

# LAB TWO – SINGLE SEGMENT

In this lab, you will learn how to use Wireshark, a software package to monitor link activity. You will also learn about ARP and how to configure the PCs. This lab uses the network configuration shown in Figure 2.2 for all parts.

Connect all four VMs to a single Ethernet segment via a single hub as shown in Figure 2.1. Configure the IP addresses for the PCs as shown in Table 2.1.

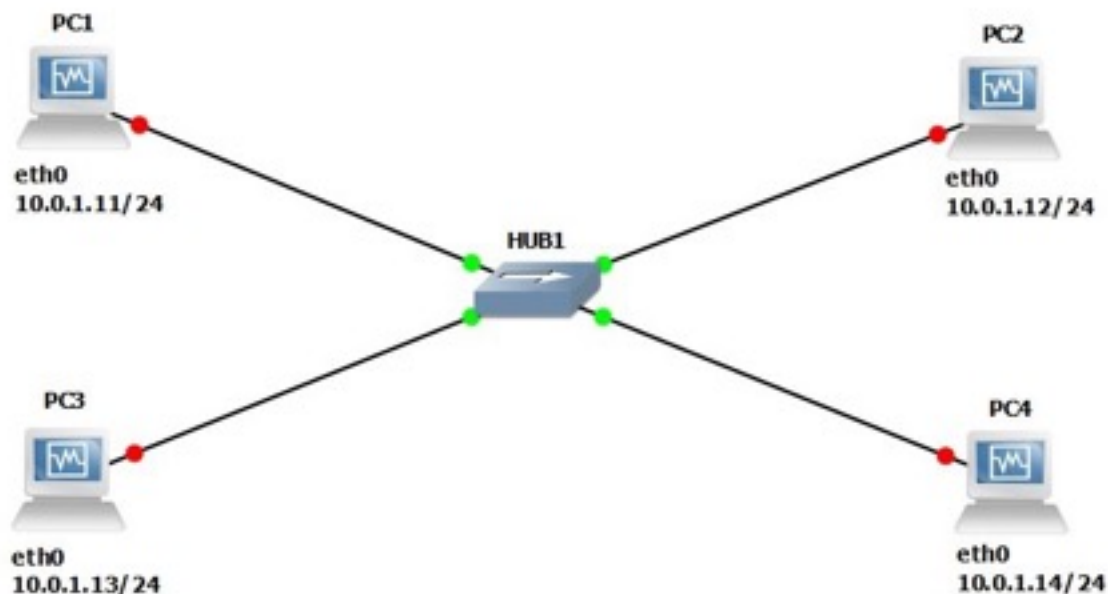


Figure 2.1 - Network Configuration for Lab 2.

VMS	IP Addresses of Ethernet Interface eth0
PC1	10.0.1.11 / 24
PC2	10.0.1.12 / 24
PC3	10.0.1.13 / 24
PC4	10.0.1.14 / 24

Table 2.1 - IP Addresses for Lab 2



**Tip:** Recall the following command to help you set up the IP addresses.

```
ifconfig interface_name A.B.C.D &X
```

## PART 1. Starting WIRESHARK

During the GNS3 installation process in Lab 1, you will automatically have installed Wireshark. For windows you do not need to download Wireshark. It comes automatically with the GNS3 installation. For Macs you are required to download it from the web ([here](#)). For Macs you also are required to download [X11](#). Wireshark will not work on a Mac without the X11 environment.

Make sure Wireshark is properly working by opening the Wireshark application on your computer. If Wireshark is not in your application folder, please download the appropriate version for your OS from the Internet.

- You have to set up four VMs on VirtualBox Manager if you haven't done so already. To do this, simply refer to Lab 1 (PART 3, Ex 3(A) Step 7-8). Make sure to have four VMs as shown in Figure 2.2.

