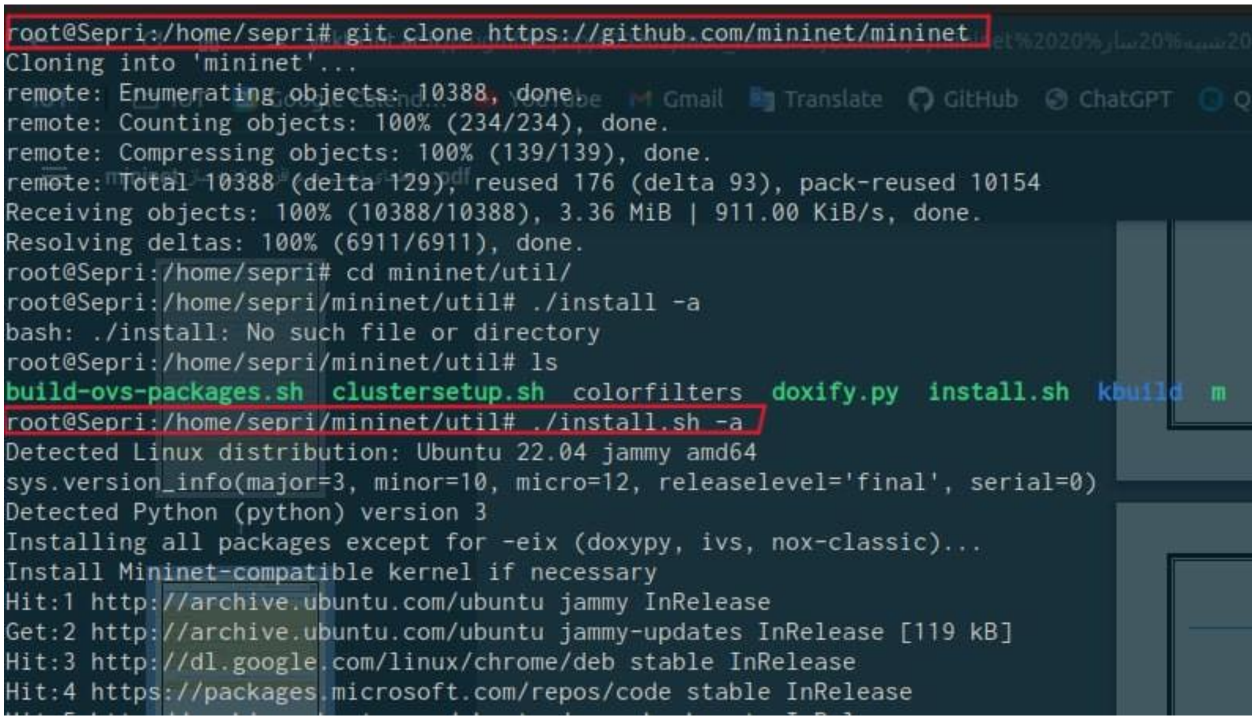
Network Project 1

Sepehr Ebadi

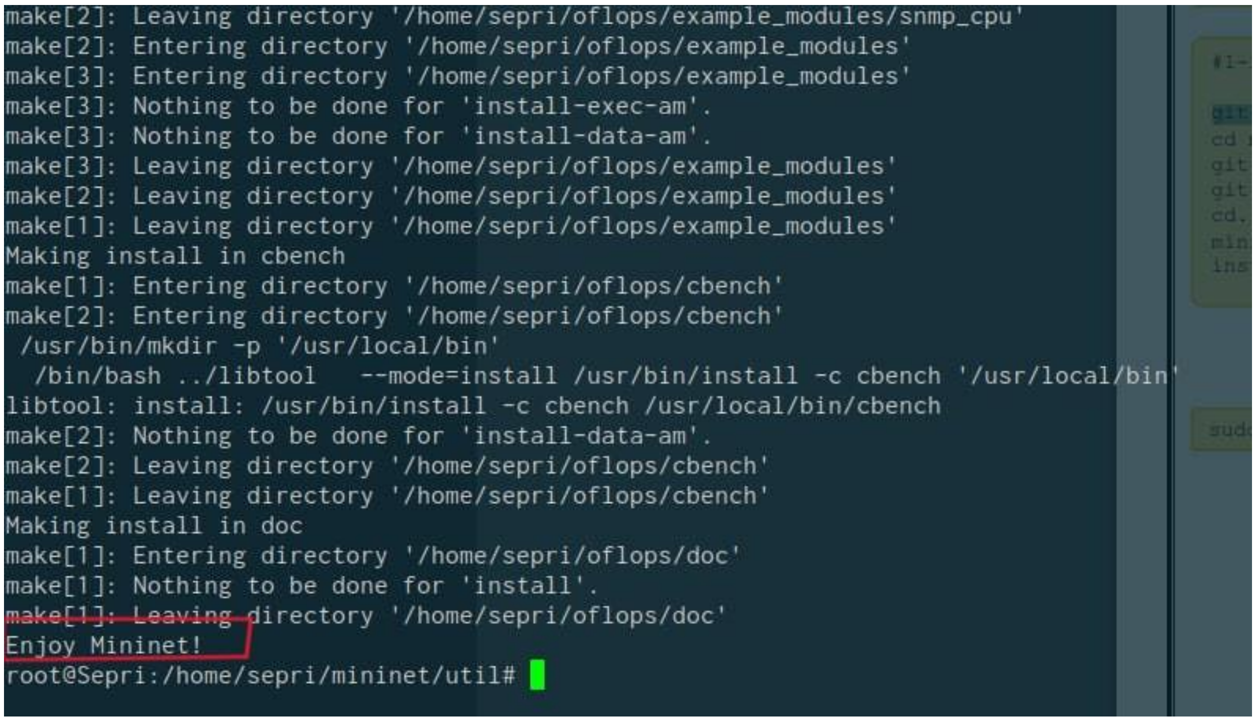
9933243

Khordad, 1403

Installation

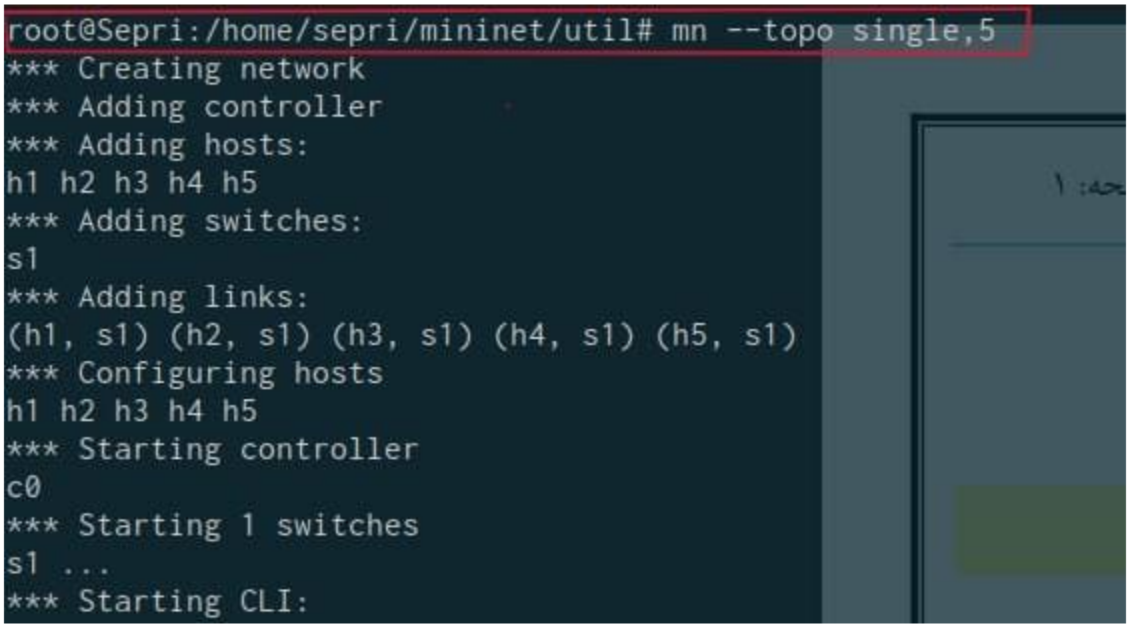


```
root@Sepri:/home/sepri# git clone https://github.com/mininet/mininet
Cloning into 'mininet'...
remote: Enumerating objects: 10388, done.
remote: Counting objects: 100% (234/234), done.
remote: Compressing objects: 100% (139/139), done.
remote: Total 10388 (delta 129), reused 176 (delta 93), pack-reused 10154
Receiving objects: 100% (10388/10388), 3.36 MiB | 911.00 KiB/s, done.
Resolving deltas: 100% (6911/6911), done.
root@Sepri:/home/sepri# cd mininet/util/
root@Sepri:/home/sepri/mininet/util# ./install -a
bash: ./install: No such file or directory
root@Sepri:/home/sepri/mininet/util# ls
build-ovs-packages.sh  clustersetup.sh  colorfilters  doxify.py  install.sh  kbuild  m
root@Sepri:/home/sepri/mininet/util# ./install.sh -a
Detected Linux distribution: Ubuntu 22.04 jammy amd64
sys.version_info(major=3, minor=10, micro=12, releaselevel='final', serial=0)
Detected Python (python) version 3
Installing all packages except for -eix (doxypy, ivs, nox-classic)...
Install Mininet-compatible kernel if necessary
