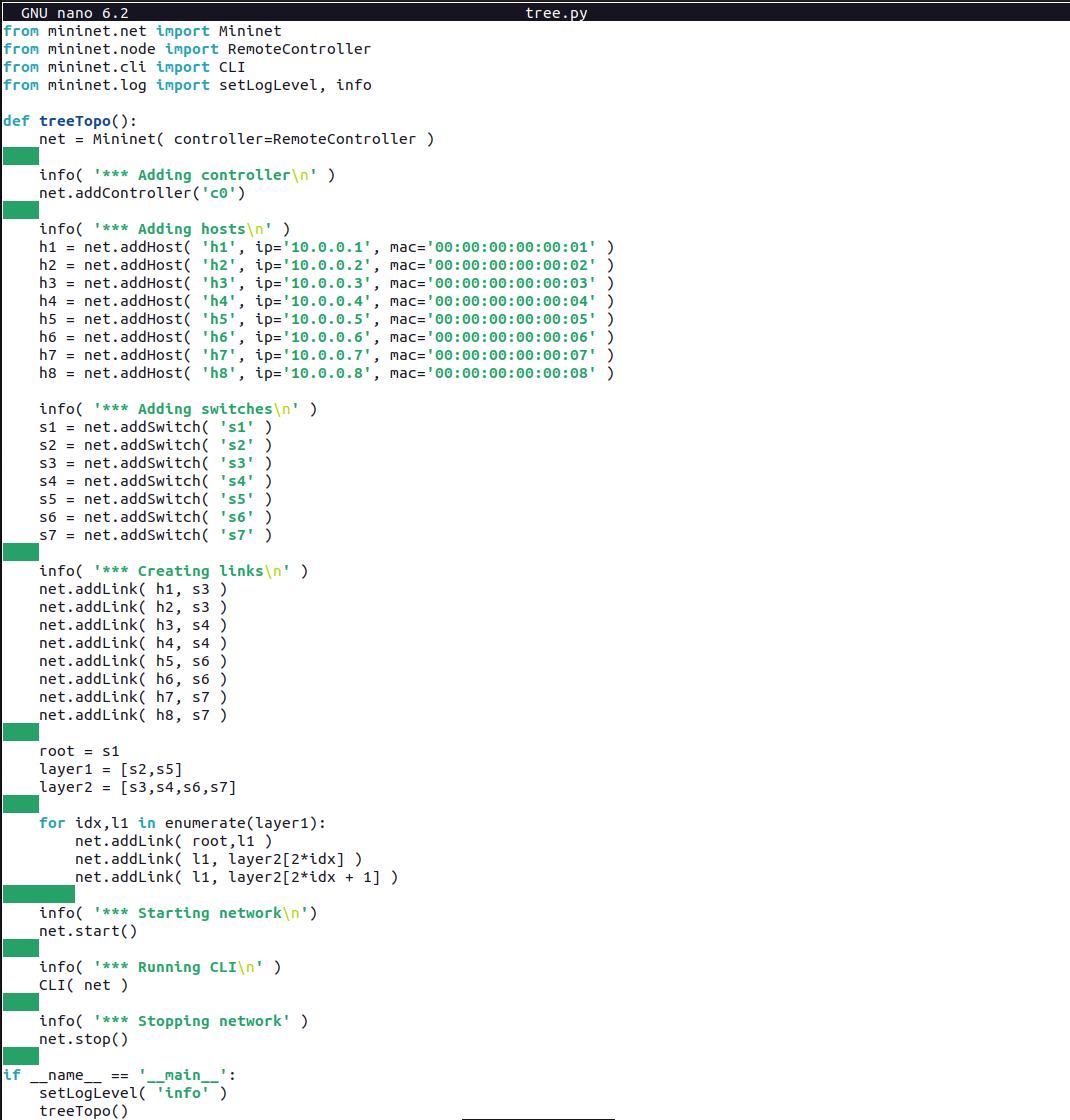
**Install mininet**



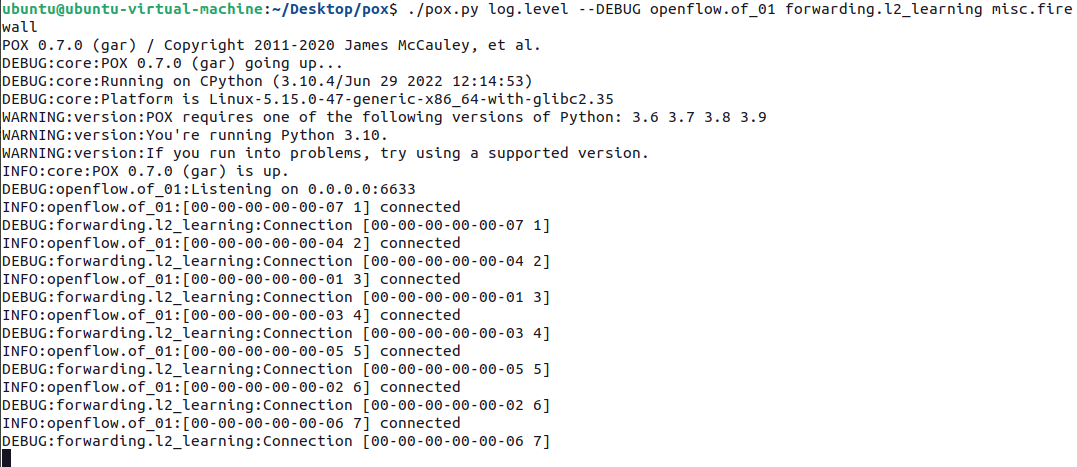
**Cd into pox/pox/misc and create the following file**



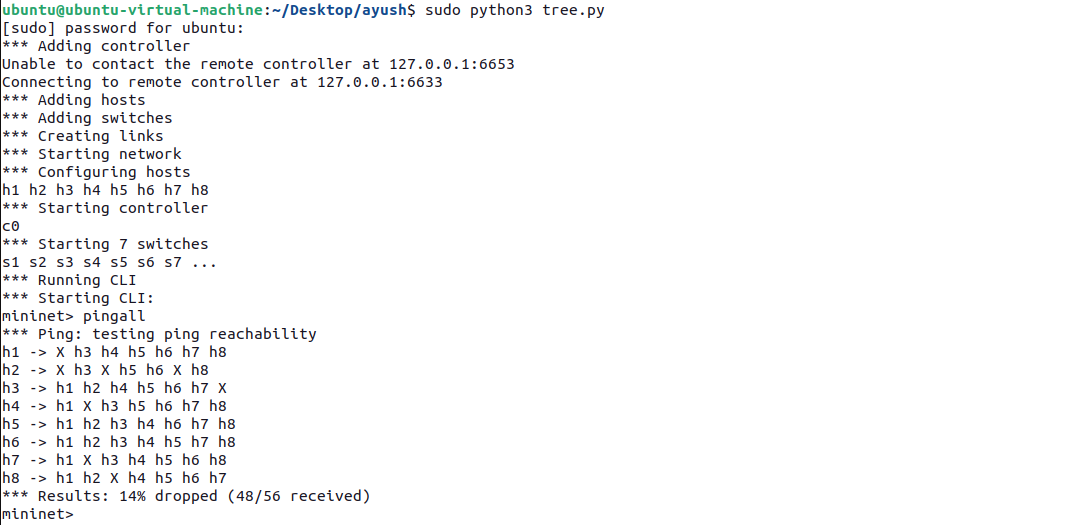
**Write the following config in any location**



**Cd into root pox repo and run the following command**



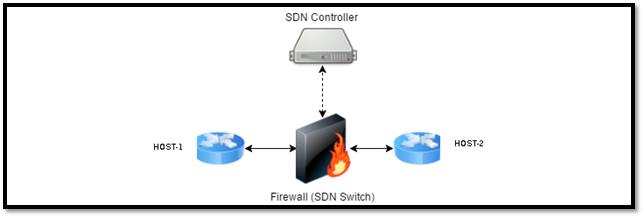
**Run the above tree.py**



**Write Up:**

**SDN Firewall**

A Firewall can be understood to be something that obstructs traffic coming its way and filters it according to some rules. A general Firewall can be employed to protect a network from the internet. For an SDN based Firewall, we are going to use the OpenFlow controller to filter traffic between hosts according to some rules and accordingly let it pass through or not. The way we will do this is by using the POX controller to establish our required policies or rules (we will come to these later) and filter traffic between hosts using the switches. An example of such an SDN Firewall with two hosts is shown in the figure below:

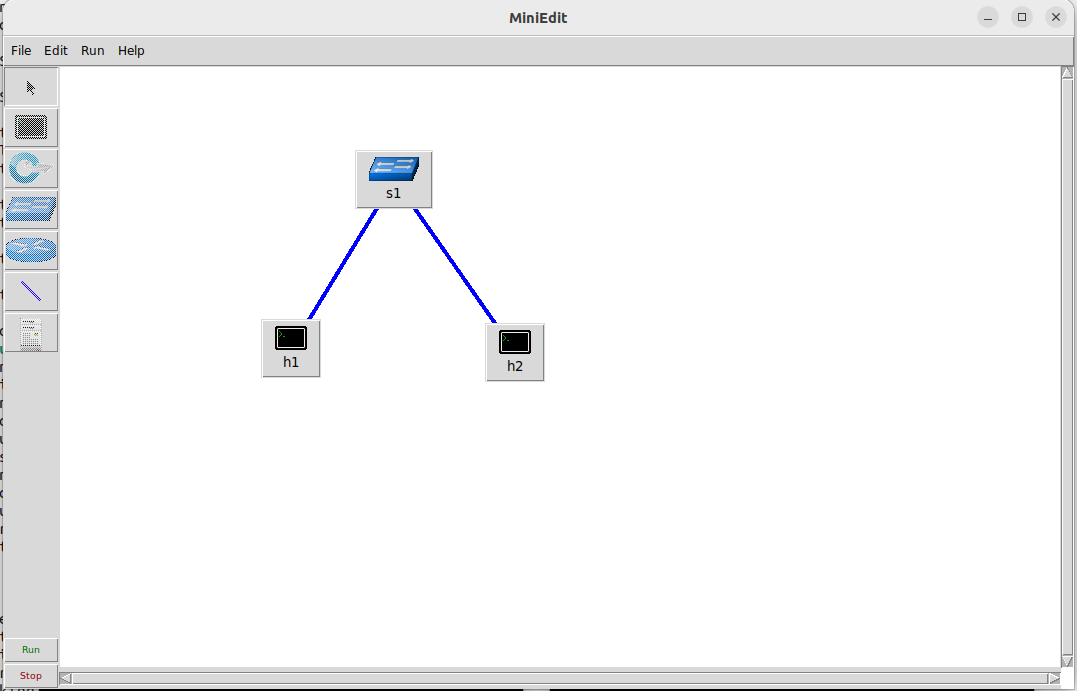


Practical No 7

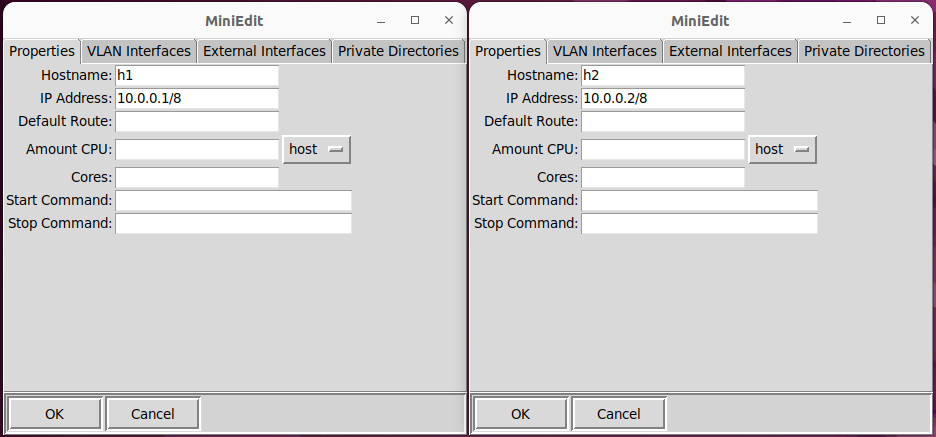
Aim : Implementing Simple Router using mininet.