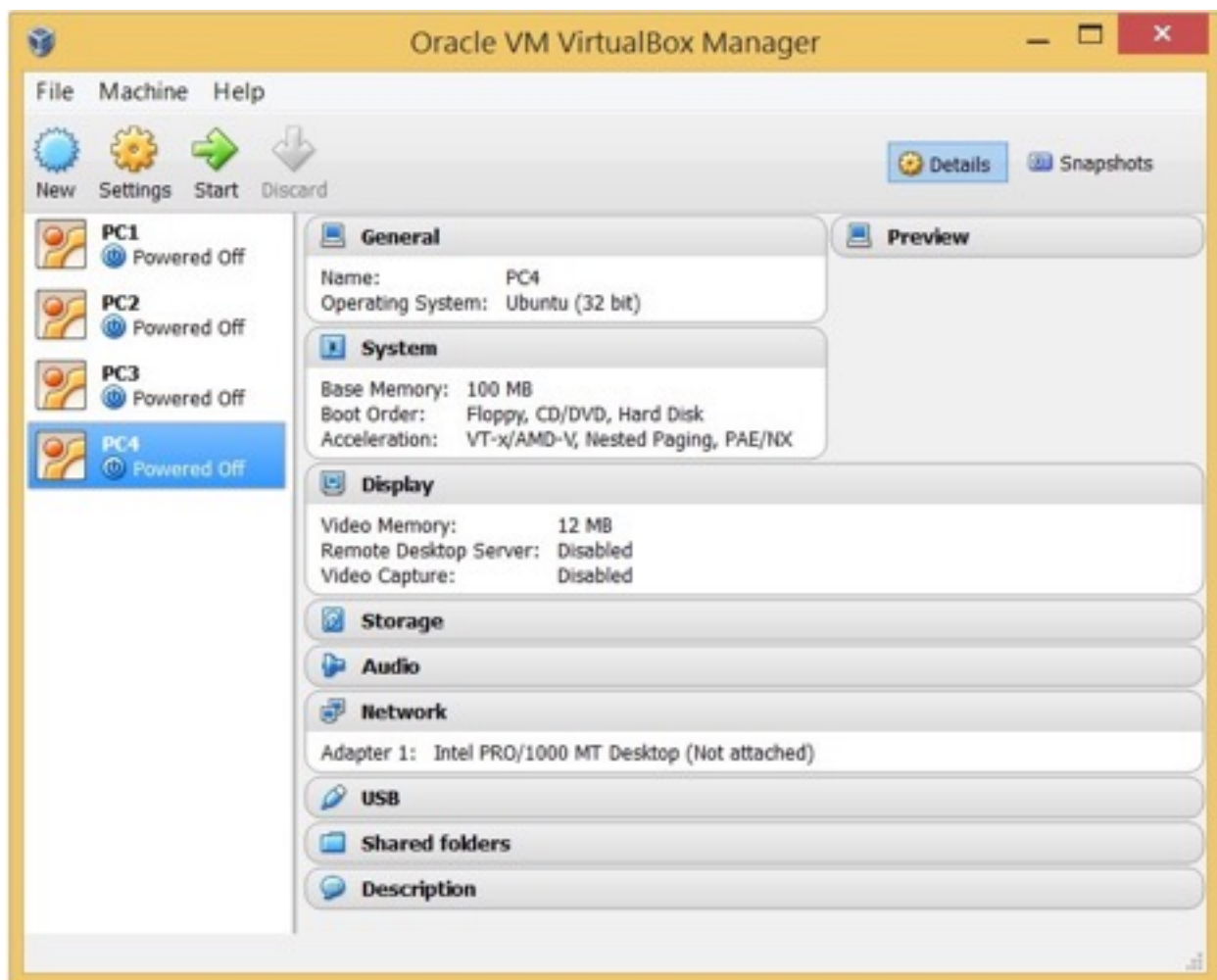


Figure 2.2 – VMs on VirtualBox Manager.

## PART 2. Capturing Traffic using WIRESHARK

In this part of the lab, you experiment with filter expressions within the Wireshark application. The filtering capabilities and options of Wireshark are described under the help tab in Wireshark.

### Exercise 1. Display filters and traffic capture with Wireshark

This exercise is mostly about the traffic capture process using Wireshark. You are introduced to the notion of capture filters.

1. Configure the network topology as shown in Figure 2.1 and configure the VMs' IP addresses with the values shown in Table 2.1.
2. Mouse right click on the link that connects PC1 and the Ethernet Hub and select "Start capture". **NOTE: Please choose the HUB side of the link for capturing, not the PC side.** Choose option Wireshark. It will initiate Wireshark and capture traffic on the link.
1. Please make sure that the Packet Capture settings are set to Wireshark Live Traffic Capture from Packet capture preferences section.

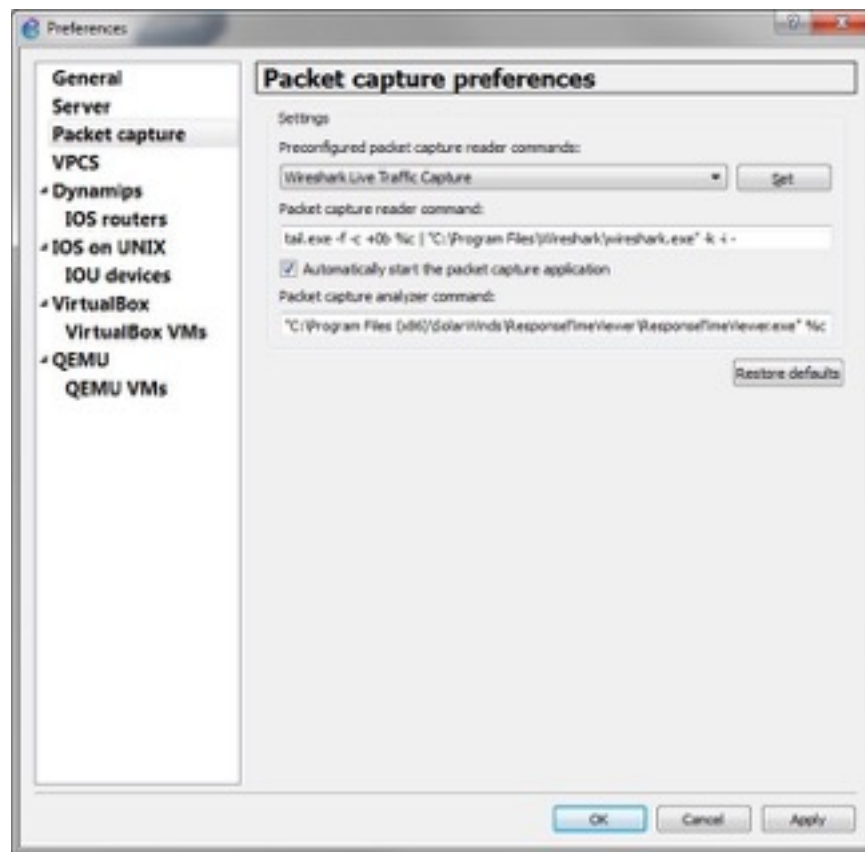


Figure 2.3 Packet capture Preferences

2. For Mac users the Wireshark does not open the capture file automatically, you need to open the \$HOME/GNS3/project folder. Then find the corresponding .pcap file in the *captures* folder. The initial size of this file will be 0. Once you start sending traffic the file will grow in size. You can open the file in Wireshark and it will keep refreshing the capture window while traffic is being captured.
3. **Setting a display filter:** From the command “Display Filters...” under the “Analyze” menu, you can set a display filter so that only the traffic that matches the filter is displayed. Set a filter so that all packets that contain the IP address of PC2 (10.0.1.12) are captured. Press “Enter/Return” after typing the filter.

