Advanced Telecommunication Networks

ELEC472

Lab Manual

Department of Electrical and Computer Engineering

Concordia University

Winter 2021

Table of Contents

Table of Contents 2
Lab Guide 4
User agreement and license 4
Report Description 5
Grading Scheme 5
Submission Conditions 5
Experiment execution 6
Manual update and corrections 6
Experiment 01: Introduction to network tools and packet tracer 7
Introduction 7
Part 1: Network Simulator Cisco Packet Tracer (PT) 7
Task 1: Explore the PT interface 8
Task 2: Adding devices to workspace 9
Task 3: Design network and setting IP address to routers and hosts 10
Task 4: Configure FTP service 18
Part 2: Using Wireshark to View Protocol Data Units 20
Objectives 20
Introduction about Wireshark 20
Installing and using Wireshark 21
Task 1: Ping PDU Capture 25
Experiment 02: Building network topology and enable RIP routing protocol 29
Introduction 29
Background 29
Task 1: Building network topology and initial configuration 30
Task 2: Configure RIP routing protocol 38
Task 3: understand RIP routing protocol 42
Experiment 03: OSPF routing protocol 43
Introduction 43
Comparing Hop Count and Bandwidth Metrics 43
Open Shortest Path First (OSPF) 44

Task1: Configure OSPF 44
Task 2: change OSPF routing table 48
Task3: change bandwidth in routers 49
Task4: verifying interfaces types and checking routing update 51
Task 5: Configuring DHCP in Router 52
Task 6: Configure NAT/PAT 54
Task7: Read OMNeT++ tutorial part as preparation for experiment 04 55
OMNeT++ Simulator tutorial 56
Introduction 56
Task 01: OMNeT++ Installation 56
Task 02: Creating New Project 60
Task 03: OMNeT++ files structures 63
Task 04: INET Framework 63
Activity 01: Add INET to OMNeT++ 63
Activity 02: Reference INET to existing project 66
Task 05: UDP simple application example 67
Task 06: Routing protocols and queuing quality 77
Experiment 04: Enable Queuing QoS 84
Introduction 84
Background 84
Task 01: Create network with best effort performance and measure the performance 88
Activity 01: build the network topology 88
Activity 02: configure general parameters 91
Activity 03: configure applications 91
Activity 04: configure network parameters and statistical measurements 92
Activity 05: Configure two types of queuing 93
Activity 06: add xml files for OSPF configuration and traffic classification 94
TASK 2: run the experiments under two configurations mode 95
Task 03: Displaying and Exporting Simulation Results Data 95
Experiment 05: Traffic shaping and policing 100
Introduction 100
Background 100

Task 01: Create network with best effort performance and measure the performance 100
Activity 01: build the network topology 100
Activity 02: configure general parameters 103
Activity 03: configure applications 103
Activity 04: configure network parameters and statistical measurements 104
Task 02: Configure shaping and policing 105
Activity 01: configure shaping 105
Activity 02: configure policing 106
Activity 03: configure queuing with and without policing and shaping 106
`**.queue.packetCapacity = 100` 106
Activity 04: add xml files for OSPF configuration and traffic classification as below figure 107
TASK 03: run the experiments under three configurations mode 108
Task 04: Displaying and Exporting Simulation Results Data 108
References 111
Appendix 111
Abbreviation 111

Lab Guide

User agreement and license

This document is based on multiple opensource or free tools and simulators. Using this document is based on opensource license, it is not allowed to redistribute or use it under any circumstances for commercial use. Tools and simulators used in this manual are as following:

1) CISCO Packet tracer simulator, under cisco network academy that has free license and access for educators (https://www.netacad.com/educators). You can access it by submitting your information and accept user agreements.
2) Network tools and services (ping, tracert, nslookup and FTP), which are free tools.
3) Wireshark, which is a free network traffic analyzer (https://www.wireshark.org/).
4) OMNeT++, which is a network modular C++ simulation library distributed under Academic Public License (https://omnetpp.org/).

Report Description

Each lab report should be divided into five parts as follow:

1. Cover page: Must include Concordia logo, student name and ID, course name and number, experiment name and date, department name.
2. Objectives: They must be stated clearly.
3. Introduction: It should be brief and written clearly in your own terms and not copied from the experiment document provided. Explain the relevant theory used in the experiments.
4. Experimental Results and Questions: Experimental results should be broken down into sections as in the lab manual. Each diagram should be discussed thoroughly, and questions should be answered in detail.
5. Discussion: You must also add a Discussion section to each part of the lab experiment for providing information about your observations and the resulted technical facts.
6. Conclusions: In this section, a detailed conclusion of activities and the technical facts that have been learned should be provided.

Grading Scheme

Each lab report will be marked out of 20. The grading scheme is as follows:

Lab preparation and participation ** 4/20

Objectives and brief introduction 2/20

Results presentation 4/20

Discussion 6/20

Conclusions 4/20

** Lab preparation and participation is evaluated by your preparation reading manual instructions and theory before conducting the experiment.