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:3 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:4 https://packages.microsoft.com/repos/code stable InRelease
```



```
make[2]: Leaving directory '/home/sepri/oflops/example_modules/snmp_cpu'
make[2]: Entering directory '/home/sepri/oflops/example_modules'
make[3]: Entering directory '/home/sepri/oflops/example_modules'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/home/sepri/oflops/example_modules'
make[2]: Leaving directory '/home/sepri/oflops/example_modules'
make[1]: Leaving directory '/home/sepri/oflops/example_modules'
Making install in cbench
make[1]: Entering directory '/home/sepri/oflops/cbench'
make[2]: Entering directory '/home/sepri/oflops/cbench'
 /usr/bin/mkdir -p '/usr/local/bin'
  /bin/bash ../libtool   --mode=install /usr/bin/install -c cbench '/usr/local/bin'
libtool: install: /usr/bin/install -c cbench /usr/local/bin/cbench
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/sepri/oflops/cbench'
make[1]: Leaving directory '/home/sepri/oflops/cbench'
Making install in doc
make[1]: Entering directory '/home/sepri/oflops/doc'
make[1]: Nothing to be done for 'install'.
make[1]: Leaving directory '/home/sepri/oflops/doc'
Enjoy Mininet!
root@Sepri:/home/sepri/mininet/util#
```

