2021

EC-330 Computer Networks

LAB MANUAL

MUHAMMAD UMAR FAROOQ, PHD

LE ASAD MANSOOR

TA SHAHID NABI

LE HAROON

Contents

LAB 1: Wireshark

Objectives

- Get basic familiarity of a network packet structure
- Understanding of network diagnostic tool wireshark

Theory

The purpose of this lab is to introduce the packet sniffer WIRESHARK. WIRESHARK would be used for the lab experiments. This document introduces the basic operation of a packet sniffer, installation, and a test run of WIRESHARK.

The Wireshark interface is one of the easiest to understand of any packet-sniffing application. It is GUI-based, with very clearly written context menus and a straightforward layout. It also provides several features designed to enhance usability, such as protocol-based color coding and detailed graphical representations of raw data.

Packet Sniffer

The basic tool for observing the messages exchanged between executing protocol entities is called a packet sniffer. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is **passive**. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent/received from/by application and protocols executing on your machine.

The first component of the packet Sniffer is the packet capture library, which receives a copy of every **link-layer frame** that is sent from or received by your computer. Messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable.

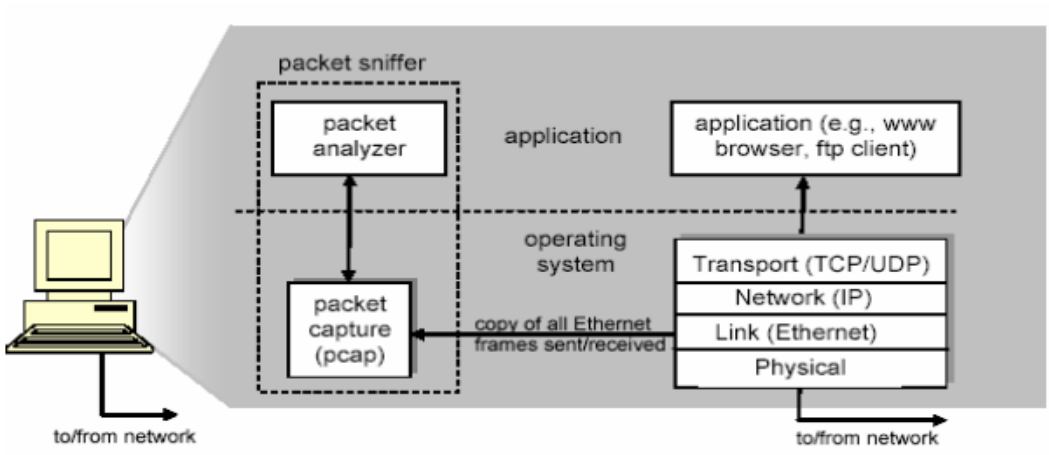


Figure 1: Packet sniffer structure

The second component of a packet sniffer is the packet analyzer, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must "understand" the structure of all messages exchanged by protocols.

Wireshark Installation

The Wireshark installation process is surprisingly simple. However, before you install Wireshark, make sure that your system meets the following requirement400 MHz processor or faster

- 128MB RAM
- At least 75MB of available storage space
- NIC that supports promiscuous mode
- WinPcap capture driver

The WinPcap capture driver is the Windows implementation of the pcap packet-capturing application-programming interface (API). Simply put, this driver interacts with your operating system to capture raw packet data, apply filters, and switch the NIC in and out of **promiscuous mode**.

Installing on Microsoft Windows Systems

The first step when installing Wireshark under Windows is to obtain the latest installation build from the official Wireshark web page, http://www.wireshark.org/. Navigate to the Downloads section on the website and choose a mirror. Once you have downloaded the package, follow these steps:

1) Double-click the .exe file to begin installation, and then click Next in the introductory window.
2) Read the licensing agreement, and then click I Agree if you agree.
3) Select the components of Wireshark you wish to install, as shown in Figure 3-1. For our purposes, you can accept the defaults by clicking Next.
4) Click Next in the Additional Tasks window.
5) Select the location where you wish to install Wireshark, and then click Next.
6) When the dialog asks whether you want to install WinPcap, make sure the Install WinPcap box is checked, as shown in Figure 3-2, and then click Install. The installation process should begin.
7) About halfway through the Wireshark installation, the WinPcap installation should start. When it does, click Next in the introductory window, read the licensing agreement, and then click I Agree.
8) WinPcap should install on your computer. After this installation is complete, click Finish.
9) Wireshark should complete its installation. When it's finished, click Next.
10) In the installation confirmation window, click Finish.

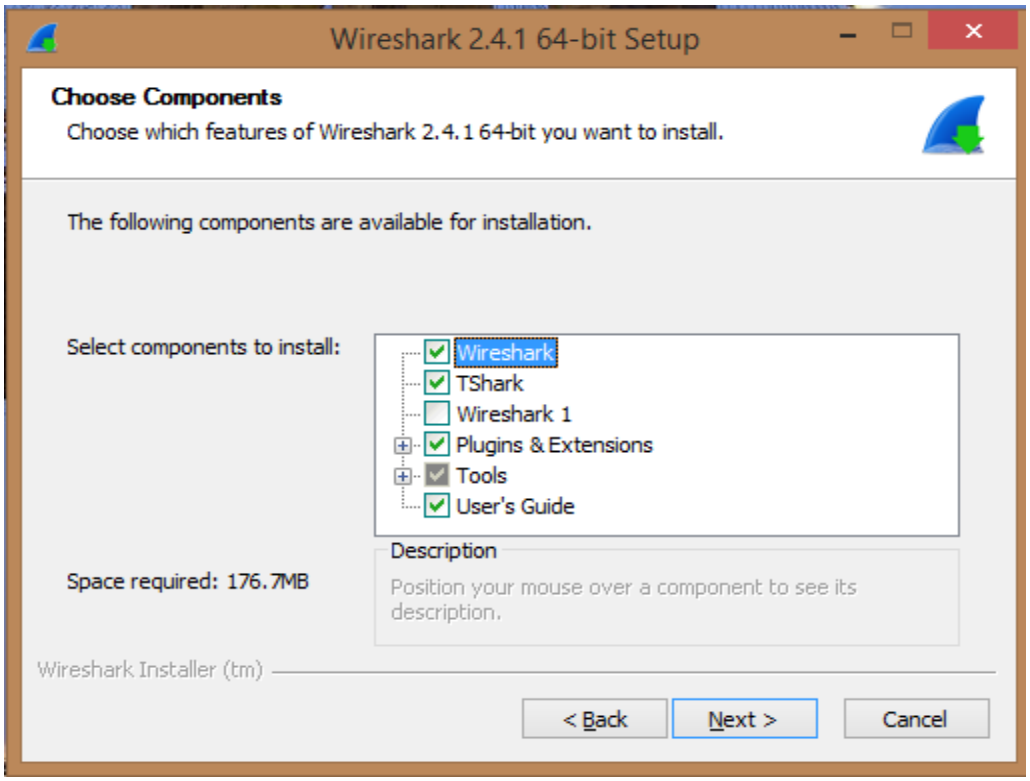


Figure 2: Installation Packages

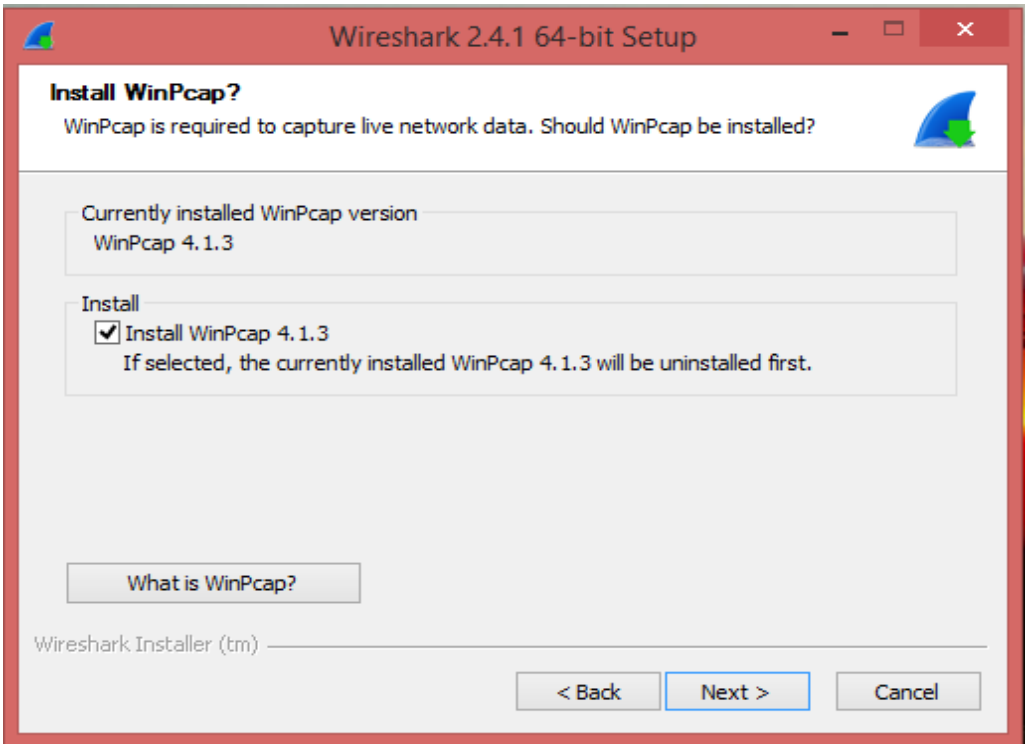


Figure 3: WinPcap Installation

Using Wireshark

1) Open Wireshark.
2) Select a Network Interface to Capture Packets through. When Wireshark is first run, a default, or blank window is shown. To list the available network interfaces, select the **Capture->Interfaces** menu option. Windows can have a long list of virtual interfaces, before the Ethernet Network Interface Card (NIC).
3) Choose the interface you wish to use, and click Start, or simply click the interface under the Interface List section of the welcome page. Data should begin filling the window.
4) Wait about a minute or so, and when you are ready to stop the capture and view your data, click the Stop button from the Capture drop-down menu.
5) Once you have completed these steps and finished the capture process, the Wireshark main window should be alive with data. In fact, you might be overwhelmed by the amount of data that appears, but it will all start to make sense very quickly as we break down the main window of Wireshark one piece at a time.

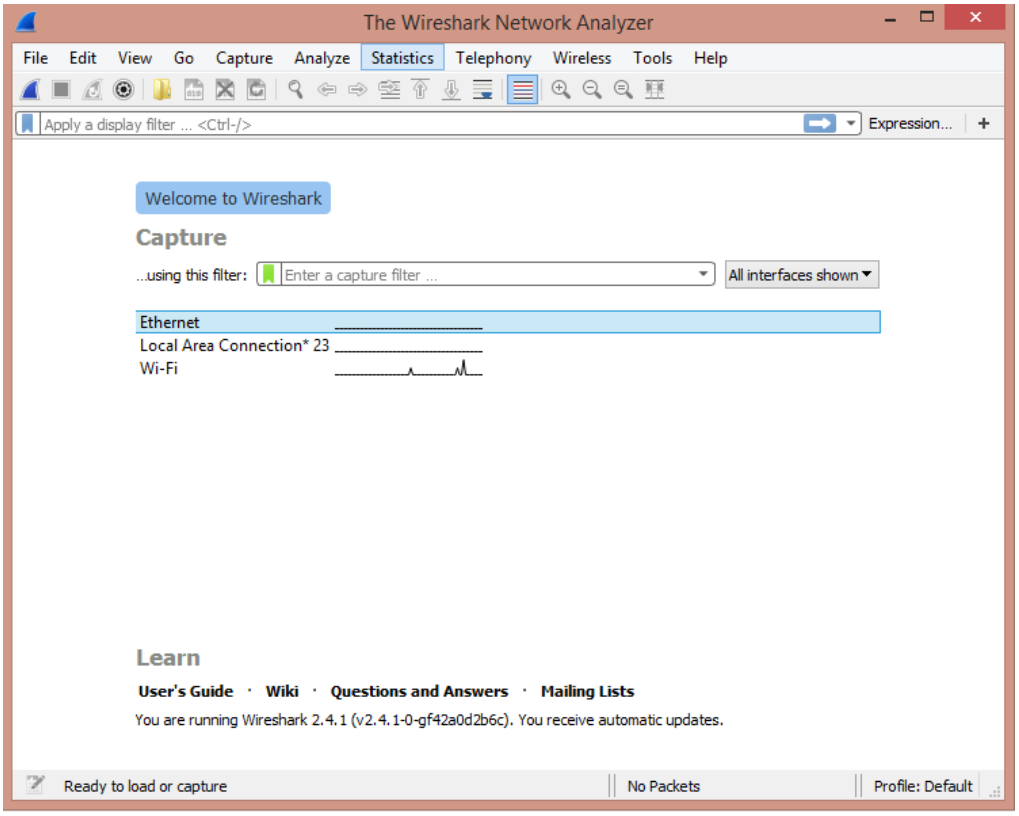


Figure 4: NIC Selection

Wireshark's Main Window

You'll spend most of your time in the Wireshark main window. This is where all of the packets you capture are displayed and broken down into a more understandable format. Using the packet capture you just made, let's take a look at Wireshark's main window, as shown in the figure below.

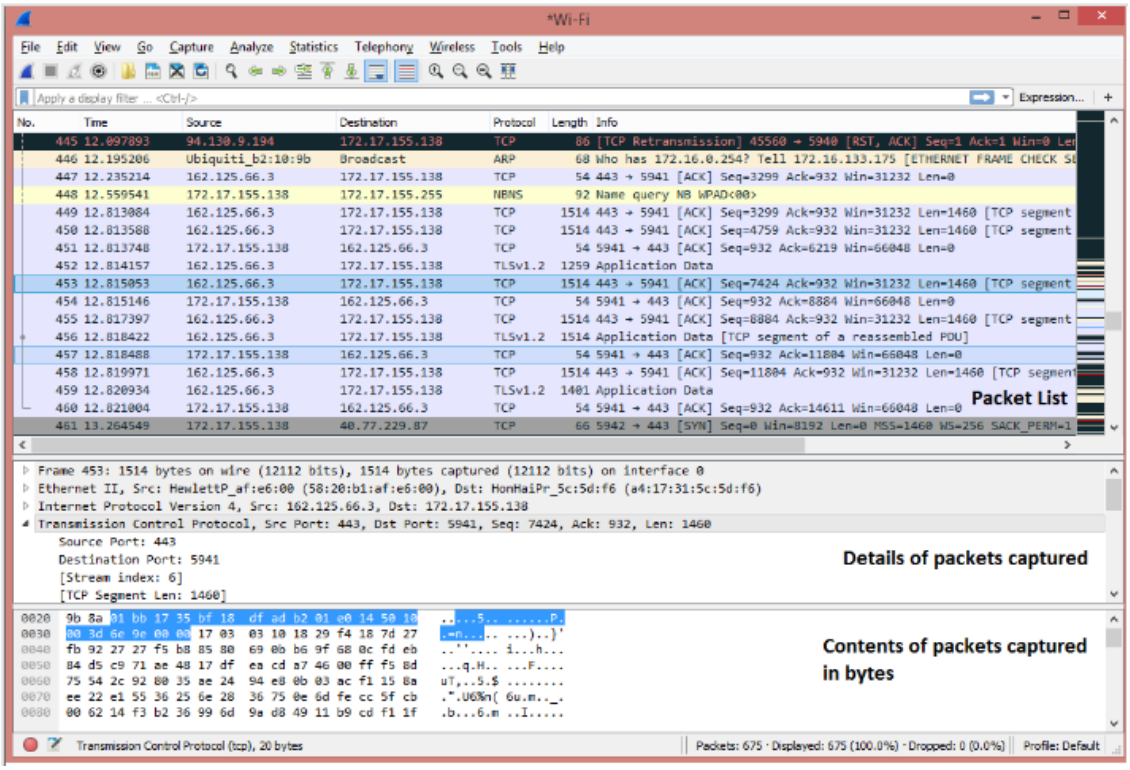


Figure 5: Wireshark Main Window

The three panes in the main window depend on one another. In order to view the details of an individual packet in the Packet Details pane, you must first select that packet by clicking it in the Packet List pane. Once you've selected your packet, you can see the bytes that correspond with a certain portion of the packet in the Packet Bytes pane when you click that portion of the packet in the Packet Details pane.