Submission Conditions

- This is an individual lab effort; lab reports must be submitted on time.
- Delay on submission costs you 20% penalty for each late day, this means after day five you will get zero.
- Word document must be submitted by email or any convenient submission method.
- Experiments simulation files must be submitted as well with lab report.
- In each experiment simulation file, a Readme file must be included that explains how to use your code and run it.
- Any new method or code you introduce must be clearly explained and shown how to use it. If it works and comply with the requirements you will get the full mark with bonus. But if it does not work you will get partial mark based on your convincing logic and code. You may also get a zero.

Experiment execution

Experiments in this manual are organized in tasks and activities, you will go through it in sequential way and for each part you must show your work by taking a snapshot using any convenient snipping tool. You should discuss the steps by describing the procedures you worked on.

Manual update and corrections

This manual version is a beta version, it may have some mistakes and errors, if you find any mistake or unclear state or description or procedure please state that with suggested correction. You will get a **bonus mark for good and correct suggestions**.

TA has the full privilege to change or override any parts on this manual and you must follow his comments and instructions.

Experiment 01: Introduction to network tools and packet tracer

Introduction

In this experiment you are going to learn about network components, devices, and tools. There are four components of a network: 1) messages, 2) devices, 3) media, and 4) services. These elements allow the communication of different types of messages (encompasses web pages, e-mail, instant messages, telephone calls, and other forms of communication enabled by the Internet). On top of that, the rules, or protocols, that tie these network elements together.

Intermediary network devices are:

1. Network Access Devices connect end users (Hubs, switches, and wireless access points)
2. Internetworking Devices connect between networks (routers)
3. Communication Servers and Modems (auxiliary services for TCP/IP stack, and communication technologies)
4. Security Devices (firewalls)
5. End devices (PC, Laptop, phones)

A few tools are used in this experiment to analyze and simulate network functionality. Two of the tools that enable you to build and interact with simulated networks are Packet Tracer software and Wireshark network protocol analyzer.

Part 1: Network Simulator Cisco Packet Tracer (PT)

The simulator that is used in this lab is Packet Tracer which is a standalone simulation and visualization program. Throughout this lab, you will be using a standard lab setup created from actual PCs, servers, routers, and switches to learn networking concepts. This method provides widest range of features and the most realistic experience. Since equipment and time are limited, this experience can be supplemented by a simulated environment.

Packet Tracer provides a rich set of protocols, equipment, and features but only a fraction of what is possible with real equipment. Packet Tracer is a supplement, but not a replacement, for experience with real equipment. You are encouraged to compare the results obtained from Packet Tracer network models with the behavior of real equipment.

Please register on cisco network academy to get your free packet tracer simulator (https://www.netacad.com/courses/packet-tracer/faq). You can enroll for a free course introduction to packet tracer English 1220 course via this link (https://www.netacad.com/portal/self-enroll/m/194032).

Task 1: Explore the PT interface

Activity 1: Workplace and Realtime mode

When Packet Tracer starts, it presents a logical view of the network in real-time mode. The main part of the PT interface is the Logical Workplace. This is the large area where devices are placed and connected as shown in Figure 1.

