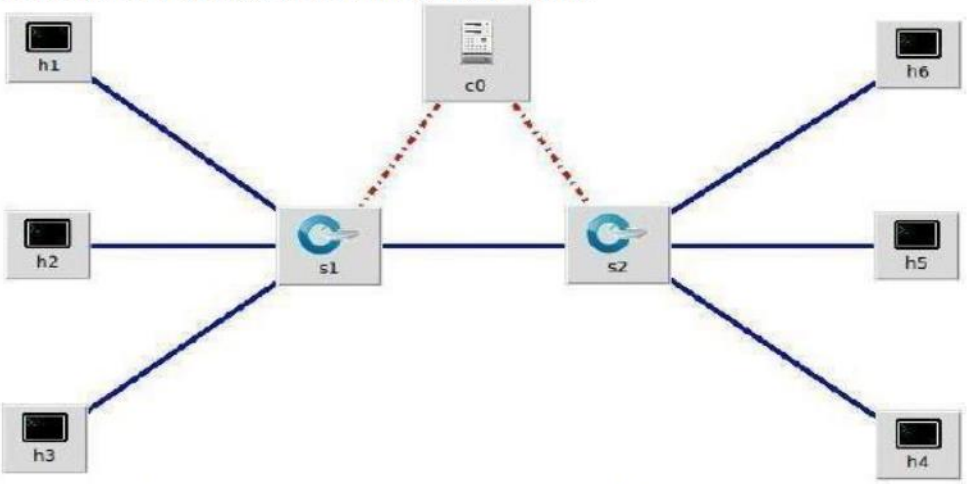
PRACTICAL-4

AIM:

Create a custom dumbbell topology as shown in the figure below.



Keep bandwidth and delay as 50 Mbps and 5 ms for all the links. Validate your topology using dump, pingall and arp.

THEORY:

Bandwidth:

- The maximum amount of data transmitted over an internet connection in a given amount of time

Mbps:

- Megabits per second (Mbps) are units of measurement for network bandwidth and throughput. They are used to show how fast a network or internet connection is.

Dump:

- It lists information about the nodes, switches and controllers in the simulated network

pingall:

- It displays the connectivity between all hosts and tells us which hosts are connected to each other

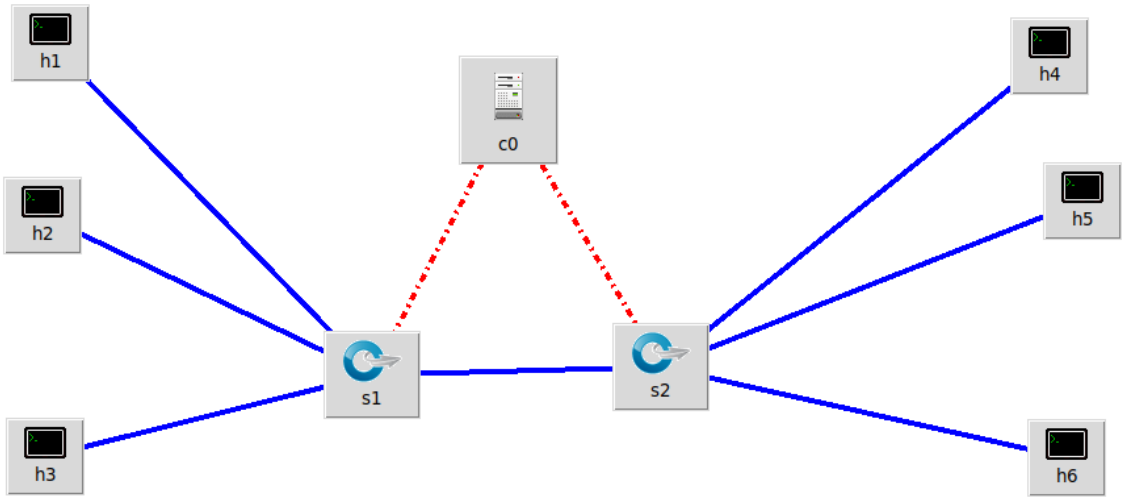
dpctl:

- It is used to view the flows in switch table

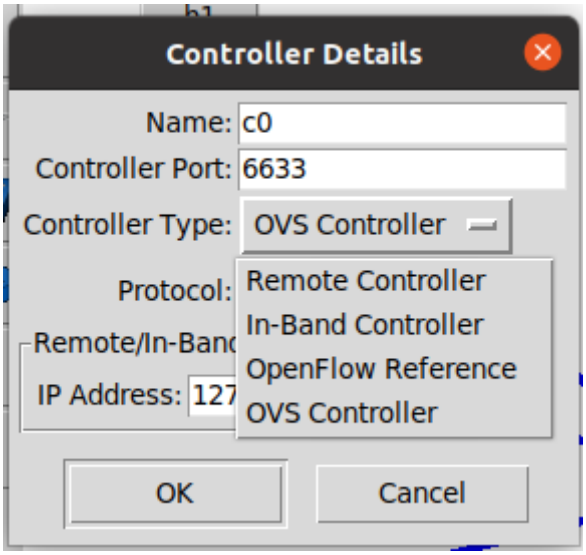
ARP:

- The Address Resolution Protocol is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address, typically an IPv4 address.