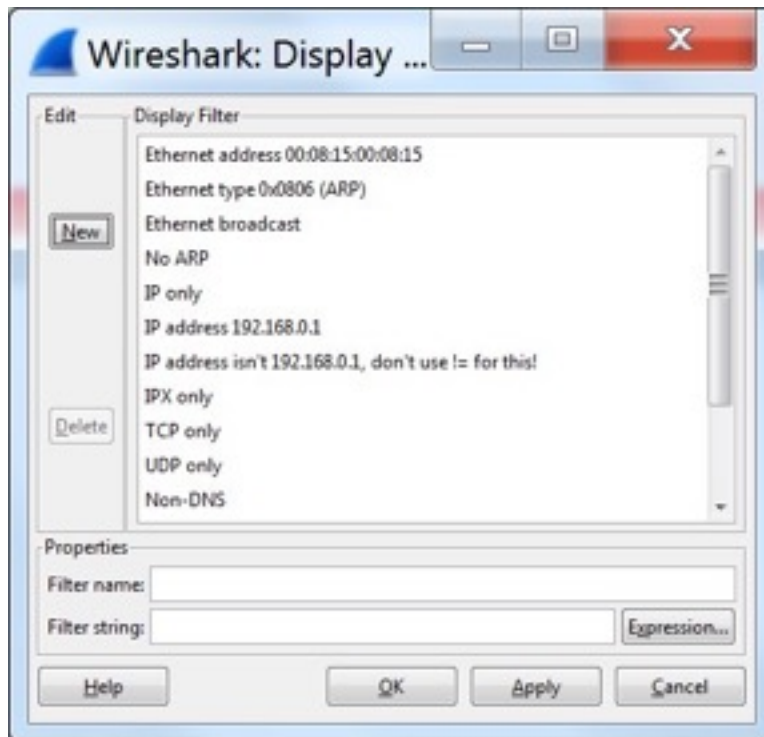


Figure 2.4 Display Filters command

4. You can also set a display filter by typing the desired display filter in the “Filter” box, which is found in the Wireshark main window as shown in Figure 2.5. Click the Clear button next to the filter box to clear any existing filter.



Figure 2.5 Filter box for setting display filters

5. In the terminal window of PC1, issue a ping command to PC2:

```
PC1% ping 10.0.1.12-c 2
```

6. Stop the capture process.
7. **Saving captured traffic:** This is done by selecting the “Print” command in the “File” menu. Be sure Output to file option is checked. (Unless asked to save the details of captured frames, selecting the summary option is usually sufficient, uncheck Packet details unless you are required to display that.) See figure 2.6 below.

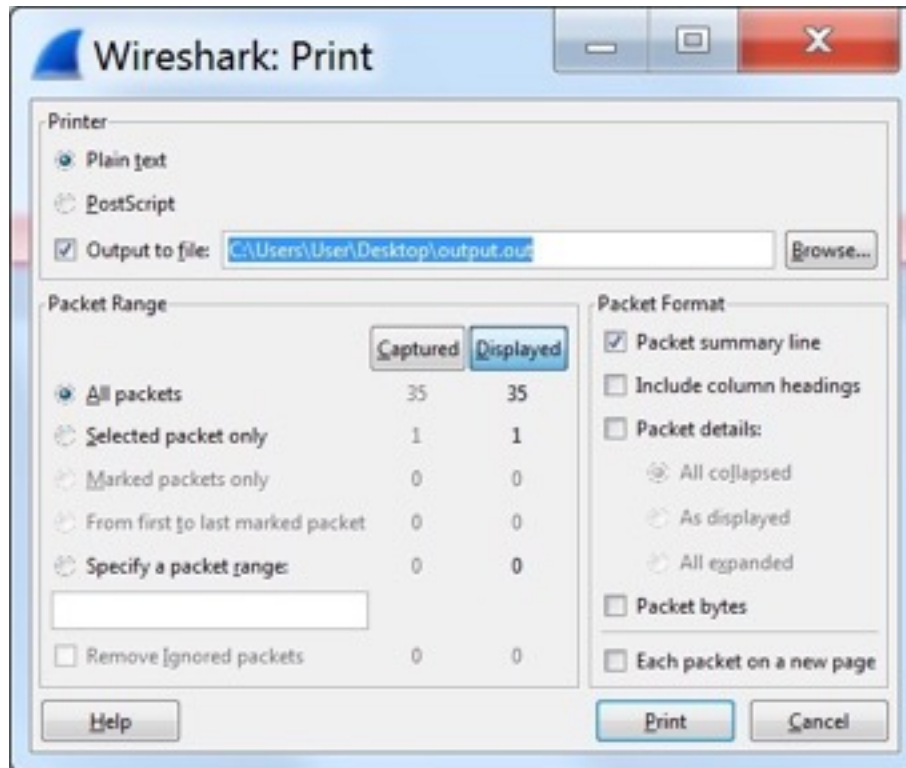


Figure 2.6 Print/Save Captured Traffic

## PART 3. Address Resolution Protocol (ARP)

This part of the lab explores the operation of the Address Resolution Protocol (ARP) that resolves a MAC address for a given IP address. The lab exercises use the Linux command `arp`, for displaying and manipulating the contents of the ARP cache. The ARP cache is a table that holds entries of the form <IP address, MAC address>. The most common uses of the `arp` command are listed below.

### COMMON USES OF THE ARP COMMAND

`arp -a`

Displays the content of the ARP cache.

