

**Based on**  
**Mastering Networks - An Internet Lab Manual**  
**by Jörg Liebeherr and Magda Al Zarki**

*Adapted for*  
*'Labo Computernetwerken'*  
*by Johan Bergs, Nicolas Letor, Michael Voorhaen and Kurt Smolderen*

Completed by  
Josse Coen      Armin Halilovic      Jonas Vanden Branden      Group 2

March 10, 2016



## Lab 2

# Single-Segment IP Networks

What you will learn in this lab:

- How to capture and filter network traffic
- How to configure a network interface for IP networking
- How to access IP statistics and settings with the netstat command
- How ARP works
- How hackers snoop passwords from the network

## 2.1 Prelab 2

### New commands

Read the manual pages of the following commands at <http://manpages.ubuntu.com/> for the operating system version “trusty 14.04 LTS”:

- arp
- ifconfig
- netstat
- tcpdump

### IP Addresses

Read the article “Understanding IP Addressing: Everything You Ever Wanted To Know” by Chuck Semeria, at <http://holdenweb.com/static/docs/3comip.pdf>

### Wireshark capture and display filters

Go to the Wireshark website and read about capture filters and display filters for Wireshark:

- <http://wiki.wireshark.org/CaptureFilters>
- [http://www.wireshark.org/docs/wsug\\_html\\_chunked/ChWorkBuildDisplayFilterSection.html](http://www.wireshark.org/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html).

## Prelab Questions

### Question 1)

Write the syntax for an `ifconfig` command that sets the IP address of the interface `eth0` to 128.143.2.3/16 with broadcast address 128.143.255.255.

`ifconfig eth0 128.143.2.3 netmask 255.255.0.0 broadcast 128.143.255.255`

### Question 2)

Write the syntax of a `tcpdump` command that captures packets containing IP datagrams with source or destination IP address equal to 10.0.1.12.

`tcpdump host 10.0.1.12`

### Question 3)

Write the syntax of a `tcpdump` command that captures packets containing ICMP messages with source or destination IP address equal to 10.0.1.12.

`tcpdump icmp and host 10.0.1.12`

### Question 4)

Write the syntax of a `tcpdump` command that captures packets containing IP datagrams between two hosts with IP addresses 10.0.1.11 and 10.0.1.12, both on interface `eth1`.

`tcpdump -i eth1 host 10.0.1.11 and 10.0.1.12`

### Question 5)

Write a `tcpdump` filter expression that captures packets containing TCP segments with source or destination IP address equal to 10.0.1.12.

`tcp and host 10.0.1.12`

### Question 6)

Write a `tcpdump` filter expression that, in addition to the constraints in Question 5, only captures packets using port number 23.

`tcp and host 10.0.1.12 and tcp port 23`

### Question 7)

Write the syntax for a `wireshark` command with capture filter so that all IP datagrams with source or destination IP address equal to 10.0.1.12 are recorded.

`wireshark -f "host 10.0.1.12"`

### Question 8)

Write the syntax for a Wireshark display filter that shows IP datagrams with destination IP address equal to 10.0.1.50 and frame sizes greater than 400 bytes.

`ip.dst == 10.0.1.50 and frame.len > 400`

### Question 9)

Write the syntax for a Wireshark display filter that shows packets containing ICMP messages with source or destination IP address equal to 10.0.1.12 and frame numbers between 15 and 30.

`icmp and ip.host == 10.0.1.12 and frame.number >= 15 and frame.number <= 30`

### Question 10)

Write the syntax for a Wireshark display filter that shows packets containing TCP segments with source or destination IP address equal to 10.0.1.12 and using port number 23.

`ip.host == 10.0.1.12 and tcp.port == 23`

### Question 11)

Write a Wireshark capture filter expression for Question 10.

`tcp and host 10.0.1.12 and port 23`

## 2.2 Lab 2

In Lab 2 you become acquainted with IP configuration issues on a single Ethernet segment. The lab also exposes you to advanced use of `tcpdump` and Wireshark.

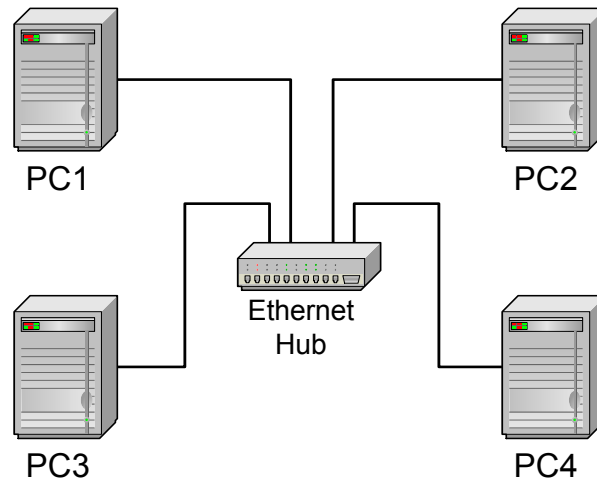


Figure 2.1: Network configuration for Lab 2.

The setup for this lab is identical as in Lab 1. All Linux PCs are connected to the same Ethernet segment by an Ethernet hub as shown in Figure 2.1. The IP addresses for the Linux PCs are configured as shown in Table 2.1 below. Configure *eth0* for each of the PCs. e.g. for PC1 use the following command.

```
PC1% ifconfig eth0 10.0.1.11 netmask 255.255.255.0 broadcast 10.0.255 up
```

As an alternative, you can use the `ip` command:

```
PC1% ip addr add 10.0.1.11/24 dev eth0
PC1% ip link set dev eth0 up
```

Linux PC	IP Addresses of Ethernet Interface eth0
PC1	10.0.1.11/24
PC2	10.0.1.12/24
PC3	10.0.1.13/24
PC3	10.0.1.14/24

Table 2.1: IPv4 addresses for Lab 2

## Part 1. Using filters in tcpdump

In the first part of the lab, you explore `tcpdump` in more detail. In particular, you learn how to write filter expressions so that `tcpdump` monitors only selected traffic flows on the network. See 2.1 for more details on the use of filters in `tcpdump`.

### Exercise 1. Writing filter expressions for tcpdump

In this exercise, you explore the use of simple filter expressions with the `tcpdump` command. Save the output for your lab report.

1. On PC1, execute a `tcpdump` command with a filter that prints all packets with PC2 as source or destination. This command is the answer to Question 2 from the Prelab. Save the output of this `tcpdump` session to a file using the `tee` or `tail` commands discussed in Lab 1.



*As in Lab 1, always use the `-n` option (i.e. `tcpdump -n`) to avoid that `tcpdump` tries to resolve hostnames.*

2. In another terminal, issue a ping command to PC2 by typing `ping -c 5 10.0.1.12` on PC1 and observe the output. Recall that the ping command to a host triggers the transmission of an ICMP Echo Request. The destination host responds with an ICMP Echo Reply message.
3. Repeat steps 1 - 2 above. In addition to the existing filter, set the filter so that only ICMP messages are captured. This command is the answer to Question 3 from the Prelab.



***Make sure to include the saved data in your lab report as they are part of your evaluation!***

## Part 2. Using filters in Wireshark

In this part of the lab, you experiment with filter expressions using the `wireshark` command. Recall that Wireshark has two types of filters: capture filters and display filters.



*There are several command line options that can be assigned when starting the `wireshark` command:*

- **Capture Filters:** A capture filter specifies the traffic to be captured by the Wireshark tool. A capture filter expression can be specified from the command line using the `-f` option or using the Wireshark GUI, under the “Capture:Start” menu. The syntax for specifying the filter expression is the same syntax as used by `tcpdump`.
- **Display Filters:** By default, Wireshark displays all captured packets. With a display filter, just those packets, which meet the requirements of the filter, are displayed. The display filter cannot be set from the command line. It must be entered in the “Filter” window at the bottom of the GUI. The syntax for setting the display filter is different from the syntax for setting a capture filter.
- **Setting an interface:** When you run Wireshark on a host with multiple network interfaces, you may specify the interface with the `-i` argument. For example, to start Wireshark to capture traffic on interface `eth1`, type

```
| wireshark -i eth1
```

If you do not specify an interface, the default is `eth0`. Alternatively, you can change the interface using the Wireshark GUI, under the “Capture:Start” menu.