TOPOLOGY:

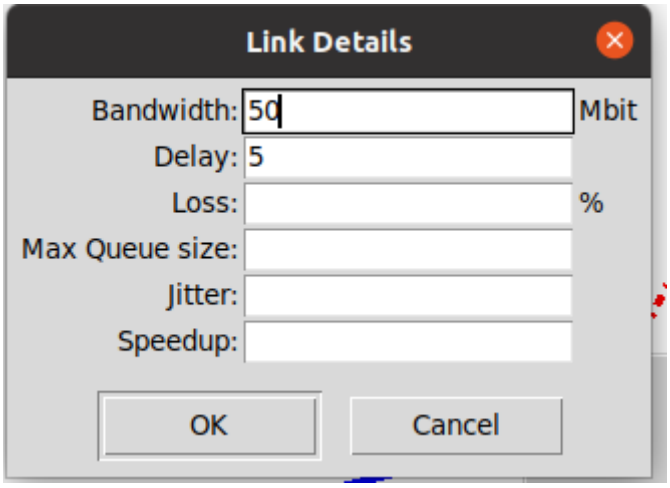


- Change the Controller Type of Controller to OVS Controller.



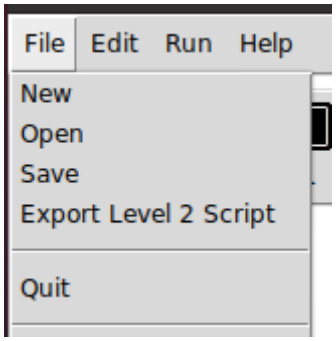
- Now, click on the blue link, and it turns green, then right click it and select properties.

- Enter the bandwidth as 50 Mbps and delay is 5 milli seconds.



- Do the same for all the link.
- Save the file.
- Then, click on file and select Export Level 2 script.
- It will generate the python file.

Note: we can create the same topology using python code.



PYTHON CODE:

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


from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call


def myNetwork():


    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')


    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                         controller=OVSController,
                         protocol='tcp',
                         port=6633)
```

```
info( '*** Add switches\n')
s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
s2 = net.addSwitch('s2', cls=OVSKernelSwitch)

info( '*** Add hosts\n')
h1 = net.addHost('h1', cls=Host, ip='10.10.10.2', defaultRoute=None)
h2 = net.addHost('h2', cls=Host, ip='10.10.10.3', defaultRoute=None)
h3 = net.addHost('h3', cls=Host, ip='10.10.10.4', defaultRoute=None)
h4 = net.addHost('h4', cls=Host, ip='10.10.10.7', defaultRoute=None)
h5 = net.addHost('h5', cls=Host, ip='10.10.10.6', defaultRoute=None)
h6 = net.addHost('h6', cls=Host, ip='10.10.10.5', defaultRoute=None)

info( '*** Add links\n')
s2s1 = {'bw':50,'delay':'5'}
net.addLink(s2, s1, cls=TCLink , **s2s1)
h1s1 = {'bw':50,'delay':'5'}
net.addLink(h1, s1, cls=TCLink , **h1s1)
h2s1 = {'bw':50,'delay':'5'}
net.addLink(h2, s1, cls=TCLink , **h2s1)
h3s1 = {'bw':50,'delay':'5'}
net.addLink(h3, s1, cls=TCLink , **h3s1)
s2h4 = {'bw':50,'delay':'5'}
net.addLink(s2, h4, cls=TCLink , **s2h4)
```

```
    s2h5 = {'bw':50,'delay':'5'}
    net.addLink(s2, h5, cls=TCLink , **s2h5)
    s2h6 = {'bw':50,'delay':'5'}
    net.addLink(s2, h6, cls=TCLink , **s2h6)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches\n')
    net.get('s1').start([c0])
    net.get('s2').start([c0])

    info( '*** Post configure switches and hosts\n')

    CLI(net)
    net.stop()