`arp -d IPAddress`

Deletes the entry with the IP address **IPAddress**

`arp -s IPAddressMACAddress`

Adds a static entry to the ARP cache that is never overwritten by network events. The MAC address is entered as 6 hexadecimal bytes separated by colons.

**Example:** `arp -s 10.0.1.1200:02:2D:0D:68:C1`

### TIME-OUTS IN THE ARP CACHE

The entries in an ARP cache have a limited lifetime. Entries are deleted unless they are refreshed. The typical lifetime of an ARP entry is 2 minutes, but much longer lifetimes (up to 20 minutes) have been observed.



### FLUSHING THE ARP CACHE

You also can clear the ARP cache with the following command

```
ip -s -s neigh flush all
```

### REFRESHING THE ARP CACHE

In Linux you will observe that a host occasionally sends out ARP requests to interfaces that are already in the ARP cache.

**Example:** Suppose that a host with IP address 10.0.1.22 has an ARP cache entry:

10.0.1.11 is at 08:00:27:53:63:1a

Then, this host occasionally sends a unicast ARP Request to MAC 08:00:27:53:63:1a of the form:

who has 10.0.1.11? Tell 10.0.1.22

to verify that the IP address 10.0.1.11 is still present before deleting the entry from the ARP cache.

## Exercise 3(A). A simple experiment with ARP

1. On PC1, view the ARP cache with `arp -a` and delete all entries with the `-d` option.
2. Start Wireshark on PC1-Hub1 link with a capture filter set to the IP address of PC2.
3. Issue a ping command from PC1 to PC2:

```
PC1% ping 10.0.1.12 -c 2
```

Observe the ARP packets in the Wireshark window. Explore the MAC addresses in the Ethernet headers of the captured packets.

Direct your attention to the following fields:

- The destination MAC address of the ARP Request packets.
  - The Type Field in the Ethernet headers of ARP packets and ICMP messages.
4. View the ARP cache again with the command `arp -a`. Note that ARP cache entries can get refreshed/deleted fairly quickly (~2 minutes).

```
PC1% arp - a
```

5. Save the results of Wireshark to a text file, using the “Packet details” option in “Print”.

### Lab Questions

- What is the destination MAC address of an ARP Request packet?  
**Broadcast address (ff:ff:ff:ff:ff:ff)**
- What are the different Type Field values in the Ethernet headers that you observed? **IP and ARP**
- Use the captured data to analyze the process in which ARP acquires the MAC address for IP address 10.0.1.12.
  - **PC1 first sends request to DEST: Broadcast address (ff:ff:ff:ff:ff:ff)**
  - **PC2 then sends its MAC address to DEST: PC1 (08:00:27:d2:fe:d0)**
  - **After the pinging is done, PC2 then sends another ARP request to DEST: PC1 (08:00:27:d2:fe:d0) to refresh its ARP cache (see note on pg6)**
  - **PC1 then sends its MAC address to DEST: PC2 (08:00:27:a9:00:08)**

### Exercise 3(B). Matching IP addresses and MAC addresses

Identify the MAC addresses of all the interfaces connected to the network, and enter them in Table 2.2. You can obtain the MAC addresses from the ARP cache of a PC by issuing a `ping` command from that host to every other host on the network. Alternatively, you can obtain the MAC addresses from the output of the `ifconfig` command in the console window of each PC.

VMS	IP Address of eth0	MAC address of eth0
PC1	10.0.1.11 / 24	08:00:27:d2:fe:d0
PC2	10.0.1.12 / 24	08:00:27:a9:00:08

PC3	10.0.1.13 / 24	08:00:27:a4:1a:79
PC4	10.0.1.14 / 24	08:00:27:dc:29:f0

Table 2.2. IP and MAC addresses.

### Exercise 3(C). ARP requests for a non-existing address

Observe what happens when an ARP Request is issued for an IP address that does not exist.

1. Start Wireshark on PC1-Hub1 link with a capture filter set to capture packets that contain the IP address of PC1.
2. Issue a ping command from PC1 to 10.0.1.22. (Note that this address does not exist in this network.)

```
PC1% ping 10.0.1.22 -c 10
```

3. Save the captured output.

### Lab Questions

- Using the saved output, describe the time interval between each ARP Request packet issued by PC1. **One second.** Observe the method used by ARP to determine the time between retransmissions of an unsuccessful ARP Request. **If it does not receive a reply in one second, it knows the ARP Request was unsuccessful.**
- Why are ARP Request packets not transmitted (i.e. not encapsulated) as IP packets? **Because an ARP Request will never be transmitted outside of a single network. ARP is a data link layer protocol and not a network layer protocol.**

## PART 4. The NETSTAT Command

The Linux command `netstat` displays information on the network configuration and activity of a Linux system, including network connections, routing tables, interface statistics, and multicast memberships. The following exercise explores how to use the `netstat` command to extract different types of information about the network configuration of a host. This list shows four important uses of the `netstat` command.

```
netstat -i
```

Displays a table with statistics of the currently configured network interfaces.

```
netstat -rn
```

Displays the kernel routing table. The `-n` option forces `netstat` to print the IP addresses. Without this option, `netstat` attempts to display the host names.



```
netstat -an  
netstat -tan  
netstat -uan
```

Displays the active network connections. The `-a` option displays all active network connections, the `-ta` option displays only information on TCP connections, and the `-tu` option displays only information on UDP traffic. Omitting the `-n` option prints host names, instead of IP addresses.

```
netstat -s
```

Displays summary statistics for each protocol that is currently running on the host.

