ELEC-E7910 - VIRTUALIZATION LAB

Note: This lab can be done at your laptops and then you can book a 1 hour lab time slot to show a demo. Or else you can also email me regarding your queries or book a (at max) 3-hour lab and do it in the lab. If you are doing it in lab then it is preferred to do this lab on your own system so that you can continue it later if not completed during lab hours.

Also, Marks for each part are mentioned with round brackets and an underline. Grading scale can be found at the end of this manual.

Table Of Content

1. Introduction

This laboratory assignment introduces you to basic network virtualization techniques and gives you a practical understanding on how to apply them. Laboratory is divided into preliminary and practical exercises that complement each other. In total, you will have to complete successfully 4 exercises to pass this lab. Each exercise is introduced at the beginning of each section.

2. Preliminary exercises (25 Marks)

Linux namespaces and open vSwitch (OVS).

(1) Explain briefly the following concepts (3)
   a) Linux namespaces
   b) Open vSwitch (OVS)
   c) Network tap
   d) Virtual Ethernet (Veth)
   e) Broadcast domain
   f) VLAN

(2) Give an example where (a) and (b) techniques can be used? (1)
(3) Open vSwitch is an alternative to Linux native bridges. Compare these two in terms of features. (2)
(4) Can Linux tap or veth interfaces be used to attach network namespaces to OVS or Linux bridges? (1)
(5) List other alternatives to Open vSwitch. (1)
(6) OVS comes with a set of different management utilities. List these utilities and briefly explain each. (2)
(7) List drawbacks of flat network design (1)
(8) Give a couple VLAN use cases? (1)
(9) How traffic can be mapped to VLANs. Based on what? (1)

Mininet emulation

(1) What is Network Emulation and how does it differ from simulation? (2)
(2) What is Mininet? (1)
(3) List programs that are similar to Mininet. (1)

Flow-Based Forwarding & Traffic shaping

(1) Explain briefly the following concepts (3)
   a) Flow
   b) Policy-based routing
   c) Openflow
(2) Explain the difference between flow and packet based forwarding. (2)

(3) Linux Traffic Control (tc) is a powerful tool that allows an administrator to queue packets differently based on attributes of the packets. Explain the following components of Linux Traffic Control (3)

a) qdisc
b) class
c) filter

3. Exercises

3.1. Linux Network Namespaces and Open vSwitch (OVS)

3.1.1. Introduction

In this exercise you will practice how to deploy virtual network environment on top of Linux operating system. The virtual network (fig2), will mimic a very simple physical LAN (fig1) that most of us use in daily basis. At the very simplest it consists of the following components.

1. **A personal computers or workstations**
2. **A physical network interface card (NIC)**, that connects computer to the network
3. **Layer 2 switch** that provides connectivity between network interfaces at the data link layer.
4. **External network** which provides an Internet access through gateway router.

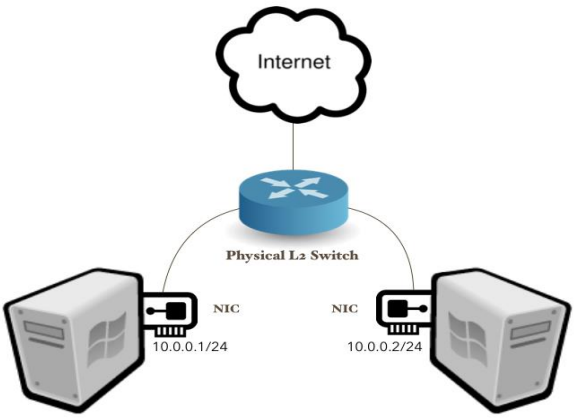


Figure 1. Physical LAN network

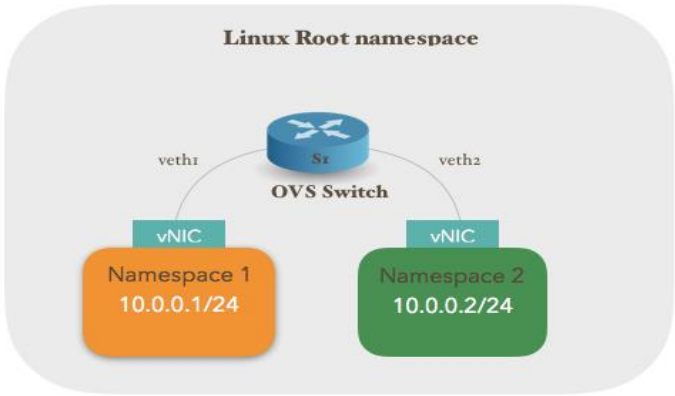


Figure 2. Virtual LAN network

To deploy physical LAN, user must power up L2 switch, connect computers to switch with Ethernet cables and finally configure proper IP addresses on NICs. In terms of deployment, the virtualized network does not differ from its physical counterpart. Same deployment procedures have to be followed, except that physical network components are now implemented in software.