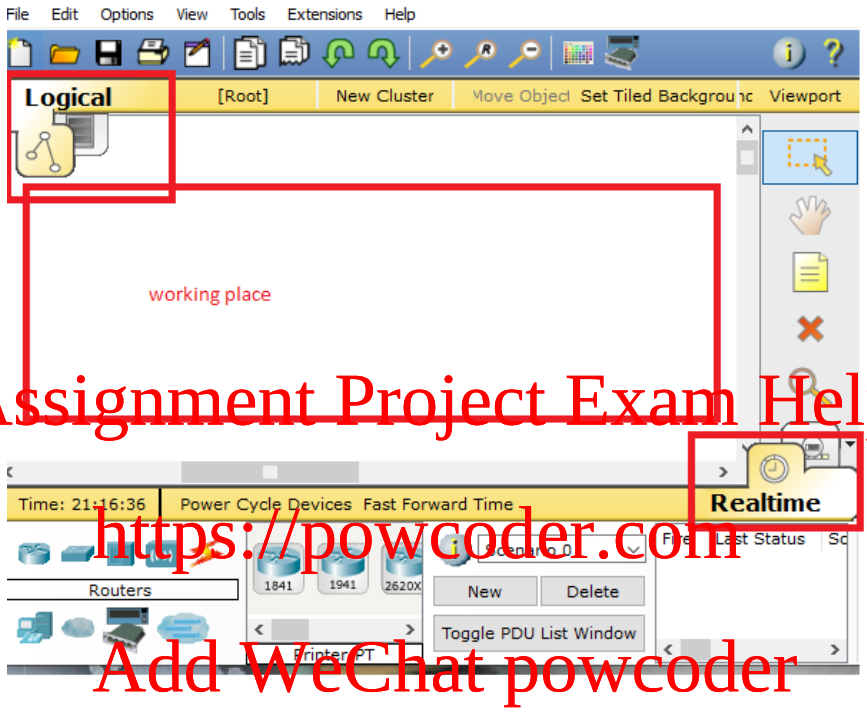


Figure 1: Packet Tracer Working Environment

Activity 2: Devices Navigation

The lower left portion of the PT interface, below the yellow bar, is the portion of the interface that you use to select and place devices into the logical workplace. The first box in the lower left contains symbols that represent groups of devices. As you move the mouse pointer over these symbols, the name of the group appears in the text box in the center. When you click on one of these symbols, the specific devices in the group appear in the box to the right. As you point to the specific devices, a description of the device appears in the text box below the specific devices. Click on each of the groups and study the various devices that are available and their symbols as in Figure 2.

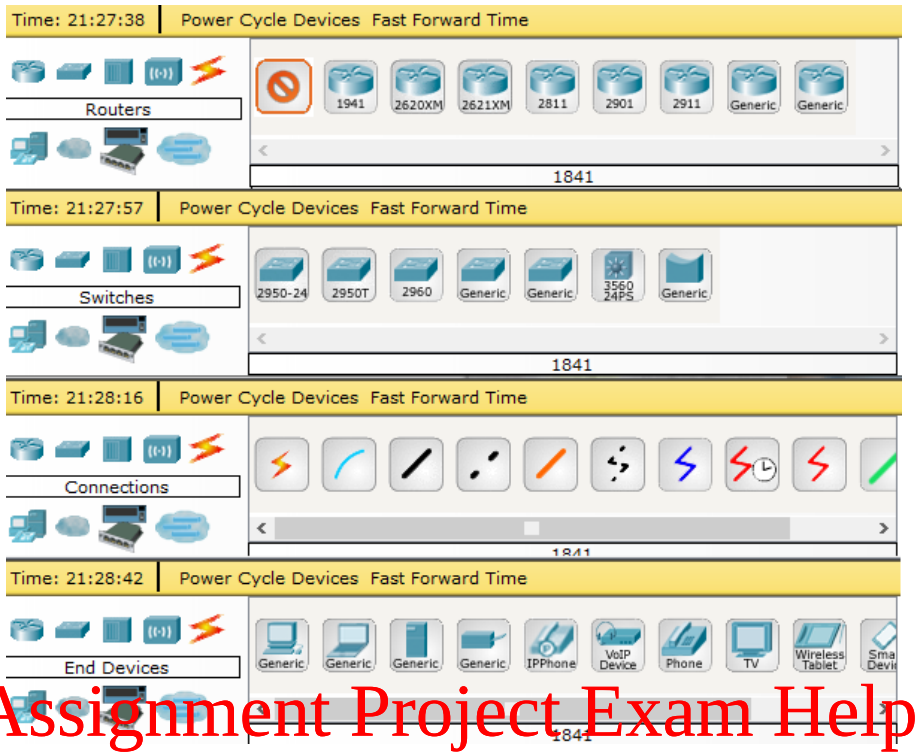


Figure 2: Devices and Connections Types in PT

Task 2: Adding devices to workspace

To add a device to the logical workplace, click on the specific device symbol, point to where you want to place the device in the logical workplace (the pointer becomes a crosshair), and click. Locate and place the following devices in a horizontal row across the logical workplace, with about an inch between them, in order from left to right: server, 2960 switch, 1841 router, hub, IP phone and PC.

Click on the connections group symbol. The specific connection symbols provide different cable types that can be used to connect devices. The first specific type, the gold lightning bolt, will automatically select the connection type based on the interfaces available on the devices. When you click on this symbol, the pointer resembles a cable connector.

To connect two devices, click the auto connection symbol, click the first device, and then click the second device. Connect devices as shown in Figure 3 part A, notes that some interfaces are not turning green (switched on). You must enable the routers interfaces as shwon in part B and C. And power on the IP phone by plugging in the power adapter as shown in parts D and E.