Packet List: The top pane displays a table containing all packets in the current capture file. It has columns containing the packet number, the relative time the packet was captured, the source and destination of the packet, the packet's protocol, and some general information found in the packet.

Packet Details: The middle pane contains a hierarchical display of the information about a single packet. This display can be collapsed and expanded to show all of the information collected about an individual packet.

Packet Bytes: The lower pane, perhaps the most confusing one displays a packet in its raw, unprocessed form; that is, it shows what the packet looks like as it travels across the wire. This is raw information with nothing warm or fuzzy to make it easier to follow.

Display Filters

Wireshark automatically generates a Display Filter, and applies it to the capture. The filter is shown in the Filter Bar, below the button toolbar. Only packets captured with a Source Port of the value selected should be displayed. The window should be similar to that shown in Figure 6.

This same process can be performed on most fields within Wireshark, and can be used to include or exclude traffic.

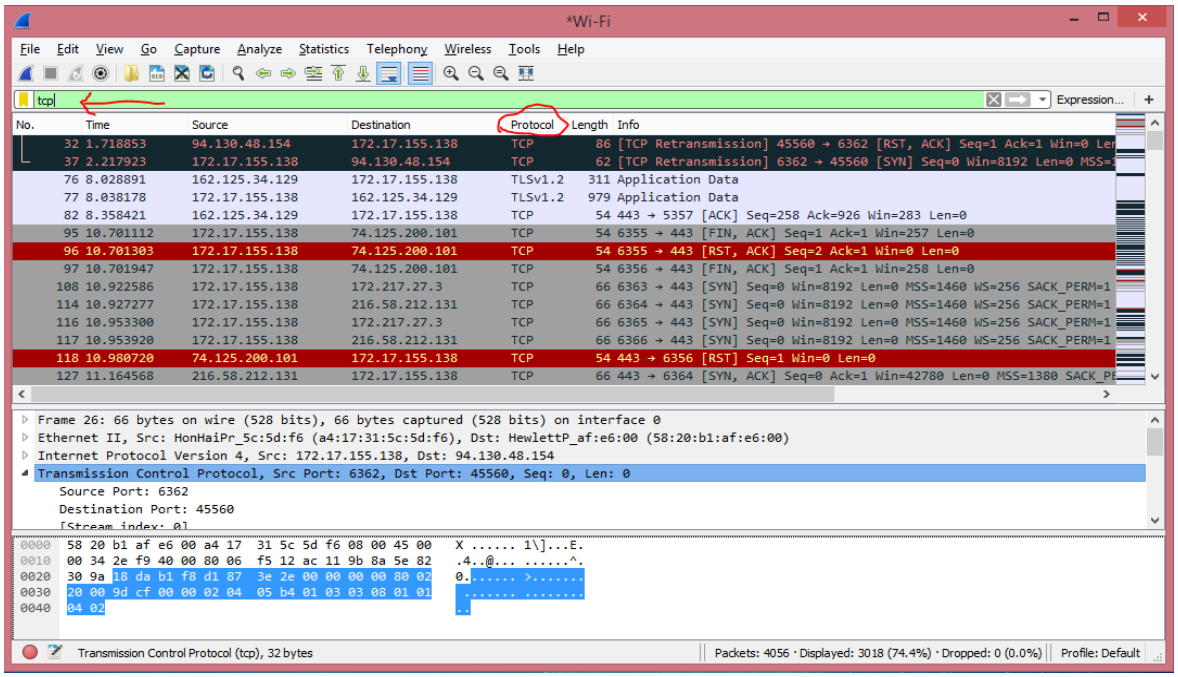


Figure 6: Applying display Filters

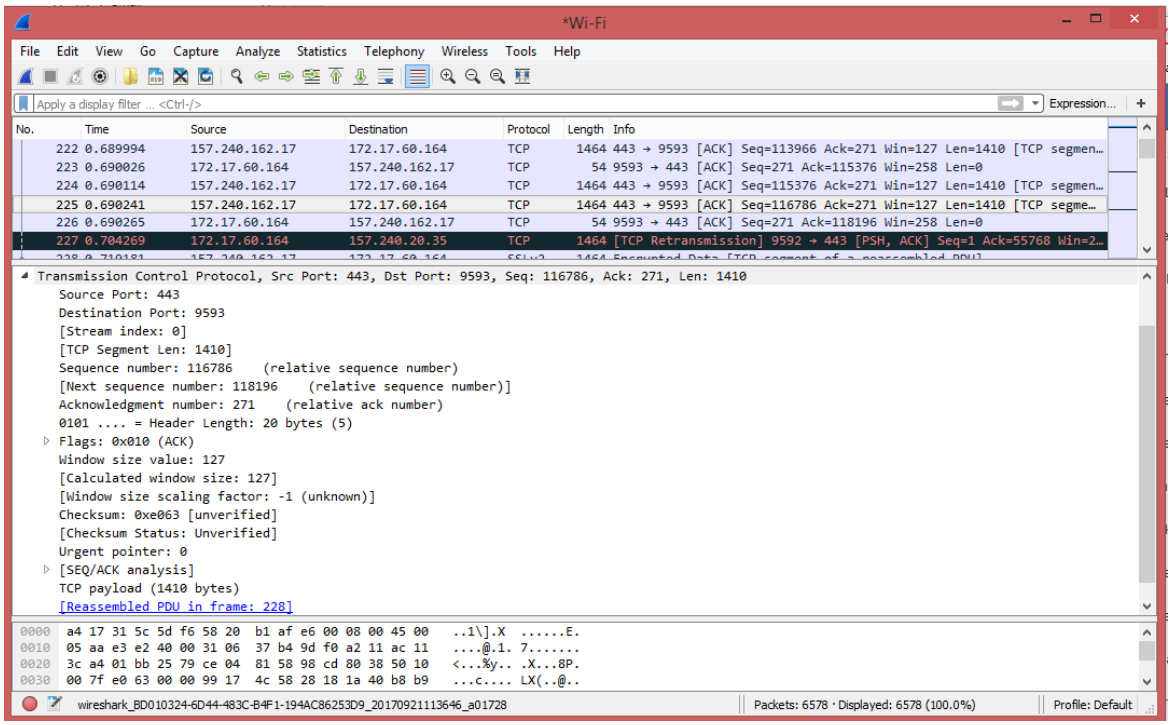


Figure 7: Selected TCP packet details

Keyboard Navigation

The table below shows some useful keyboard navigation shortcuts from which you should be familiar to use this software.

Accelerator	Description
Tab, **Shift+Tab**	Move between screen elements, e.g. from the toolbars to the packet list to the packet detail.
Down	Move to the next packet or detail item.
Up	Move to the previous packet or detail item.
Ctrl+Down, **F8**	Move to the next packet, even if the packet list isn't focused.
Ctrl+Up, **F7**	Move to the previous packet, even if the packet list isn't focused.
Ctrl+.	Move to the next packet of the conversation (TCP, UDP or IP).
Ctrl+,	Move to the previous packet of the conversation (TCP, UDP or IP).
Alt+Right or **Option+Right** (macOS)	Move to the next packet in the selection history.
Alt+Left or **Option+Right** (macOS)	Move to the previous packet in the selection history.
Left	In the packet detail, closes the selected tree item. If it's already closed, jumps to the parent node.
Right	In the packet detail, opens the selected tree item.
Shift+Right	In the packet detail, opens the selected tree item and all of its subtrees.
Ctrl+Right	In the packet detail, opens all tree items.
Ctrl+Left	In the packet detail, closes all tree items.
Backspace	In the packet detail, jumps to the parent node.
Return, **Enter**	In the packet detail, toggles the selected tree item.

Figure 7: Keyboard shortcuts

Lab Tasks:

1.1) Open wireshark and get acquainted with the tool. Clear your web browser's history and visit some webpage (for instance www.google.com) and capture the packets exchanged.
1.2) Navigate through different packets to identify the packets having different headers (protocol layers)
1.3) Use filters to show only the incoming packets from the webserver.
1.4) Use filters to show the packets not exchanged with the webserver.
1.5) Use filters to show the packets only exchanged within your IP address.

LAB 2: Shell Commands

Objectives:

- Getting familiar with Command Prompt (Windows) and Terminal (Linux)
- Introduction to some Basic Networking Commands in Windows and Linux
- Getting familiar with Linux Programming Environment.

Common Windows Commands

Any relatively modern Operating System (Windows, Ubuntu, OS X, Fedora, you name it) has a **Command Prompt** or a **Terminal**. This utility can be used to quickly check some of the network settings and can also be used to edit them to some extent. Knowing these small but useful commands can be very helpful if you want to quickly lookup some networking detail of your machine or the network that you are on.

Open Command Prompt in Windows and start typing these commands. Some of the commands to be covered in this lab are as follows:

- **ipconfig,**
  ipconfig command (and its variations) will display the IP configuration of the machine that you are currently on.
- **ipconfig /all**
  ipconfig /all will provide a more detailed account of the same settings.
- **ipconfig /release**
  Ipconfig /release command will release all matching connections. In simple words, it will disconnect all the active network connections on the machine (LAN or WiFi).
- **ipconfig /renew**
  ipconfig /renew will renew the connections i.e. it will refresh the existing settings with new ones.
  These commands can be the first step in quickly diagnosing and fixing a network problem. In addition, they also provide a plethora of information about the IP configuration of the machine on which they are run.
- **arp –a**
  This command is useful if you want to know that which IP address on the network is bound to which Physical address or more commonly known as MAC. It will print a list of all the IP address and their corresponding MAC address.
- **ping**
  ping is the command that can be used to check whether a particular device is online or not. A device can be pinged by using the ping command as follows: ping Device_IP_Address. When a device is pinged, a small amount of data is sent from your device to the device that you are pinging. The receiving device also sends some data to your machine. The time it takes for the data packet to make the roundtrip is displayed on the terminal. You can try pinging one of Google's server. Use ping 8.8.8.8.

- **pathping**

  pathping is an extension of the ping command. Using pathping, you can actually see the network path or route that the data packets take to reach their destination. This can be useful if you want to see the exact path from your machine to the machine that you are pinging.
- **tracert**

  tracert commands not only displays the route that the data packets take to reach the destination but this command also pings every router on the way. The command then displays the stats from each router that it encounters in the way. Try the command tracert 8.8.4.4 and check what routers are in the way.
- **netstat**

  Using the netstat command, you can see all the internet connections from your PC. The command will display the protocol that is being used, the address along with the port number on your PC, the address with the port number on the terminating end and the state of the connections. This command is very useful if you want to see a list of all the connections from your PC. It can also be used to weed out suspicious connections from your device.
- **nslookup**

  Using the nslookup command you can view all the IP address that are associated with a website. This command can be useful if you want to look up the IP address of a specific server that a company uses. Type nslookup google.com to view all the IP address that Google uses.

Common Linux Commands

Open the Virtual box and start the Ubuntu operating system installed in the virtual box. Once you see the desktop/ welcome screen, open up the Terminal. Now run these commands on the terminal and learn how each of the following command works.

- **pwd**

  Print working directory Pathnames enable you to work out where you are in relation to the whole file-system

- **ls**

  This command lists the contents of your current working directory. When you first login, your current working directory is your home directory. Your home directory has the same name as your username.

- **mkdir** testDirectory

  This command maes a new directory in your current directory, named as testDirectory in this case. To check whether the directory was created type ls after executing this command.

- **cd** testDirectory

It changes your current working directory from home to testDirectory. In Linux dot (.) means current directory, so if you type **cd .** it means to stay in the current directory. Aslo check how **cd ..** works**.**

- **rm**
  Remove a file from a directory. This command can only remove a file and an empty folder. If a folder contains files or subfolders, this command will give you an error. Then delete a folder containing subfiles and folders use **rm -r**
- **su**
  This command switched the user from less privileged mode to more privileged mode. Type **exit** to go back to less privileged mode.

- **sudo passwd root**
  In Ubuntu there is no root password by default. Use this command to set the root password.

- **man** command name
  This command opens the help manual for the command you have typed. In terminal use this command to check the functionality of the following commands from help manual. **cp, mv, grep, halt, ps, getpid**

- **Ifconfig**
  List IP addresses for all devices on the local machine By default it displays the status of the currently active interfaces.

- **Netstat**
  The netstat command can be used to display the configuration information and statistics on a network interface. The same command is also used to display the host routing table. Several netstat options are listed below that will be used frequently in the experiments. **netstat -a:** Shows the state of all sockets, routing table entries, and interfaces. **netstat -r:** Displays the routing table. **netstat -*i*** : Displays the interface information. **netstat -*n*:** Displays numbers instead of names. **netstat -*s*:** Displays per-protocol statistics.

- **tcpdump**
  it captures and displays packets on the LAN segment. By analyzing the traffic flows and the packet header fields, a great deal of information can be gained about the behavior of the protocols and their operation within the network. Problems in the network can also be identified. A packet filter can be defined in the command line with different options to obtain a desired output.

    a) **tcpdump –i eth0:** -i command specifies to listen packets from a particular interface. Eh0 in this case

b) **tcpdump –D:** Lists the available interfaces from where packets can be listened.
c) **tcpdump –c 5:** Only capture 5 incomming packets
d) **tcpdump -i eth0 -c 5 host 192.168.159.128:** listen 5 packets exchanged with host of this particular ip address, on interface eth0.
e) **tcpdump -i eth0 -c 5 src host 192.168.159.128:** listen 5 packets incoming from the host of this particular ip address, on interface eth0.
f) **tcpdump -i eth0 -c 5 dst host 192.168.159.128:** listen 5 packets outgoing to the host of this particular ip address, on interface eth0.
g) **tcpdump -i eth0 tcp:** Only capture tcp packets coming from eth0
h) **tcpdump -i eth0 arp:** Only capture arp packets coming from eth0

- **sudo apt-get install g++**
  This command installs the g++ compiler for compiling C++ codes. To check that the right version of g++ is installed on your machine, give the following command **g++ -v**

- **sudo apt-get install geany**
  You can install Geany the code editor from terminal by using this command. Once Geany is installed, run the following command in a terminal. This will incorporate a terminal within Geany and will come in handy when compiling and running the codes.

Lab Tasks

2.1) Read the main pages for the following programs and learn the usage of these commands

Arp
Arping
Ifconfig
Tcpdump
dig
Netstat
Route
Ping

2.2) Capture Network Packets using tcpdump and classify it based on TCP and UDP protocols.
2.3) List the number of received packets, accepted and discarded packets during next 5 minutes
2.4) List the number of current TCP connections on your machine

2.5) Displays all active TCP connections and the TCP and UDP ports on which the computer is listening.

Reference link:

https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/netstat

LAB 3: Understanding Network Devices and Building a Small Network using CISCO Packet Tracer

Objectives:

- Introduction to basic Networking Devices
- Introduction to Networking simulation tool CISCO Packet Tracer
- Simulation of Local Area Network by using star topology

Prerequisites:

- Knowledge of IP Addresses Classes and their Range
- Knowledge of the essential Networking Commands.
- Familiarity with Command Prompt (Windows)
- CISCO packet Tracer installed

Network Devices:

Some of the basic networking devices that we will be using in this course are as follows:

- Hub
- Switch
- Router
- Networking Cables
- RJ-45 Network Cable Tester

Network Hub

Network Hub is a networking device which is used to connect multiple network hosts. A network hub is also used to do data transfer. The data is transferred in terms of packets on a computer network. So when a host sends a data packet to a network hub, the hub copies the data packet to all of its ports connected to. Like this, all the ports know about the data and the port for whom the packet is intended, claims the packet. However, because of its working mechanism, a hub is not so secure and safe. Moreover, copying the data packets on all the interfaces or ports makes it slower and more congested which led to the use of network switch.

Figure 1: Network Hub

Network Switch:

Switch is the device used to connect devices in a same network. Switch is more intelligent than a hub. While hub just does the work of data forwarding, a switch does 'filter and forwarding' which is a more intelligent way of dealing with the data packets. Therefore, when a packet is received at one of the interfaces of the switch, it filters the packet and sends only to the interface of the intended receiver.

Figure 2: Network Switch

Network Router:

The router is used to connect multiple networks to one another. Multiple routers can be connected to form a bigger network. A router can be connected to another router using a Serial DCE or a crossover Ethernet cable. Routers are network devices that literally route data around the network. By examining data as it arrives, the router can determine the destination address for the data; then, by using tables of defined routes, the router determines the best way for the data to continue its journey.