In this exercise, the virtual network will be deployed using three virtualization tools/software.

1. Namespaces
2. Open vSwitch (OVS)
3. Virtual Ethernet devices (veth).

These tools are commonly used by other programs, such as OpenStack, to implement large and complex virtual systems. At the end of this exercise, you should be fairly familiar with these tools and feel comfortable using them. As a part of this exercise, you will also practice to segment broadcast domain

into separate VLANs using port-based approach. This will be done for illustration purpose, to compare it later with the flow-based VLAN method, which will be used in the exercise 4.

3.1.2. Exercises

Tip! Before you start, take a snapshot of your virtual machine, so you can roll back to initial state in case of a problem.

1. Table 1 includes commands that will be used to set up virtual network environment. Get familiar with these commands and briefly explain what is the outcome/output of each command. (3)

COMMANDS
ip netns list
ip netns add <NS_NAME>
ip netns exec <NS_NAME> [command]
ip netns exec <NS_NAME> bash
ovs-vsctl show
ovs-vsctl add-br <BR_NAME>
ovs-vsctl add-port <BR_NAME> port
ip link show
ip link add int1 type veth peer name int2
ip link set

Table.1

Tip! Most of these commands require root privileges to execute. You may either prefix these commands with sudo or enter root mode by executing sudo su command.

2. Write down link layer data (ip link) of root namespace. You will need it in task 5. (1)

3. Create two network namespaces “h1” and “h2”. In physical LAN these namespaces would represent networking environment inside personal computers/workstations. Check link layer data of h1. Could you connect this namespace? Explain. (1)

4. Create Open vSwitch and name it “S1”. Check and write down OVS database content. (1)

5. Create two veth pipes with the following interfaces: (1)
   - Pipe one: “h1-veth0” <—> “s1-veth1”
   - Pipe two: “h2-veth0” <—> “s1-veth2”.

   What would these “pipes” represent in physical LAN?
   Write down link layer data of h1, h2, and root namespace.

Tip! Root namespace can be recognized from physical interfaces (e.g. eth0) that are included in link layer output.

6. Attach h1-veth0 interface to h1 namespace and h2-veth0 interface to h2 namespace and configure them with the following IP addresses. (2)

   h1: 10.0.0.1/24
   h2: 10.0.0.2/24

   Provide the commands that you used.

7. Check h1, h2, and root namespace link layer data and compare the results to output obtained in task 5. (1)

8. Enter h1 bash shell and ping h2 (10.0.0.2) namespace. Does it work? If not, why? (1)

9. How could you fix the problem? (1)

10. Fix the problem and verify connectivity between namespaces. Provide commands. (1)

11. Check the OVS database and link layer information in root namespace. How does it differ from task 4 and task 7 outputs? (1)

Tip! It’s good idea to take another snapshot in this at this point.

12. Expand your virtual network to match the topology shown in Figure 3. Complete the following steps and provide the commands that you used. (6)
    A. Create two new namespaces: SERVER and BLACKHAT
    B. Create new Open vSwitch and name it “S2”
    C. Create and use two veth links to connect SERVER to s2 and BLACKHAT to s1
    D. Create another veth link and interconnect switches.
    E. Assign same overlapping IP address (10.0.0.100/24) for BLACKHAT and SERVER.
    F. Set all created links up.