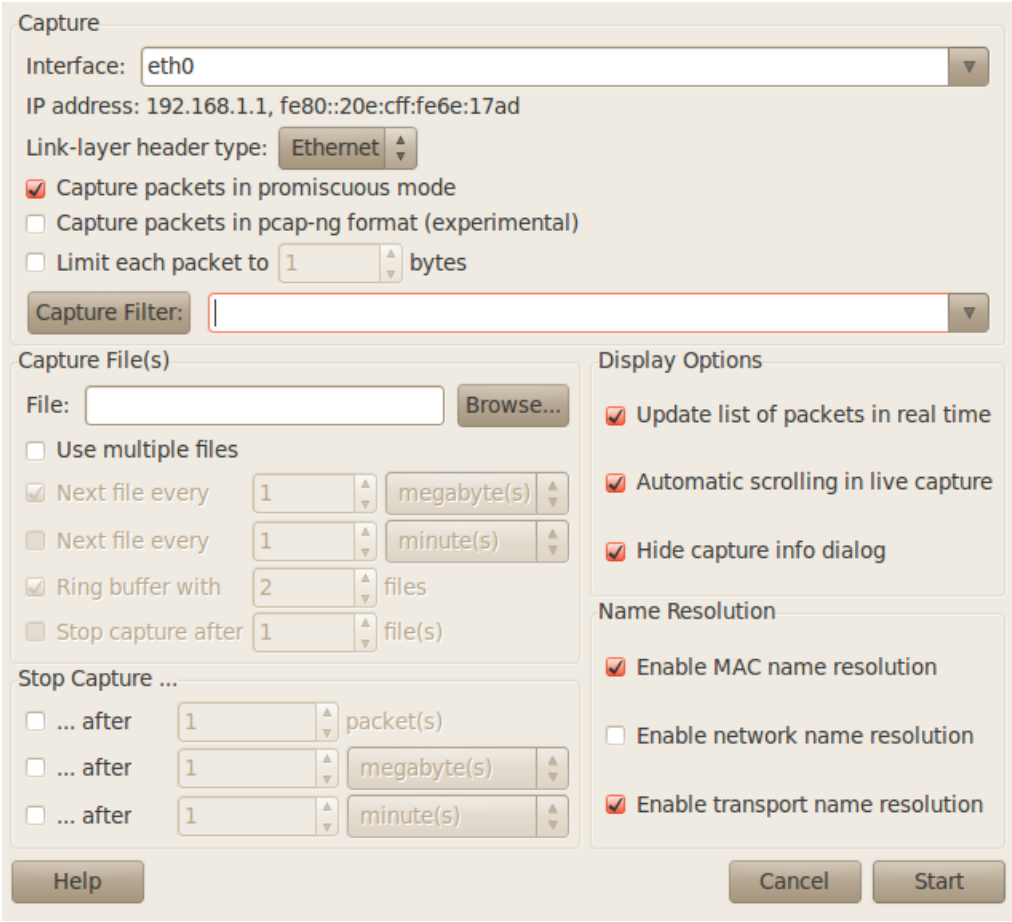
### Exercise 2-A. Setting capture filters in Wireshark

This exercise is a review of the traffic capture capabilities of Wireshark. As a new feature, you are introduced to the notion of capture filters.

1. Start Wireshark on PC1 and set the same capture preferences as in Lab 1 and as shown again in Figure 2.2 for your convenience. You should always set these same preferences for all your experiments.



**Selecting capture preferences in wireshark**



- Select *eth0* in “Interface”.
- Select “Capture packets in promiscuous mode”.
- Select “Update list of packets in real time”.
- Select “Automatic scrolling in live capture”.
- Unselect “Enable MAC name resolution”.
- Unselect “Enable network name resolution”.
- Unselect “Enable transport name resolution”.

Figure 2.2: General capture settings for Wireshark

2. Setting a capture filter: In the window “Capture Preferences”, set a filter so that all packets that contain the IP address of PC2 are recorded. The filter is set in the “Filter” box under “Capture Preferences” (see Figure 2.2). The required filter expression is the answer to Question 7 from the Prelab.
3. Start the capture by clicking “OK” in the “Capture Preferences” window.
4. In another terminal window of PC1, issue a `ping` command to PC2

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

5. Stop the capture process of Wireshark.
6. Save the results of the capture. This is done by selecting “Print” in the “File” menu as described in Lab 1. (As instructed in Lab 1, unless asked to save the details of captured frames, selecting the summary option is usually sufficient.)



***Make sure to include the saved data in your lab report as they are part of your evaluation!***

### Exercise 2-B. Working with display filters

Next you set display filters, which allow you to select a subset of the captured data for display in the main window of Wireshark.

1. In the Wireshark main window on PC1 from Exercise 2-A, set the display options as listed below. You can find the display options in the “Capture Options” window (select “Start” in the “Capture” menu)(see Figure 2.2):
  - Select “Update list of packets in real time”.
  - Select “Automatic Scrolling in live capture”.
  - Unselect “Enable MAC name resolution”.
  - Unselect “Enable network name resolution”.
  - Unselect “Enable transport name resolution”.
2. Setting a display filter: Type the desired display filter in the field next to the “Filter” box, which is located at the bottom of the Wireshark main window, as shown in Figure 2.3. Click the “Reset” button next to the “Filter” box to clear any existing filter:



Figure 2.3: Filter box for setting display filters.

Enter a display filter so that all IP datagrams with destination IP address 10.0.1.12 are shown. Refer to Question 8 from the Prelab.

3. Observe the changes in the display panel of Wireshark. Only packets with 10.0.1.12 in the IP destination address field are now being displayed.
4. Save the displayed data, by selecting “File:Print”. Note that the “Print” command only saves packets that are currently being displayed. If a display filter is used, the saved data is limited to the packets that match the display filter.
5. Repeat the above exercise with a display filter that lists only IP datagrams with source IP address equal to 10.0.1.12. Save the results.



***Make sure to include the saved data in your lab report as they are part of your evaluation!***

**Exercise 2-C. More complex capture and display filters**

In this exercise, you learn how to use more sophisticated filters to restrict the packets being captured and displayed.

1. Start Wireshark on PC1 and start to capture traffic using the same settings as in Exercise 2-A. Do not set any capture or display filters!

2. From a new terminal on PC1, execute the `ping` command for PC2

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

3. At the same time, start a Telnet session from PC1 to PC2 in another terminal by typing

```
| PC1% telnet 10.0.1.12
```

and log in as *telecomlabo*. After you logged in successfully to PC2, logout with the command `exit`.

4. Stop the traffic capture of wireshark.
5. Apply a set of display filters to the captured traffic and save the output to a text file. Select the option "Print summary" in the "Print" window.
  - Display packets that contain ICMP messages with the IP address of PC2 either in the IP destination address or IP source address. Refer to Question 9 from the Prelab. Save the output.
  - Display packets that contain TCP traffic with the IP address of PC2 either in the IP destination address or IP source address. Refer to Question 10 from the Prelab. Save the output.
  - Display packets that, in addition to the constraints in the previous filter expression, use the port number 23. Refer to Question 10 from the Prelab. Save the output.



***Make sure to include the saved data in your lab report as they are part of your evaluation!***

### Part 3. ARP - Address Resolution Protocol

This part of the lab explores the operation of the Address Resolution Protocol (ARP) which 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 as follows:*

`arp -a`

*Displays the content of the ARP cache.*

`arp -d <IPAddress>`

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

`arp -s <IPAddress><MAC_Address>`

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

*Example: `arp -s 00:02:2D: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. You may want to verify when your Linux system does remove ARP entries automatically after a certain amount of time.*



#### **Refreshing the ARP cache:**

*In Linux, you will observe that occasionally, a host sends out ARP requests to interfaces that are already in the ARP cache. Example: Suppose that a host with IP address 10.0.1.12 has an ARP cache entry: "<10.0.1.11>is-at <00:02:83:39:2C:42>". Then, this host occasionally sends a unicast ARP Request to MAC address 00:02:83:39:2C:42 of the form "Who has 10.0.1.11? Tell 10.0.1.12" 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 with a capture filter set to the IP address of PC2.
3. Issue a ping command from PC1 to PC2:

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

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 get refreshed/ deleted fairly quickly (approx. 2 minutes).
  5. Save the results of Wireshark to a pcap dump file.