Figure 3: Network Routers

Network Cables

There are two type of network cables that are most widely used to interconnect the networking devices.

- Cross Over Cables
- Straight Through Cables

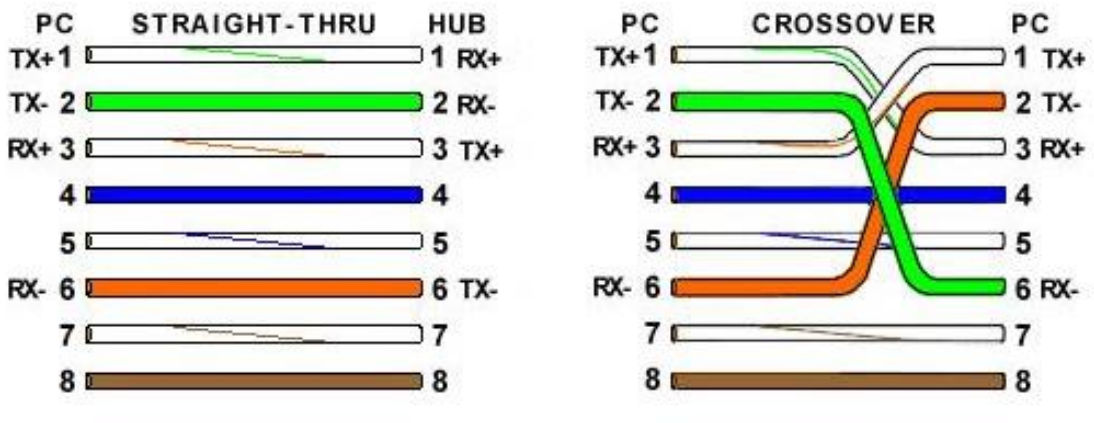


Figure 4: Network Cable Configuration

Cross over cables are used to connect the similar devices like switch with a switch, PC with a PC, hub with a hub. Straight Through cables are used to connect different type of devices, like PC with a Switch, PC with a router etc. Note that some devices such as routers will have advanced circuitry, which enables them to use both crossover and straight-through cables.

	Hub	Switch	Router	PC
Hub	Crossover	Crossover	Straight	Straight
Switch	Crossover	Crossover	Straight	Straight
Router	Straight	Straight	Crossover	Crossover
PC	Straight	Straight	Crossover	Crossover

Table 1: Type of network cable connection

RJ-45 Network Cable Tester:

An RJ-45 cable tester is used to ensure that the cable is not faulty and the connection is working as intended. Majority of the testers consist of two main components: the tester itself and the remote. Once the cable is connected to both the tester and the remote, users should turn the tester on to send a signal up the cable, which lights up the LEDs on both the tester and remote. It not only works as a continuity tester, but more importantly, shows the user exactly what kind of network cable they have and indicates any potential problem.

Figure 5: RJ-45 Network Cable Tester

Crimping Tool:

A crimping tool, shown below is used to make Ethernet cables. This tool is the only thing we need to completely strip the Ethernet wire and add an RJ-45 connector on its end.

Figure 6: Crimping Tool

Cisco Packet Tracer

Cisco Packet Tracer is a simulation software for prototyping networks before they are created in real life. The simulation allows for testing the networks to check whether it will perform as needed and to solve any problems beforehand. All the devices that Cisco manufactures like switches and routers etc. can be found in the Cisco Packet Tracer.

Procedure:

Install CISCO packet Tracer 7.0 in your systems. Once the Cisco Packet Tracer has been installed, running the software will display the following screen. (A Cisco account can be created if needed. Otherwise, a guest session will be more than enough for all the simulation purposes.)

Any network topology – from simple to complex – can be created by simply dragging and dropping the required devices and connecting them as needed. A topology that has been created can be saved as .pkt file for future use. The Simulation Mode of Cisco Packet Tracer allows to track the data packets sent from one PC to the next at every stage.

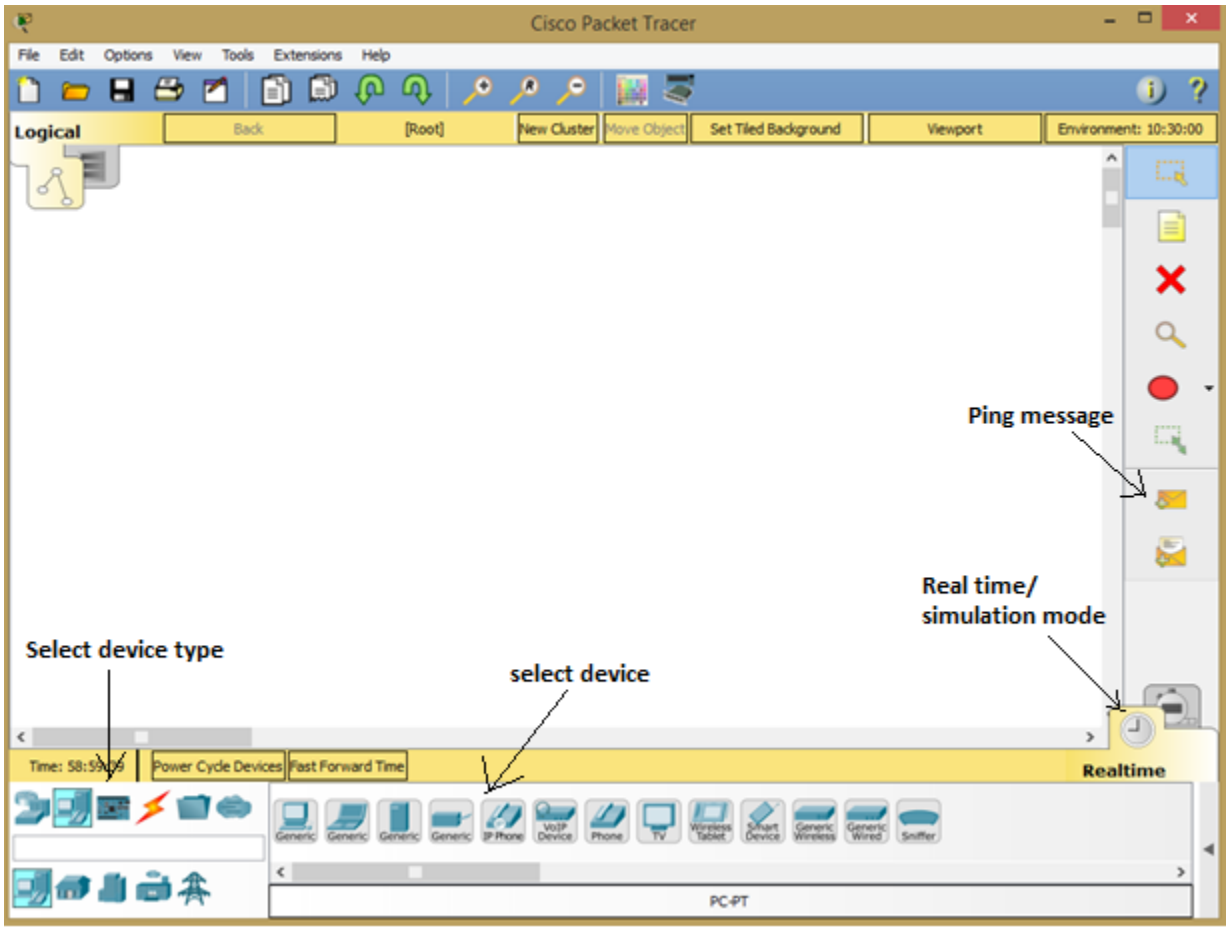


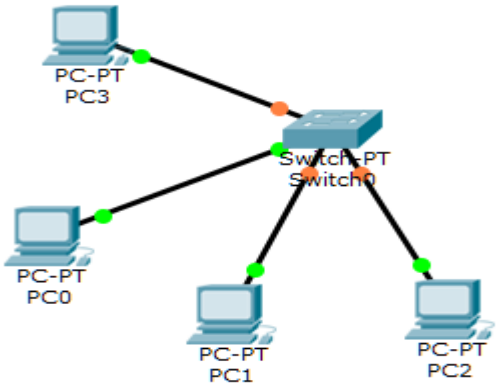
Figure 7: CISCO Packet Tracer Main Window

Creating a Local Area Network in Cisco Packet Tracer:

Procedure:

a) Drag and drop a generic Switch from the Switch Hub onto the canvas.

b) Drag and drop 4 PCs or laptops and connect them using a straight through cable with the switch as shown here:



c) Stright through cables are used when connecting different types of devices i.e. a PC to a Switch or a Switch to a Router. When two same type of devices are to be connected, a crossover cable is used.

d) Once the required topolgy has been created, the lights at the end of each wire between the PC and the Switch will turn green indicating that the link is up i.e. the link is functioning properly.

e) Even though the lights have turned green, the PCs won't be able to communicate with eachother because the PCs themselves haven't been configured. In order to configure the PCs, we will use the IP Configuration utitly provided by the OS running on the PC in Cisco Packet Tracer. Click on the PC then navigate to the Desktop tab and then click on the icon labelled IP Configuration. You should see a window as shown below:

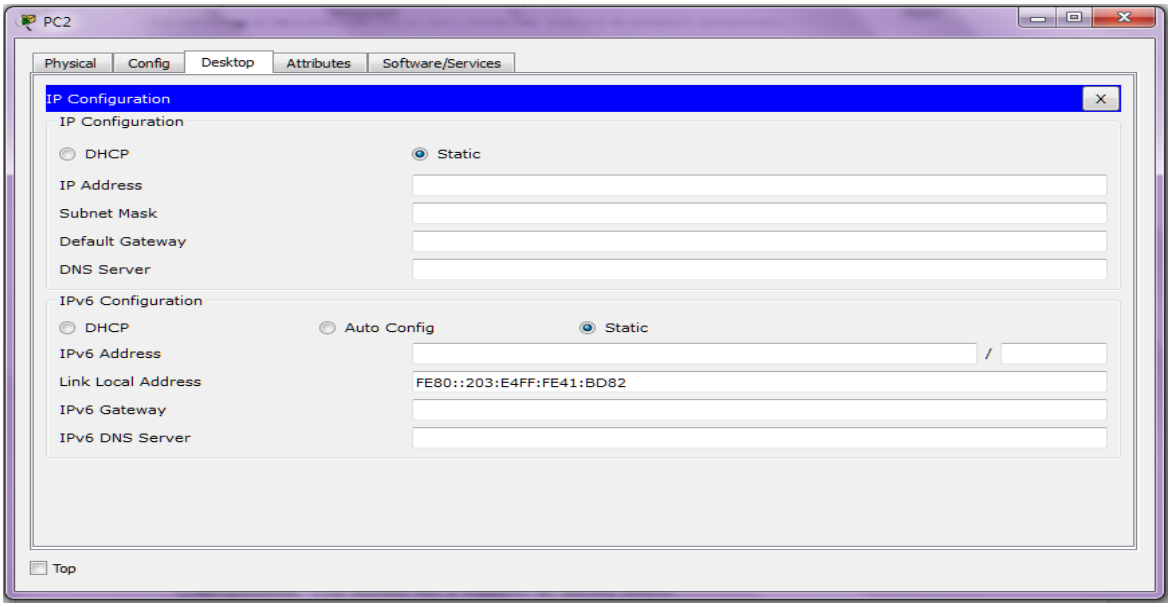


Figure 8: IP Configuration procedure of PC

The first four rows are where you will enter the required settings. The IP address will contain the IP address of that particular PC. You can use the default subnet mask for the class of the IP address that you are using. For now, you can skip the Default Gateway and DNS Server settings as we are just configuring a Local Area Network. Remember that all the PCs on this LAN should share the same network address.

f) Once you have configured the PCs, you can check your connection by pinging one PC by another. To do that, click on a PC, navigate to Desktop tab and click on Command Prompt. Type ping and the IP address of the PC that you want to ping. The results should appear below the command.

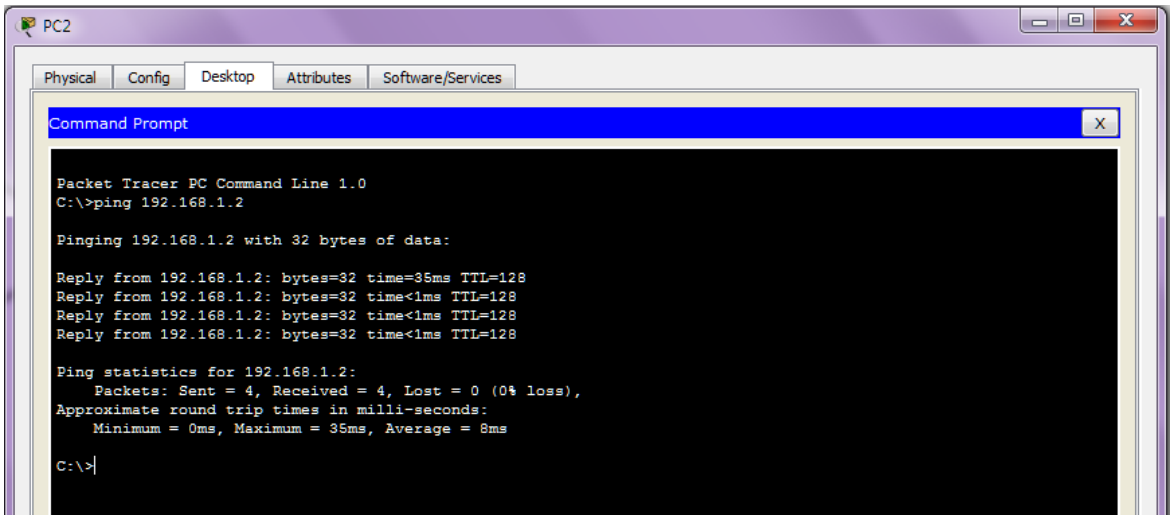


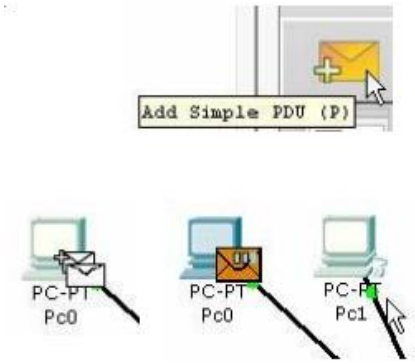
Figure 9: Running Ping Command

Packet Tracer allows you either to issue the command from the command prompt or to use the Add Simple PDU tool. Both methods will be used.

To enter Simulation Mode click the Simulation Mode tab in the lower right hand corner of the interface. In order to view only the “pings”, in the Event List, click on ALL/NONE to clear all protocols, and then click on ICMP to select only that protocol.

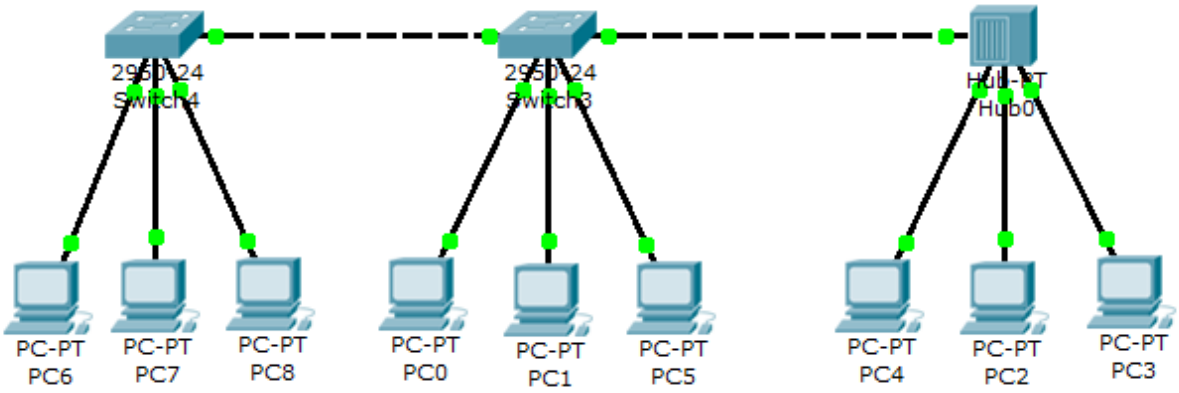
Figure 10: Network packet simulation filters

Using the Add Simple PDU perform a ping from PC0 to PC1. Choose the Add Simple PDU tool from the toolbox. Click once on PC0, the device issuing the ping (ICMP Echo Request) and then click once on PC1 (the destination of the ICMP Echo Request). Run the simulation using the Capture / Forward button.



