BITS Pilani - Hyderabad Campus

CS F303 (Computer Networks)

Second Semester 2023-24, Lab Sheet 8

Subnet and Routing

1. Overview

In this lab, we will implement subnets and understand the routing in networks. The subnets are used to divide a big network into smaller networks for efficient networking whereas routing is a process of selecting a path for traffic between multiple hosts in the network. We will utilize Mininet as a virtualized environment for simulating hosts, switches and routers.

2. Deployment of Subnet with Mininet

In this lab, we will be using Mininet to create a subnet and understand the routing.

- To setup and use Mininet, follow the previous labsheets.
- Create a standard topology as shown in Figure 1 using the code given in the next section and place it in mininet/examples folder.

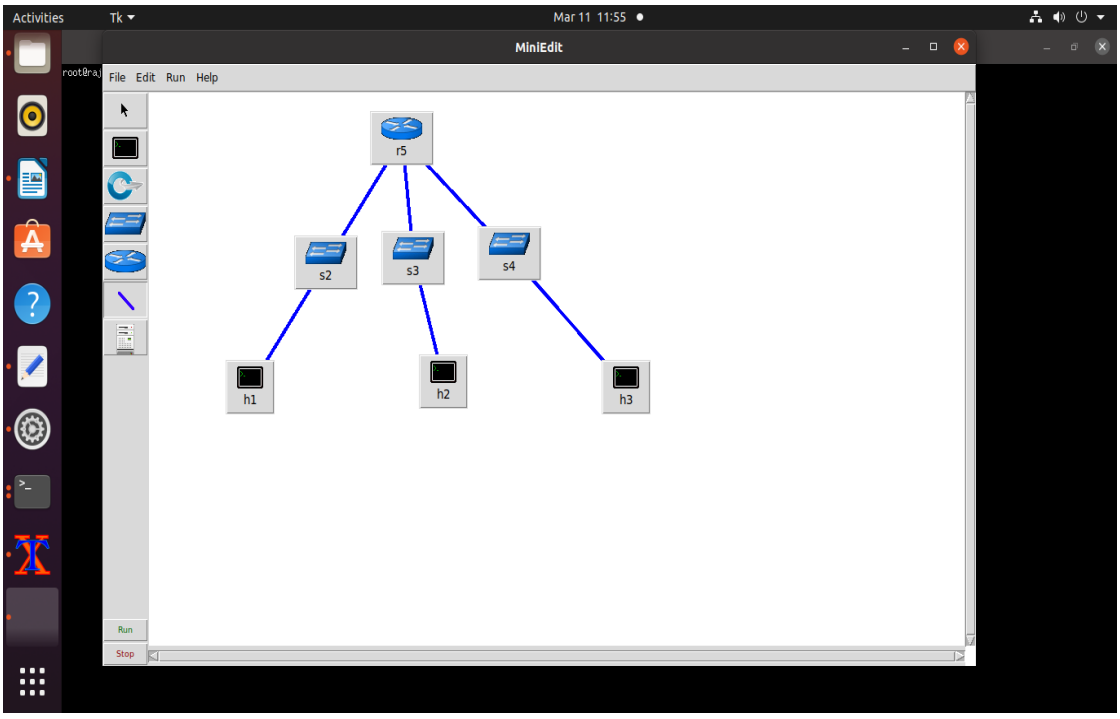

Fig 1: Topology with different subnets

3. Program (Subnetrouting.py)

```
#!/usr/bin/env python


from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Node
from mininet.log import setLogLevel, info
from mininet.cli import CLI


class LinuxRouter( Node ):
        "A Node with IP forwarding enabled."

        # pylint: disable=arguments-differ
        def config( self, **params ):
        super( LinuxRouter, self).config( **params )
        # Enable forwarding on the router
        self.cmd( 'sysctl net.ipv4.ip_forward=1' )

        def terminate( self ):
        self.cmd( 'sysctl net.ipv4.ip_forward=0' )
        super( LinuxRouter, self ).terminate()


class NetworkTopo( Topo ):
        "A LinuxRouter connecting three IP subnets"

        # pylint: disable=arguments-differ
        def build( self, **_opts ):

        defaultIP = '192.168.1.1/24'  # IP address for r0-eth1
        router = self.addNode( 'r0', cls=LinuxRouter, ip=defaultIP )

        s1, s2, s3 = [ self.addSwitch( s ) for s in ( 's1', 's2', 's3' ) ]

        self.addLink( s1, router, intfName2='r0-eth1',
                params2={ 'ip' : defaultIP } )  # for clarity
        self.addLink( s2, router, intfName2='r0-eth2',
                params2={ 'ip' : '172.16.0.1/12' } )