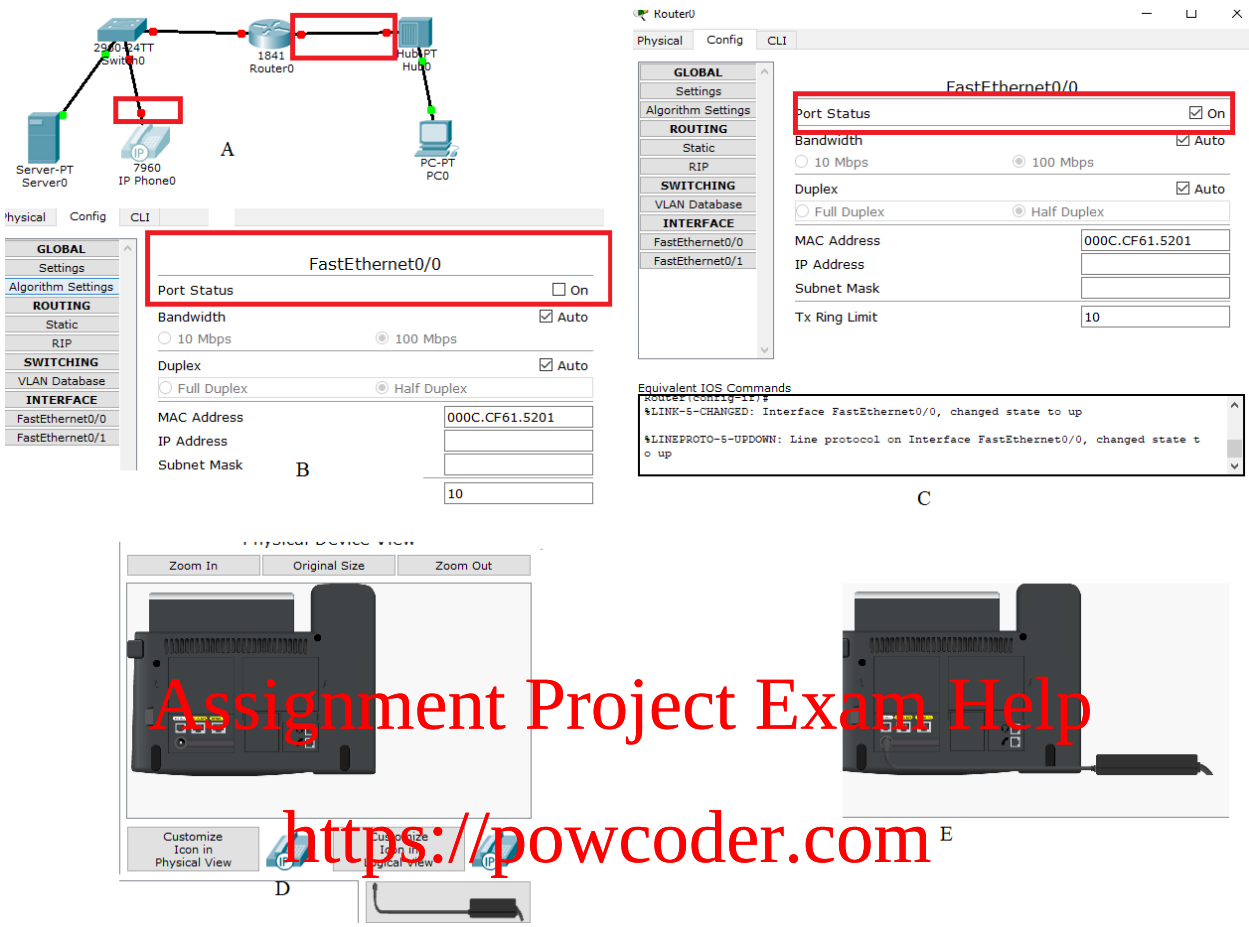


Figure 3: Connecting and enabling interfaces

Task 3: Design network and setting IP address to routers and hosts

Activity 1: Adding router 1841 model to workspace and adding WAN links.

1) From devices, left bottom, add router 1841 by dragging and dropping to workspace. Double click on router icon and a dialog window will appear shows physical view for the router.
2) Power off the router by clicking on the power button, then from modules menu choose *HWIC-2T* a brief description and image of the interface will be shown. Drag and drop the image of the interface to one of the empty module slots on the router image, then switch on the router by clicking the power button again. Figure 4 explains that.