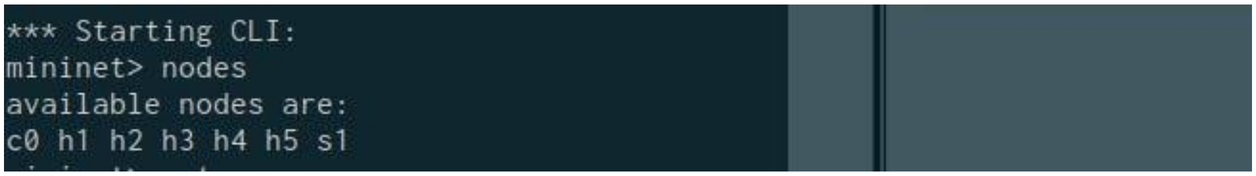
Single Topology



```
root@Sepri:/home/sepri/mininet/util# mn --topo single,5
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Nodes

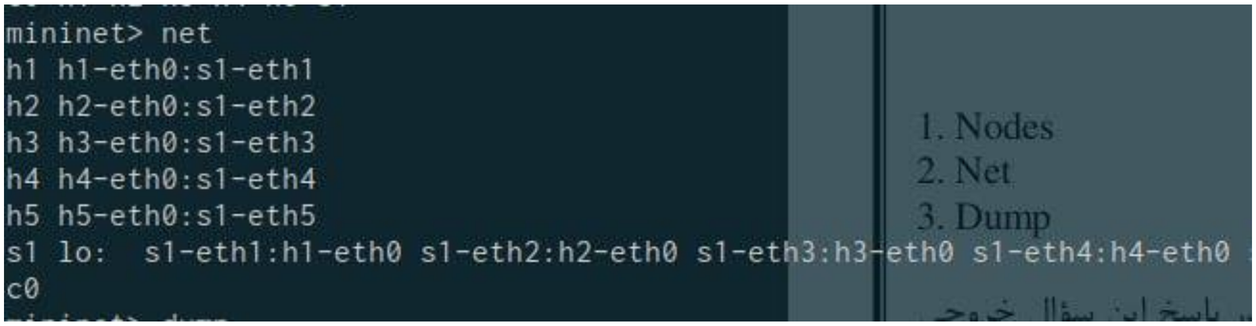
Display all the nodes controller, switch and hosts of the topology In the Following example, we have 5 hosts from h1 to h5, one controller c0, and one switch s1.



```
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 s1
```

Net

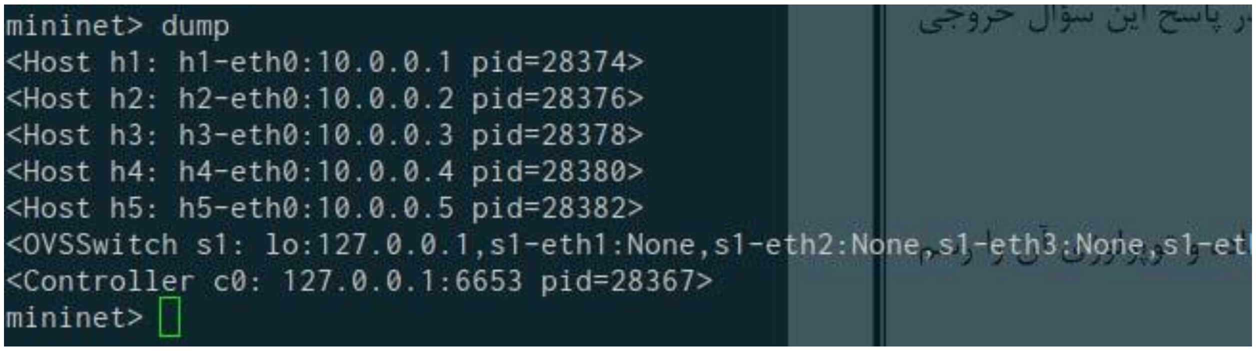
Show the links connection. In the following example, hosts 1 to 5 are connected to switch S1 through eth0. Switch S1 is connected to hosts h1 to h5 through interfaces eth1 to eth5. In conclusion, all the hosts are connected to each other through switch S1.