        self.addLink( s3, router, intfName2='r0-eth3',
                params2={ 'ip' : '10.0.0.1/8' } )

        h1 = self.addHost( 'h1', ip='192.168.1.100/24',
                           defaultRoute='via 192.168.1.1' )
        h2 = self.addHost( 'h2', ip='172.16.0.100/12',
                           defaultRoute='via 172.16.0.1' )
        h3 = self.addHost( 'h3', ip='10.0.0.100/8',
                           defaultRoute='via 10.0.0.1' )

        for h, s in [ (h1, s1), (h2, s2), (h3, s3) ]:
```

```
        self.addLink( h, s )


def run():
        "Test linux router"
        topo = NetworkTopo()
        net = Mininet( topo=topo,
                waitConnected=True )  # controller is used by s1-s3
        net.start()
        info( '*** Routing Table on Router:\n' )
        info( net[ 'r0' ].cmd( 'route' ) )
        CLI( net )
        net.stop()


if __name__ == '__main__':
        setLogLevel( 'info' )
        run()
```

This code converts a Node into a router using IP forwarding already built into Linux. The example topology creates a router and three IP subnets:

- 192.168.1.0/24 (r0-eth1, IP: 192.168.1.1)
- 172.16.0.0/12 (r0-eth2, IP: 172.16.0.1)
- 10.0.0.0/8 (r0-eth3, IP: 10.0.0.1)

Each subnet consists of a single host connected to a single switch:

- r0-eth1 - s1-eth1 - h1-eth0 (IP: 192.168.1.100)
- r0-eth2 - s2-eth1 - h2-eth0 (IP: 172.16.0.100)
- r0-eth3 - s3-eth1 - h3-eth0 (IP: 10.0.0.100)

The example relies on default routing entries that are automatically created for each router interface, as well as 'defaultRoute' parameters for the host interfaces. Additional routes may be added to the router or hosts by executing 'ip route' or 'route' commands on the router or hosts.

4. Implementation of Code and Analysis

Execute the program with command sudo python subnetrouting.py. This will create the required Fig 1 topology. Now, run the mininet command to verify the topology and the network connectivity using:

- Nodes (Figure 2)
- net (Figure 3)

Use xterm to go to the each of the machine and check its configuration using ifconfig command (Figure 4).