Steps :

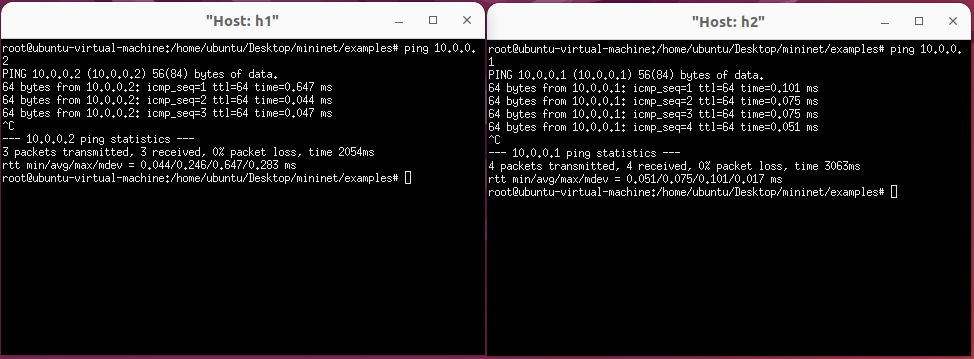
Create the following topology



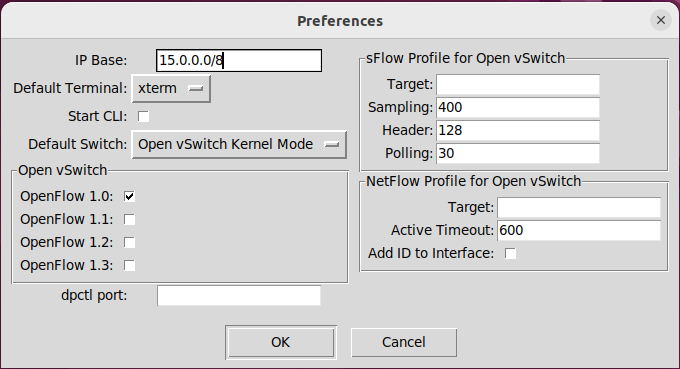
Configure the two hosts



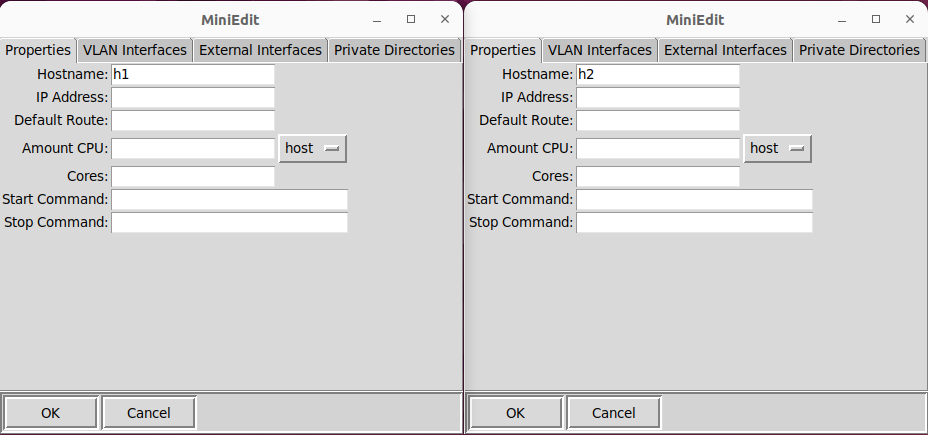
Try pinging the two hosts



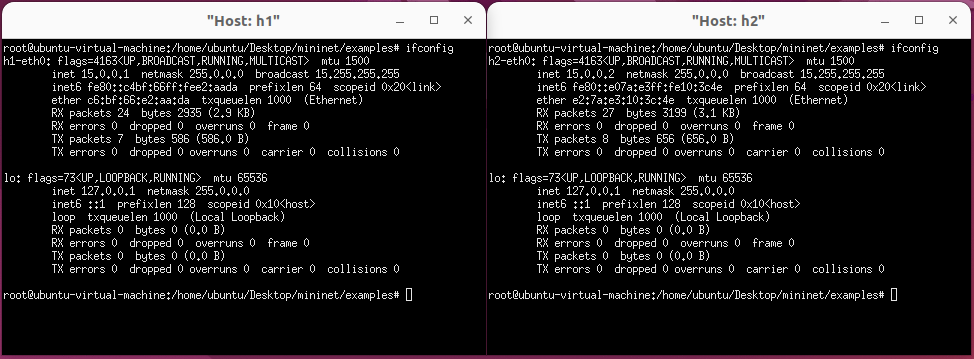
Make the following changes under Edit > Preferences