Use the saved data to answer to the following questions:

**Question 3.A.1)**

What is the destination MAC address of an ARP Request packet?

"ff:ff:ff:ff:ff:ff", this is the broadcast MAC address. (send to everyone)

**Question 3.A.2)**

What are the different values of the Type field in the Ethernet headers that you observed?

0x0806, which stands for ARP. Both ARP-request and response have ARP as type value.  
0x0800, which stands for IP. The ping request and responses both have (IP) as type.

**Question 3.A.3)**

Use the captured data to discuss the process in which ARP acquires the MAC address for IP address 10.0.1.12.

An ARP request message is broadcasted, asking for the MAC address of a node with a certain IP address X. The node which has IP address X will then respond with an ARP reply message, which contains its MAC address.

**Exercise 3-B. Matching IP addresses and MAC addresses**

Identify the MAC addresses of all interfaces connected to the network, and enter them in Table 2.2. You can obtain the MAC addresses from the ARP cache of each PC. You can fill up the ARP cache at a host, 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 -a` command explained in Part 5.

Linux PC	IP Address of eth0	MAC Address of eth0
PC1	10.0.1.11/24	
PC2	10.0.1.12/24	
PC3	10.0.1.13/24	
PC3	10.0.1.14/24	

Table 2.2: IP and MAC addresses.

**Question 3.B.1)**

Include the completed Table 2.2 in your lab report.

Linux PC	IP Address of eth0	MAC Address of eth0
PC1	10.0.1.11/24	68:05:ca:36:33:a0
PC2	10.0.1.12/24	68:05:ca:36:31:f0
PC3	10.0.1.13/24	68:05:ca:36:39:c7
PC3	10.0.1.14/24	68:05:ca:36:e1:2f

Table 2.3: solutions 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. On PC1, start Wireshark with a capture filter set to capture packets that contain the IP address of PC1:

```
PC1% wireshark -f 'host 10.0.1.11'
```

2. Establish a Telnet session from PC1 to 10.0.1.10 (Note that this address does not exist on this network)

```
PC1% telnet 10.0.1.10
```

Observe the time interval and the frequency with which PC1 transmits ARP Request packets. Repeat the experiment a number of times to discover the pattern.

3. Save the captured output.

### Question 3.C.1)

Using the saved output, describe the time interval between each ARP Request packet issued by PC1. Describe the method used by ARP to determine the time between retransmissions of an unsuccessful ARP Request. Include relevant data to support your answer.

The time interval between requests is 1 second, 6 requests are sent.

No.	Time	Source	Destination	Protocol	Length
1	0.000000000	68:05:ca:36:39:c7	ff:ff:ff:ff:ff:ff	ARP	42
3	0.998636000	68:05:ca:36:39:c7	ff:ff:ff:ff:ff:ff	ARP	42
5	1.998631000	68:05:ca:36:39:c7	ff:ff:ff:ff:ff:ff	ARP	42
7	2.998653000	68:05:ca:36:39:c7	ff:ff:ff:ff:ff:ff	ARP	42
	3.998634000	68:05:ca:36:39:c7	ff:ff:ff:ff:ff:ff	ARP	42
	4.998629000	68:05:ca:36:39:c7	ff:ff:ff:ff:ff:ff	ARP	42

### Question 3.C.2)

Why are ARP Request packets not transmitted (i.e. not encapsulated) as IP packets? Explain your answer.

ARP Request packets are sent because they are used to resolve IP-addresses to their corresponding MAC-address. IP routing does not apply here, since we are transmitting packets on a single segment network.

## 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, masquerade connections, 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.



*The most common uses of the `netstat` command are as follows:*

`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 hostnames.*

`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 `-ua` option displays only information on UDP traffic. Omitting the `-n` options prints hostnames and names of servers, instead of IP addresses and ports numbers.*

`netstat -s`

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

### Exercise 4. Basic usage of the netstat command

On PC1, try the different variations of the `netstat` command listed above and save the output to a file.