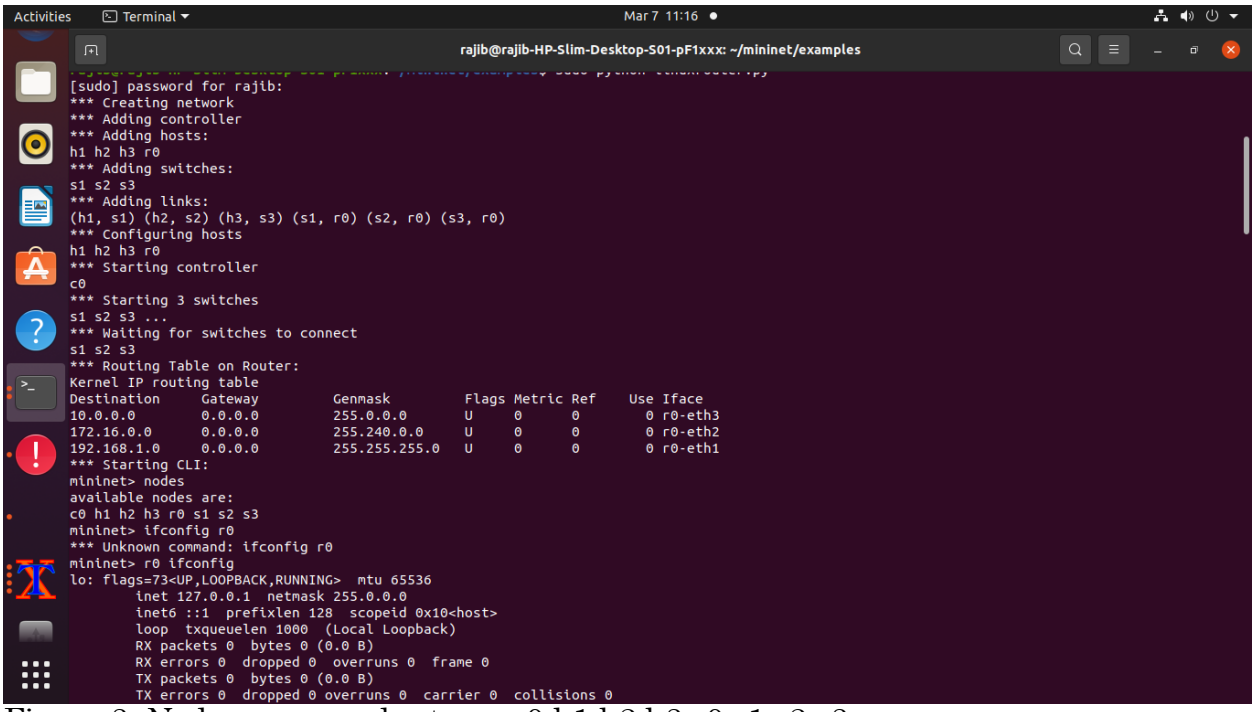

Figure 3: Nodes command return: c0 h1 h2 h3 r0 s1 s2 s3

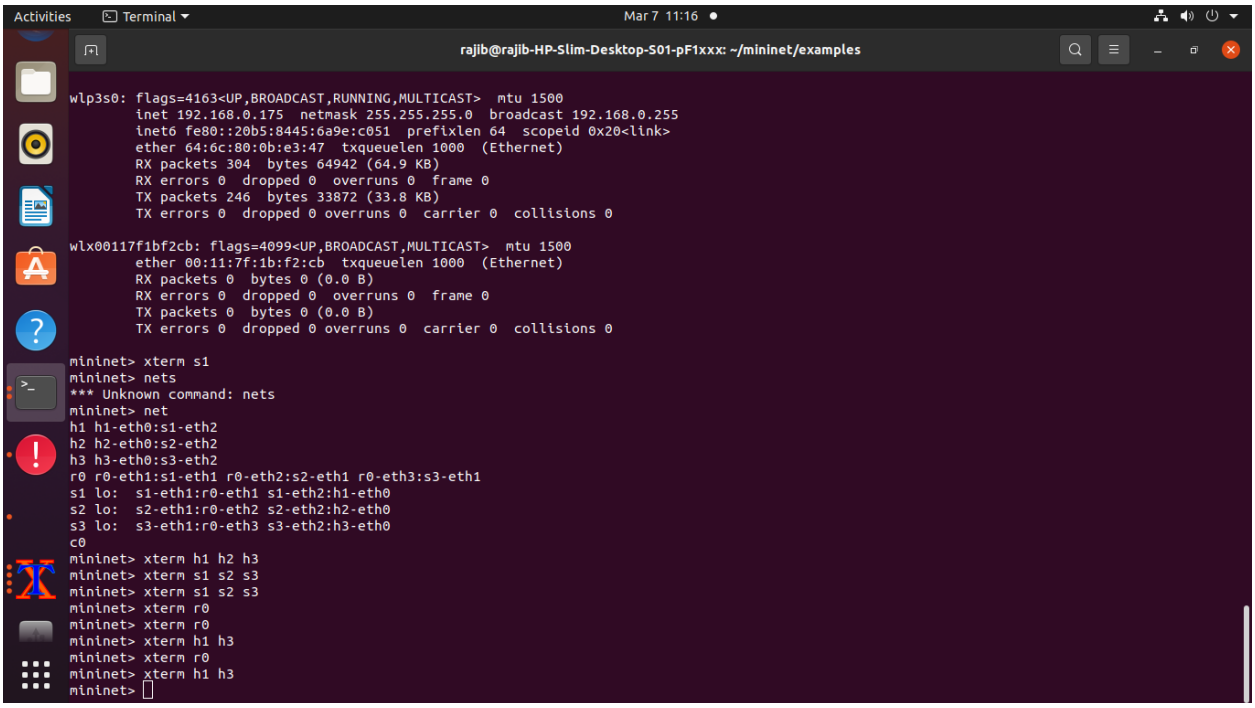

Figure 3: nets command output to see the topology and various interfaces

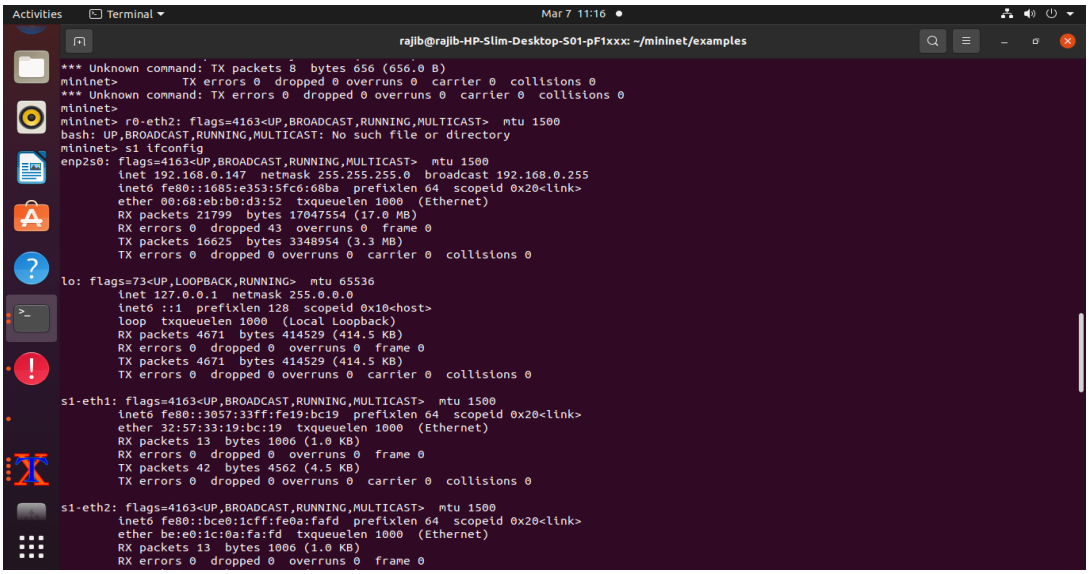

Figure 4: Xterm with ifconfig command output

You can also Ping from h1 machine to h3 machine to check the reachability and run the ip