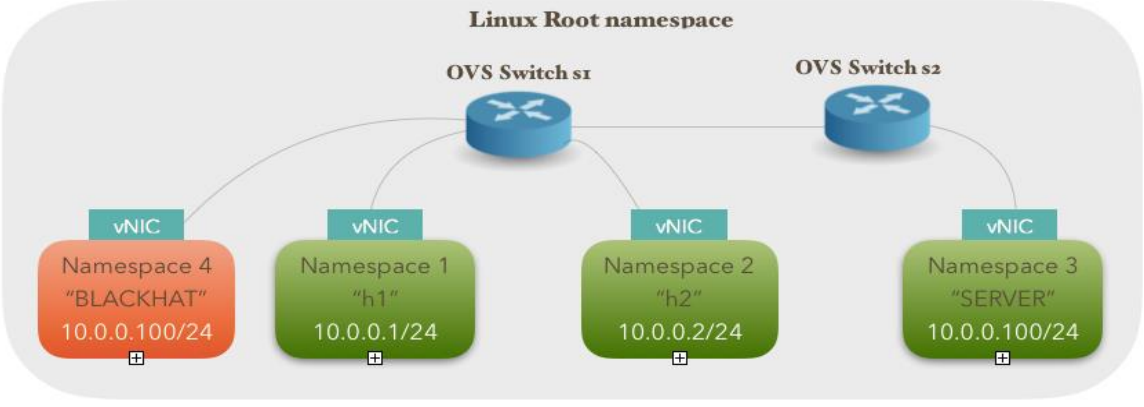


Fig.3 Expanded virtual network

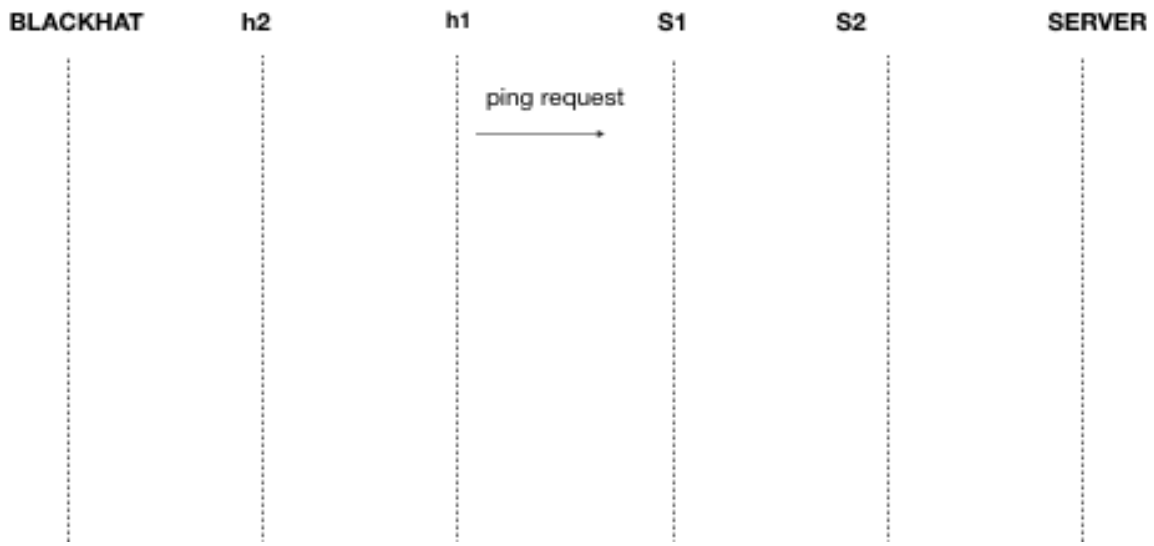
13. Check MAC addresses of each network interface and write them down. (1)

14. Open terminal and execute sudo wireshark & command. This will open wireshark in separate window. Remember to use sudo prefix, otherwise wireshark won't list all interfaces. Start capturing packets from S1/S2 network interfaces. Keep wireshark running on the background.

15. Enter h1 namespace and check its ARP cache with arp -a command. If it's not empty, delete the arp entries with the following command: arp -d x.x.x.x , where x.x.x.x represents IP address. (1)

16. Ping SERVER from h1. Did you receive reply from SERVER? (1)

17. Stop Wireshark capturing and sketch the packet flow in the below graph? (2)



18. Analyze the wireshark dump and explain what happened. What is the problem? (1)

19. OVS supports port based VLAN tagging. Use this feature to fix the problem. Provide the commands that you used. (1)

20. Repeat tasks from 14 to 17. Analyze the wireshark dump and explain what happened. (3)

21. Test connectivity between all hosts and write the results down in table X. Remember to check sources from arp table. (1)

	h1	h2	BLACKHAT	SERVER
h1				
h2				
BLACKHAT				
SERVER				

22. Can h1 communicate with h2? If not, what would it require to fix that? (1)

3.2. Mininet emulation

3.2.1. Introduction

Mininet is a network emulator, which provides an easy way to deploy virtual networks, using virtual hosts, switches, controllers, and links. The objective of this exercise is to introduce you to the mininet platform and its key features. During this exercise you will practice how to deploy standard and custom network topologies, using built-in and customized scripts.