Lab Tasks:

Build the following network in Packet Tracer:



3.1) Assign first 6 PC's with class C addresses and last three with Class B addresses. Now send packets using PDU tool from PC0 to PC3. Explain if the packet transmits?

3.2) Now assign all the PC's with class C addresses and again send packets using PDU tool from PC0 to PC3. Explain does this ping this time?

3.3) Using PDU tool, send the packets from PC6 to PC4 and observe the transmission of switch and the hub. How they differ?

LAB 4: Connecting Multiple Networks in CISCO Packet Tracer

Objectives:

- Understanding of the working of router
- Configuration of different ports of the router
- Connect multiple networks by using Routers

Prerequisites:

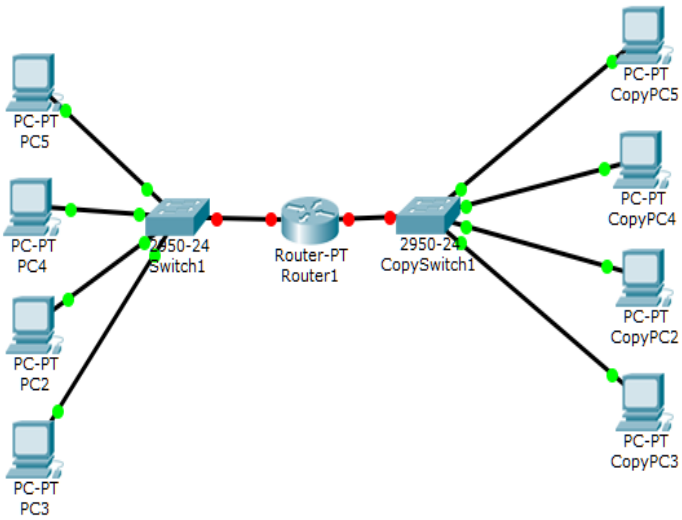
- Knowledge of essential Networking commands
- Knowledge of the basic Networking devices.
- Familiarity with CISCO Packet Tracer
- Understanding of the working of LAN

Procedure:

a) We are now familiar with the creating a single network for simulation in Cisco Packet Tracer. For this task, create a local area network following the same procedure of the last lab.

b) Once a single local network has been created, build a similar network on the other side of the router.

c) Select a generic router and place it between the two networks.



d) Now connect the switches with the router by connecting straight through cable with Fast Ethernet port of the switch and Fast Ethernet port of the router. You can add PT-ROUTER-NM-1CFE port if Fast Ethernet port is not available.

e) Once the first network has been created, Assign IP addresses to PCs of both the networks. Remember that these are different networks, so the network address of second network will be different from the first one.

f) Also assign first address of the network as default gateway to all the PCs in that network. e.g 192.168.1.1 can be a default gateway.

g) For configuring the router, click on the router and select the Config tab. Navigate to FastEthernet port on which connection was made. You should see something like this:

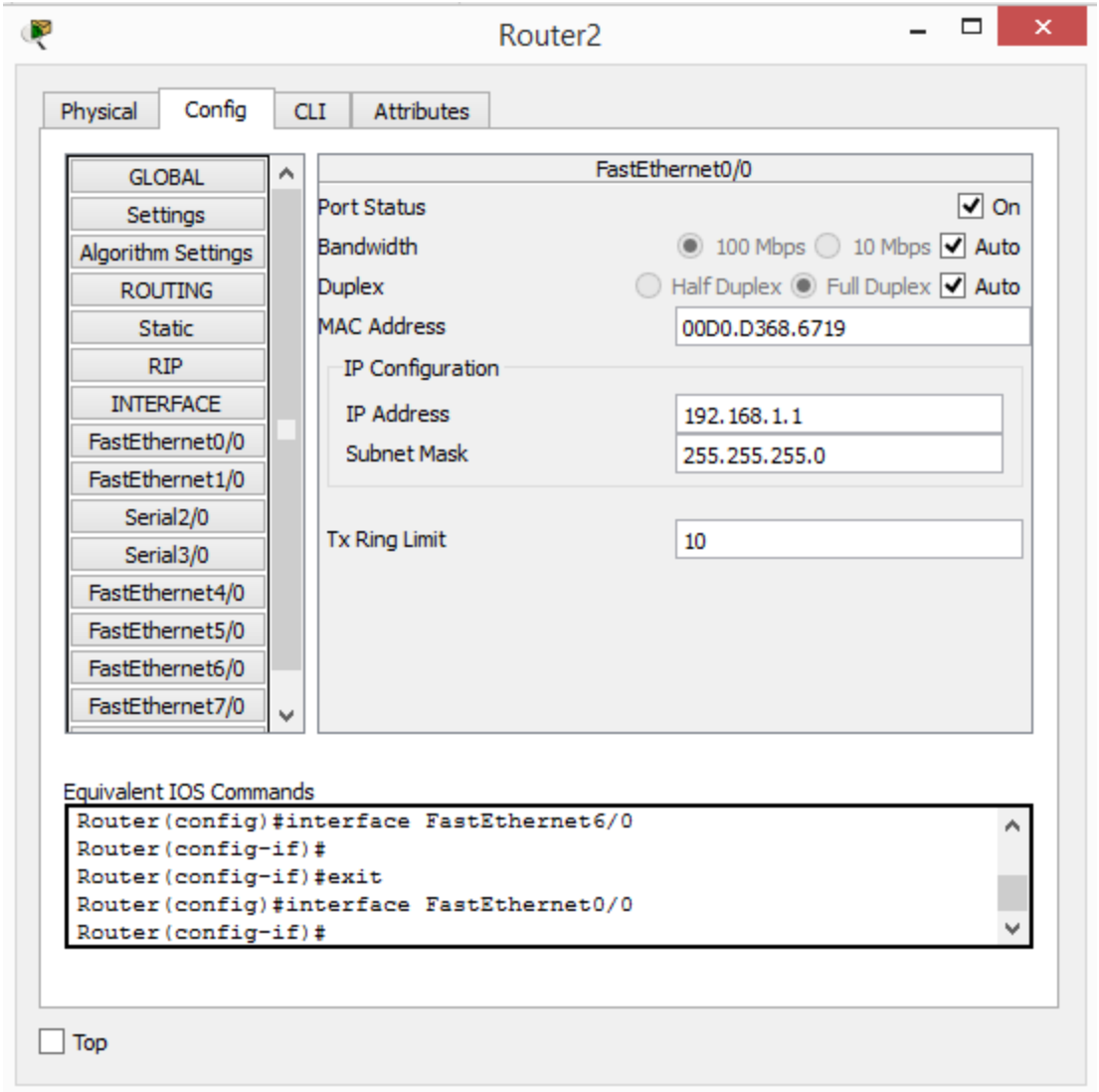


Figure 1: Fast Ethernet port configuration of the Router

h) Assign the default gateway of this network as an IP address of this Fast Ethernet port. Change the port status to ON.

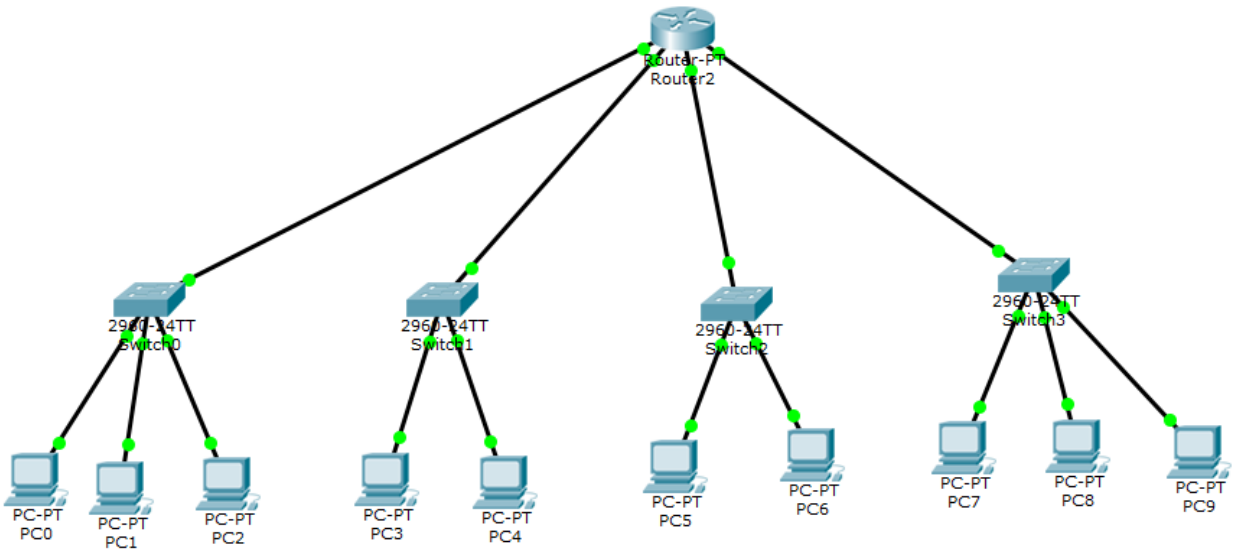
i) Repeat the steps for the second Fast Ethernet interface on the router. Remember to configure it in accordance to the network that it is attached to.

j) Once again open the PC settings and check the correct Default Gateway for both the networks.

k) Ping the PC in one network from the PC in the 2nd network to check whether the configuration has been done properly or not. If it has been done successfully, then the ping should be successful.

Lab Tasks:

Build the following network in Packet Tracer:



4.1) Create 4 different networks under 4 switches by assigning different class IP addresses to the PCs. Select PC1, go to command prompt and ping with PC 6, Is it successful?

4.2) Use inspect tool to check routing table, ARP table and other tables available in the router and examine how these tables are updated.

4.3) Now send packets using PDU tool from PC7 to PC3. Explain does this ping this time?

4.4) Now assign a same network IP address to last 5 PCs and Using PDU tool, send the packets from PC5 to PC9. Is it successful this time? Justify your answer with reason.

4.5) Write a report (1-2 pages) on types of memories and interfaces in a router.

LAB 5: Socket Programming

Objectives

- Get basic understanding of Socket
- Learn about Berkley Sockets
- Use TCP sockets to build client-server applications

Theory

The purpose of this lab is to learn about TCP sockets. A socket is basically an end point of a network communication path. This is an interface, where any application hands over data (for transmission to other applications) to the operating system and receives any incoming data from the operating system. For any application, to be able to communicate over network (or Internet), it needs to create at-lease one socket. Since, socket is an endpoint of a network communication path, there must be more than one endpoints (one at each application belonging to the same communication path). For instance, when you send an email to a friend, the email client at your computer creates a socket to send that email to the mail server, which has already created a socket to receive that email message.

Usually in a network communication session, one application acts as a server application and other application act as client. The server application creates a socket and wait for the client to connect. Therefore client application must know the IP address and port number of server application to start communication. On receiving any message from client, server learns about client's IP address and port number.

Berkley Sockets

The standard API for network programming in C is Berkeley Sockets. This API was first introduced in 4.3BSD Unix, and is now available on all Unix-like platforms including Linux, MacOS X, FreeBSD, and Solaris. A very similar network API is available on Windows. The BSD sockets API is written in the C programming language. Most other programming languages provide similar interfaces, typically written as a wrapper library based on the C API.

Procedure

Sockets provide a standard interface between the network and application. There are two types of socket: a stream socket provides a reliable byte stream transport service, while a datagram socket provides unreliable delivery of individual data packets. When used in the Internet environment, stream sockets correspond to TCP/IP connections and datagram sockets to UDP/IP datagrams. This exercise will only consider TCP/IP stream sockets. To use the sockets API, you must first include the appropriate header files:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
```

Once the headers have been included, you can create a socket by calling the socket function. This function returns an integer known as a file descriptor that identifies the newly created socket.

```
...
int fd = socket(AF_INET, SOCK_STREAM, 0);        //defined in socket.h
if (fd == -1) {
// an error occurred
...
}
...
```

AF INET parameter to the socket() call indicates that the Internet address family (IPv4) is to be used; AF INET6 would create an IPv6 socket.

SOCK_STREAM parameter indicates that a TCP stream socket is desired.

Third parameter (protocol) is not used with TCP or UDP sockets, and is set to zero.

Return Value: If an error occurs, -1 is returned and the global variable errno is set to indicate the type of error. If a positive number is returned, it acts as descriptor (file handler).

Hint: Type man socket on linux shell to check more details about socket

The socket that is created is unbound (no port number is allocated), and not yet connected to the network. In a TCP communication session, the server creates a socket, binds it to a well-known port, and listens for connections; meanwhile, the client creates a socket, and connects to the specified port on the server. Once the connection is established, either client or server can write data into the connection, where it is available for the other party to read.

TCP Server Socket

To make a newly created socket into a TCP server, it is first necessary to bind that socket to the appropriate well-known port. This is done using the bind() function, specifying the file descriptor, address, and port on which you wish to listen for connections:

```
...
if (bind(fd, (struct sockaddr *) &addr, sizeof(addr)) == -1) {
// an error occurred
...
}
...
```

The addr parameter to the bind() function is specified to be a pointer to a variable of type struct sockaddr. This structure is defined in the <sys/socket.h> header as follows:

```
struct sockaddr {
uint8_t sa_len;
sa_family_t sa_family;
char sa_data[22];
}
```

The struct sockaddr is intended to be generic. The sa_len and sa_family fields hold the size and type of the address, while the sa_data field is large enough to hold any type of address. It treats the address as an opaque piece of binary data.

Applications do not use a struct sockaddr directly. Rather, if they use IPv4 addresses, they instead use a struct sockaddr_in:

```
struct sockaddr_in {
uint8_t sin_len;
sa_family_t sin_family;
in_port_t sin_port;
struct in_addr sin_addr;
char sin_pad[16];
}
```

```
struct in_addr {
in_addr_t s_addr;
}
```

These structures are defined in the header, so you don't need to define them yourself. You will note that the struct sockaddr_in is exactly the same size in bytes as the struct sockaddr, and has the sin_len and sin_family fields in exactly the same places as the sa_len and sa_family fields. The sa_data field of the struct sockaddr is replaced by the sin_port, sin_addr and sin_pad fields, which have the same total size. A struct sockaddr in can therefore be freely cast to a struct sockaddr, and vice versa, since they have the same size and the common fields are laid out in memory in the same way. These definitions allow for a primitive form on sub-classing, where the Sockets API takes a generic superclass (struct sockaddr) while the program uses a subclass specific to IPv4 (struct sockaddr_in) or IPv6 (struct socaddr_in6) and casts to/from struct sockaddr as appropriate.

When creating a server, you need only specify the address family (IPv4 or IPv6) and the port to bind, allowing the system to listen on any available address. For example, an IPv4 server on port 80 would use:

```
struct sockaddr_in addr;
...
addr.sin_addr.s_addr = INADDR_ANY;
addr.sin_family = AF_INET;
addr.sin_port = htons(80);
if (bind(fd, (struct sockaddr *) &addr, sizeof(addr)) == -1) {
... // an error occurred
}
...
```

Once a socket has been bound to a port, you can instruct the operating system to begin listening for connections on that port using the listen() function:

```
...
int backlog = 10;
...
if (listen(fd, backlog) == -1) {
// an error occurred
..
}
```

The value of the backlog parameter to the listen() function specifies how many simultaneous clients can be queued up waiting for their connections to be accepted by the server before the server returns a "connection refused" error to new clients (note: this is not the maximum number of clients that can be connected at once, but rather the maximum number of clients that can be waiting for the server to accept() their connection at once; a value of 10 is reasonable unless your server is very busy and also slow to accept connections).

Finally, you can accept a new connection by calling the accept() function: If there are no clients waiting, then the accept() function will block until a client tries to connect. The cliaddr parameter will be filled in with the address of the client, and the cliaddr_len parameter will be set to the length of that address. The accept() function returns a new file descriptor, called connfd in this example, that represents the connection to this particular client (i.e., it's a new socket, connected to the client at the client at cliaddr). The original listening socket, and the corresponding file descriptor, remains untouched, and you can call accept() again on it to accept the next