```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
h5 h5-eth0:s1-eth5
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0
c0
```

Dump

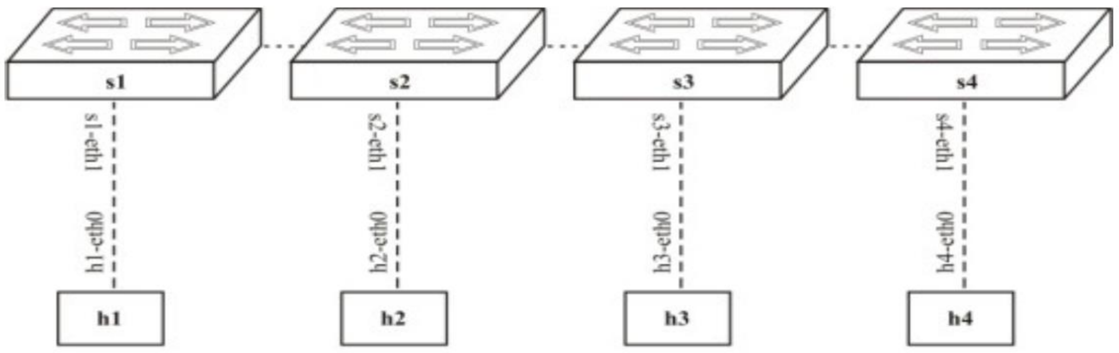
Provide all the information about the nodes, such as the node's name, the network interface with its IP address, and the process ID. For instance, in the following example host h1-eth IP is 10.0.0.1 with PID 28374, switch s1 loop back interface IP is 127.0.0.1 and controller c0 IP is 127.0.0.1:6653 with PID 28367

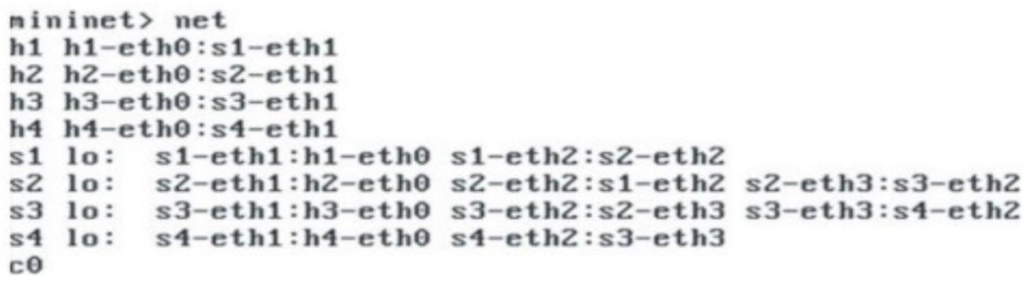