## Exercise 4. Using netstat commands

1. Display information on the network interfaces by typing

```
PC1% netstat -in
```

2. Display the content of the IP routing table by typing

```
PC1% netstat -rn
```

3. Display information on TCP and UDP ports that are currently in use by typing

```
PC1% netstat -a
```

4. Display the statistics of various networking protocols by typing

```
PC1% netstat -s
```

### NOTE

The values of the statistics displayed by some of the `netstat` commands are reset each time a host is rebooted. Therefore, if you are doing this exercise immediately after rebooting the VM, the output of `netstat` may not be very useful.

## Lab Questions

Using the `netstat` output, answer the following questions.

- What are the network interfaces of PC1 and what are the MTU (Maximum Transmission Unit) values of the interfaces? (Type `netstat -in`)

Eth0 has MTU 1500

Lo has MTU 65536

- How many IP datagrams, ICMP messages, UDP datagrams, and TCP segments has PC1 transmitted and received since it was last rebooted. (type `netstat -s > output.txt`)

IP: 230 transmitted, 170 received

ICMP: 182 transmitted, 122 received

UDP: 48 transmitted, 48 received

TCP: 0 transmitted, 0 received

- Explain the role of interface `lo`, the loopback interface. In the `netstat -in` output, why are the values of RX-OK (packets received) and TX-OK (packets transmitted) different for interface `eth0` but identical for interface `lo`?

The loopback interface (`lo`) is a virtual network interface implemented in software only and is not connected to any hardware. It is fully integrated into the computer system's internal network infrastructure. Any traffic that a computer program sends to the loopback interface is immediately received on the same interface. Therefore, the RX-OK and TX-OK are the same for interface `lo`.

For `eth0`, RX-OK and TX-OK are measuring completely different things. One is the packets that are received. The other one is the packets that are sent.

## PART 5. Configuring IP Interfaces in LINUX

The `ifconfig` command is used to configure parameters of network interfaces, including enabling and disabling interfaces and setting the IP address. The `ifconfig` command is usually run when a system boots up. In this case, the parameters are read from a file. Once the Linux system is running, the `ifconfig` command can be used to modify the network configuration parameters. This list shows how `ifconfig` is used to query the status of network interfaces.

`ifconfig`

Displays the configuration parameters of all active interfaces.

`ifconfig interface`

Displays the configuration parameters of a single interface. For example, `ifconfig eth0` displays information on interface `eth0`.

`ifconfig eth0 down`

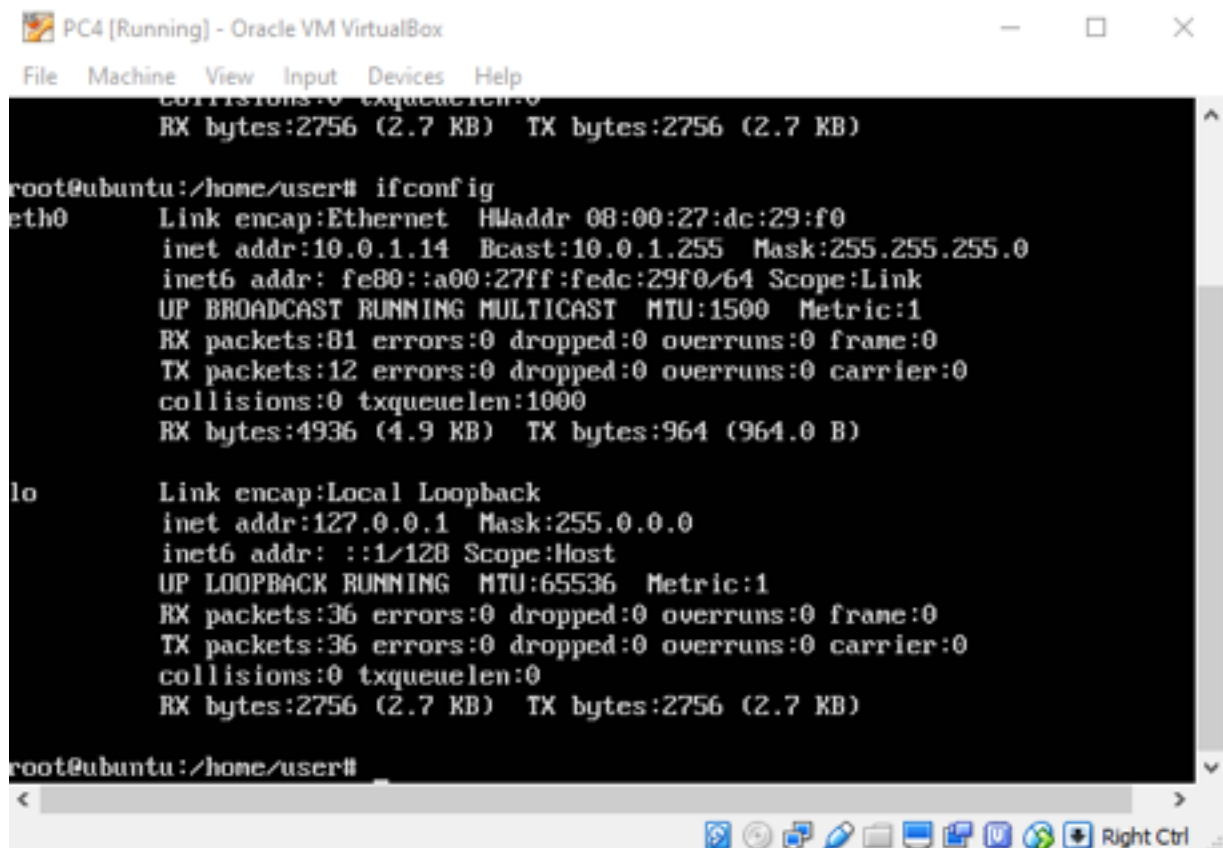
Disables the `eth0` interface. No traffic is sent or received on a disabled interface.

```
ifconfig interface up  
    Enables an interface.
```

```
ifconfig eth0 10.0.1.8 netmask 255.255.255.0 broadcast  
10.0.1.255 Assigns interface eth0 the IP address  
10.0.1.8/24 and a broadcast address of 10.0.1.255.
```