```
int connfd;
struct sockaddr_in cliaddr;
socklen_t cliaddr_len = sizeof(cliaddr);
...
connfd = accept(fd, (struct sockaddr *) &cliaddr, &cliaddr_len);
if (connfd == -1) {
// an error occurred
...
}
```

connection. A server application will call accept() in a loop to accept new connections from

clients, then pass each connfd to a new thread for processing. Since there is only one listening socket, there's no benefit to calling accept() from multiple threads.

TCP Client Socket

A TCP client can connect to a server by calling the connect() function. This function takes three parameters: the file descriptor of a newly created socket, the address and port of the server it should connect to (cast to a struct sockaddr), and the size of the address:

```
struct sockaddr_in s_addr;
...
if (connect(fd, (struct sockaddr *) &s_addr, sizeof(s_addr)) == -1) {
... // an error occurred
}
...
```

here s_addr parameter carries the server's IP address and port number

```
struct sockaddr_in s_addr;
s_addr.sin_family  = AF_INET;   //Protocol family
s_addr.sin_port     = htons(80); //Remember the port number in server application!
inet_aton("127.0.0.1", &s_addr.sin_addr);          //Replace 127.0.0.1 with server IP address
```

Sending and Receiving Data

Once client and server are connected, you can send and receive data over the connection using the send and receive functions (alternatively, write() and read() functions can also be used). The send() function takes file descriptor (fd), a pointer to the data (char *), and the size of the data in bytes. It returns the number of bytes sent, or -1 if an error occurs:

```
int size;
char buffer[MAX_SIZE];
n = send(fd,buffer, BUFLEN,0);
n = recv(fd, buffer, BUFLEN, 0);
```

LAB Tasks

5.1) Create a simple TCP client server application to facilitate chat between a TCP client and a TCP server

5.2) Create a simple UDP chat application

5.3) When server receives a message from client, display client's IP address and port number along with the message!

LAB 6: Full Duplex Chat Application

Objectives

- Get familiar with use of threading
- Use POSIX threads to simultaneously send and receive data
- Create Full duplex chat application with help from send and receive threads

Theory

The purpose of this lab is to be able to use POSIX threads to create full duplex chat application between client and server. You are required to use the same TCP socket codes (client.c and server.c) learnt in LAB 5. Soft copy of these codes is also attached with the LAB manual. The TCP socket codes written in LAB 5 had a limitation. Both client and server were not able to simultaneously send and receive messages to each other. The reason is blocking function calls. Any application calling recv on socket blocks until some data is available at socket to collect. This stops the application from sending any data until it has received some data on socket. Therefore, each application (client and server) is configured to receive data in a separate thread. This only blocks that particular thread. So main application can continue to send data.

Threads

A thread is as an independent stream of instructions that can be scheduled to run as such by the operating system. To the software developer, the concept of a "procedure" that runs independently from its main program may best describe a thread. To go one step further, imagine a main program (a.out) that contains a number of procedures. Then imagine all of these procedures being able to be scheduled to run simultaneously and/or independently by the operating system. That would describe a "multi-threaded" program. For instance, a word processor performs grammar and spelling checking, formatting and automatic saving of document, all at the same time!

Creating and Running a Hello Word Thread

This exercise will guide you about developing a simple hello word thread by using POSIX threads (pthreads) API. To use POSIX threads, you must first include the pthread.h header file. Once the headers have been included, you can create a thread by calling the pthread_create function. The pthread_create function takes the function name to be run as a separate thread, and starts an independent thread. Suppose, we have to run a helloPrint function

```
void* helloPrint(void *info) {
        long val = (long) info;
        while (1) {
                cout<<"Output from thread "<<val<<endl;
                sleep(2);
        }
}
int main() {
        …
        pthread_create(&thread, NULL, helloPrint, (void*) val);
        …
}
```

Creating a Chat Application

To turn your TCP client server codes into full duplex chat application, add the following thread function in both, client.cpp and server.cpp to receive messages.

```
void *recvMsg(void *socket) {
        long fd = (long) socket;
        char buffer[100];

        while (1) {
                recv(fd, buffer, 100, 0);
                cout<<"Message from Server : "<<buffer<<endl;
        }
}
```

In your main module, call the thread function and pass the socket descriptor as parameter to the thread function.