3.2.2. Exercises

Tip! Roll back the initial VM snapshot, so you can start from the clean table.

1. Before you start emulation, check the following information.
    a. OVS database configuration: ovs-vsctl show
    b. System link layer information: ip link
    c. List of namespaces: ip netns

2. Start mininet by executing sudo mn command in shell prompt. This will start mininet with a default topology. "**mininet>**" indicates that you are inside mininet CLI where you can execute mininet commands.

3. Table 2 includes most popular mininet commands. Briefly explain what each command does when executed in mininet CLI? (2)

COMMANDS
nodes
net
dump
pingall
iperfudp
sh

Table 2

4. Sketch the default network topology, deployed with "**sudo mn**". (1)

5. Repeat exercise 1 from mininet console while running default topology and explain how mininet has emulated network. (2)

6. Why the list of namespaces is empty? How mininet emulates hosts? (1)

7. Shut down mininet by executing exit command. If mininet does not function properly, you can always clean it up by executing mn -c command in shell prompt after shutdown.

8. Mininet comes with built-in scripts that can deploy standard topologies such as single switch, linear, and tree topology. The desired topologies and options can be specified and passed to mininet through parameters during startup. Use following parameters and to deploy virtual networks. Sketch their topologies and describe what does the arguments "tree,3" and "fanout=2" do? (2)

    d) sudo mn --topo linear,3
    e) sudo mn --topo tree,3
    f) sudo mn --topo tree,3,fanout=2

9. There is a set of other options that you can specify during the startup. Below are some example deployments. Try to execute them and describe what each option has done. (4)

    a) sudo mn --topo single,3 —mac --switch ovsk
    b) sudo mn --topo single,3 --controller remote -x
    c) sudo mn --topo tree,3 --mac —arp
    d) sudo mn --topo linear —controller=remote,ip=127.0.0.1,port=6633

10. Deploy default network topology and try to ping google.com from h1. Does it work? If not, how can you fix that during mininet startup? (1)

11. Deploy the following topology sudo mn --controller remote and ping h2 from h1? Does it work? If not, explain why? (1)