```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=28374>
<Host h2: h2-eth0:10.0.0.2 pid=28376>
<Host h3: h3-eth0:10.0.0.3 pid=28378>
<Host h4: h4-eth0:10.0.0.4 pid=28380>
<Host h5: h5-eth0:10.0.0.5 pid=28382>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-et
<Controller c0: 127.0.0.1:6653 pid=28367>
mininet> 
```

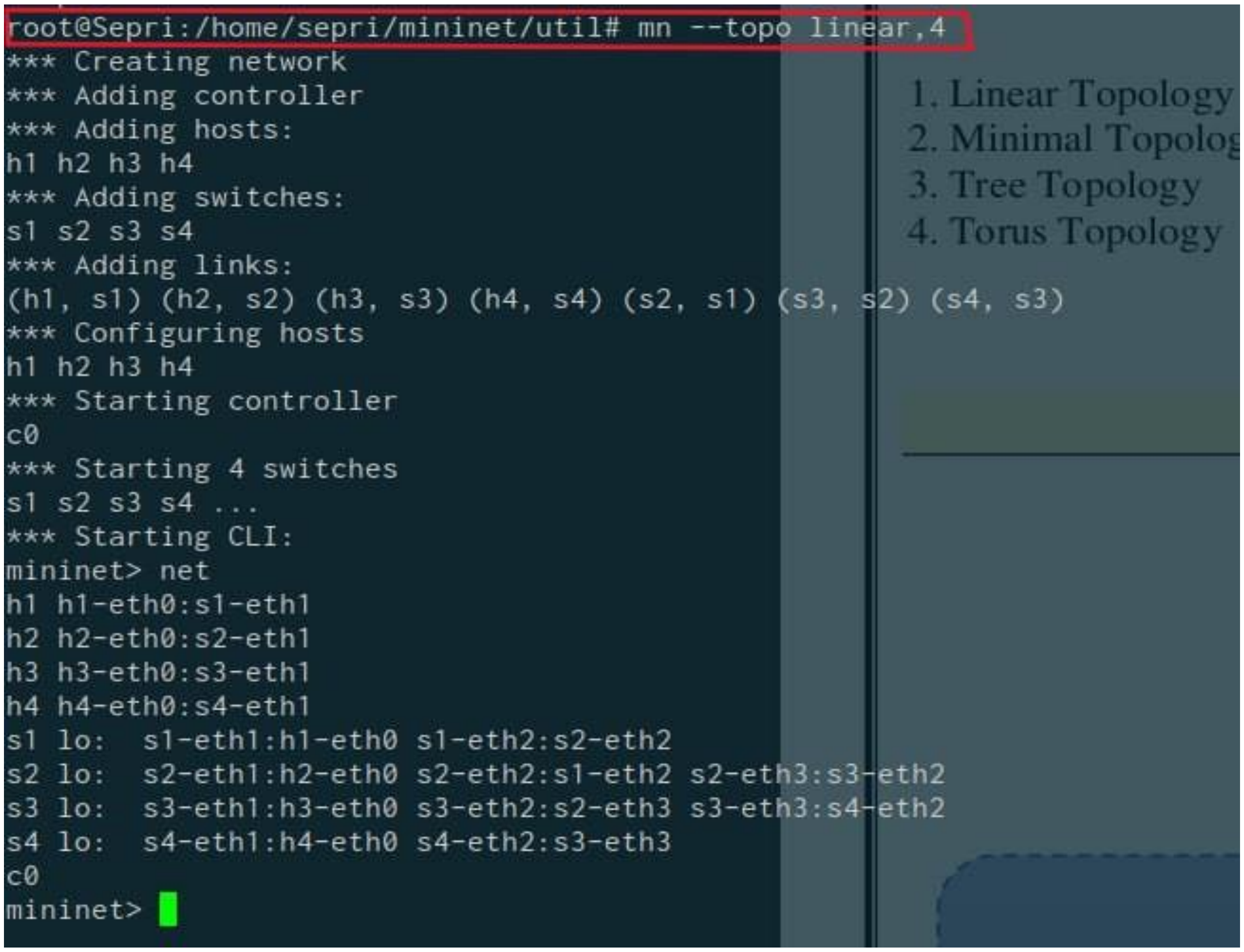
Linear Topology

A linear topology is a type of network topology in which all the network nodes are arranged in a straight line, with each node connected to its nearest neighbors. The first and last nodes in the topology have only one neighbor, while the intermediate nodes have two neighbors. In a linear topology, data can be transmitted in only one direction, from one end of the topology to the other. This makes it easy to add or remove nodes, but it also means that the failure of any one node can result in the entire network being disrupted. Linear topologies are commonly used in small networks, such as home or office networks. They are easy to set up and maintain, and can be implemented using a variety of technologies, including wired and wireless connections. Linear topologies are not well-suited for large networks or networks with high traffic, as they can become congested and suffer from latency and packet loss. In such cases, more complex topologies, such as ring, mesh, or tree topologies, may be more appropriate.