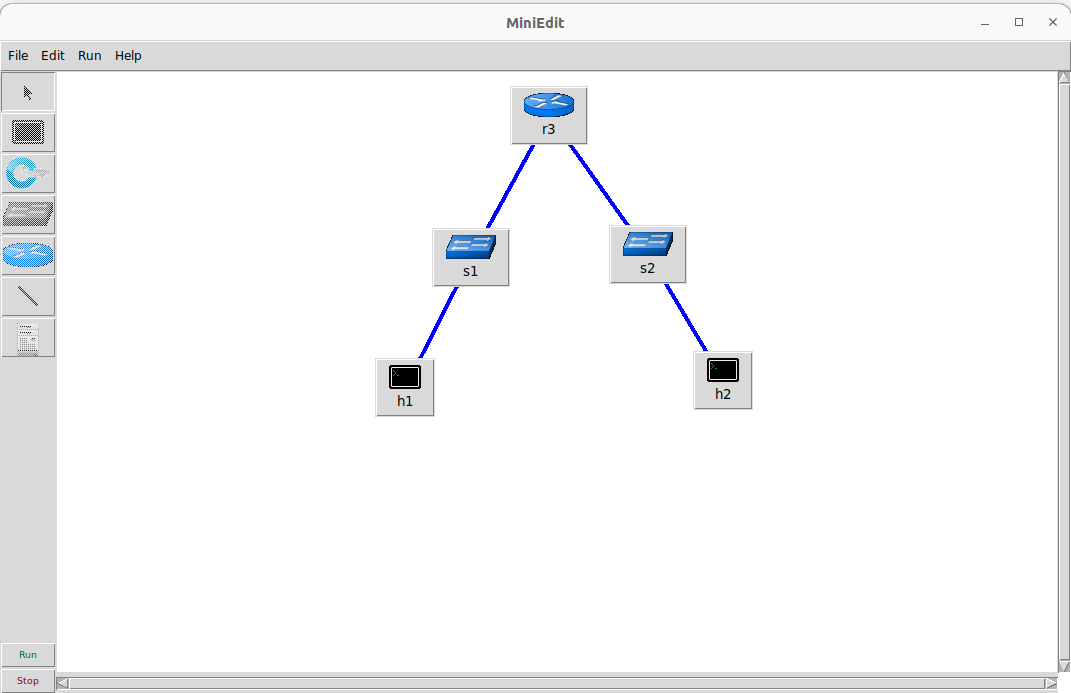
Make the following changes in the two hosts



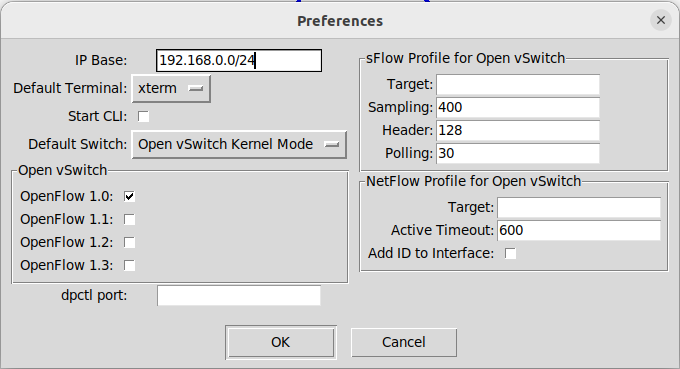
Check network info of the two hosts via ifconfg