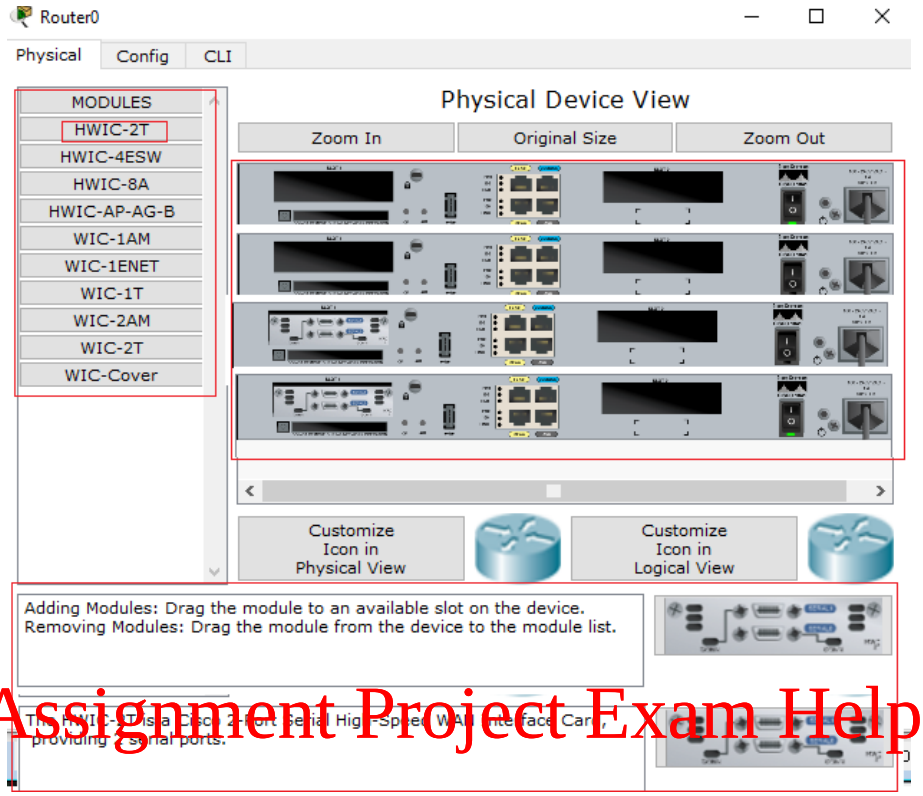


Figure 4: adding interface to router

3) From the router menu choose CLI tab, a question will show asking to configure the router using dialog configuration please choose no. You will be at *user mode* and see the router name and the "greater than" sign (*Router>*). To switch to exec privilege mode, type *enable* then enter you will see the greater than sign changed to # sign. The last mode that allows you to configure the router is global configuration mode switch to it by typing *configure terminal* command, the prompt will change to *Router(config)# as* Figure 5 shows.

4) To return to exec privilege mode type exit and the *Router#* symbol will be displayed on the command prompt. In this mode you can use *show* command that allows you to get information about router. Type *show running-config* to display the running configuration that are set on the router. You can see the interfaces that are installed on the router as in *Figure 6*.

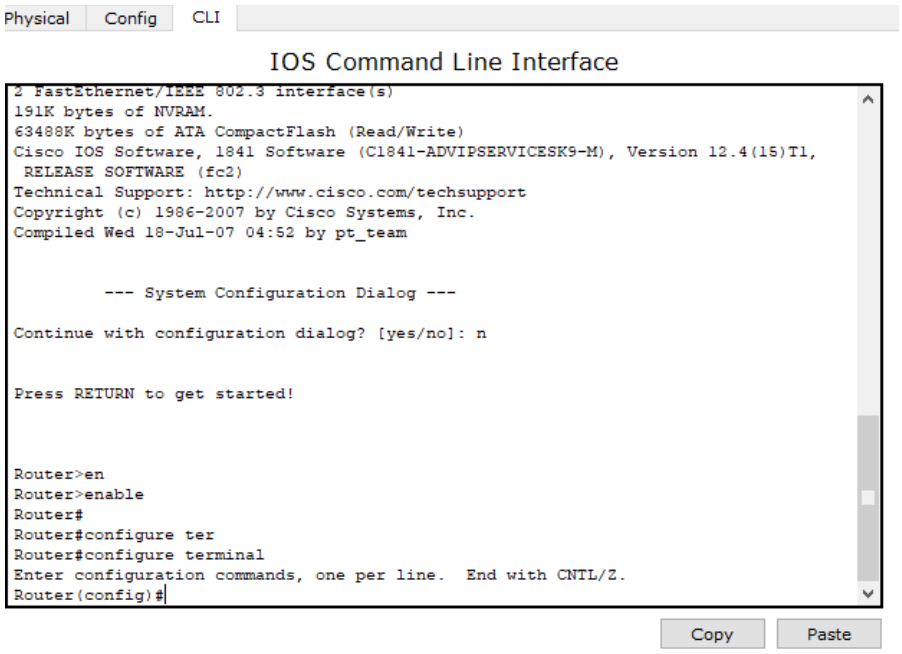


Figure 5: router configuration modes


```
Building configuration...

Current configuration : 489 bytes
!
version 12.4
no service timestamps log datetime msec
no service timestamps debug datetime msec
no service password-encryption
!
hostname Router
!
!
!
!
interface FastEthernet0/0
 no ip address
 duplex auto
 speed auto
 shutdown
!
interface FastEthernet0/1
 no ip address
 duplex auto
 speed auto
 shutdown
!
interface Serial0/1/0
 no ip address
 shutdown
!
interface Serial0/1/1
 no ip address
 shutdown
!
interface Vlan1
 no ip address
 shutdown
!
ip classless
```

Figure 6: show running-config command

Now you can make the following network topology with two 1841 routers, two 2960 switches, two PCs, and one server. For the routers, add HWIC interfaces on both routers and connect the network by appropriate links using automatic connection type as shown in Figure 7.