```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo:  s4-eth1:h4-eth0 s4-eth2:s3-eth3
c0
```

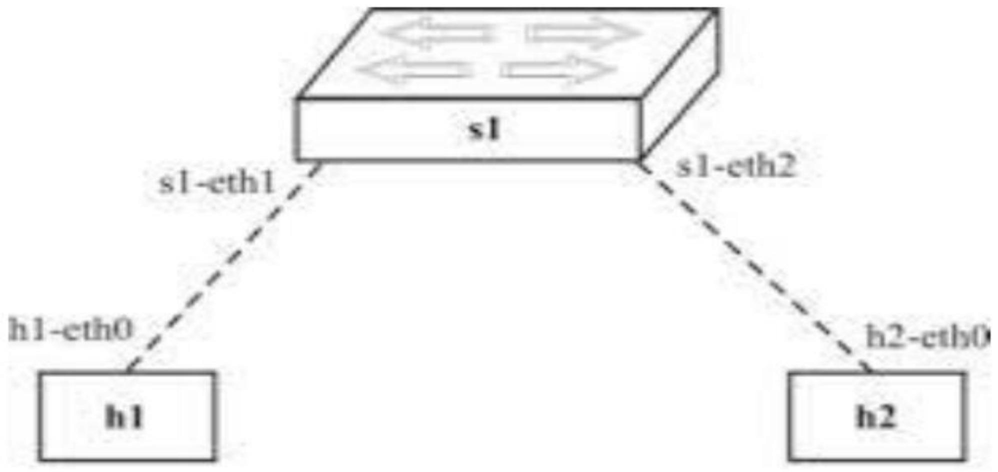


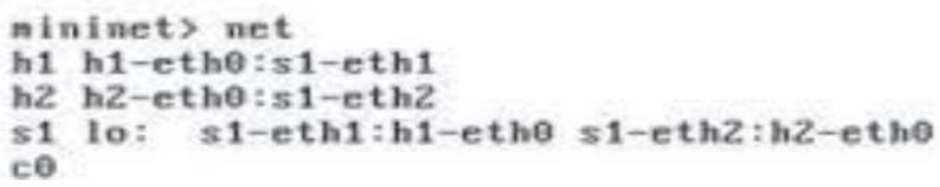
```
root@Sepri:/home/sepri/mininet/util# mn --topo linear,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo:  s4-eth1:h4-eth0 s4-eth2:s3-eth3
c0
mininet>
```

1. Linear Topology
2. Minimal Topolog
3. Tree Topology
4. Torus Topology

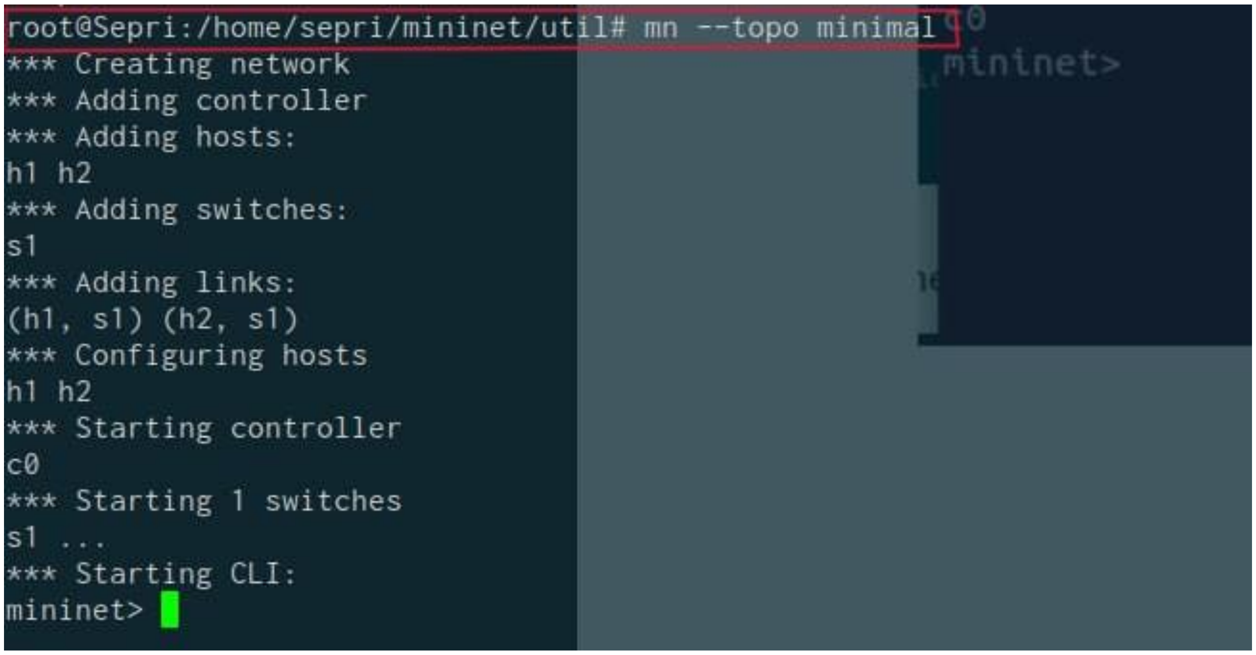
Minimal Topology

A minimal topology is a type of network topology that consists of the smallest number of nodes required to establish network connectivity between them. In other words, a minimal topology is the simplest possible network topology that can achieve a particular level of connectivity. The exact number of nodes required for a minimal topology depends on the specific connectivity requirements of the network. For example, if the goal is to connect two devices together, a minimal