## Exercise 5. Changing the IP address of an interface

1. On PC4, run `ifconfig` and screenshot the output.



The screenshot shows a terminal window titled "PC4 [Running] - Oracle VM VirtualBox". The terminal output displays the configuration for two network interfaces: `eth0` and `lo`. The `eth0` interface is configured with IP address `10.0.1.14`, netmask `255.255.255.0`, and broadcast address `10.0.1.255`. The `lo` interface is the local loopback interface with IP address `127.0.0.1` and netmask `255.0.0.0`. The terminal also shows statistics for RX and TX bytes and packets for both interfaces.

```
PC4 [Running] - Oracle VM VirtualBox  
File Machine View Input Devices Help  
RX bytes:2756 (2.7 KB) TX bytes:2756 (2.7 KB)  
root@ubuntu:/home/user# ifconfig  
eth0      Link encap:Ethernet  HWaddr 08:00:27:dc:29:f0  
          inet addr:10.0.1.14  Bcast:10.0.1.255  Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fedc:29f0/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:81 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:4936 (4.9 KB)  TX bytes:964 (964.0 B)  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:2756 (2.7 KB)  TX bytes:2756 (2.7 KB)  
  
root@ubuntu:/home/user#
```

2. Change the IP address of interface `eth0` of PC4 to `10.0.1.11/24`.
3. Run `ifconfig` again and screenshot the output.

```
PC4 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

root@ubuntu:/home/user# ifconfig eth0 10.0.1.11/24
root@ubuntu:/home/user# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:dc:29:f0
          inet addr:10.0.1.11  Bcast:10.0.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fedc:29f0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:81 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4936 (4.9 KB)  TX bytes:964 (964.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2756 (2.7 KB)  TX bytes:2756 (2.7 KB)

root@ubuntu:/home/user#
```

**Tip:** If you are not able to screenshot all the output on the screen (too much data), you should use the command `ifconfig interface` for each interface so that you can capture each one separately.

## Lab Questions

- Explain the fields of the `ifconfig` output.

`Ifconfig` gives info on all network devices. For each device, for example `eth0`, it gives the link type, the hardware address (MAC address), IP address, netmask and the packets sending and receiving information.

## PART 6. DUPLICATE IP Addresses

In this part of the lab, you observe the effects of having more than one host with the same (duplicate) IP address in a network. After completing Exercise 5, the IP addresses of the Ethernet interfaces on the four PCs are as shown in table 2.3 below. Note that PC1 and P4 are assigned the same IP address.

VMS	IP Address of eth0
-----	--------------------

PC1	10.0.1.11 / 24
PC2	10.0.1.12 / 24
PC3	10.0.1.13 / 24
PC4	10.0.1.11 / 24

Table 2.3. IP addresses.