route from the router machine and route commands to see the rules of routing table (Figure 5).

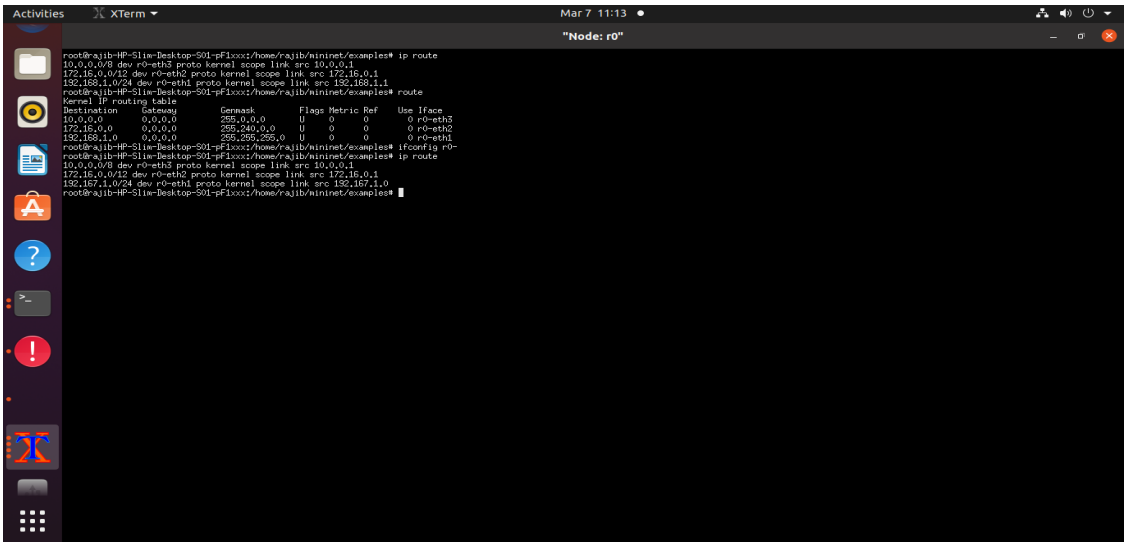

Figure 5: ip route and route command output

5. Lab Exercise

While running the program (subnetrouting.py), a topology is created where each subnet is reachable to another. Change one of the ip table route in the router to make one of the subnets unreachable.