topology would consist of a single link connecting those devices. If the goal is to connect three devices, a minimal topology would consist of a triangle topology, in which each device is connected to the other two devices. Minimal topologies are often used in situations where resources are limited or where network complexity needs to be minimized. They can be implemented using a variety of technologies, including wired and wireless connections. While minimal topologies can be simple and easy to set up, they can also be less resilient than more complex topologies. For example, if a single link or node fails in a minimal topology, the entire network may be disrupted. As such, more complex topologies, such as mesh or tree topologies, may be more appropriate for larger or more critical networks.





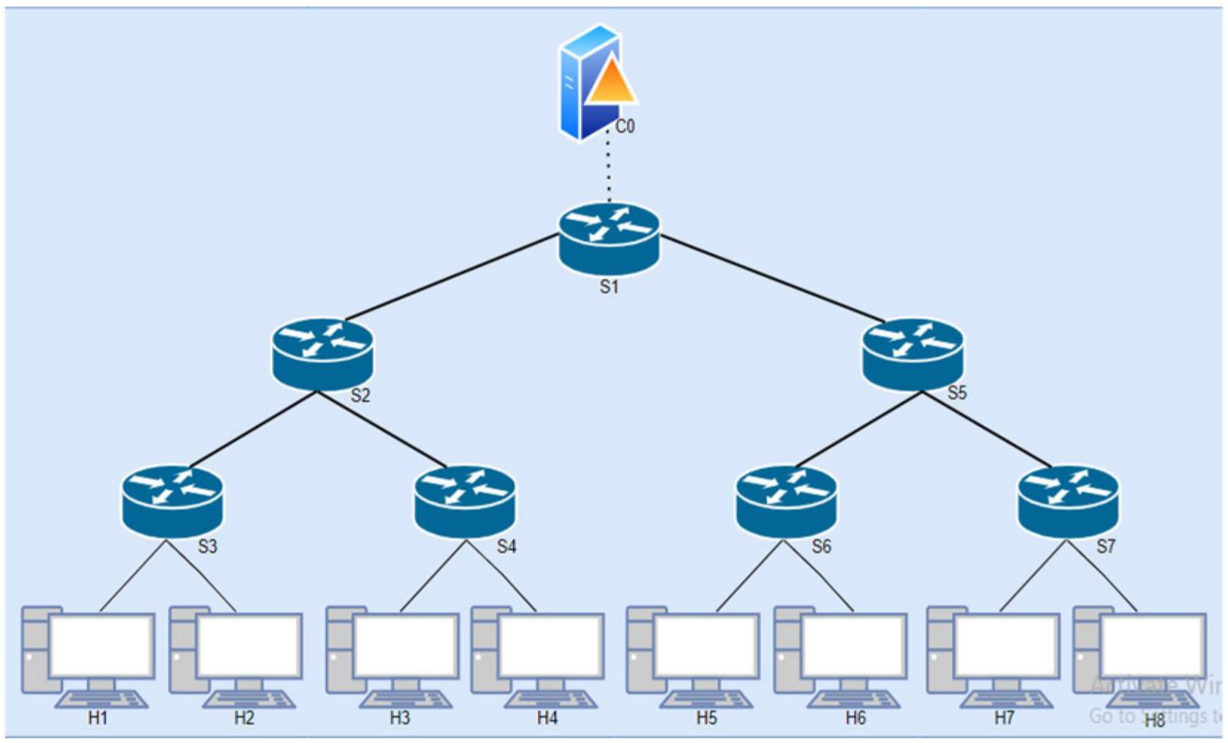
```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
```

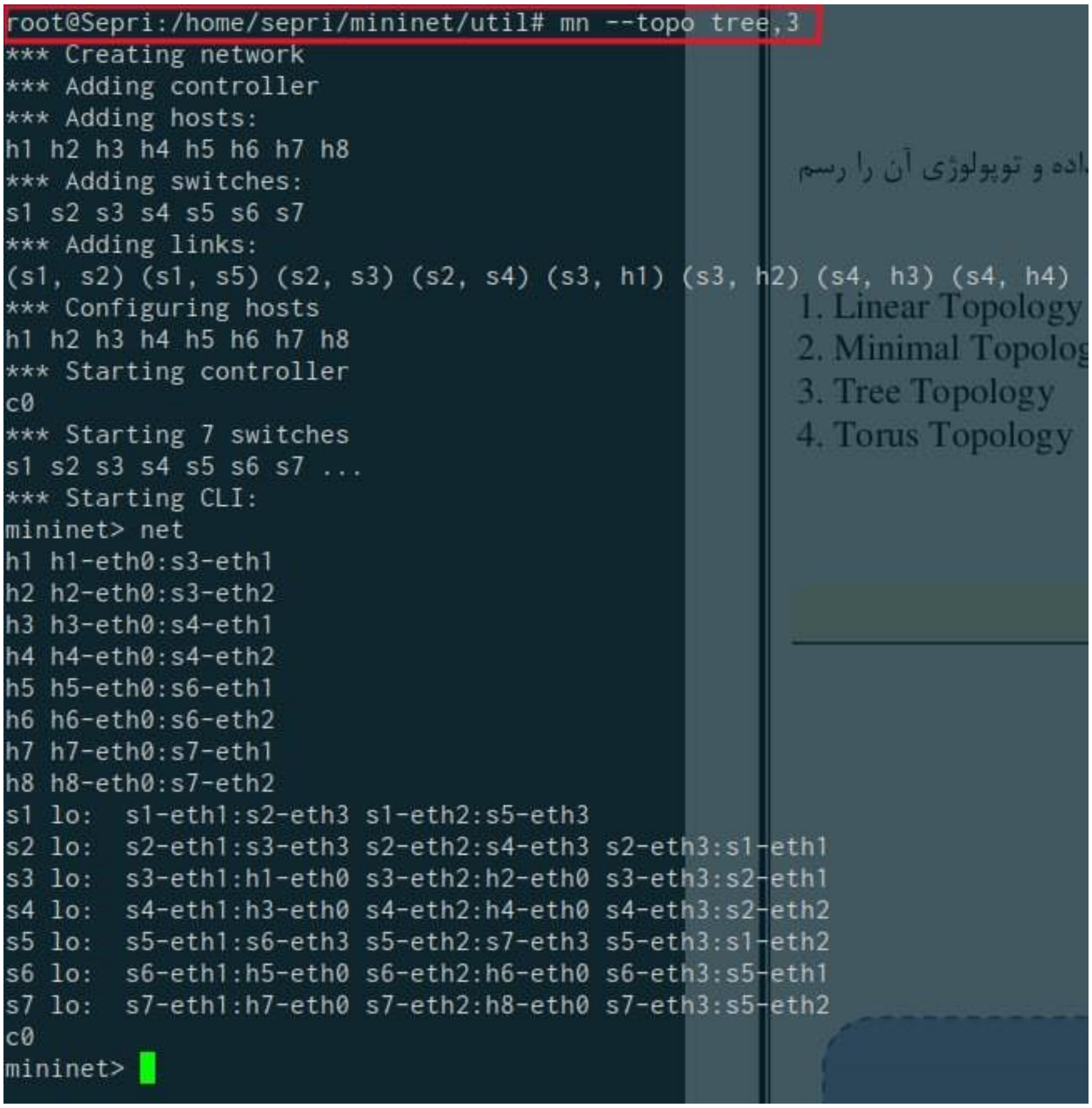