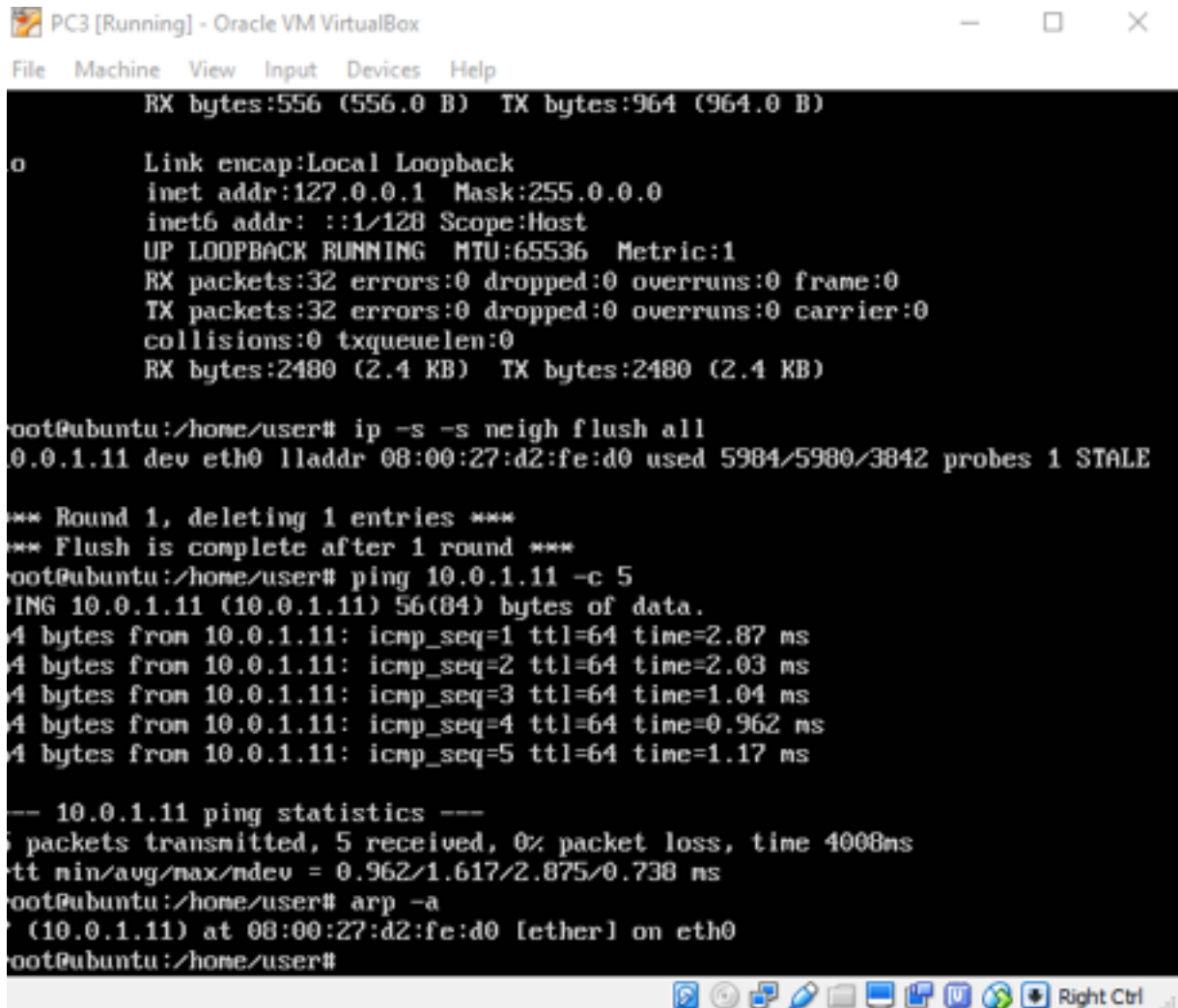
## Exercise 6

1. Delete all entries in the ARP cache on all PCs.
2. Run Wireshark on PC3-Hub1 link and capture the network traffic to and from the duplicate IP address 10.0.1.11.
3. From PC3, issue a ping command to the duplicate IP address, 10.0.1.11, by typing

```
PC3% ping 10.0.1.11 -c 5
```

4. Stop Wireshark, save all ARP packets and screenshot the ARP cache of PC3 using the arp -a command:

```
PC3% arp - a
```



```
PC3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

RX bytes:556 (556.0 B) TX bytes:964 (964.0 B)

o Link encap:Local Loopback
  inet addr:127.0.0.1 Mask:255.0.0.0
  inet6 addr: ::1/128 Scope:Host
  UP LOOPBACK RUNNING MTU:65536 Metric:1
  RX packets:32 errors:0 dropped:0 overruns:0 frame:0
  TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:0
  RX bytes:2480 (2.4 KB) TX bytes:2480 (2.4 KB)

root@ubuntu:/home/user# ip -s -s neigh flush all
10.0.1.11 dev eth0 lladdr 08:00:27:d2:fe:d0 used 5984/5980/3842 probes 1 STALE

*** Round 1, deleting 1 entries ***
*** Flush is complete after 1 round ***
root@ubuntu:/home/user# ping 10.0.1.11 -c 5
PING 10.0.1.11 (10.0.1.11) 56(84) bytes of data.
4 bytes from 10.0.1.11: icmp_seq=1 ttl=64 time=2.87 ms
4 bytes from 10.0.1.11: icmp_seq=2 ttl=64 time=2.03 ms
4 bytes from 10.0.1.11: icmp_seq=3 ttl=64 time=1.04 ms
4 bytes from 10.0.1.11: icmp_seq=4 ttl=64 time=0.962 ms
4 bytes from 10.0.1.11: icmp_seq=5 ttl=64 time=1.17 ms

-- 10.0.1.11 ping statistics --
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/ndev = 0.962/1.617/2.875/0.738 ms
root@ubuntu:/home/user# arp -a
(10.0.1.11) at 08:00:27:d2:fe:d0 [ether] on eth0
root@ubuntu:/home/user#
```

5. When you are done with the exercise, reset the IP address of PC4 to its original value as given in Table 2.1.

## Lab Questions

- Explain how the ping packets were issued by the hosts with duplicate addresses. **PC3 only pinged PC1, which was the first to respond to the ARP Request sent to both PC1 and PC3. It looks like PC3 rejected the connection between PC4 as it is not in the ARP cache.**
- Did the ping command result in error messages? **The ARP response from PC4 gave a warning DUPLICATE IP ADDRESS**
- How can duplicate IP addresses be used to compromise the data security? **If I know the IP of a host on a network I can change my IP to it and pretend I am the host. Any packets (say, packets that included user/password) that were sent are now compromised.**
- Give an example. Use the ARP cache and the captured packets to support your explanation. **See above.**

## PART 7. Changing NETMASKS

In this part of the lab, you test the effects of changing the netmask of a network configuration. In the table below, two hosts (PC2 and PC4) have been assigned different network prefixes.

### Exercise 7. Setting different IP address masks

1. Setup the interfaces of the hosts as follows:

VMS	IP Address of eth0	Network Mask
PC1	10.0.1.100 / 24	255.255.255.0
PC2	10.0.1.101 / 28	255.255.255.240
PC3	10.0.1.120 / 24	255.255.255.0
PC4	10.0.1.121 / 28	255.255.255.240

Table 2.4 IP addresses for Part 7.

2. Run Wireshark on PC1-Hub1 link and capture the packets for the following scenarios

a. From PC1 ping PC3: **PC1%** ping 10.0.1.120 -c 1

Success, a normal ping procedure (ARP broadcast, ARP Response, ping request, ping reply, ARP Refresh Request, ARP Response)

b. From PC1 ping PC2: **PC1%** ping 10.0.1.101 -c 1

Success, a normal ping procedure (ARP broadcast, ARP Response, ping request, ping reply, ARP Refresh Request, ARP Response)

c. From PC1 ping PC4: **PC1%** ping 10.0.1.121 -c 1

Unsuccessful, 3 ARP packets from PC1 to ff:ff:ff:ff:ff:ff  
Error: Destination host unreachable

d. From PC4 ping PC1: **PC4%** ping 10.0.1.100 -c 1

Connect: network unreachable

No wireshark output

e. From PC2 ping PC4: **PC2% ping 10.0.1.121 -c 1**  
**Connect: network unreachable**  
**No wireshark output**

f. From PC2 ping PC3: **PC2% ping 10.0.1.120 -c 1**  
**Connect: network unreachable**  
**no wireshark output**

3. Save the Wireshark output to a text file (using the “Packet Summary” option from “Print”), and save the output of the ping commands. Note that not all of the above scenarios are successful. Save all the output including any error messages.
4. When you are done with the exercise, reset the interfaces to their original values as given in Table 2.1. (Note that /24 corresponds to network mask 255.255.255.0. and /28 to network mask 255.255.255.240.)

