Name (please print): _____

Networks Lab: Single-Segment IP Networks

Objectives:

- 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.

Prelab Questions

Read on the Internet about the following:

- Network commands in Unix: `arp`, `ifconfig` and `netstat`
- `tcpdump`: Read about the various arguments available for `tcpdump`.
- `wireshark`: Read about capture filters and display filters in `wireshark`.

Answer the following questions in the space provided below each one. Use extra sheets of paper if needed and attach them to this document. Submit the answers with your lab report.

1. Write the syntax for an `ifconfig` command that sets the IP address of the interface eth0 to 128.143.2.3116 with broadcast address 128.143.255.255.

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

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

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 I0.0.1.12, both on interface eth1.

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

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

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

LAB

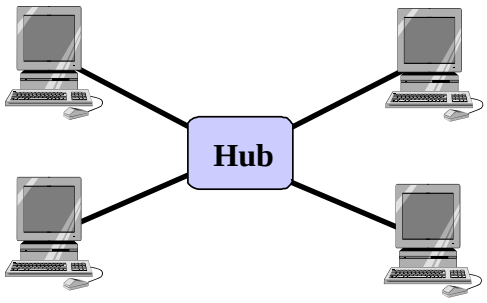
In this lab you become acquainted with IP configuration issues on a single Ethernet segment. The lab also exposes you to advanced issues in `tcpdump` and `wireshark`.

RECALL:

- Before you get started, reboot the Linux PCs.
- During the lab, you need to save data to files. Save all files in the directory `/labdata`.
- Save your file to a memory stick before the end of the lab. You will need the files when you prepare your lab report.

SETUP FOR LAB

- In G291 there are four groups of Linux PCs connected to the same Ethernet segment by an Ethernet hub, as shown in the Figure below.



  You may use any of the four groups to complete this lab. In each group, the four machines have names starting with A, B, C and D. For example, the A-machines are Alpha, Abner, Algol and Atlanta, one in each group. You can easily identify the machines by the labels posted on them.

- On each machine, the root name is root and the password is password. Easy to remember ☺.

- The key combination to enter the lab is 3125.

IMPORTANT NOTE:

- DO NOT connect the machines to the Internet, no matter the circumstances. The lab is intended to operate without Internet access.

Part 0. 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 command are read from a file. Once the Linux system is running, the `ifconfig` command can be used to modify the network configuration parameters. The list below shows how `ifconfig` is used to query the status of network interfaces.

`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 example in the list below 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.

Exercise 0. Changing the IP address of an interface.

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

1. Save the IP addresses of the Linux PCs in a file.
2. Change the IP addresses for the Linux PCs to the values displayed in the table below

Linux PC	IP Address
PC1	`10.0.1.11/24`
PC2	`10.0.1.12/24`
PC3	`10.0.1.13/24`
PC4	`10.0.1.14/24`

3. Throughout the lab, you will work with the IP values from the table above. Before leaving the lab, restore the IP addresses of the PCs to the original values.

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

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.

Exercise 1. 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` Unix commands. Use the Unix manual pages to learn how to use these commands, e.g.

   ```
   man tcpdump
   ```

   at the shell prompt will tell you everything you need to know about `tcpdump`.

2. In another terminal window, issue a `ping` command to PC2 by typing

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

   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. 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.

Lab Report. Include the saved data in your lab report.

Part 2. Using filters in wireshark

In this part of the lab, you experiment with filter expressions in wireshark. Recall that wireshark has two types of filters: capture filters and display filters. A capture filter specifies the traffic to be captured by the wireshark tool. By default, wireshark displays all captured packets. With a display filter, just the packets that meet the requirements of the filter are displayed.

Exercise 2a. This exercise is a review of the traffic capture capabilities of wireshark.

1. Start wireshark on PC1 and set the following capture preferences:

   - 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.

You should always set these same preferences for all your experiments.

2. **Setting a capture filter**: In the window *Capture Filters*, set a filter so that all packets that contain the IP address of PC2 are recorded. Apply the filter using *Capture Options*.

   The required filter expression is the answer to question 7 from the prelab.

3. Start the capture by clicking OK in the *Capture Options* 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. Save the results of the capture. This is done by selecting *Print* in the *File* menu.

Exercise 2b. 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, set the following display options:

   - Select Automatic scrolling in live capture.
   - Unselect Enable MAC name resolution.
   - Unselect Enable network name resolution.
   - Unselect Enable transport name resolution.