12. Mininet provides a simple Python API, which can be used to deploy custom topologies. Custom topologies are written in python file, where you can specify hosts, switches, links, and controllers and pass it to mininet through startup parameters. You can find a simple topo-2sw-2host.py example script in /home/ubuntu/mininet/custom/ directory. Copy the example scripts to a new file and modify it to represent same topology as shown in figure 3. Provide code of modified script. (2)

```
from mininet.topo import Topo

class MyTopo( Topo ):
   "Simple topology example."

   def __init__( self ):
      "Create custom topo."

      # Initialize topology
      Topo.__init__( self )

      # Add hosts and switches
      leftHost = self.addHost( 'h1' )
      rightHost = self.addHost( 'h2' )
      leftSwitch = self.addSwitch( 's3' )
      rightSwitch = self.addSwitch( 's4' )

      # Add links
      self.addLink( leftHost, leftSwitch )
      self.addLink( leftSwitch, rightSwitch )
      self.addLink( rightSwitch, rightHost )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Deploy modified topology and verify connectivity. To start up mininet with the custom topology, use the following command.

sudo mn --custom /home/ubuntu/mininet/custom/<script name>.py --topo mytopo

3.3. Flow-Based Forwarding and Linux Traffic Control

3.3.1. Introduction

Mininet is often used as a tool to deploy and run experiences with Software-Defined Networking (SDN). In general, SDN represents a new network architecture where network is controlled through a centralized controller, which interacts and programs the network via open interfaces. One of the most popular southbound interfaces in SDN is Openflow, which defines both the communications protocol between controller and data-plane, as well as the flow table structure of the data plane.

In this exercise you will learn how to manually configure flow tables using Ovs-ofctl management utility. This exercise is divided in two parts.

1. In the first part, you will deploy the custom topology that you created in exercise 3.2 task 12 and configure it using flow-based VLANs.
2. In the second part, you will deploy much larger topology that is shown in figure 4 and configure it to fulfill network requirements. In order to build more realistic network environment, you will use Linux Traffic Control (tc) and Network Emulation (netem) tools to add network impairments, such as latency, jitter, loss, duplication, corruption and reordering to the network.

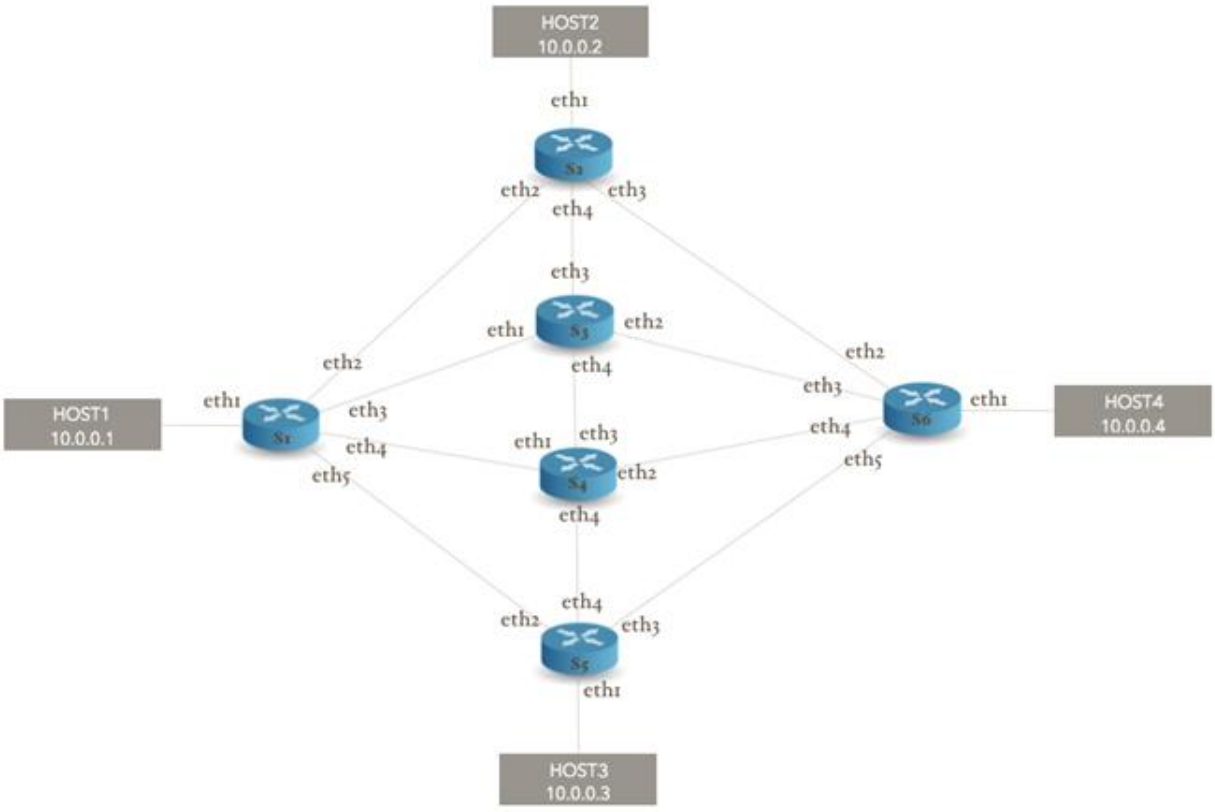


Figure 4.

3.3.2. Exercises

PART1

1. Table 3 includes functions that will be used in this exercise. Find the corresponding ovs-ofctl command for each function. (3)

FUNCTION
Show OpenFlow features and port descriptions of switch <s1>
Show all flow entries of <s1>
Print port statistics of <s1>
Add a static <flow> to <s1>
Add a static <flow> with priority 100 to <s1>
Delete particular <flow> from <s1>
Delete all flows from <s1>
Add a flow rule to <s1> that matches source input port and performs an <action>
Add a flow rule to <s1> that matches an Ethernet destination address <mac> and performs an <action>
Add a flow rule to <s1> that matches an ip source <ip1> + destination <ip2> address and performs an <action>
Add a flow rule to <s1> that matches source input <port1>, encapsulates packet to VLAN = 100 frame and sends to output <port2>

Table 3.

2. Deploy the custom mininet topology that you created in exercise 3 task 12. Use the following parameters.
   sudo mn --custom /home/ubuntu/mininet/custom/<script name>.py --mac --topo mytopo

3. Dump and explain flow entries of s1 and s2. (1)

4. Deploy same topology, but now use the following parameters. (1)

   sudo mn --custom /home/ubuntu/mininet/custom/<your_topo>.py --topo mytopo --mac --controller remote