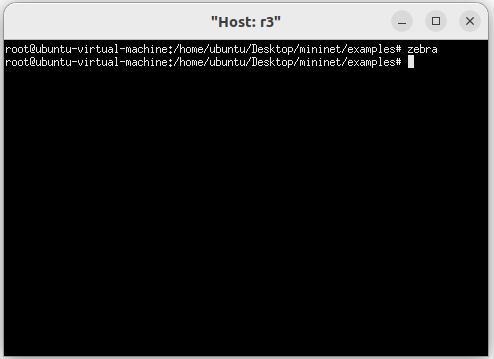
Create the following topology



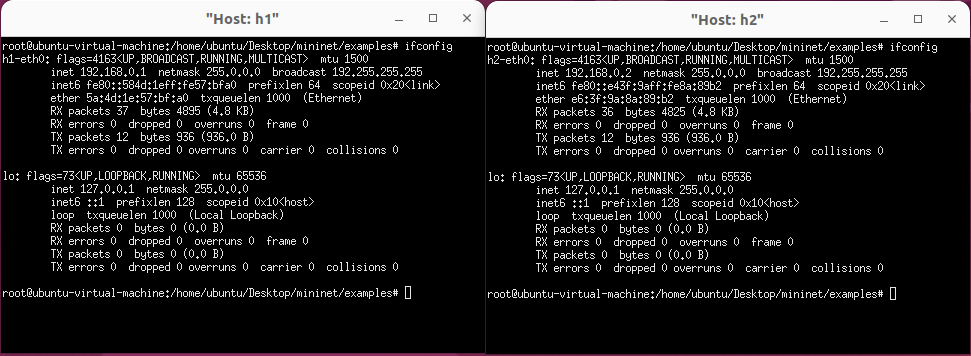
Make the following changes under Edit > Preferences



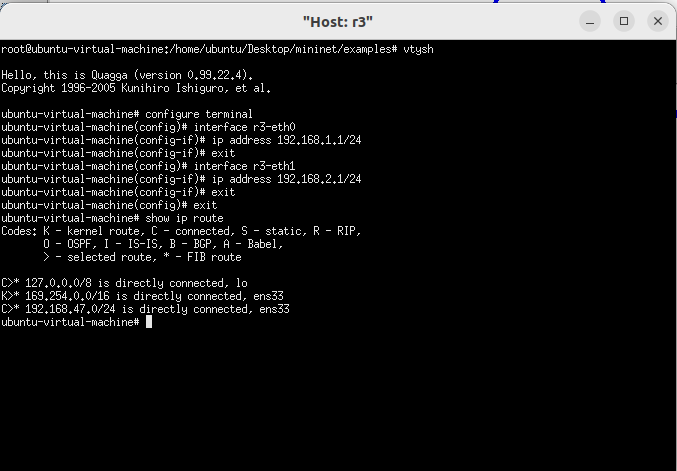
Check whether zebra command is working



Check network info of the two hosts via ifconfig



Configure the router



**Write Up:**

Sometimes simulations are not possible or not practical, and network experiments must be run on actual machines. One can always use a set of interconnected virtual machines, but even pared-down virtual machines consume sufficient resources that it is hard to create a network of more than a handful of nodes. Mininet is a system that supports the creation of lightweight logical nodes that can be connected into networks. These nodes are sometimes called containers, or, more accurately, network namespaces. Virtual-machine technology is not used. These containers consume sufficiently few resources that networks of over a thousand nodes have been created, running on a single laptop. While Mininet was originally developed as a testbed for software-defined networking (3.4 Software-Defined Networking), it works just as well for demonstrations and experiments involving traditional networking.

A Mininet container is a process (or group of processes) that no longer has access to all the host system’s “native” network interfaces, much as a process that has executed the chroot() system call no longer has access to the full filesystem. Mininet containers then are assigned virtual Ethernet interfaces (see the ip-link man page entries for veth), which are connected to other containers through virtual Ethernet links. The use of veth links ensures that the virtual links behave like Ethernet, though it may be necessary to disable TSO (17.5 TCP Offloading) to view Ethernet packets in WireShark as they would appear on the (virtual) wire. Any process started within a Mininet container inherits the container’s view of network interfaces.

For efficiency, Mininet containers all share the same filesystem by default. This makes setup simple, but sometimes causes problems with applications that expect individualized configuration files in specified locations. Mininet containers can be configured so that each container has at least one private directory, eg for configuration files. See 30.6 Quagga Routing and BGP for an example, though mostly we avoid this feature.

Mininet is a form of network emulation, as opposed to simulation. An important advantage of emulation is that all network software, at any layer, is simply run “as is”. In a simulator environment, on the other hand, applications and protocol implementations need to be ported to run within the simulator before they can be used. A drawback of emulation is that as the network gets large and complex the emulation may slow down. In particular, it is not possible to emulate link speeds faster than the underlying hardware can support. (It is also not possible to emulate non-Linux network software.)