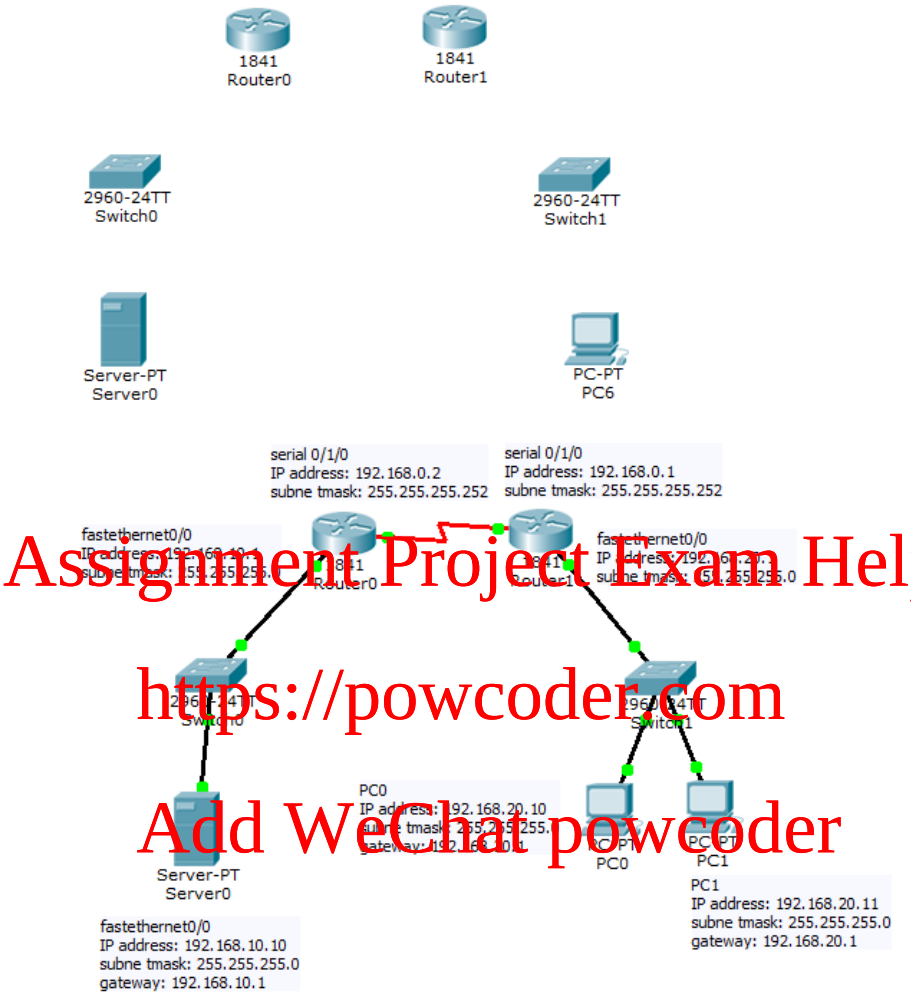


Figure 7: Network topology and IP plan

Activity 2: Assigning IP address.

On router 0, at CLI tab, apply the following commands:

```
Press RETURN to get started!
Router>enable
Router#configure terminal
Router(config)#hostname Router0

Router0(config)#interface serial 0/1/0
Router0(config-if)#no shutdown
Router0(config-if)#ip address 192.168.0.2 255.255.255.252
```

```
Router0(config)#exit

Router0(config)#interface fastEthernet 0/0
Router0(config-if)#no shutdown
Router0(config-if)#ip address 192.168.10.1 255.255.255.0
Router0(config)#exit

Router0(config)#ip route 192.168.20.0 255.255.255.0 192.168.0.1
Router0(config)#exit
Router0#copy running-config startup-config
```

On router 1, at CLI tab, apply the following commands:

```
Press RETURN to get started!
Router>enable
Router#configure terminal
Router(config)#hostname Router0

Router0(config)#interface serial 0/1/0
Router0(config-if)#no shutdown
Router0(config-if)#ip address 192.168.0.1 255.255.255.252
Router0(config)#exit

Router0(config)#interface fastEthernet 0/0
Router0(config-if)#no shutdown
Router(config-if)#ip address 192.168.20.1 255.255.255.0
Router1(config-if)#ip helper-address 192.168.10.10
Router0(config)#exit

Router0(config)#ip route 192.168.10.0 255.255.255.0 192.168.0.2
Router0(config)#exit
Router0#copy running-config startup-config
```


To validate your work, the ping command on routers must work as following:

```
router0#ping 192.168.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/8/22 ms

router0#ping 192.168.0.2

Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 192.168.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/8/14 ms

router0#ping 192.168.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/5/8 ms

router0#ping 192.168.20.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.20.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 5/6/8 ms
```



```
Router1#ping 192.168.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 6/12/23 ms
```



```
Router1#ping 192.168.20.1
```



```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.20.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/8/17 ms

Router1#ping 192.168.0.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/8/25 ms

Router1#ping 192.168.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/9 ms
```

On server, choose Desktop tab then choose IP configuration icon setting to set the IP address of the server as shown in Figure 8.