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
```



*The values of the statistics displayed by some of the `netstat` commands are reset each time a host is rebooted.*

#### Question 4.1)

Attach the saved output to your report. Using the saved output, answer the following questions.

`netstat -i:`

1	Kernel Interface table											
3	Iface	MTU	Met	RX-OK	RX-ERR	RX-DRP	RX-OVR	TX-OK	TX-ERR	TX-DRP	TX-OVR	Flg
	eth0	1500	0	52819	0	0	0	52760	0	0	0	BMRU
	lo	65536	0	956	0	0	0	956	0	0	0	LRU

netstat -rn:

2	Kernel IP routing table							
	Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
	10.0.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0

netstat -an:

1	Active Internet connections (servers and established)						
3	Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	
	tcp	0	0	*:ftp	:::	LISTEN	
	tcp	0	0	*:ssh	:::	LISTEN	
5	tcp	0	0	localhost:ipp	:::	LISTEN	
	tcp	0	0	*:telnet	:::	LISTEN	
7	tcp6	0	0	:::ssh	:::*	LISTEN	
	tcp6	0	0	ip6-localhost:ipp	:::*	LISTEN	
9	tcp6	1	0	ip6-localhost:46681	ip6-localhost:ipp	CLOSE_WAIT	
	udp	0	0	*:60385	:::		
11	udp	0	0	*:bootpc	:::		
	udp	0	0	*:ipp	:::		
13	udp	0	0	*:mdns	:::		
	udp	0	0	*:30159	:::		
15	udp6	0	0	:::48348	:::*		
	udp6	0	0	:::13209	:::*		
17	udp6	0	0	:::mdns	:::*		
19	Active UNIX domain sockets (servers and established)						
	Proto	RefCnt	Flags	Type	State	l-Node	Path
21	unix	2	[ ACC ]	STREAM	LISTENING	14898	@/tmp/.ICE-unix/2049
	unix	2	[ ACC ]	STREAM	LISTENING	13329	/var/run/acpid.socket
	unix	2	[ ACC ]	STREAM	LISTENING	10782	/var/run/sdp
23	unix	2	[ ACC ]	STREAM	LISTENING	13950	@/tmp/.X11-unix/X0
	unix	2	[ ACC ]	STREAM	LISTENING	13951	/tmp/.X11-unix/X0
25	unix	2	[ ACC ]	STREAM	LISTENING	14899	/tmp/.ICE-unix/2049
	unix	2	[ ACC ]	STREAM	LISTENING	14637	/var/run/cups/cups.sock
27	unix	2	[ ACC ]	STREAM	LISTENING	19004	/run/user/112/pulse/
	native						
	unix	2	[ ACC ]	STREAM	LISTENING	15431	/run/user/1000/keyring-
	n5bKb6/ssh						
29	unix	2	[ ACC ]	STREAM	LISTENING	15435	/run/user/1000/keyring-
	n5bKb6/pkcs11						
	unix	2	[ ACC ]	STREAM	LISTENING	15438	/run/user/1000/keyring-
	n5bKb6/gpg						
31	unix	2	[ ACC ]	STREAM	LISTENING	857	/var/run/dbus/
	system_bus_socket						
	unix	2	[ ACC ]	STREAM	LISTENING	9885	@/com/ubuntu/upstart
33	unix	2	[ ACC ]	STREAM	LISTENING	24933	@/dbus-vfs-daemon/socket
	-Ar7V7Shu						
	unix	2	[ ACC ]	STREAM	LISTENING	10861	/var/run/uml-utilities/
	uml_switch.ctl						
35	unix	2	[ ACC ]	STREAM	LISTENING	15331	@/com/ubuntu/upstart-
	session/112/2402						
	unix	2	[ ACC ]	STREAM	LISTENING	14975	/run/user/1000/pulse/
	native						
37	unix	2	[ ACC ]	SEQPACKET	LISTENING	638	/run/udev/control
	unix	2	[ ACC ]	STREAM	LISTENING	12166	/var/run/avahi-daemon/
	socket						
39	unix	2	[ ACC ]	STREAM	LISTENING	16545	@/com/ubuntu/upstart-
	session/1000/2155						



41	unix	2	[ ]	DGRAM		10862	@/%ÃŽi
	unix	2	[ ACC ]	STREAM	LISTENING	15416	@/com/ubuntu/upstart-
	session/1000/1934						
	unix	2	[ ACC ]	STREAM	LISTENING	12830	@/tmp/dbus-OAyeDpuWXb
43	unix	19	[ ]	DGRAM		10203	/dev/log
	unix	2	[ ACC ]	STREAM	LISTENING	23024	@/tmp/dbus-PGSuh9PnQT
45	unix	2	[ ACC ]	STREAM	LISTENING	14836	/run/user/1000/keyring-
	n5bKb6/control						
	unix	3	[ ]	STREAM	CONNECTED	16618	@/tmp/dbus-OAyeDpuWXb
47	unix	3	[ ]	STREAM	CONNECTED	15730	
	unix	3	[ ]	STREAM	CONNECTED	14601	/var/run/dbus/
	system_bus_socket						
49	unix	2	[ ]	DGRAM		22794	
	unix	3	[ ]	STREAM	CONNECTED	13035	
51	unix	3	[ ]	STREAM	CONNECTED	11244	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15553	
53	unix	3	[ ]	STREAM	CONNECTED	19198	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	16476	
55	unix	3	[ ]	STREAM	CONNECTED	15440	@/com/ubuntu/upstart-
	session/1000/1934						
	unix	3	[ ]	STREAM	CONNECTED	15157	/var/run/dbus/
	system_bus_socket						
57	unix	3	[ ]	STREAM	CONNECTED	10679	@/com/ubuntu/upstart
	unix	3	[ ]	STREAM	CONNECTED	16410	@/tmp/.X11-unix/X0
59	unix	3	[ ]	STREAM	CONNECTED	12962	
	unix	3	[ ]	STREAM	CONNECTED	18989	/var/run/dbus/
	system_bus_socket						
61	unix	2	[ ]	STREAM	CONNECTED	19022	
	unix	3	[ ]	STREAM	CONNECTED	10759	
63	unix	3	[ ]	STREAM	CONNECTED	13038	
	unix	3	[ ]	STREAM	CONNECTED	13033	@/tmp/dbus-OAyeDpuWXb
65	unix	3	[ ]	STREAM	CONNECTED	13000	@/tmp/.ICE-unix/2049
	unix	3	[ ]	STREAM	CONNECTED	15627	
67	unix	3	[ ]	STREAM	CONNECTED	12762	
	unix	3	[ ]	STREAM	CONNECTED	13039	@/tmp/.ICE-unix/2049
69	unix	3	[ ]	STREAM	CONNECTED	15547	
	unix	3	[ ]	STREAM	CONNECTED	12945	/var/run/dbus/
	system_bus_socket						
71	unix	2	[ ]	DGRAM		12160	
	unix	3	[ ]	STREAM	CONNECTED	16431	
73	unix	3	[ ]	STREAM	CONNECTED	20744	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	16459	
75	unix	3	[ ]	STREAM	CONNECTED	15676	/run/user/1000/pulse/
	native						
	unix	3	[ ]	STREAM	CONNECTED	20743	@/tmp/.X11-unix/X0
77	unix	3	[ ]	STREAM	CONNECTED	19261	
	unix	3	[ ]	STREAM	CONNECTED	14803	/var/run/dbus/
	system_bus_socket						
79	unix	3	[ ]	STREAM	CONNECTED	15674	
	unix	3	[ ]	STREAM	CONNECTED	12089	
81	unix	3	[ ]	STREAM	CONNECTED	13029	
	unix	3	[ ]	STREAM	CONNECTED	12855	
83	unix	3	[ ]	STREAM	CONNECTED	12854	
	unix	3	[ ]	STREAM	CONNECTED	12755	/var/run/dbus/
	system_bus_socket						
85	unix	3	[ ]	STREAM	CONNECTED	12502	/var/run/acpid.socket
	unix	3	[ ]	STREAM	CONNECTED	16605	
87	unix	3	[ ]	STREAM	CONNECTED	16539	
	unix	3	[ ]	STREAM	CONNECTED	16509	
89	unix	3	[ ]	STREAM	CONNECTED	15724	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	14870	
91	unix	3	[ ]	STREAM	CONNECTED	14361	
	unix	3	[ ]	STREAM	CONNECTED	14952	/var/run/dbus/
	system_bus_socket						
93	unix	3	[ ]	STREAM	CONNECTED	15741	
	unix	3	[ ]	STREAM	CONNECTED	15649	
95	unix	3	[ ]	STREAM	CONNECTED	15082	

97	unix	3	[ ]	STREAM	CONNECTED	16429	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	14849	/var/run/dbus/
system_bus_socket							
99	unix	3	[ ]	STREAM	CONNECTED	13047	
	unix	3	[ ]	STREAM	CONNECTED	16512	
101	unix	3	[ ]	STREAM	CONNECTED	15545	
	unix	3	[ ]	STREAM	CONNECTED	15651	
103	unix	3	[ ]	STREAM	CONNECTED	15546	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15516	@/tmp/.X11-unix/X0
105	unix	3	[ ]	STREAM	CONNECTED	15721	
	unix	3	[ ]	STREAM	CONNECTED	16537	@/tmp/.X11-unix/X0
107	unix	3	[ ]	STREAM	CONNECTED	13007	
	unix	3	[ ]	STREAM	CONNECTED	15241	@/tmp/.ICE-unix/2049
109	unix	3	[ ]	STREAM	CONNECTED	16409	
	unix	3	[ ]	STREAM	CONNECTED	15460	
111	unix	3	[ ]	STREAM	CONNECTED	16430	
	unix	2	[ ]	STREAM	CONNECTED	19260	@/tmp/dbus-OAyeDpuWXb
-ckMYcNJw							
113	unix	3	[ ]	STREAM	CONNECTED	20726	@/dbus-vfs-daemon/socket
	unix	3	[ ]	STREAM	CONNECTED	16540	