```
Pthread_t thread;
Pthread_create(&thread, NULL, recvMsg, (void*) fd);
```

Here, fd is the socket descriptor. In server.cpp, it is the socket descriptor returned from the accept call (connfd). Whereas, in client.cpp, it is the descriptor used in connect call (fd).

LAB Tasks

6.1) Create a simple TCP server, that simultaneously communicates with two TCP clients.

LAB 7: DHCP

Objectives:

- Understanding how DHCP works
- Configuration of DHCP server
- Adding multiples scopes in the DHCP server

Prerequisites:

- Knowledge of configuring a Local Area Network
- Understanding of the working of Router
- CISCO packet Tracer installed

Running DHCP Service on a Server

DHCP which allows dynamic assignment of IP addresses as well as remote booting of diskless workstations. In addition to supporting the dynamic assignment of IP addresses, DHCP supplies all configuration data required by TCP/IP, plus additional data required for specific services.

As noted, this functionality simplifies tasks for the network administrator, who can now manually configure just one computer—the DHCP server. Whenever a new computer starts on a network segment that is served by the DHCP server (or an existing computer is restarted), the computer asks for a unique IP address and the DHCP server assigns one from the pool of available addresses.

As figure 1 shows, this process requires four steps:

1. The DHCP client asks for an IP address (a DHCP Discover message).
2. The DHCP Server offers an address (a DHCP Offer message).
3. The DHCP client accepts the offer and requests the address (a DHCP Request message).
4. The DHCP Server officially assigns the address to the client (a DHCP Acknowledge message).

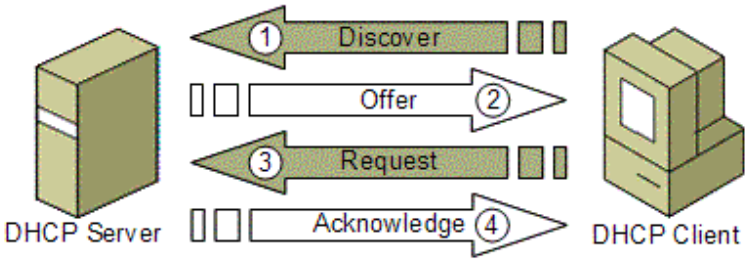


Figure 1. DHCP DORA Packets exchanged

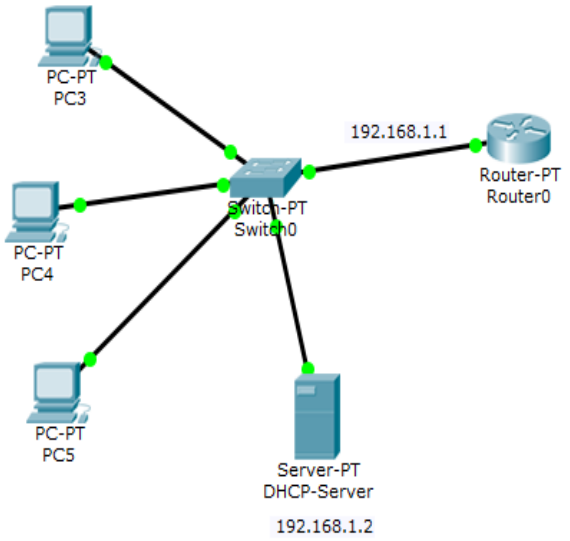
The DHCP server places an administrator-defined time limit, called a lease, on the address assignment. Halfway through the lease period, the DHCP client requests a lease renewal, and the DHCP server extends the lease. As a result, when a computer stops using its assigned IP address (for example, upon relocation to another network segment), the lease expires and the address returns to the pool for reassignment.

DHCP Scope is a range of IP addresses that you configure in your DHCP server as range of addresses that designed for distribution to the clients. For example, if you set a scope with a range from 10.0.0.100-10.0.0.200, you can easily provide only from this range IP addresses to your clients. You can also create more than one scope, but it's recommended to check that your scopes aren't duplicating one with each other's. In some cases, you'll need to prevent the client using some addresses, for example if your scope is from 10.0.0.1 up to 10.0.0.100, and your servers using 10.0.0.1-10.0.0.10, you can exclude these IP addresses from the scope and exclude the DHCP to distribute them to the clients, in most of the DHCP servers this option called exclude.

Procedure:

a) Create the topology as shown here. Remember that there needs to be at least one server in the network on which the DHCP runs.

b) For configuring the networks this time, you only need to give a static IP address to the DHCP servers. The PCs will obtain the IP addresses automatically thanks to the DHCP running on the servers. In addition, you will also have to configure the ports on the router as you have done before. Assign static address 192.168.1.2 to DHCP server and set 192.168.1.1 as its default gateway.



c) For running DHCP on the server, click on the server and navigate to the services tab. There you will see a bunch of services that can run on the server. Click on the DHCP and you should see the following interface.

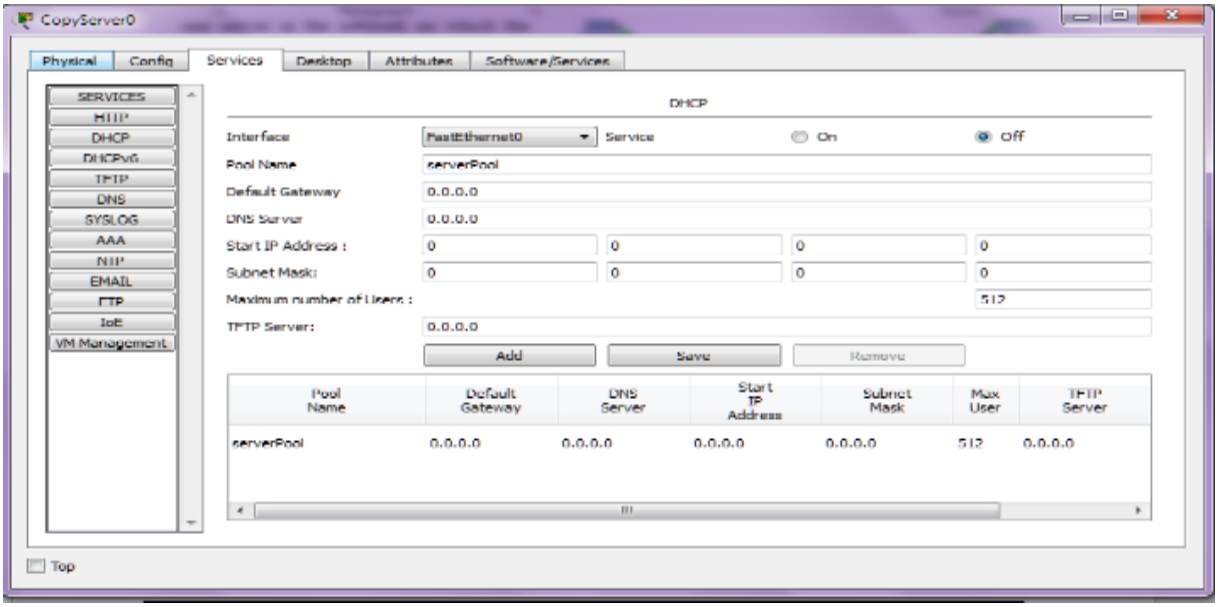


Figure 2: DHCP Configuration

d) The details that you need to provide to set up DHCP service are: Pool Name, Default Gateway, DNS Server, Starting IP address, Subnet mask and maximum number of users. Default Gateway and DNS Server are needed because when a host device request automatic IP assignment, these bits of information are also required for the hosts to communicate with other hosts. You can also limit the number of users by providing the exact number of maximum number of users.

e) Provide all the required information needed to configure the DHCP service. Repeat the process for the other server on the other network.

f) In the PC configuration, select DHCP (as shown below) to automatically get the IP address

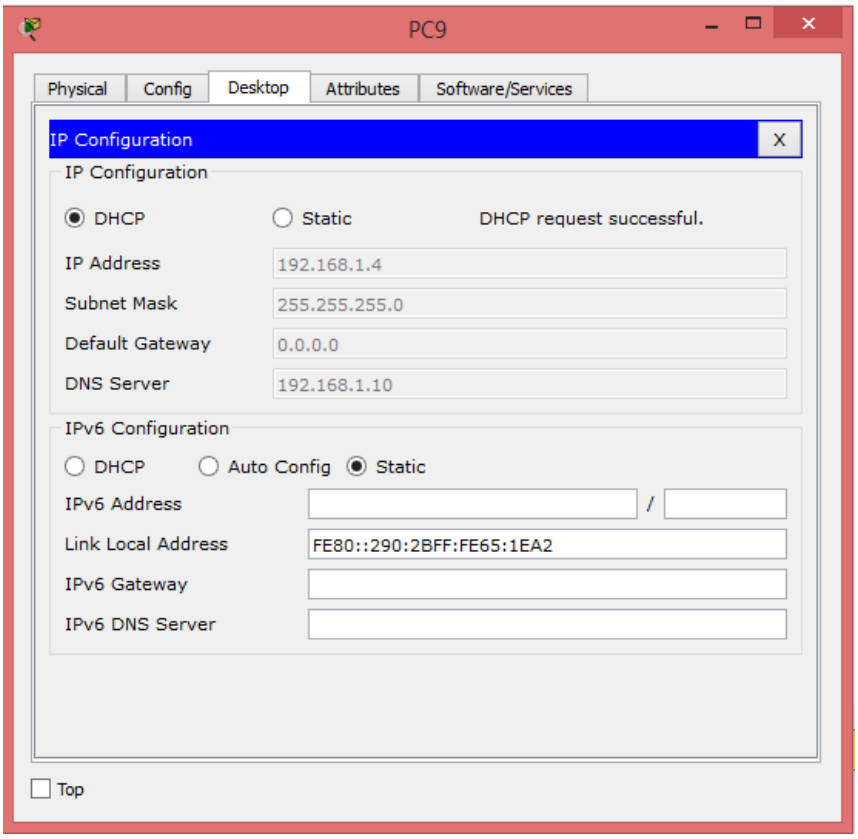
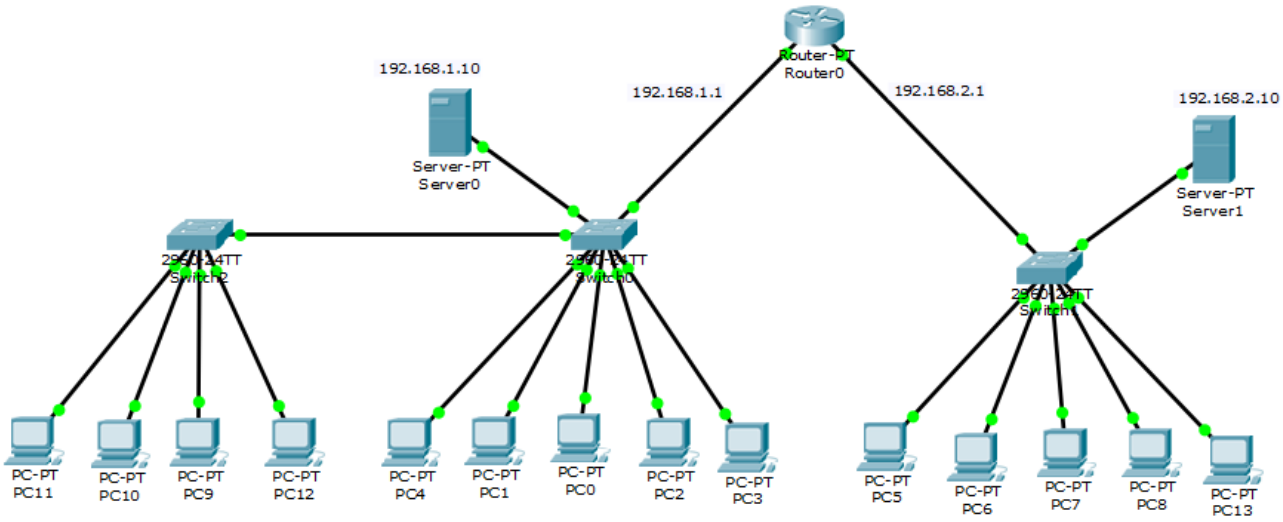


Figure 3: Dynamic Allocation of IP address

Lab Tasks:

Build the following network in Packet Tracer. Assign the static IP address and their corresponding default gateway to each DHCP server as shown in the figure below. Create the pools to define the scope of DHCP server by mentioning the starting IP address and maximum number of users.



7.1) Go to simulation mode, enable the DHCP filter, now dynamically allocate IP address to each of the PC in the two networks and monitor the packets exchanged during the process.

7.2) For the network 192.168.1.0 on the left side of the router, consider a scenario in which the PC should be assigned an IP address from one of the following ranges **192.168.1.71 to 192.168.1.73** or from **192.168.1.31 to 192.168.1.33** or from **192.168.1.201 to 192.168.1.203**. Reconfigure the server and reallocate the IP address to each PC in the network. List the IP address issued to each PC in the network along with its PC number.

7.3) Add another DHCP server in the left network. Connect it with switch 2, Assign a static IP address **192.168.1.50** to the server. Add server pool with starting range **192.168.1.101** and maximum count 5 in DHCP server 2. Add server pool with starting range **192.168.1.201** and maximum count 5 in DHCP Server 0. Once the servers are configured, Go to simulation mode, enable the DHCP filter, and dynamically allocate IP address to each of the PC in the network and monitor the packets exchanged with both of the servers during the process. List the IP address issued to each PC in the network along with its PC number.

LAB 8: Dynamic Routing

Objectives:

- Understanding the working of dynamic routing protocols
- Configuration of RIP Protocol on Router
- Configuration of OSPF Protocol on Router

Prerequisites:

- Knowledge of essential Router configuration commands in CLI
- Familiarity with CISCO Packet Tracer
- Understanding of the working of Router

Theory

Routing protocols are used to facilitate the exchange of routing information between routers. Routing protocols allow routers to dynamically learn information about remote networks and automatically add this information to their own routing tables. Routing protocols determine the best path to each network, which is then added to the routing table. One of the primary benefits of using a dynamic routing protocol is that routers exchange routing information whenever there is a topology change. This exchange allows routers to automatically learn about new networks and also to find alternate paths if there is a link failure to a current network.

Dynamic routing protocols are typically used in larger networks to ease the administrative and operational overhead of using only static routes. Typically, a network uses a combination of both a dynamic routing protocol and static routes. In most networks, a single dynamic routing protocol is used; however, there are cases where different parts of the network can use different routing protocols.

Although many networks will use only a single routing protocol or use only static routes, it is important for a network professional to understand the concepts and operations of all the different routing protocols. A network professional must be able to make an informed decision regarding when to use a dynamic routing protocol and which routing protocol is the best choice for a particular environment.

Dynamic routing protocols have evolved over several years to meet the demands of changing network requirements. Although many organizations have migrated to more recent routing protocols such as Enhanced Interior Gateway Routing Protocol (EIGRP) and Open Shortest Path First (OSPF), many of the earlier routing protocols, such as Routing Information Protocol (RIP), are still in use today.

One of the earliest routing protocols was RIP. RIP has evolved into a newer version: RIPv2. However, the newer version of RIP still does not scale to larger network implementations. To address the needs of larger networks, two advanced routing protocols were developed: OSPF and Intermediate System–to–Intermediate System (IS-IS). Cisco developed Interior Gateway

Routing Protocol (IGRP) and Enhanced IGRP (EIGRP). EIGRP also scales well in larger network implementations.

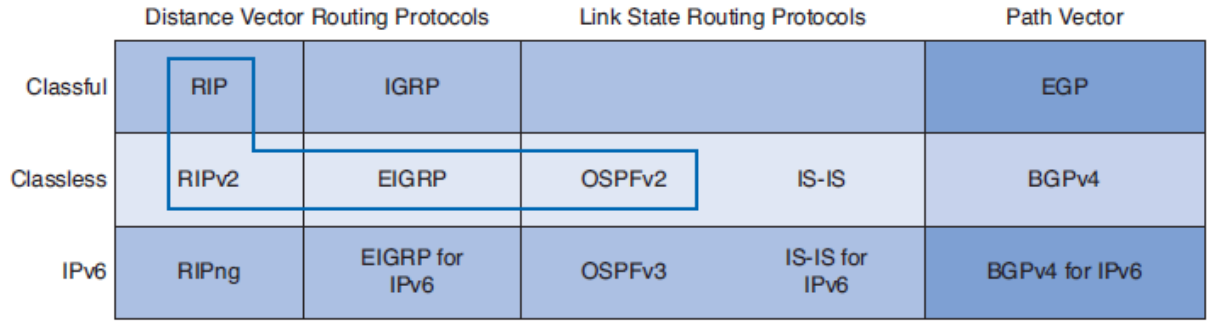


Figure 1: Routing Protocols classification

Additionally, there was the need to interconnect different internetworks and provide routing among them. Border Gateway Protocol (BGP) is now used between Internet service providers (ISP) as well as between ISPs and their larger private clients to exchange routing information.

Interior and Exterior Gateway Protocols

Interior Gateway Protocol (IGP) is a Routing Protocol which is used to find network path information within an Autonomous System. Interior gateways are gateways that belong to the same autonomous system. Known Interior Gateway Protocol (IGP) Routing Protocols are Routing Information Protocol (RIP), Interior Gateway Routing Protocol (IGRP), and Open Shortest Path First (OSPF)

Exterior Gateway Protocol (EGP) is a Routing Protocol which is used to find network path information between different Autonomous Systems. Exterior Gateway Protocol (EGP) is commonly used in the Internet to exchange routing table information. There is only one Exterior Gateway Protocol (EGP) exists now and it is Border Gateway Protocol (BGP).

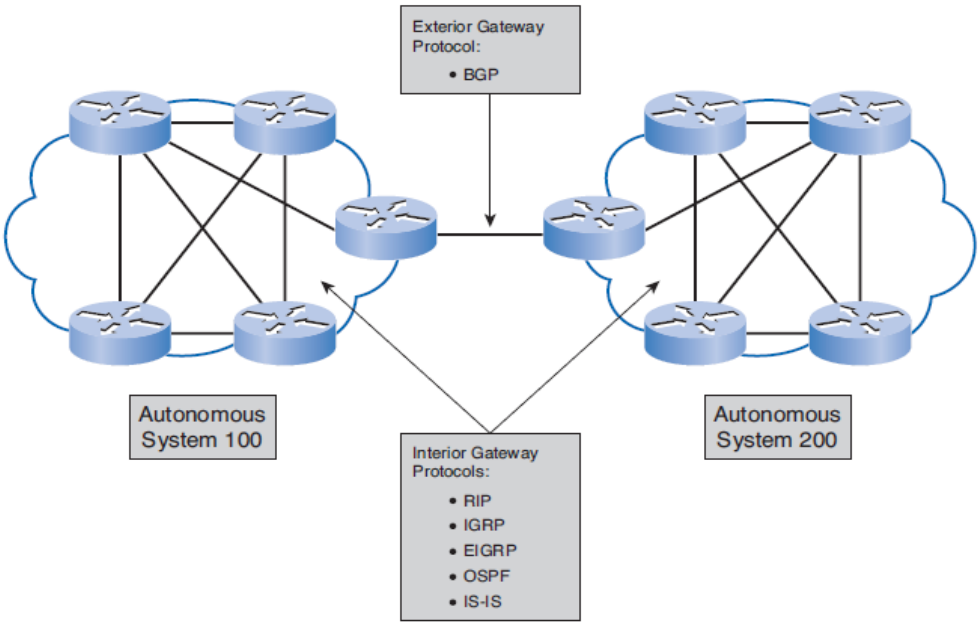


Figure 2: IGP and EGP Routing Protocols

An Autonomous System is defined as a group of routers under a common administrative domain running a common routing protocol. Autonomous system may refer to a routing domain, in this context, an autonomous system is a system of one or more IGPs that is autonomous from other systems of IGPs. Autonomous system may also refer to a process domain, or a single IGP process that is autonomous from other IGP processes. For example, a system of OSPF-speaking routers may be referred to as an OSPF autonomous system.

Routing Information Protocol (RIP)

RIP is a distance-vector routing protocol that uses hop count as the metric for path selection. When RIP is enabled on an interface, the interface exchanges RIP broadcasts with neighboring devices to dynamically learn about and advertise routes. RIP is one of the most enduring of all routing protocols. RIP has four basic components: routing update process, RIP routing metrics, routing stability, and routing timers. Devices that support RIP send routing-update messages at regular intervals and when the network topology changes. These RIP packets include information about the networks that the devices can reach, as well as the number of routers or gateways that a packet must travel through to reach the destination address. RIP generates more traffic than OSPF, but is easier to configure.

RIP uses a single routing metric (hop count) to measure the distance between the source and a destination network. Each hop in a path from source to destination is assigned a hop count value, which is typically 1. When a router receives a routing update that contains a new or changed destination network entry, the router adds 1 to the metric value indicated in the update and enters the network in the routing table. The IP address of the sender is used as the next hop.

RIP sends routing-update messages at regular intervals and when the network topology changes. When a router receives a routing update that includes changes to an entry, it updates its routing table to reflect the new route. The metric value for the path is increased by 1, and the sender is indicated as the next hop. RIP routers maintain only the best route (the route with the lowest metric value) to a destination. After updating its routing table, the router immediately begins transmitting routing updates to inform other network routers of the change. These updates are sent independently of the regularly scheduled updates that RIP routers send.

RIP Configuration Commands

You can only enable one RIP routing process on the ASA. After you enable the RIP routing process, you must define the interfaces that will participate in that routing process using the network command. By default, the ASA sends RIP Version 1 updates and accepts RIP Version 1 and Version 2 updates. To enable the RIP routing process, enter the following command.

Command	Function
`hostname(config)# router rip`	Starts the RIP routing process and places you in router configuration mode.

`network` ***`network_address`***
`Example:`
`hostname(config)# router rip`
	Specifies the interfaces that will participate in the RIP routing process. If an interface belongs to a network defined by this command, the interface will participate in the RIP routing process. If an

hostname(config-router)# **network 10.0.0.0**	interface does not belong to a network defined by this command, the interface will not send or receive RIP updates.
version [1 | 2] **Example:** hostname(config-router):# **version [1]**	Specifies the version of RIP used by the ASA. You can override this setting on a per-interface basis. In this example, Version 1 is entered.
passive-interface [**default** | *if_name*] **Example:** hostname(config-router)# **passive-interface [default]**	Specifies an interface to operate in passive mode. Using the **default** keyword causes all interfaces to operate in passive mode. Specifying an interface name sets only that interface to passive mode. In passive mode, RIP routing updates are accepted by, but not sent out of, the specified interface. You can enter this command for each interface that you want to set to passive mode.
show rip database	Display the contents of the RIP routing database.
show running-config router rip	Displays the RIP commands
no router rip **Example:** hostname(config)# **clear rip**	Removes the entire RIP configuration that you have enabled. After the configuration is cleared, you must reconfigure RIP again using the **router rip** command.

Table1: RIP Configuration Commands

RIP Configuration in Packet Tracer

Configure the following network in CISCO Packet Tracer, as we did in the previous lab. Don't forget to configure clock rate on Serial DTE interface.

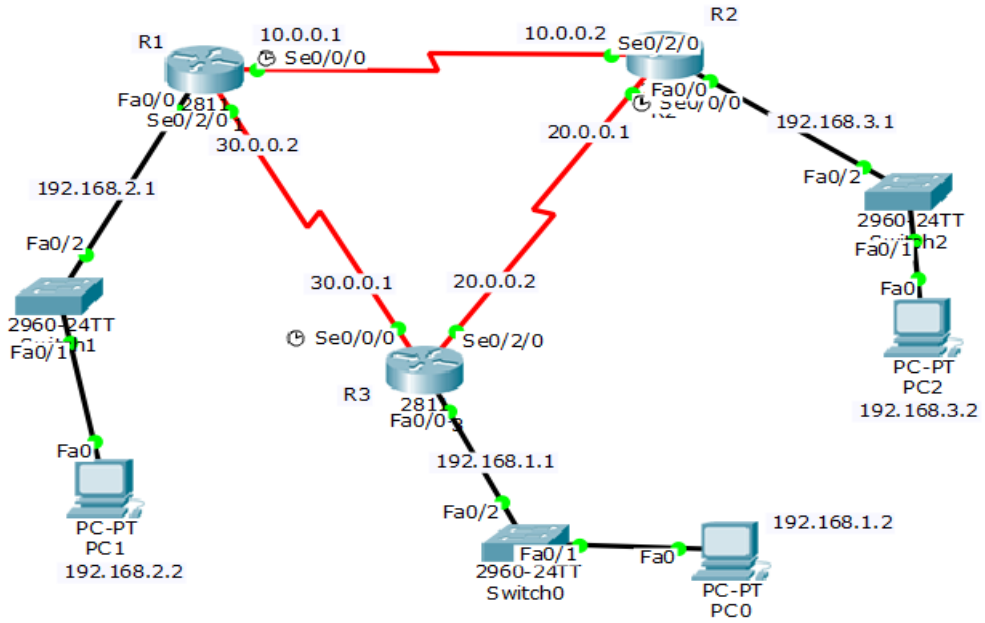


Figure 3: RIP Configuration Network

- Once the network is configured our next step is to configure dynamic routing protocol RIP in the network
- First of all, click on Router R1, open CLI tab, go to configuration mode and enter the following command

  R1(config)#router rip
- Once you enter the above command you will enter in router mode to configure RIP.
- Now in this mode,

  R1(config-router)#
- In RIP each router advertises the information of its directly connected networks to other routers. For doing this, we add directly connected networks to its routing table. Enter the directly connected networks in Router R1 table.

  R1(config-router)#**network 192.168.2.0**
  R1(config-router)#**network 10.0.0.0**
  R1(config-router)#**network 30.0.0.0**
- Similarly add directly connected networks on Router R2 routing table.

  R2(config)#router rip
  R2(config-router)#**network 192.168.3.0**
  R2(config-router)#**network 10.0.0.0**
  R2(config-router)#**network 20.0.0.0**
- Similarly for Router R3, add directly connected networks

  R3(config)#router rip
  R3(config-router)#**network 192.168.1.0**
  R3(config-router)#**network 20.0.0.0**
  R3(config-router)#**network 30.0.0.0**
- Once we have successfully added the directly connected networks, each router will start advertising its directly connected networks to other routers in the network.
- You can check the routing table by entering the command **`show ip route`** or **`show ip protocols`**
- You have successfully configured RIP protocol, Ping from one PC to another to check if it is successful.

Open Shortest Path First (OSPF)

OSPF is an interior gateway routing protocol that uses link states rather than distance vectors for path selection. OSPF propagates link-state advertisements rather than routing table updates. Because only LSAs are exchanged instead of the entire routing tables, OSPF networks converge more quickly than RIP networks. OSPF uses a link-state algorithm to build and calculate the shortest path to all known destinations. Each router in an OSPF area contains an identical link-state database, which is a list of each of the router usable interfaces and reachable neighbors.

The advantages of OSPF over RIP are, firstly OSPF link-state database updates are sent less frequently than RIP updates, and the link-state database is updated instantly rather than

gradually as stale information is timed out. Secondly, Routing decisions are based on cost, which is an indication of the overhead required to send packets across a certain interface. The ASA calculates the cost of an interface based on link bandwidth rather than the number of hops to the destination. The cost can be configured to specify preferred paths. The disadvantage of shortest path first algorithms is that they require a lot of CPU cycles and memory.

A router that has interfaces in multiple areas is called an Area Border Router (ABR). A router that acts as a gateway to redistribute traffic between routers using OSPF and routers using other routing protocols is called an Autonomous System Boundary Router (ASBR). OSPF supports MD5 and clear text neighbor authentication. Authentication should be used with all routing protocols when possible because route redistribution between OSPF and other protocols (like RIP) can potentially be used by attackers to subvert routing information.

An ABR uses LSAs to send information about available routes to other OSPF routers. Using ABR type 3 LSA filtering, you can have separate private and public areas with the ASA acting as an ABR. Type 3 LSAs (inter-area routes) can be filtered from one area to other. This lets you use NAT and OSPF together without advertising private networks.

OSPF States

When OSPF adjacency is formed, a router goes through several state changes before it becomes fully adjacent with its neighbor. The adjacency building process takes effect after multiple stages have been fulfilled. Routers that become adjacent will have the exact link-state database. The following is a brief summary of the states an interface passes through before becoming adjacent to another router.

Down: No information has been received from anybody on the segment.

Attempt: On non-broadcast multi-access clouds such as Frame Relay and X.25, this state indicates that no recent information has been received from the neighbor. An effort should be made to contact the neighbor by sending Hello packets at the reduced rate PollInterval.

Init: The interface has detected a Hello packet coming from a neighbor but bi-directional communication has not yet been established.

Two-way: There is bi-directional communication with a neighbor. The router has seen itself in the Hello packets coming from a neighbor. At the end of this stage the DR and BDR election would have been done. At the end of the 2way stage, routers will decide whether to proceed in building an adjacency or not. The decision is based on whether one of the routers is a DR or BDR or the link is a point-to-point or a virtual link.

Exstart: Routers are trying to establish the initial sequence number that is going to be used in the information exchange packets. The sequence number insures that routers always get the most recent information. One router will become the primary and the other will become secondary. The primary router will poll the secondary for information.

Exchange: Routers will describe their entire link-state database by sending database description packets. At this state, packets could be flooded to other interfaces on the router.

Loading: At this state, routers are finalizing the information exchange. Routers have built a link-state request list and a link-state retransmission list. Any information that looks incomplete or

outdated will be put on the request list. Any update that is sent will be put on the retransmission list until it gets acknowledged.

Full: At this state, the adjacency is complete. The neighboring routers are fully adjacent. Adjacent routers will have a similar link-state database.

OSPF Configuration Commands

To enable OSPF, you need to create an OSPF routing process, specify the range of IP addresses associated with the routing process, then assign area IDs associated with that range of IP addresses. To enable OSPF, perform the following detailed steps.

Command	Function
router ospf *process_id* **Example:** hostname(config)# **router ospf** ***2***	This creates an OSPF routing process, and the user enters router configuration mode for this OSPF process. The *process_id* is an internally used identifier for this routing process. It can be any positive integer. This ID does not have to match the ID on any other device; it is for internal use only. You can use a maximum of two processes.
network *ip_address wild-card-mask* **area** *area_id* **Example:** hostname(config)# **router ospf 2** hostname(config-router)# **network 10.0.0.0 0.255.255.255 area 0**	This step defines the IP addresses on which OSPF runs and to define the area ID for that interface.
ospf hello-interval *seconds* **Example:** hostname(config-interface)# **ospf hello-interval** ***10***	This allows you to specify the length of time between the hello packets that the ASA sends on an OSPF interface. The value must be the same for all nodes on the network. In this example, the hello-interval is set to 10.
ospf dead-interval *seconds* **Example:** hostname(config-interface)# **ospf dead-interval** ***40***	This allows you to set the number of seconds that a device must wait before it declares a neighbor OSPF router down because it has not received a hello packet. The value must be the same for all nodes on the network. In this example, the dead-interval is set to 40.
ospf retransmit-interval *seconds* **Example:** hostname(config-interface)# **ospf retransmit-interval 15**	This allows you to specify the number of seconds between LSA retransmissions for adjacencies belonging to an OSPF interface. The value for *seconds* must be greater than the expected round-trip delay between any two routers on the attached network. The range is from 1 to 65535 seconds. The default value is 5 seconds. In this example, the retransmit-interval value is set to 15.
summary-address *ip_address mask* [**not-advertise**] [**tag** *tag*] **Example:** hostname(config)# **router ospf 1** hostname(config-router)# **summary-address 10.1.0.0 255.255.0.0**	This step sets the summary address. In this example, the summary address 10.1.0.0 includes address 10.1.1.0, 10.1.2.0, 10.1.3.0, and so on. Only the address 10.1.0.0 is advertised in an external link-state advertisement

ospf authentication-key *key* **Example:** hostname(config-interface)# **ospf authentication-key cisco**	This allows you to assign a password to be used by neighboring OSPF routers on a network segment that is using the OSPF simple password authentication. The *key* can be any continuous string of characters up to 8 bytes in length. The password created by this command is used as a key that is inserted directly into the OSPF header when the ASA software originates routing protocol packets. All neighboring routers on the same network must have the same password to be able to exchange OSPF information.
show ospf [*process-id [area-id]*]	Displays general information about OSPF routing processes.
show ospf border-routers	Displays the internal OSPF routing table entries to the ABR and ASBR.
show ip ospf neighbor	It shows the IP addresses of all the neighbors of a particular router
show ospf interface [*if_name*]	**show ospf interface** [*if_name*]
clear ospf *pid* **Example:** hostname(config)# **clear ospf 2**	This remove entire OSPF configuration you have enabled. Once this is cleared, you must reconfigure OSPF again using the **router ospf** command.

Table1: OSPF Configuration

OSPF Configuration in Packet Tracer

We will use same network to configure **OSPF** protocol. Configure it in the same way, save it in a new file and remove RIP protocol from routers.

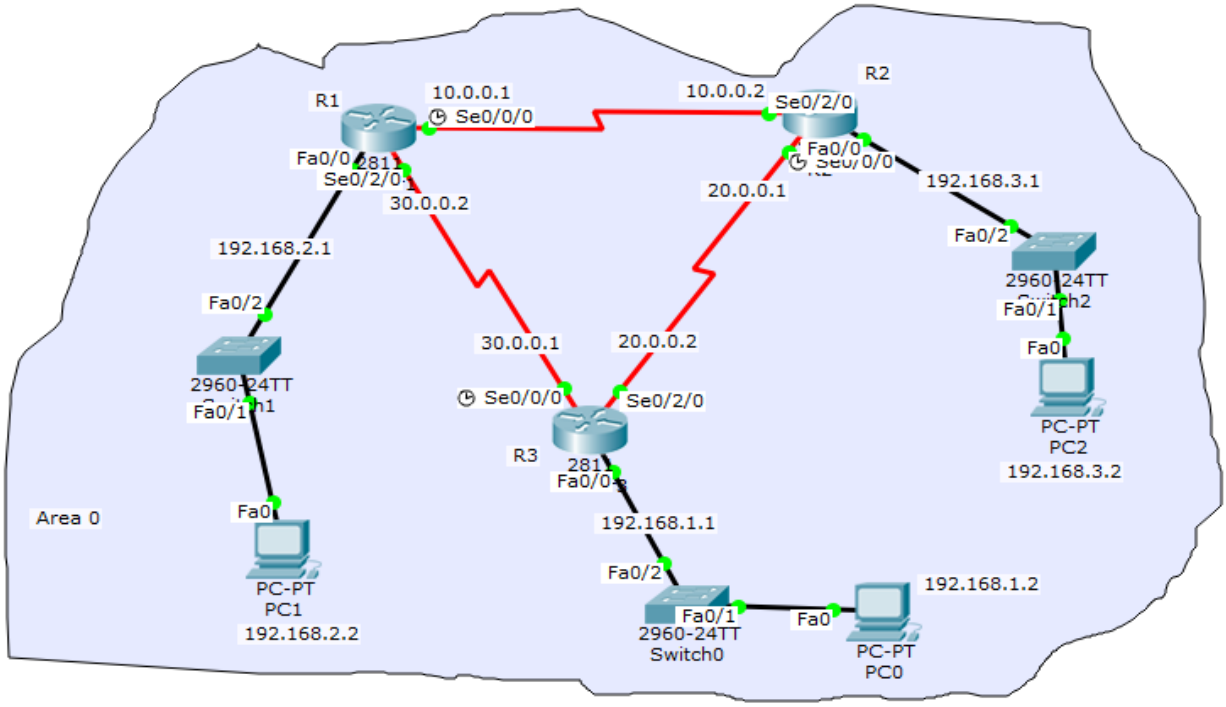


Figure 3: OSPF Configuration Network

- Now let's configure OSPF on these routers, click on Router R1, open CLI tab and enter the following commands.
- We will go to router mode to configure