2. Enter a display filter so that all IP datagrams with destination IP address 10.0.1.12 are shown.
3. Observe the changes in the display panel of wireshark. Only packets with 10.0.1.12 in the IP destination address field should now being displayed.
4. Save the displayed data, by selecting *File/Print*. Note that the *Print* command saves only 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 previous exercise with a display filter that lists only IP datagrams with a source IP address equal to 10.0.1.12. Save the results.

Lab Report Include the saved data in your lab report.

Exercise 2c. More complex capture and display filters. In 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 2a. Do not set any capture or display filters yet.
2. From a new terminal window on PCl, 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 window by typing

   ```
   PC1% te1net 10.0.1.12
   ```

   and log in as root. After you log in successfully to PC2, log out 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.

a. Display packets that contain ICMP messages with the IP address of PC2 either in the IP destination address or IP source address. Save the output.
b. Display packets that contain TCP traffic with the IP address of PC2 either in the IP destination address or IP source address. Save the output.
c. Display packets that, in addition to the constraints in the previous filter expression, use the port number 23. Save the output.

Lab Report Include the saved data in your lab report.

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 <IPaddress, MACaddress>.

COMMON USES OF THE ARP CACHE:

1. Display the contents of the ARP cache:
   ```
   arp -a
   ```
2. Delete the entry with the IP address IPAddress.
   ```
   arp -d IPAddress
   ```
3. Add 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:
   ```
   arp -s IPaddress  MACAddress
   ```
   Example: `arp -s 10.0.1.12 00:02:2D:0D:68:Cl`

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 a host occasionally sends out ARP requests that are already in the ARP cache.
Example: Suppose that a host with IP address 10.0.1.12 has on ARP cache
```
<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 3a. 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 a with a capture filter set to the IP address of PC2.
3. Issue a ping command from PC1 to PC2:
   ```
   PCl% 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:

1. The destination MAC address of the ARP Request packets
2. 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 deleted fairly quickly (about 2 minutes).
5. Save the results of wireshark to a text file, using the Print detail option.

Lab Report Use the saved data to answer the following questions:

- What is the destination MAC address of an ARP Request packet?
- What are the different values of the Type field in the Ethernet headers that you observed?
- Use the captured data to discuss the process in which ARP acquires the MAC address for IP address 10.0.1.12.

Exercise 3b. Matching IP addresses and MAC addresses

Identify the MAC addresses of all interfaces connected to the network and enter them in the table below. 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.

IP and MAC addresses.

Linux PC	IP Address	MAC Address
PC1	`10.0.1.11/24`	
PC2	`10.0.1.12/24`	
PC3	`10.0.1.13/24`	
PC4	`10.0.1.14/24`	

Lab Report Include the completed table in your lab report.