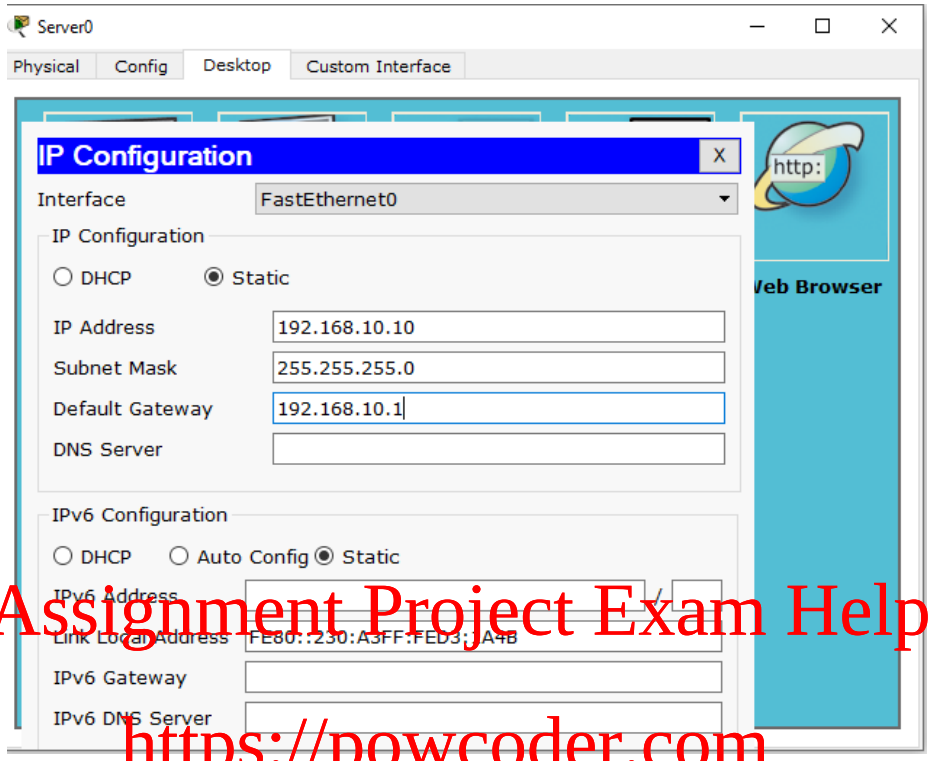


Figure 8: static IP address

1) Configure DHCP pool, choose config tab from the server main menu. Under services choose DHCP then set the information as in Figure 9, after that save by clicking on the save button. Don't forget to enable the DHCP service by turning it on at service.

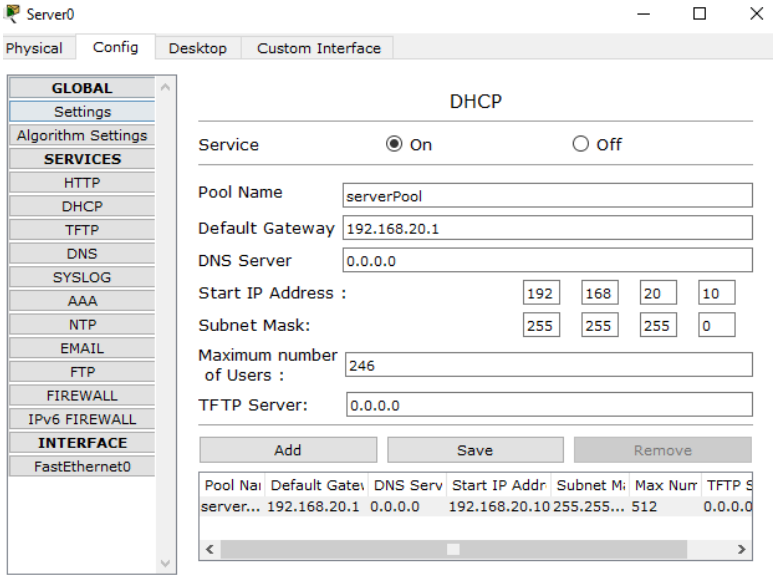


Figure 9: DHCP server configuration

2) Configure PCs' IP addresses by using DCHP service, on PC choose Desktop tab then IP configuration choose DHCP the PC will automatically get an IP from the DHCP server as figure 10.