   Dump the flow entries again. What do you notice?

5. Add the flow entries to s1 and s2 that makes them act as a normal L2 switch. Use priority 100. What command did you use? (1)

6. Change SERVER and BLACKHAT IP addresses to 10.0.0.100/24. (1)

7. Configure similar VLAN isolation as in exercise 1 task 19, but now use flow-based VLAN approach. Complete the following steps: (4)

   A. Add a flow rule to <s1> that tags all frames coming from port1 with VLAN=20 and forwards them to S2. Use priority 500.
   B. Add a flow rule to <s2> that matches VLAN=20 frames coming from port 2 and forwards them to port 1. Use priority 500.
   C. Add a flow rule to <s2> that matches VLAN=20 frames coming from port 1 and forwards them to port 1. Use priority 500.
   D. Add a flow rule to <s1> that matches VLAN=20 frames coming from port 2 and forwards them to port 1. Use priority 500.

   Provide the commands that you used and flow-dumps of s1/s2.
   Can h1 ping SERVER? If not, how can you fix it ?

   Tip! The following sources are helpful:
   http://www.pica8.com/document/v2.3/html/ovs-commands-reference/

8. Test connectivity between all hosts (pingall) and write the results down in table 4. (1)

	h1	h2	BLACKHAT	SERVER
h1				
h2				
BLACKHAT				
SERVER				

Table 4.

9. How could you configure flow tables differently to gain table 5 connectivity? Provide solution and commands. (2)

	h1	**h2**	**BLACKHAT**	**SERVER**
h1	ok	ok	-	ok
h2	ok	ok	ok	-
BLACKHAT	-	ok	ok	-
SERVER	ok	-	-	ok

Table 5.

PART2

1. Use the same method as in exercise 2 task 12 and create new script that will deploy topology shown in figure 4. Provide the python code. (3)

2. Deploy created network topology. Use the following parameters.
   sudo mn --custom /home/ubuntu/mininet/custom/<your_topo_2>.py --topo mytopo --mac --controller remote

3. Use Linux Traffic Control and netem tools to emulate network delay, throughput and loss as defined.
   Tip! Use hierarchical token bucket (HTB). (10)

   A. Limit the output rate to 100Mbps from S1 to S2 and add 100ms delay.
   B. Limit the output rate to 50Mbps from S2 to S6.
   C. Limit the output rate to 50Mbps from S2 to S3 and and add 50ms delay.
   D. Limit the output rate to 10Mbps from S1 to S4 and add 100ms delay
   E. Limit the output rate to 10Mbps from S4 to S5
   F. Limit the output rate to 10Mbps from S5 to S6, add 50ms delay, and 10% packet loss
   G. Emulate 100ms delay from S1 to S3
   H. Emulate 50ms delay from S3 to S4
   I. Emulate 150ms delay from S4 to S6.
   J. Emulate 10% packet loss from S6 to S4

4. Now, use the table 3 commands and configure the network according to following policies.
   Traffic between hosts must always take a shortest path while fulfilling the requirements below.
   Use priority 100 in all flow entries. (8)

   A. Arp messages must always follow the shortest path.
      In h1-h4 case choose s1-s3-s6 path
      In h2-h3 case choose s2-s6-s5 path

   B. Traffic between host1 and host2 must take a path with minimum delay
   C. Traffic from host1 to host4 must take a path with the maximum throughput. However, traffic from host4 to host1 must follow S6->S4->S1 path.
   D. Traffic between host1 and host3 must follow the shortest path.
   E. Traffic between host2 and host3 must take a path with the maximum throughput.
   F. Traffic between host2 and host4 must take a path with a minimum packet loss.
   G. Traffic between host3 and host4 must take a path with minimum delay and packet loss.

5. Measure max throughout, average delay and packet loss between hosts and write down the results in table 6. At least 30packets must be exchanged when measuring delay and packet loss. (2)

	h1	h2	h3	h4
h1				
h2				
h3				
h4				

Table 6.

Grading:

Total Lab = 110

- Prelab = 25 marks
- Lab = 85 Marks

Number	Grade
Less than 55	0
55 to 60	1
60 to 70	2
70 to 80	3
80 to 90	4
90 to 110	5