Exercise 3c. ARP Requests to a nonexisting 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. Try to establish a telnet session from PC1 to 10.0.1.10. (Note that this address does not exist on this network.)

```
PC1% te1net 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.

Lab Report

- Using the saved output, describe the time interval between each ARP Request 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.
- Why are ARP Request packets not transmitted (i.e., not encapsulated) like IP packets? Explain your answer.

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.
The following 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`
Display 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.
On PC1, try the different variations of the `netstat` command 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 protocols that are currently in use by typing
   `PCl% netstat -a`
4. Display the statistics of various networking protocols by typing
   `PCl% netstat -s`
   Note: The values of the statistics displayed by some of the `netstat` command are reset each time a host is rebooted.

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

- What are the network interfaces of PC1 and what are the MTU (maximum transmission unit) values of the interfaces?
- How many IP datagrams, ICMP messages, UDP datagrams, and TCP segments has PC1 transmitted and received since it was last rebooted?

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?

Part 5. Duplicate IP Addresses

Use the Use the `ifconfig` command to modify the IP address of the eth0 interface of PC3.

1. On PC3, run `ifconfig -a` and save the output.
2. Change the IP address of interface eth0 of PC3 to `10.0.1.12/24.`
3. Run ifconfig -a again and save the output.
4. At this point, the IP addresses of the four PCs are as follows:

Linux PC	IP Address
PC1	`10.0.1.11/24`
PC2	`10.0.1.12/24`
PC3	`10.0.1.12/24`
PC4	`10.0.1.14/24`

Exercise 5. What happens when two hosts have identical IP addresses?

1. Delete all entries in the ARP cache on all PCs.
2. Run wireshark on PC1 and capture the network traffic to and from the duplicate IP address `10.0.1.12.`
3. From PC1, start a telnet session to the duplicate IP address, `10.0.1.12`, by typing
   ```
   PC1% telnet 10.0.1.12
   ```
   and log in as root user.
4. 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: (a) issue the command hostname, (b) inspect the ARP cache on PC1, or (c) interpret the captured wireshark packets.
5. Stop the traffic capture in wireshark.
6. Save all ARP packets and the first few TCP packets captured by wireshark. Also save the ARP cache of PC1 using the `arp -a` command.
7. When you are done with the exercise, reset the IP address of PC3 to its original value.

Lab Report 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.

Part 6. 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

Linux PC	IP Address	Network Mask
PC1	`10.0.1.100/24`	`255.255.255.0`
PC2	`10.0.1.101/24`	`255.255.255.240`
PC3	`10.0.1.120/24`	`255.255.255.0`
PC4	`10.0.1.121/24`	`255.255.255.240`

Exercise 6. What happens when two hosts have identical IP addresses?

1. Set up the interfaces of the hosts as shown in the table above. Note that the netmasks of the hosts are different.
2. Run wireshark on PC1 and capture the packets for the following ping commands:
    a. From PC1 to PC3: `ping -c 10.0.1.120`
    b. From PC1 to PC2: `ping -c 10.0.1.101`
    c. From PC1 to PC4: `ping -c 10.0.1.121`
    d. From PC4 to PC1: `ping -c 10.0.1.100`
    e. From PC2 to PC4: `ping -c 10.0.1.121`
    f. From PC2 to PC3: `ping -c 10.0.1.120`

    Explain how PC1 sees the traffic between PC2 and PC4 and PC2 and PC3.
3. Save the wireshark output to a text file (setting the Print summary option), and save the output of the ping commands. Note that not all of the previous 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. (Recall that the /24 corresponds to netmask 255.255.255.0.)

Lab Report. 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?

Part 7. 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 host name, with an IP address. On the Internet, the resolution between host names and IP addresses is generally done by the Domain Name System (DNS). 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 host names in the Internet. All hosts on the Internet had to occasionally synchronize with the content of other */etc/hosts* files.

Exercise 7. Associating names with IP addresses.
In this exercise you manipulate the static mapping of host names and IP addresses using the */etc/hosts* file.

1. On PC1, inspect the content of file */etc/hosts* with *gedit*.
2. On PCl, 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 host names with the IP addresses and save the changes.
5. Repeat Step 3. You should now be able to ping directly using 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.

Lab Report. Explain why a static mapping of names and IP addresses is impractical when the number of hosts is large. What will be the result of the host name resolution when multiple IP addresses are associated with the same host name in the /etc/hosts file?

Part 8. 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 Socket Secure Shell (ssh) for file transfers and remote login.

Exercise 8a. Snoop Passwords from an FTP Session.
Capture traffic from an FTP session between two hosts.

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 PCI, initiate an FTP session to PC2 by typing
   ```
   PC1% ftp 10.0.1.12
   ```
3. Log in as root.
4. Inspect the payload of packets with FTP payload that are sent from PC1 to PC2. FTP sessions use TCP connections for data transfer.
5. Save the details of the packets that transmit the login name and password. As a hint, you can set the display filter in wireshark to show only the desired packet(s).

Lab Report.

- Using the saved output, identify the port numbers of the FTP client and the FTP server.
- Identify the login name and the password, shown in plain text in the payload of the packets that you captured.

Exercise 8b. Snoop Passwords from a Telnet Session.
Repeat the previous 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.

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

Exercise 8c. 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, three packets are transmitted. Determine why this occurs.
4. Save the wireshark output to a text file.

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

Troubleshooting

1. Not all machines run a telnet or an ftp server. To activate the servers on a particular machine, run the following set of commands on that machine:

```
/etc/rc.d/init/sshd start
yum install pure-ftpd
/etc/init.d/pure-ftpd start
yum install telnet-server
/etc/init.d/telnet-server start
```

2. You may need to disable the firewall to allow incoming or outgoing telnet connection requests. The firewall sometimes comes back on by itself, so instead of enabling it, you should add exceptions for ftp, ssh and telnet ports to it.

HAVE FUN!

FEEDBACK FROM LAB

- Complete this feedback form at the completion of the lab exercises and submit the form when submitting your lab report.
- The feedback is anonymous. Do not put your name on this form and keep it separate from your lab report.
- For each exercise, please record the following:

Difficulty (-2,-1,0,1,2)	**Interest Level (-1,-1,0,1,2)**
-2 = too easy 0 = just right 2 = too hard	-2 = low interest 0 = just right 2 = high interest

Part 0. Configuring IP interfaces		
Part 1. Using filters in tcpdump		
Part 2. Using filters in wireshark		
Part 3. ARP-Address Resolution Protocol		
Part 4. The netstat command		
Part 5. Duplicate IP addresses		
Part 6. Changing netmasks		
Part 7. Static mapping of IP Adresses and host names		
Part 8. Experiments with FTP and Telnet		