```
root@Sepri:/home/sepri/mininet/util# mn --topo minimal
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Tree Topology

A tree topology is a type of network topology in which network nodes are arranged in a hierarchical structure, resembling the shape of a tree. In a tree topology, nodes are connected to a central node, which is sometimes called the root node. The root node is connected to one or more nodes, which are in turn connected to additional nodes, creating a branching structure. The tree topology is commonly used in wide area networks (WANs) and local area networks (LANs), and is often used in corporate and enterprise networks. In a LAN environment, the root node is typically a switch or a router, and the end nodes are usually computers, printers, or other network devices. One of the advantages of the tree topology is that it is scalable and can support a large number of nodes. Additionally, it provides a clear hierarchical structure, which makes it easy to manage and troubleshoot network issues. However, one of the disadvantages of the tree topology is that it can be less resilient than other topologies, such as mesh or ring topologies, because a failure of the root node or a single link can cause the entire network to fail. Furthermore, the tree topology can become inefficient as the network grows larger, leading to longer paths between nodes and increased latency. Overall, the tree topology is a commonly used and versatile network topology that provides a balance between scalability and manageability.





```
root@Sepri:/home/sepri/mininet/util# mn --topo tree,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s5-eth3
s2 lo:  s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h1-eth0 s3-eth2:h2-eth0 s3-eth3:s2-eth1
s4 lo:  s4-eth1:h3-eth0 s4-eth2:h4-eth0 s4-eth3:s2-eth2
s5 lo:  s5-eth1:s6-eth3 s5-eth2:s7-eth3 s5-eth3:s1-eth2
s6 lo:  s6-eth1:h5-eth0 s6-eth2:h6-eth0 s6-eth3:s5-eth1
s7 lo:  s7-eth1:h7-eth0 s7-eth2:h8-eth0 s7-eth3:s5-eth2
c0
mininet> 
```

Finding problems with using tcpdump

tcpdump is a powerful tool for network troubleshooting and analysis. In Mininet, we can use tcpdump to capture network traffic and diagnose problems in network.

steps to use tcpdump in Mininet:

1. Open Mininet network topology and start the simulation.

2. Open a terminal window on the host machine.

3. Type the following command to list the available network interfaces: $ifconfig

4. Identify the interface that corresponds to the Mininet network, typically it will have an IP address in the 10.0.0.0/8 range.

5. Use the following command to capture network traffic on the identified interface: $ sudo tcpdump -i <interface> -n -s0 -w <file.pcap>

where:

- <interface> is the name of the network interface you identified in step 4.
- -n option is used to display IP addresses instead of hostnames.
- -s0 option is used to capture the entire packet without truncation.
- <file.pcap> is the name of the file where the captured packets will be saved.

6.Reproduce the problem you are experiencing in your Mininet network.

7. Stop the capture by pressing CTRL-C in the terminal window.

8. Open the captured file in a packet analysis tool like Wireshark to analyze the network traffic and diagnose the problem.

Using tcpdump in Mininet can be very helpful in troubleshooting network problems, as it allows you to capture and analyze network traffic in real-time.