115	unix	3	[ ]	STREAM	CONNECTED	16426	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	13048	
117	unix	3	[ ]	STREAM	CONNECTED	15065	
	unix	3	[ ]	STREAM	CONNECTED	14602	
119	unix	3	[ ]	STREAM	CONNECTED	14002	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	15659	
system_bus_socket							
121	unix	3	[ ]	STREAM	CONNECTED	12872	/var/run/dbus/
	unix	3	[ ]	STREAM	CONNECTED	16546	
123	unix	3	[ ]	STREAM	CONNECTED	15653	
	unix	3	[ ]	STREAM	CONNECTED	13011	
125	unix	3	[ ]	STREAM	CONNECTED	15417	
	unix	3	[ ]	STREAM	CONNECTED	24701	/run/user/1000/keyring-
n5bKb6/pkcs11							
127	unix	3	[ ]	STREAM	CONNECTED	15763	
	unix	3	[ ]	STREAM	CONNECTED	15544	@/tmp/.X11-unix/X0
129	unix	3	[ ]	STREAM	CONNECTED	10850	/var/run/dbus/
	unix	3	[ ]	STREAM	CONNECTED	12848	
131	unix	3	[ ]	STREAM	CONNECTED	23726	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	15457	
133	unix	3	[ ]	STREAM	CONNECTED	14124	
	unix	3	[ ]	STREAM	CONNECTED	13043	@/tmp/.X11-unix/X0
135	unix	3	[ ]	STREAM	CONNECTED	15171	
	unix	3	[ ]	STREAM	CONNECTED	15175	@/tmp/dbus-OAyeDpuWXb
137	unix	3	[ ]	STREAM	CONNECTED	16623	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15514	
system_bus_socket							
139	unix	3	[ ]	STREAM	CONNECTED	10888	/var/run/dbus/
	unix	3	[ ]	STREAM	CONNECTED	13359	
141	unix	3	[ ]	DGRAM	CONNECTED	9980	
	unix	3	[ ]	STREAM	CONNECTED	18425	
143	unix	3	[ ]	STREAM	CONNECTED	12970	
	unix	3	[ ]	STREAM	CONNECTED	24168	@/tmp/dbus-OAyeDpuWXb
145	unix	3	[ ]	STREAM	CONNECTED	897	
	unix	3	[ ]	STREAM	CONNECTED	13115	
147	unix	3	[ ]	STREAM	CONNECTED	15034	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15532	
149	unix	3	[ ]	STREAM	CONNECTED	12983	
	unix	2	[ ]	DGRAM	CONNECTED	15560	@/tmp/.X11-unix/X0
151	unix	3	[ ]	STREAM	CONNECTED	21789	
	unix	3	[ ]	STREAM	CONNECTED	15490	
session/1000/1934							
153	unix	3	[ ]	STREAM	CONNECTED	11241	@/com/ubuntu/upstart-
	unix	3	[ ]	STREAM	CONNECTED	20686	
system_bus_socket							
	unix	3	[ ]	STREAM	CONNECTED	10849	/var/run/dbus/
	unix	3	[ ]	STREAM	CONNECTED		

155	unix	3	[ ]	STREAM	CONNECTED	24724	
	unix	3	[ ]	STREAM	CONNECTED	15472	
157	unix	3	[ ]	STREAM	CONNECTED	15458	
	unix	3	[ ]	STREAM	CONNECTED	15566	@/tmp/dbus-OAyeDpuWXb
159	unix	2	[ ]	STREAM	CONNECTED	21787	
	unix	3	[ ]	STREAM	CONNECTED	12997	@/tmp/.ICE-unix/2049
161	unix	3	[ ]	STREAM	CONNECTED	25782	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	20746	
163	unix	3	[ ]	STREAM	CONNECTED	13053	
	unix	3	[ ]	STREAM	CONNECTED	15332	
165	unix	3	[ ]	STREAM	CONNECTED	15731	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	892	
167	unix	3	[ ]	STREAM	CONNECTED	17179	
	unix	3	[ ]	STREAM	CONNECTED	16477	@/tmp/dbus-OAyeDpuWXb
169	unix	3	[ ]	STREAM	CONNECTED	12985	
	unix	3	[ ]	STREAM	CONNECTED	11238	
171	unix	3	[ ]	STREAM	CONNECTED	14166	
	unix	3	[ ]	STREAM	CONNECTED	15518	
173	unix	3	[ ]	STREAM	CONNECTED	13058	
	unix	3	[ ]	STREAM	CONNECTED	20559	
175	unix	3	[ ]	STREAM	CONNECTED	15660	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	25088	
177	unix	3	[ ]	STREAM	CONNECTED	15044	
	unix	3	[ ]	STREAM	CONNECTED	14873	@/tmp/dbus-OAyeDpuWXb
179	unix	3	[ ]	STREAM	CONNECTED	23036	@/tmp/dbus-PGSuh9PnQT
	unix	3	[ ]	STREAM	CONNECTED	16675	
181	unix	3	[ ]	STREAM	CONNECTED	15468	
	unix	3	[ ]	STREAM	CONNECTED	20687	@/dbus-vfs-daemon/socket
	-vHTyGqrl						
183	unix	3	[ ]	STREAM	CONNECTED	15161	/var/run/sdp
	unix	3	[ ]	STREAM	CONNECTED	15675	/var/run/dbus/
	system_bus_socket						
185	unix	3	[ ]	STREAM	CONNECTED	10224	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	24818	
187	unix	3	[ ]	STREAM	CONNECTED	12966	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	10226	/var/run/dbus/
	system_bus_socket						
189	unix	3	[ ]	DGRAM		9979	
	unix	2	[ ]	STREAM	CONNECTED	24935	@/dbus-vfs-daemon/socket
	-Ar7V7Shu						
191	unix	3	[ ]	STREAM	CONNECTED	15167	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	22142	
193	unix	3	[ ]	STREAM	CONNECTED	15040	
	unix	2	[ ]	DGRAM		14158	
195	unix	3	[ ]	STREAM	CONNECTED	10973	
	unix	3	[ ]	STREAM	CONNECTED	11235	
197	unix	2	[ ]	STREAM	CONNECTED	17280	
	unix	3	[ ]	STREAM	CONNECTED	15121	
199	unix	3	[ ]	STREAM	CONNECTED	16783	
	unix	3	[ ]	STREAM	CONNECTED	13119	
201	unix	3	[ ]	STREAM	CONNECTED	12963	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	12485	
203	unix	3	[ ]	STREAM	CONNECTED	22168	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	22143	
205	unix	3	[ ]	STREAM	CONNECTED	19182	@/dbus-vfs-daemon/socket
	-mO3Q1TcU						
	unix	3	[ ]	STREAM	CONNECTED	16517	
207	unix	3	[ ]	STREAM	CONNECTED	12876	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	13118	@/dbus-vfs-daemon/socket
	-QO5OKfs4						
209	unix	3	[ ]	STREAM	CONNECTED	14634	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	12885	
211	unix	2	[ ]	DGRAM		951	

	unix	3	[ ]	STREAM	CONNECTED	13042	@/tmp/dbus-OAyeDpuWXb
213	unix	3	[ ]	STREAM	CONNECTED	13034	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	12974	@/tmp/.X11-unix/X0
215	unix	3	[ ]	STREAM	CONNECTED	15469	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	20747	@/tmp/.X11-unix/X0
217	unix	3	[ ]	STREAM	CONNECTED	15596	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	15551	
219	unix	2	[ ]	STREAM	CONNECTED	17217	@/dbus-vfs-daemon/socket
	-kPxvkJE6						
	unix	3	[ ]	STREAM	CONNECTED	16483	
221	unix	3	[ ]	STREAM	CONNECTED	19609	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	10220	/var/run/dbus/
	system_bus_socket						
223	unix	3	[ ]	STREAM	CONNECTED	18380	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	13106	/var/run/dbus/
	system_bus_socket						
225	unix	3	[ ]	STREAM	CONNECTED	16506	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15548	@/tmp/dbus-OAyeDpuWXb
227	unix	3	[ ]	STREAM	CONNECTED	15433	
	unix	3	[ ]	STREAM	CONNECTED	12706	
229	unix	3	[ ]	STREAM	CONNECTED	13981	
	unix	3	[ ]	STREAM	CONNECTED	19904	
231	unix	3	[ ]	STREAM	CONNECTED	14969	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	15039	@/tmp/dbus-OAyeDpuWXb
233	unix	3	[ ]	STREAM	CONNECTED	10211	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	23715	
235	unix	3	[ ]	STREAM	CONNECTED	15733	
	unix	3	[ ]	STREAM	CONNECTED	14887	
237	unix	3	[ ]	STREAM	CONNECTED	13027	
	unix	3	[ ]	STREAM	CONNECTED	15537	
239	unix	3	[ ]	STREAM	CONNECTED	14890	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	14871	
241	unix	3	[ ]	STREAM	CONNECTED	12162	
	unix	3	[ ]	STREAM	CONNECTED	16606	
243	unix	3	[ ]	STREAM	CONNECTED	15550	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	19981	