### Lab Questions

- Use your output data and ping results to explain what happened in each of the ping commands. **Explained above**
- Which ping operations were successful and which were unsuccessful? Why? **Explained above.**

Of the six total ping commands, the only two that worked were from PC1 to PC2 and PC1 to PC3. Only four bits are reserved for the computer on PC2 and PC4 since their netmasks are 255.255.255.240, and that is inadequate for representing the values 120 and 121. Therefore, PC2 and PC4 believe IP addresses at 10.0.1.16 and greater belong to a different network. Their routing tables do not provide information to reach that network, and so they determine the network is not reachable.

## PART 8. Static Mapping of IP Addresses and Host Names

Since it is easier to memorize names than IP addresses, there are mechanisms to associate a symbolic name, called *hostname*, with an IP address. The term hostname is ambiguous on multi-homed systems, since a system can have one hostname for each network interface.

On the Internet, the resolution between hostnames and IP addresses is generally done by the Domain Name System (DNS), which is the topic of Lab 8. This experiment illustrates another, simpler method to map IP addresses and domain names using the host file `/etc/hosts`.

Before DNS became available, the `/etc/hosts` file was the only method to resolve hostnames in the Internet. All hosts on the Internet had to occasionally synchronize with the content of other `/etc/hosts` files.

### Exercise 8. Associating names with IP addresses



In this exercise, you manipulate the static mapping of hostnames and IP addresses using the `/etc/hosts` file.

1. On PC1, inspect the content of file `/etc/hosts` with `vi`.

```
PC1% vi /etc/hosts
```

Linux has a wide variety of editors that can be used to modify text files. The native text editor for Ubuntu is `vi`. The UNIX `vi` editor has two modes of operation, **command** and **insert modes**. The command mode takes actions such as saving and quitting from a file, and the insert mode permits the insertion of characters. To get in the insert mode, press `i` and enter with the characters in the line, and to exit from the insert mode and enter in the command mode, press `esc` as long as the text editor is on insert mode.

#### Quitting from and saving a file

In command mode, type `:q!` to quit from a file and not to save any edits. Type in command mode `:wq` to save a file and exit from the text editor.

2. On PC1, issue a `ping` command to PC2.

```
PC1% ping 10.0.1.12-c 5
```

3. Repeat Step 2, but use symbolic names instead of IP addresses (e.g., PC2 instead of `10.0.1.12`). You should see that the symbolic name is unreachable at this point.
4. On PC1, edit the file `/etc/hosts` and associate hostnames with the IP addresses of the 4 PC's and save the changes. Use the names PC1, PC2, etc., as used throughout this lab to refer to the PCs. For example, PC2 information should be inserted as shown in Figure 2.6.

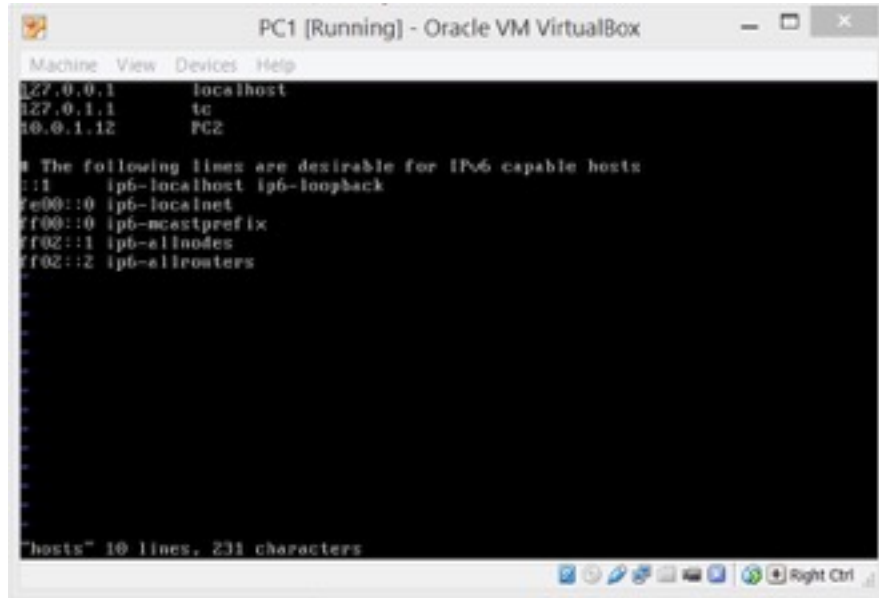


Figure 2.7 How to insert PC2 information on /etc/hosts from PC1

5. You should now be able to ping directly using PC2, PC3, and PC4, as in

```
PC1% ping PC2 -c 5
PC1% ping PC3 -c 5
PC1% ping PC4 -c 5
```

6. Reset the /etc/hosts file to its original state. That is, remove the changes you have made in this exercise, and save the file.

#### Deleting characters on vi

On command mode, go to the character you want to delete and press **d** and then the right arrow **→** or the left arrow **←**. To delete the whole line press **d** two times.