```
R1(config)#router  ospf      10
                          Protocol-name   Process-ID
```

- For every Router we have to enter a process ID for OSPF, if not entered the highest IP address on any interface of that router is taken as its process ID.
- Now in second step, we add the address to directly connected networks, in OSPF LSDB. Each directly connected address is entered along with its wildcard mask. Wildcard mask is bitwise inverse of subnet mask. The 3rd parameter is to add Area ID. All the networks in the same area must have same area ID. We can also configure OSPF for multiple areas.

```
R1(config-router)#network Network-Address wildcard-mask area ID
```

- In Router R1 add following commands to add its directly connected networksin OSPF database, assuming all networks are in area 0.

```
R1(config)#router ospf 10
R1(config-router)#network 192.168.2.0 0.0.0.255 area 0
R1(config-router)#network 10.0.0.0 0.255.255.255 area 0
R1(config-router)#network 30.0.0.0 0.255.255.255 area 0
```

- Similarly for Router R2, add its directly connected networks of area 0, in its OSPF database.

```
R2(config)#router ospf 10
R2(config-router)#network 192.168.3.0 0.0.0.255 area 0
R2(config-router)#network 10.0.0.0 0.255.255.255 area 0
R2(config-router)#network 20.0.0.0 0.255.255.255 area 0
```

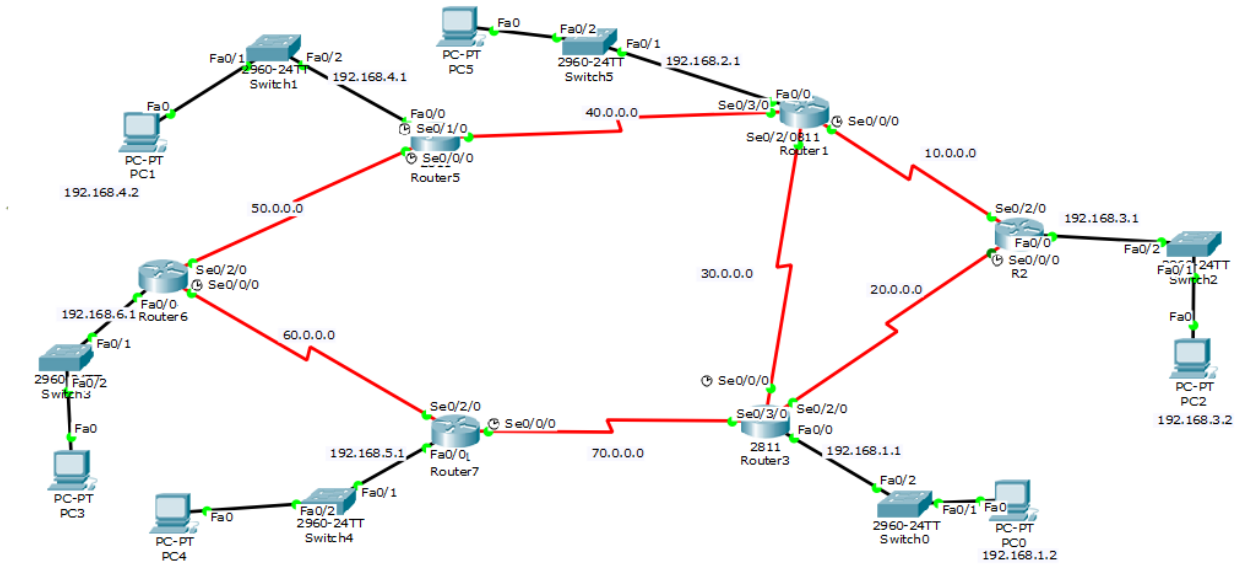
- Similarly for Router R3, add its directly connected networks of area 0, in its OSPF database.