245	unix	3	[ ]	STREAM	CONNECTED	13041	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15594	@/tmp/dbus-OAyeDpuWXb
247	unix	3	[ ]	STREAM	CONNECTED	12999	@/tmp/.ICE-unix/2049
	unix	2	[ ]	DGRAM		14264	
249	unix	3	[ ]	STREAM	CONNECTED	12163	
	unix	3	[ ]	STREAM	CONNECTED	13065	@/tmp/dbus-OAyeDpuWXb
251	unix	3	[ ]	STREAM	CONNECTED	15597	
	unix	3	[ ]	STREAM	CONNECTED	15441	@/com/ubuntu/upstart-
	session/1000/1934						
253	unix	3	[ ]	STREAM	CONNECTED	15094	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	16433	
255	unix	3	[ ]	STREAM	CONNECTED	15081	
	unix	3	[ ]	STREAM	CONNECTED	15538	
257	unix	3	[ ]	STREAM	CONNECTED	19985	
	unix	3	[ ]	STREAM	CONNECTED	15066	/run/user/1000/keyring-
	n5bKb6/pkcs11						
259	unix	3	[ ]	STREAM	CONNECTED	11127	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	13054	/var/run/dbus/
	system_bus_socket						
261	unix	3	[ ]	STREAM	CONNECTED	15387	
	unix	2	[ ]	DGRAM		15734	
263	unix	3	[ ]	STREAM	CONNECTED	11262	
	unix	3	[ ]	STREAM	CONNECTED	12998	@/tmp/.ICE-unix/2049
265	unix	3	[ ]	STREAM	CONNECTED	12944	
	unix	3	[ ]	STREAM	CONNECTED	13093	/var/run/dbus/
	system_bus_socket						

267	unix	3	[ ]	STREAM	CONNECTED	15090	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	15586	@/tmp/dbus-OAyeDpuWXb
269	unix	3	[ ]	STREAM	CONNECTED	15543	
	unix	3	[ ]	STREAM	CONNECTED	23012	
271	unix	3	[ ]	STREAM	CONNECTED	16559	
	unix	3	[ ]	STREAM	CONNECTED	18872	
273	unix	3	[ ]	STREAM	CONNECTED	15723	/run/user/1000/pulse/
	native						
	unix	3	[ ]	STREAM	CONNECTED	839	@/com/ubuntu/upstart
275	unix	3	[ ]	STREAM	CONNECTED	19982	
	unix	3	[ ]	STREAM	CONNECTED	13040	@/tmp/.ICE-unix/2049
277	unix	3	[ ]	STREAM	CONNECTED	16432	
	unix	3	[ ]	STREAM	CONNECTED	14874	
279	unix	2	[ ]	DGRAM		12710	
	unix	3	[ ]	STREAM	CONNECTED	12168	
281	unix	3	[ ]	STREAM	CONNECTED	22137	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	14979	
283	unix	3	[ ]	STREAM	CONNECTED	13012	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	12980	@/tmp/dbus-OAyeDpuWXb
285	unix	3	[ ]	STREAM	CONNECTED	12939	
	unix	3	[ ]	STREAM	CONNECTED	16510	@/tmp/.X11-unix/X0
287	unix	3	[ ]	STREAM	CONNECTED	12844	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	16616	@/tmp/dbus-OAyeDpuWXb
289	unix	3	[ ]	STREAM	CONNECTED	15623	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	14409	
291	unix	3	[ ]	STREAM	CONNECTED	19928	
	unix	3	[ ]	STREAM	CONNECTED	16542	@/tmp/dbus-OAyeDpuWXb
293	unix	3	[ ]	STREAM	CONNECTED	15095	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15652	@/tmp/dbus-OAyeDpuWXb
295	unix	3	[ ]	STREAM	CONNECTED	15043	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15577	
297	unix	2	[ ]	DGRAM		15437	
	unix	3	[ ]	STREAM	CONNECTED	16692	@/tmp/.X11-unix/X0
299	unix	3	[ ]	STREAM	CONNECTED	13050	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	15083	
301	unix	3	[ ]	STREAM	CONNECTED	12754	
	unix	3	[ ]	STREAM	CONNECTED	16541	@/tmp/dbus-OAyeDpuWXb
303	unix	3	[ ]	STREAM	CONNECTED	16408	
	unix	3	[ ]	STREAM	CONNECTED	22144	@/tmp/.X11-unix/X0
305	unix	3	[ ]	STREAM	CONNECTED	10751	
	unix	3	[ ]	STREAM	CONNECTED	24816	@/dbus-vfs-daemon/socket
	-p9go0eNP						
307	unix	2	[ ]	DGRAM		17290	
	unix	3	[ ]	STREAM	CONNECTED	13112	
309	unix	3	[ ]	STREAM	CONNECTED	16693	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	9942	
311	unix	3	[ ]	STREAM	CONNECTED	14877	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	10209	
313	unix	3	[ ]	STREAM	CONNECTED	24700	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	12843	
315	unix	3	[ ]	STREAM	CONNECTED	12593	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	12487	
317	unix	3	[ ]	STREAM	CONNECTED	25087	
	unix	3	[ ]	STREAM	CONNECTED	10453	@/com/ubuntu/upstart
319	unix	3	[ ]	STREAM	CONNECTED	20377	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	22156	
321	unix	3	[ ]	STREAM	CONNECTED	12868	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	14876	/var/run/dbus/
	system_bus_socket						
323	unix	3	[ ]	STREAM	CONNECTED	15114	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	12996	@/tmp/.ICE-unix/2049
325	unix	3	[ ]	STREAM	CONNECTED	10898	/var/run/dbus/
	system_bus_socket						
	unix	2	[ ]	DGRAM		12979	

327	unix	3	[ ]	STREAM	CONNECTED	16626	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	16413	@/tmp/dbus-OAyeDpuWXb
329	unix	3	[ ]	STREAM	CONNECTED	11220	
	unix	3	[ ]	STREAM	CONNECTED	12984	/run/user/1000/pulse/
	native						
331	unix	3	[ ]	STREAM	CONNECTED	14953	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	15179	
333	unix	2	[ ]	DGRAM		12384	
	unix	3	[ ]	STREAM	CONNECTED	24819	@/dbus-vfs-daemon/socket
	-sQ94RQHx						
335	unix	3	[ ]	STREAM	CONNECTED	18426	
	unix	3	[ ]	STREAM	CONNECTED	14962	
337	unix	3	[ ]	STREAM	CONNECTED	23188	
	unix	3	[ ]	STREAM	CONNECTED	12946	@/tmp/dbus-OAyeDpuWXb
339	unix	3	[ ]	STREAM	CONNECTED	12882	@/tmp/.X11-unix/X0
	unix	3	[ ]	STREAM	CONNECTED	11240	@/com/ubuntu/upstart-
	session/1000/1934						
341	unix	3	[ ]	STREAM	CONNECTED	24169	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	14562	/var/run/dbus/
	system_bus_socket						
343	unix	3	[ ]	STREAM	CONNECTED	16627	@/dbus-vfs-daemon/socket
	-Klqa6Af4						
	unix	3	[ ]	STREAM	CONNECTED	15160	
345	unix	3	[ ]	STREAM	CONNECTED	13006	/var/run/dbus/
	system_bus_socket						
	unix	3	[ ]	STREAM	CONNECTED	22152	
347	unix	3	[ ]	STREAM	CONNECTED	10210	
	unix	3	[ ]	STREAM	CONNECTED	14939	@/tmp/dbus-OAyeDpuWXb
349	unix	3	[ ]	STREAM	CONNECTED	12871	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	22161	
351	unix	2	[ ]	DGRAM		10208	
	unix	3	[ ]	STREAM	CONNECTED	15084	@/tmp/dbus-OAyeDpuWXb
353	unix	3	[ ]	STREAM	CONNECTED	15515	
	unix	2	[ ]	DGRAM		896	
355	unix	3	[ ]	STREAM	CONNECTED	25086	
	unix	3	[ ]	STREAM	CONNECTED	15588	
357	unix	2	[ ]	DGRAM		17282	
	unix	3	[ ]	STREAM	CONNECTED	19707	/run/user/112/pulse/
	native						
359	unix	3	[ ]	STREAM	CONNECTED	18871	
	unix	3	[ ]	STREAM	CONNECTED	11072	/var/run/dbus/
	system_bus_socket						
361	unix	3	[ ]	STREAM	CONNECTED	12439	
	unix	3	[ ]	STREAM	CONNECTED	22155	
363	unix	3	[ ]	STREAM	CONNECTED	21267	
	unix	3	[ ]	STREAM	CONNECTED	13017	/var/run/dbus/
	system_bus_socket						
365	unix	3	[ ]	STREAM	CONNECTED	13111	
	unix	3	[ ]	STREAM	CONNECTED	18421	
367	unix	3	[ ]	STREAM	CONNECTED	15585	
	unix	3	[ ]	STREAM	CONNECTED	14394	/var/run/dbus/
	system_bus_socket						
369	unix	2	[ ]	DGRAM		14165	
	unix	3	[ ]	STREAM	CONNECTED	12971	@/tmp/.X11-unix/X0
371	unix	2	[ ]	DGRAM		873	
	unix	3	[ ]	STREAM	CONNECTED	19257	@/tmp/dbus-OAyeDpuWXb
373	unix	3	[ ]	STREAM	CONNECTED	19181	@/tmp/dbus-OAyeDpuWXb
	unix	3	[ ]	STREAM	CONNECTED	13105	

netstat -s:

```

Ip:
2   53549 total packets received
    2 with invalid addresses
4   0 forwarded
    0 incoming packets discarded

```

```

6      53547 incoming packets delivered
      53518 requests sent out
8      4 outgoing packets dropped
      112 dropped because of missing route
10     Icmp:
      52988 ICMP messages received
12      9 input ICMP message failed.
      ICMP input histogram:
14          destination unreachable: 352
          echo requests: 13
16          echo replies: 52623
      52993 ICMP messages sent
18      0 ICMP messages failed
      ICMP output histogram:
20          destination unreachable: 352
          echo request: 52628
22          echo replies: 13
     IcmpMsg:
24         InType0: 52623
         InType3: 352
26         InType8: 13
         OutType0: 13
28         OutType3: 352
         OutType8: 52628
30     Tcp:
      67 active connections openings
32      34 passive connection openings
      33 failed connection attempts
34      0 connection resets received
      0 connections established
36      596 segments received
      597 segments send out
38      1 segments retransmitted
      0 bad segments received.
40      32 resets sent
     Udp:
42      504 packets received
      8 packets to unknown port received.
44      0 packet receive errors
      483 packets sent
46      IgnoredMulti: 12
     UdpLite:
48     TcpExt:
      34 TCP sockets finished time wait in fast timer
50      4 delayed acks sent
      2 delayed acks further delayed because of locked socket
52      159 packet headers predicted
      169 acknowledgments not containing data payload received
54      41 predicted acknowledgments
      1 other TCP timeouts
56      TCPRcvCoalesce: 21
      TCPSpuriousRtxHostQueues: 1
58      TCPSynRetrans: 1
      TCPOrigDataSent: 264
60     IpExt:
      InMcastPkts: 489
62      OutMcastPkts: 130
      InBcastPkts: 14
64      InOctets: 4637350
      OutOctets: 4516426
66      InMcastOctets: 166812
      OutMcastOctets: 26326
68      InBcastOctets: 5108
      InNoECTPkts: 53549

```

**Question 4.1.a)**

What are the network interfaces of PC1 and what are the MTU (Maximum Transmission Unit) values of the interfaces?

*eth0 and loopback (lo) interfaces with respectively 1500 and 65536 as MTU.*

**Question 4.1.b)**

How many IP datagrams, ICMP messages, UDP datagrams, and TCP segments has PC1 transmitted and received since it was last rebooted?

protocol	received	sent
IP	53549	53518
ICMP	52988	52993
UDP	504	483
TCP	596	597

Table 2.4: solutions IP and MAC addresses.

**Question 4.2)**

Explain the role of interface lo, the loopback interface. In the output of "netstat -in", 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 is the interface used by hosts to communicate with themselves. Therefore, every packet sent to the loopback interface is also received by the loopback interface.*

*This is not the case for interface "eth0". Thus, RX-OK and TX-OK are identical for interface lo but not for interface "eth0".*



## Part 5. Configuring IP interfaces in Linux

The `ifconfig` command is used to configure parameters of network interfaces on a Linux system, such as enabling and disabling of interfaces and setting the IP address. The `ifconfig` command is usually run when a system boots up. In this case, the parameters of the commands are read from a file. Once the Linux system is running, the `ifconfig` command can be used to modify the network configuration parameters.



*The most common uses of the `ifconfig` command to query the status of network interfaces are as follows:*

`ifconfig`

*Displays the configuration parameters of all active interfaces.*

`ifconfig -a`

*Displays the configuration parameters of all network interfaces, including the inactive interfaces.*

`ifconfig <interface>`

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



*There are numerous options for configuring a network interface with `ifconfig`. The following example shows how to enable and disable an interface and how to change the IP configuration.*

`ifconfig eth0 down`

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

`ifconfig eth0 up`

*Enables the `eth0` 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. The interface should be disabled before a new IP address is assigned, and should be enabled after the IP address has been modified.*

`ifconfig eth0 down 10.0.1.8 netmask 255.255.255.0 broadcast 10.0.1.255 up`

*Performs all three commands above in sequence. Interface `eth0` is disabled, an IP address and a broadcast address are assigned, and the interface is enabled.*

`ifconfig eth0 mtu 500`

*Sets the MTU of interface `eth0` to 500 bytes.*

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

Use the `ifconfig` command to modify the IP address of the `eth0` interface of PC4.

1. On PC4, run `ifconfig -a` and save the output.
2. Change the IP address of interface `eth0` of PC4 to 10.0.1.11/24.
3. Run `ifconfig -a` again and save the output.

**Question 5.1)**

Attach the saved files to your report and explain the fields of the `ifconfig` output.

Link encap: Name of link layer protocol

HWaddr: MAC address

inet addr: IPv4 address

Bcast: Broadcast IP address

Mask: Subnet-mask

inet6 addr: IPv6 address

Scope: IPv6 scope

MTU: Maximum Transmission unit

Metric: Priority of the device

RX packets: Received packets through this interface

errors: Number of errors with incoming packets

dropped: Number of incoming packets dropped

overruns: Number of incoming packets dropped because capacity of interface exceeded

frame: 0

TX packets: Transmitted packets through this interface

errors: Number of errors with outgoing packets

dropped: Number of outgoing packets dropped

overruns: Number of outgoing packets dropped because capacity of interface exceeded

carrier:0

collisions: Number of collisions

txqueuelen: Length of the transmission-queue

RX bytes: Number of bits received

TX bytes: Number of bits transmitted

Interrupt: The interrupt line used by the interface

Memory: Memory used by this device

## Part 6. Duplicate IP addresses

In this part of the lab, you observe what happens when two hosts have identical IP addresses.

### Exercise 6. Duplicate IP addresses

1. After completing Exercise 5, the IP addresses of the Ethernet interfaces on the four PCs are as shown in the Table 2.5. Note that PC1 and P4 are assigned the same IP address.

Linux PC	IP Addresses of Ethernet Interface eth0
PC1	10.0.1.11/24
PC2	10.0.1.12/24
PC3	10.0.1.13/24
PC3	10.0.1.11/24

Table 2.5: IP addresses for Part 6

2. Delete all entries in the ARP cache on all PCs.
3. Run Wireshark on PC3 and capture the network traffic to and from the duplicate IP address 10.0.1.11
4. From PC3, start a Telnet session to the duplicate IP address, 10.0.1.11, by typing
 

```
PC3% telnet 10.0.1.11
```

 and log in as *telecomlabo* user, using the *mvkbf1n* password.
5. Once you have logged in, determine the name of the host to which you are connected. The name of the host can be determined in several ways: (1) issue the command `hostname`, (2) inspect the ARP cache on PC3, or (3) interpret the captured Wireshark packets.
6. Stop the traffic capture in Wireshark.
7. Save all ARP packets and the first few TCP packets captured by Wireshark. Also save the ARP cache of PC3 using the `arp -a` command.
8. When you are done with the exercise, reset the IP address of PC4 to its original value as given in Table 2.1.

### Question 6.1)

Explain why the Telnet session was established to one of the hosts with the duplicate address and not the other. Explain why the Telnet session was established at all, and did not result in an error message. Use the ARP cache and the captured packets to support your explanation.

In the PCAP dump we see that the ARP request gets duplicate replies. However, the first host who replies has its MAC address saved in the local ARP table and is used for further messaging.

Even though there are duplicate IP addresses in the network, the MAC addresses are unique, so only one of the hosts will respond.

## 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.



*If you are having difficulties understanding the concept of netmasks, try using a subnet calculator tool. You can find one at <http://subnet-calculator.com> or most operating systems have native tools or widgets that can do the same.*

### Exercise 7.

1. Setup the interfaces of the hosts as shown in Table 2.6. Note that the netmasks of the hosts are different.

Linux PC	IP Addresses of eth0	Netmask
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
PC3	10.0.1.121/28	255.255.255.240

Table 2.6: IP addresses for Part 7

2. Run Wireshark on PC1 and capture the packets for the following ping commands
  - a. From PC1 to PC3:  

```
| PC1% ping -c 1 10.0.1.120
```
  - b. From PC1 to PC2:  

```
| PC1% ping -c 1 10.0.1.101
```
  - c. From PC1 to PC4:  

```
| PC1% ping -c 1 10.0.1.121
```
  - d. From PC4 to PC1:  

```
| PC1% ping -c 1 10.0.1.100
```
  - e. From PC2 to PC4:  

```
| PC1% ping -c 1 10.0.1.121
```
  - f. From PC2 to PC3:  

```
| PC1% ping -c 1 10.0.1.120
```
3. Save the Wireshark output, and save the output of the ping commands. Note that not all of the above scenarios are successful. Save all 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.

### Question 7.1)

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



**You get credits for answering the 'why?' question, not for stating the obvious.**

## PC1