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

- Now, we will check for the dump command.

```
mininet> dump
<Host h1: h1-eth0:10.10.10.2 pid=4513>
<Host h2: h2-eth0:10.10.10.3 pid=4515>
<Host h3: h3-eth0:10.10.10.4 pid=4517>
<Host h4: h4-eth0:10.10.10.7 pid=4519>
<Host h5: h5-eth0:10.10.10.6 pid=4521>
<Host h6: h6-eth0:10.10.10.5 pid=4523>
<customOvs s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None
pid=4525>
<customOvs s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None
pid=4528>
<OVSController c0: 127.0.0.1:6633 pid=4508>
mininet>
```

- Now, we will check for pingall command.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results: 0% dropped (30/30 received)
```

- Now, we will check for arp command.

- Syntax: <hostname> arp

```
mininet> h1 arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.10.5               ether   82:cb:e3:62:4c:cd   C                     h1-et
h0
10.10.10.4               ether   ca:1b:a3:7b:b0:7c   C                     h1-et
h0
10.10.10.7               ether   92:d3:e0:ab:70:71   C                     h1-et
h0
10.10.10.6               ether   d6:ac:1f:74:e9:9b   C                     h1-et
h0
10.10.10.3               ether   1e:a8:f0:c2:72:4c   C                     h1-et
h0
```

- Now, we will perform the commands for iperf and link

```
mininet> iperf s1 s2
*** Iperf: testing TCP bandwidth between s1 and s2
*** Results: ['9.12 Gbits/sec', '9.14 Gbits/sec']
```

```
mininet> links
s2-eth1<->s1-eth1 (OK OK)
h1-eth0<->s1-eth2 (OK OK)
h2-eth0<->s1-eth3 (OK OK)
h3-eth0<->s1-eth4 (OK OK)
s2-eth2<->h4-eth0 (OK OK)
s2-eth3<->h5-eth0 (OK OK)
s2-eth4<->h6-eth0 (OK OK)
```

- Command: dpctl show

```
mininet> dpctl show
*** s1 ------------------------------------------------------------------------
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(s1-eth1): addr:16:e1:08:ec:94:79
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s1-eth2): addr:5e:0a:d6:cd:c4:8a
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s1-eth3): addr:5e:d2:85:4d:2a:c0
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 4(s1-eth4): addr:5e:00:60:0d:98:2f
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s1): addr:86:b9:c4:fe:9d:47
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

```
*** s2 ----------------------------------------------------------------------
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000002
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(s2-eth1): addr:ae:43:60:36:ed:c5
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s2-eth2): addr:3a:6b:a9:7f:00:ea
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s2-eth3): addr:56:c8:85:9b:d4:cd
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 4(s2-eth4): addr:da:b3:48:10:64:fa
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s2): addr:42:2f:ea:91:93:40
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

- Command: dpctl dump-desc

```
mininet> dpctl dump-desc
*** s1 ----------------------------
OFPST_DESC reply (xid=0x2):
Manufacturer: Nicira, Inc.
Hardware: Open vSwitch
Software: 2.13.3
Serial Num: None
DP Description: s1
*** s2 ----------------------------
OFPST_DESC reply (xid=0x2):
Manufacturer: Nicira, Inc.
Hardware: Open vSwitch
Software: 2.13.3
Serial Num: None
DP Description: s2
```

- Command: dpctl dump-ports

```
mininet> dpctl dump-ports
*** s1 ----------------------------------------------------------------------
OFPST_PORT reply (xid=0x2): 5 ports
  port LOCAL: rx pkts=0, bytes=0, drop=87, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth4": rx pkts=12, bytes=936, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=103, bytes=10106, drop=0, errs=0, coll=0
  port  "s1-eth1": rx pkts=57, bytes=5486, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=58, bytes=5556, drop=0, errs=0, coll=0
  port  "s1-eth2": rx pkts=11, bytes=866, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=104, bytes=10176, drop=0, errs=0, coll=0
  port  "s1-eth3": rx pkts=11, bytes=866, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=104, bytes=10176, drop=0, errs=0, coll=0
*** s2 ----------------------------------------------------------------------
OFPST_PORT reply (xid=0x2): 5 ports
  port LOCAL: rx pkts=0, bytes=0, drop=86, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s2-eth1": rx pkts=58, bytes=5556, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=57, bytes=5486, drop=0, errs=0, coll=0
  port  "s2-eth4": rx pkts=11, bytes=866, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=103, bytes=10090, drop=0, errs=0, coll=0
  port  "s2-eth2": rx pkts=11, bytes=866, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=103, bytes=10090, drop=0, errs=0, coll=0
  port  "s2-eth3": rx pkts=11, bytes=866, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=103, bytes=10090, drop=0, errs=0, coll=0
```

- Command: dpctl dump-ports-desc

CONCLUSION:

- By performing the above practical, I learnt how to create topology in mini-edit, how to execute the commands of pingall, arp, dump.

- I also learnt about how to alter the link configuration between the devices.