```
R3(config)#router ospf 10
R3(config-router)#network 192.168.1.0 0.0.0.255 area 0
R3(config-router)#network 20.0.0.0 0.255.255.255 area 0
R3(config-router)#network 30.0.0.0 0.255.255.255 area 0
```

- You have successfully configured OSPF routing protocol on the network, you can check the routing table of the routers by entering the command **`show ip route`**
- Try to ping from one PC to another, you should be successful if you have correctly configured the network.

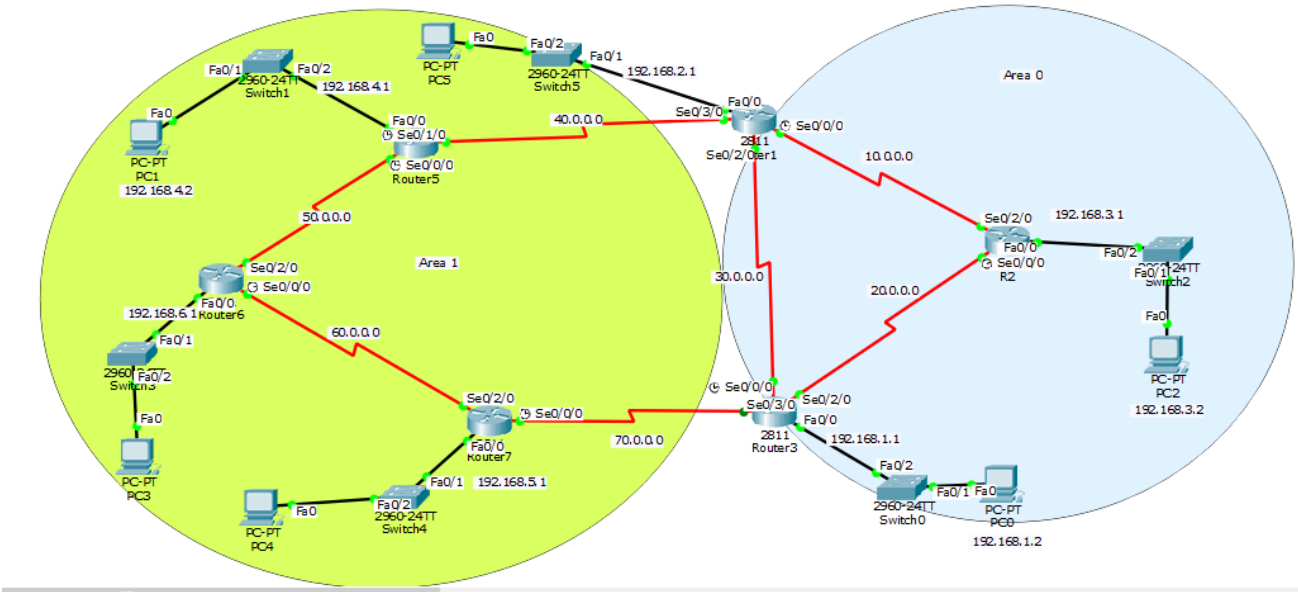
LAB TASKS

8.1) Build and configure the following network. Configure RIP protocol on all the routers in the network.



Now go to simulation mode and check the path followed by the packet, when we ping from PC2 to PC3, PC0 to PC1 and PC5 to PC4. Explain why the packet follows a particular path.

8.2) Now reconfigure the same network by OSPF for Area 0 and Area 1, as shown in the figure on the next page.



8.3) Now go to simulation mode and check the path followed by the packet, when we ping from PC2 to PC3, PC0 to PC1 and PC5 to PC4. Explain why the packet follows a particular path.

LAB 9: CISCO Router Configuration

Objectives:

- Learn to make a network using Router and Switches
- Configuration of a router using Hyper Terminal
- Communicate via LAN and WAN network

Prerequisites:

- Knowledge of essential Networking commands
- Knowledge of working on Command line interface
- Understanding of the working of Router

Theory:

In previous labs we have worked a lot with switches and routers on CISCO Packet Tracer and we already know how the switch and router works. We also know the name and functionality of the ports available in switch and routers. Now it's time to apply our knowledge and establish a network using real router and switches.

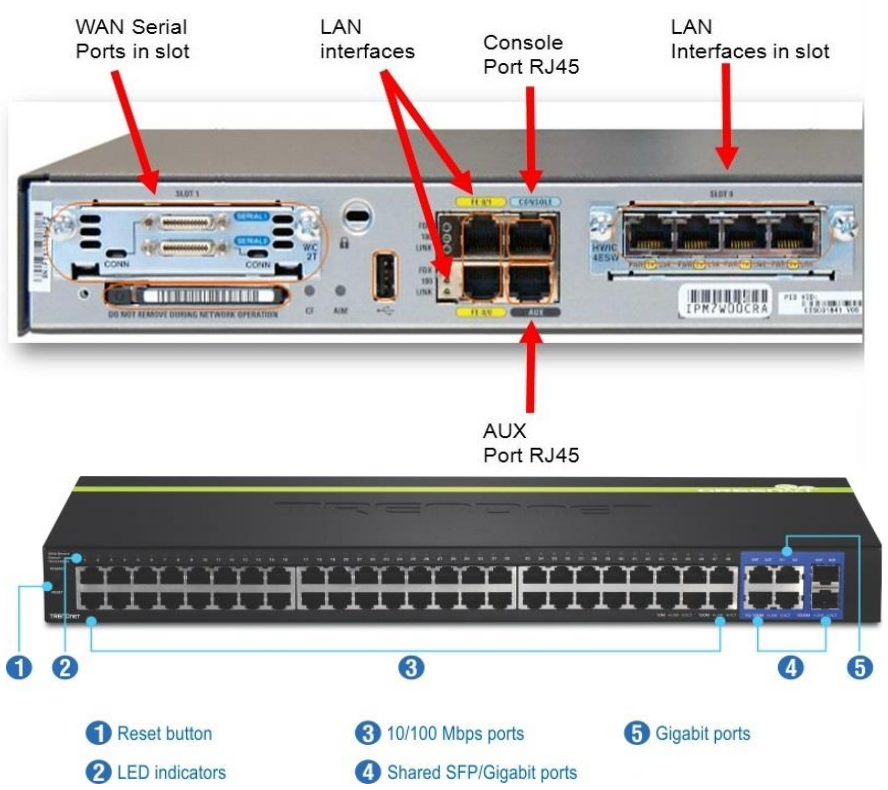


Figure 1: Available Ports in Router and Switch

Just for a brief overview, Switch is a device which connects different devices on a same network by using Fast Ethernet ports. Router is a device which connects two different networks. Fast Ethernet (LAN) port is used to make a connection between Router and other network devices (switch). Serial (WAN) Port is used to interconnect multiple Routers.

Establishing a Local Area Network

Let's start with making a LAN network. Make the connections between switch and PCs. Connect one end of a cable to Ethernet port of the PC and second end to any one of the Fast Ethernet ports available on the switch. Assign the IP address of same network to each of the PCs as shown in the diagram.

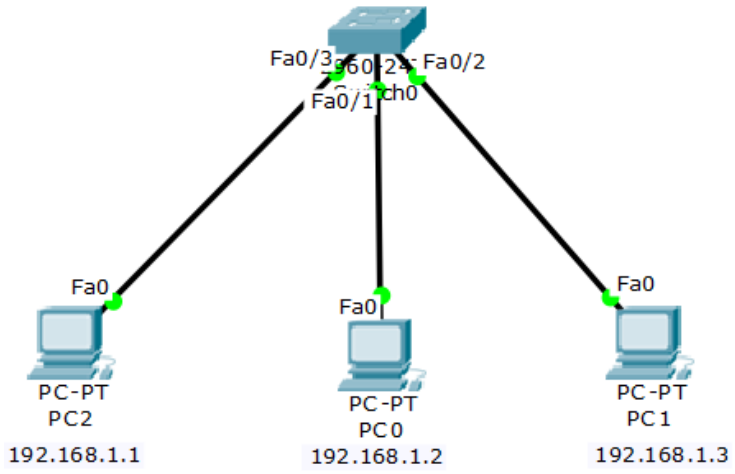


Figure 2: LAN Network

Once the IP address are assigned, open command Prompt to ping from one PC to another and check if the connection is successfully established.

Establishing a Wide Area Network

Now let's move one step further and establish a Wide Area Network using Router. We will use Router to connect two different networks. Establish another LAN network using a switch and assign a different network IP address. Once the two LAN networks are established make a connection between router and switches. For each of the switches, connect one end of the Ethernet cable to one of the available fast Ethernet ports of a switch and second end of the cable to Ethernet (LAN) Port available on the Router as shown in the diagram below.

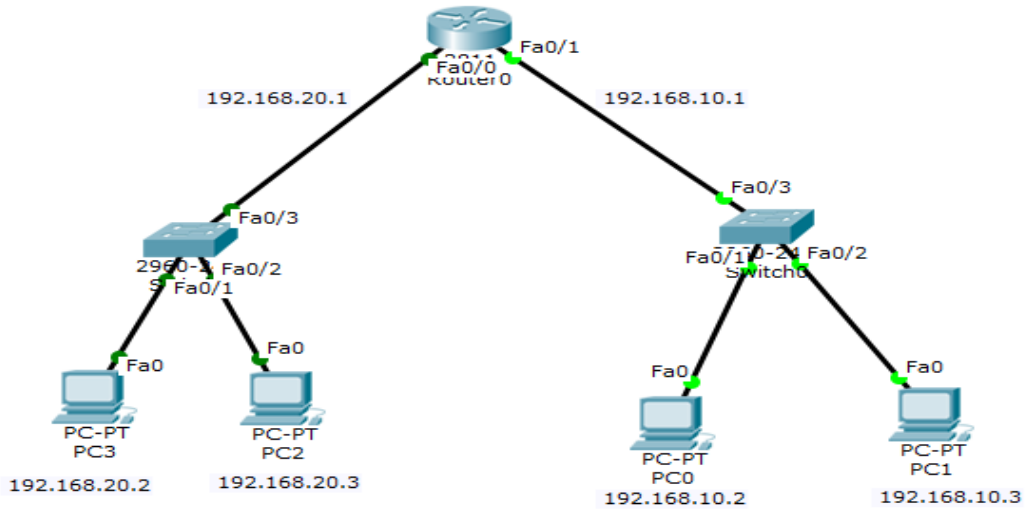


Figure 3: WAN Network

Router Configuration

Once you have made connections, the last but most important step is to configure a Router. For Router configuration, we will use console port of the router. Use a console cable which has console connector at one end and MD-9 connector at another end. Connect console interface of the cable to console port of the router and MD-9 interface to PC. If your PC / Laptop does not have MD-9 interface you can use MD-9 to USB converter for making a connection to PC.

Now after making a connection between router and PC via console cable, we will use any hyper-terminal software for configuration of Router using command Line Interface. There are many software available like Hyper Terminal, Putty, Tera Term etc. We will be using Tera Term in this lab. Install the Tera Term software by using the package provided or download it from the internet. Once you have completed the installation, open it and you will see the following interface.

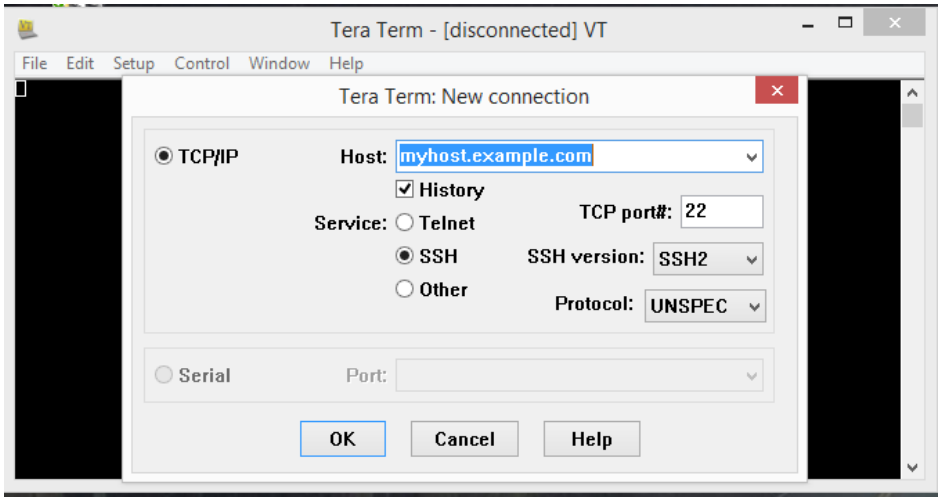


Figure 3: Tera Term interface

As you can see there are two interfaces available for communication. TCP/IP and Serial. Select a Serial Interface, select a COM port from the list and press OK. After this the router will start a booting process.

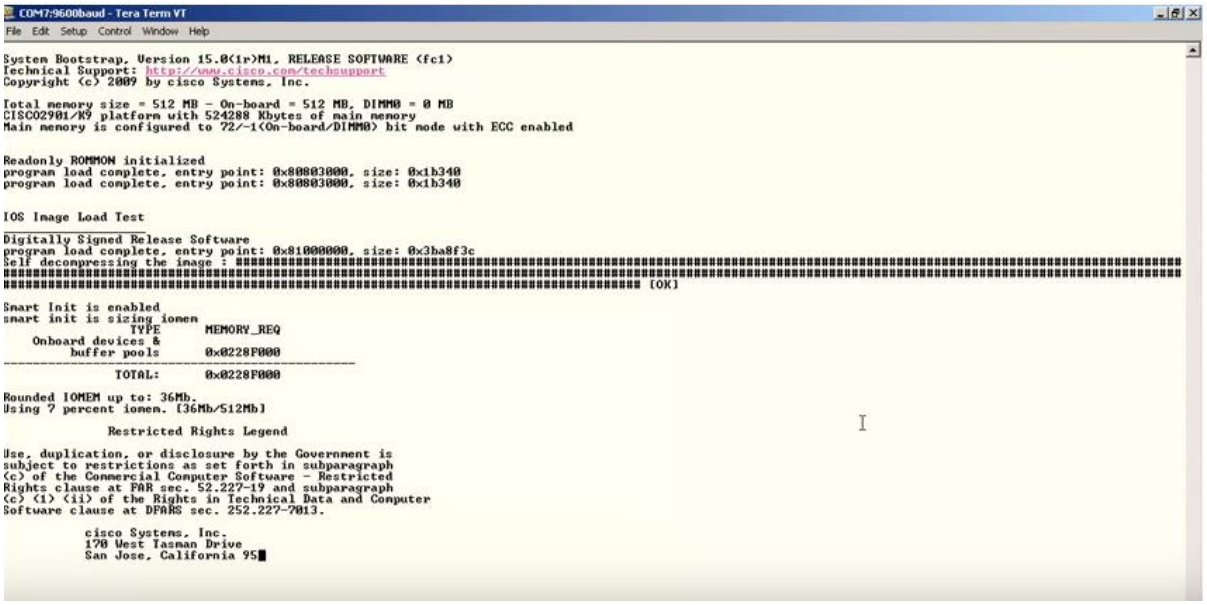


Figure 4: Router boot loading

Once the router has completed the booting process, start configuring the router in CLI mode, as we did in the previous labs. For the above example the router will be configured by using the commands as shown in the diagram below.

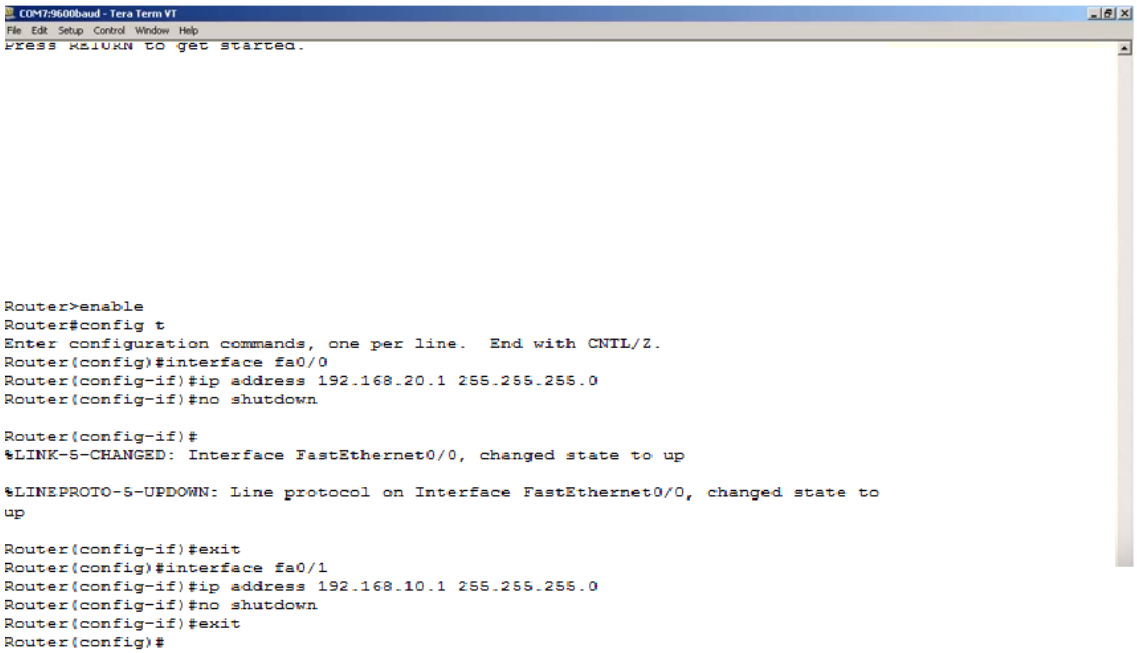


Figure 5: Router configuration in CLI using Tera Term

To confirm that the router is correctly configured, check the IP addresses assigned by using the command show ip interface brief or show running configuration. If you have successfully configured, you should be able to ping from one network's PC to another. Open command prompt and check if the ping is successful.

LAB TASKS

9.1) Build and configure the network router to form a network as shown in Figure 3.

LAB 10: Raw Sockets

OBJECTIVE

- Get basic understanding of raw sockets
- Use raw sockets to receive and process network messages

Note: this tutorial also includes 1-2 open ended labs

Theory

The purpose of this lab is to learn the basics of using raw sockets. Raw sockets can be used for injecting or receiving any IP protocol based datagram without using any socket functions like bind, connect and accept etc. such functionality allows building network sniffing tools like wireshark, intrusion detection systems and port scanning tools like nmap. Basically, you can send any packet at any time. However, standard socket functions like bind, connect, send etc. allow you to have no direct control over the packet. Raw sockets theoretically enable you to simulate the behavior of your OS's IP stack, and also to send stateless traffic (datagrams that don't belong to a valid connection).

Procedure

The basic concept of low level (raw/packet) sockets is to send/receive a single packet at one time, with all the protocol headers filled in by the user program (instead of the kernel). Unix provides two kinds of sockets that permit direct access to the network. One is SOCK_PACKET, which receives and sends data on the device link layer. This means, the NIC specific header is included in the data that will be written or read. For most networks, this is the Ethernet header. Of course, all subsequent protocol headers will also be included in the data. On the other hand, SOCK_RAW includes IP headers and all subsequent protocol headers and data.

The standard command for creating raw sockets is.

```
socket (PF_INET, SOCK_RAW, IPPROTO_UDP);
socket (PF_INET, SOCK_RAW, IPPROTO_TCP);
socket (PF_INET, SOCK_RAW, IPPROTO_ICMP);
```

From the moment that it is created, you can send any IP packets over it, and receive any IP packets that the host received after that socket was created if you read() from it. Note that even though the socket is an interface to the IP header, it is transport layer specific. That means, for listening to TCP, UDP and ICMP traffic, you have to create 3 separate raw sockets, using IPPROTO_TCP, IPPROTO_UDP and IPPROTO_ICMP (the protocol numbers are 0 or 6 for tcp, 17 for udp and 1 for icmp).

On the other hand, Packet sockets are used to receive or send raw packets at the device driver (Layer 2) level. They allow the user to implement protocol modules in user space on top of the physical layer. The socket_type is either SOCK_RAW for raw packets including the link level

header or SOCK_DGRAM for cooked packets with the link leveheader removed. Packet socket can be created using

Socket(PF_PACKET, SOCK_RAW, htons(ETH_P_IP));

ETH_P_IP captures all IP packets; whereas, ETH_P_ALL passes all packets to application, before sending them to the linux kernel. Common headers needed for using raw or packet sockets with device binding options are

```
#include <arpa/inet.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/ether.h>
```

Only processes with effective uid 0 or the CAP_NET_RAW capability may open packet sockets. SOCK_RAW packets are passed to and from the device driver without any changes in the packet data.

Socket Options

Packet sockets can be used to configure many options like physical layer multicasting and promiscuous mode etc. It works by calling setsockopt() on a packet socket for SOL_PACKET. For instance, the following function binds network interface indicated by ifname for listening.

```
if ( setsockopt(sockfd,
                SOL_SOCKET,
                SO_BINDTODEVICE,
                ifName,
                IFNAMSIZ-1) == -1)     {
          perror("SO_BINDTODEVICE");
          close(sockfd);
          exit(EXIT_FAILURE);
}
```

LAB TASKS

10.1) 1-2 Open ended labs!