```
1 student@lab2pc1:~$ ping -c1 10.0.1.120
PING 10.0.1.120 (10.0.1.120) 56(84) bytes of data.
3 64 bytes from 10.0.1.120: icmp_seq=1 ttl=64 time=0.537 ms

5 — 10.0.1.120 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
7 rtt min/avg/max/mdev = 0.537/0.537/0.537/0.000 ms

9

11 student@lab2pc1:~$ ping -c1 10.0.1.101
PING 10.0.1.101 (10.0.1.101) 56(84) bytes of data.
13 64 bytes from 10.0.1.101: icmp_seq=1 ttl=64 time=0.589 ms

15 — 10.0.1.101 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
17 rtt min/avg/max/mdev = 0.589/0.589/0.589/0.000 ms

19

21 student@lab2pc1:~$ ping -c1 10.0.1.121
PING 10.0.1.121 (10.0.1.121) 56(84) bytes of data.
23 From 10.0.1.100 icmp_seq=1 Destination Host Unreachable

25 — 10.0.1.121 ping statistics —
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

## PC2

```
student@lab2pc1:~$ ping -c1 10.0.1.121
2 connect: Network is unreachable

4 student@lab2pc1:~$ ping -c1 10.0.1.120
connect: Network is unreachable
```

## PC4

```
1 student@lab2pc1:~$ ping -c1 10.0.1.100
connect: Network is unreachable
```

a and b succeed, all of the others fail.

PC1 and PC3 have the same netmask and are in the same subnet. PC1 successfully retrieves PC3's hardware address through an ARP request.

PC1 and PC2 don't have the same netmask, but PC1's subnet would match PC2's subnet if they both had netmask /24, so PC1 sends an ARP request to PC2. PC2 receives the ARP request and "coincidentally", they would also be in the same subnet if they both had netmask /28, so PC2 responds to the ARP request and the ping succeeds.

PC1 and PC4 don't have the same netmask, but they would be in the same subnet if they both had netmask /24, so PC1 sends an ARP request to PC4. PC4 receives the request, but from its point of view, PC1 is not in the same subnet, even if they both had netmask /28, so it doesn't respond to the ARP request and the ping fails.

Because of the reason PC4 didn't respond to PC1's ARP request, the ping fails right away with a "Network is unreachable" error.

PC2 and PC4 have the same netmask, but are not in the same subnet, so we get a "Network

is unreachable" error again.

For the ping from PC2 to PC3, the situation is the same as with PC4 to PC1.

## Part 8. Static mapping of IP addresses and hostnames

Since it is easier to memorize names than IP addresses, there are mechanisms to associate a symbolic name, called *hostname*, with an IP address. On the Internet, the resolution between hostnames and IP addresses is generally done by the Domain Name System (DNS), which will not be discussed in this course. 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 a text editor.
2. On PC1, issue a `ping` command to PC2

```
| PC1% ping 10.0.1.12
```

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 and save the changes. Use the names PC1, PC2, etc., as used throughout this lab to refer to the PCs.
5. Repeat Step 3. You should now be able to ping directly using the hostnames "PC2", "PC3", "PC4", as in:

```
| PC1% ping PC2
| PC1% ping PC3
| PC1% ping PC4
```

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.

#### Question 8.1)

Explain why a static mapping of names and IP addresses is impractical when the number of hosts is large.

With many hosts, the static mapping of hosts' addresses and names becomes impractical as there are many host files to update, each time a new host is introduced or removed.

Say there are 1000 hosts and a new host enters the network, each of those 1000 hosts' files should be updated for the network name resolving to become synchronised.

This synchronisation becomes too complex very fast.

#### Question 8.2)

What will be the result of the hostname resolution when multiple IP addresses are associated with the same hostname in the `/etc/hosts` file?

When multiple addresses are associated with a single hostname, only the first occurring instance of the hostname is resolved.

## Part 9. Experiments with FTP and Telnet

A severe security problem with the file transfer protocol (FTP) is that the login and password information are transmitted as plain text (not encrypted). Sometimes malicious users exploit this by snooping passwords on the network.

Here you learn how easy it is to crack passwords by snooping traffic from FTP and Telnet sessions.



***The use of applications that do not encrypt passwords, such as FTP and Telnet, is strongly discouraged. On the Internet, you should use protocols such as Secure Shell (SSH) tools for file transfers and remote login.***

### Exercise 9-A. Snoop Passwords from an FTP session

Capture traffic from an FTP session between two hosts.

The ftp server installed on the lab PC's is vsftpd, which is not started by default. Use the following command to start it on PC2:

```
|PC2% service vsftpd start
```

1. On PC1, run the Wireshark command with capture filters set to capture traffic between PC1 and PC2. The capture filter is

```
|host 10.0.1.11 and host 10.0.1.12
```

2. On PC1, initiate a FTP session to PC2 by typing

```
|PC1% ftp 10.0.1.12
```

3. Log in as *telecomlabo* user.
4. Inspect the payload of packets with FTP payload that are sent from PC1 to PC2. FTP sessions use TCP connections for data transfer.



*In Wireshark, there is a simple method to view the payload sent in a TCP connection. Simply select a packet that contains a TCP segment in the main window of Wireshark, and then click on "Follow TCP Stream" in the "Tools" menu of the Wireshark window. This will create a new window that displays only the payload of the selected TCP connection.*

5. Save the details of the packets, i.e., select "Print details" in the "Print" window of Wireshark, which transmit the login name and password. As a hint, you can set the display filter in Wireshark to show only the desired packet(s). Refer to Question 9 from the Prelab.

#### Question 9.A.1)

Using the saved output, identify the port numbers of the FTP client and the FTP server.



```

Source: 10.0.1.11 (10.0.1.11)
2 Destination: 10.0.1.12 (10.0.1.12)
...
4 Transmission Control Protocol, Src Port: 41694 (41694), Dst Port: 21 (21), Seq: 1,
  Ack: 1, Len: 0

```

In the TCP header, we can see that the source port is 41694 and the destination port is 21. With the destination being the server and source the client.

### Question 9.A.2)

Identify the login name and the password, shown in plain text in the payload of the packets that you captured.

No.	Time	Source	Destination	Protocol	Length
2	8 6.633875000	10.0.1.11	10.0.1.12	FTP	80
	Request: USER student				

No.	Time	Source	Destination	Protocol	Length
1	14 11.825979000	10.0.1.11	10.0.1.12	FTP	80
3	Request: PASS mvkbj1n				

In packets 8 and 14 we find the username and passwords sent in plaintext.  
 Username: student  
 Password: mvkbj1n

### Exercise 9-B. Snoop Passwords from a Telnet session

Repeat the above exercise with the `telnet` command instead of `ftp`. On PC1, establish a Telnet session to PC2, and save the Wireshark output of packets used to transmit the login name and password.

### Question 9.B)

Does Telnet have the same security flaws as FTP? Support your answer using the saved output.

Yes, it does.

In telnet, the the text is sent character per character.

The user login and password are divided between subsequent packets in one direction but are still plaintext and thus readable when monitoring the network.

Packets 32, 35, 38, 41, 44, 47, 50 contain respectively 's','t','u','d','e','n','t'.

Packets 58, 61, 64, 67, 70, 73, 76 contain respectively 'm','v','k','b','j','1','n'.

**Exercise 9-C. Observing traffic from a Telnet session**

This exercise uses the Telnet session established in the previous exercise.

1. Run Wireshark on PC1, and start to capture traffic. If the Wireshark window from the previous exercise is still open, make sure that Wireshark is capturing traffic.
2. If the Telnet session from the previous exercise is still in place, skip to the next step. Otherwise, follow the steps from the previous exercise and log in from PC1 to PC2 with the `telnet` command.
3. Once you are logged in, type a few characters. Observe the number of packets, captured by Wireshark, for each character typed. Observe that for each key you type, there are three packets transmitted. Determine why this occurs.
4. Save the Wireshark output to a text file (using the “Print Summary” option).

**Question 9.C)**

Attach the saved output to your report. Explain why three packets are sent in a telnet session for each character typed on the terminal.

In the first packet, the character is sent from the client to the telnet server.

The second packet is a telnet message with empty data, which can be seen as an ACK which says that the character is received.

The third and last packet is a TCP ACK message from the client to the server which acknowledges the second packet's arrival.