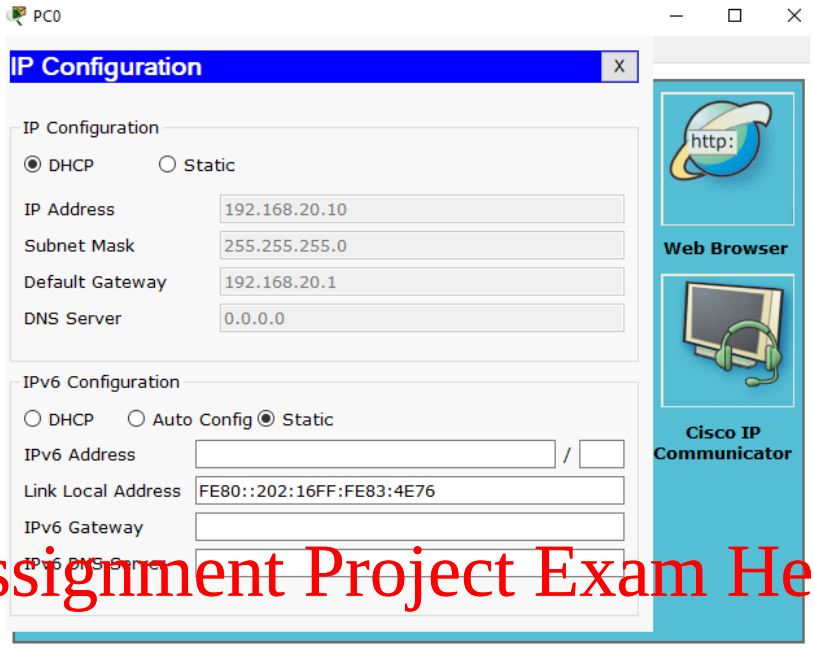


Figure 10: set PC IP suing DHCP

3) To validate the IP configuration, ping from any device in the network to another must be reachable. From PC desktop click on command prompt. Ping the DHCP server as in Figure 11.

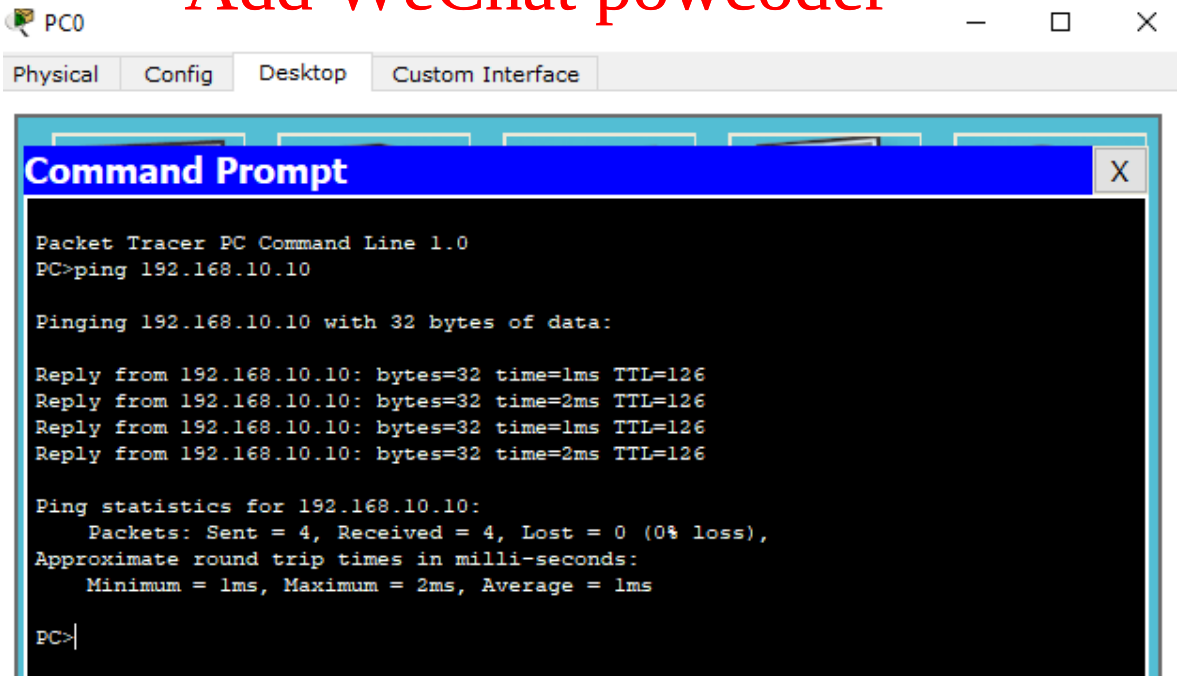


Figure 11: ping DHCP server from pc

4) Use the command ‘tracert’. The tracert command traces the path by sending Internet Control Message Protocol (ICMP) Echo Request and Echo Reply messages (similar to the ping command) to produce command-line report output about each router that is crossed and the roundtrip time (RTT) for each hop. Packet filtering policies on routers, firewalls, or other types of security gateways might prevent the forwarding of this traffic. To trace a path by using the tracert command. Open Command Prompt, and tracert the server from the PC as Figure 12.

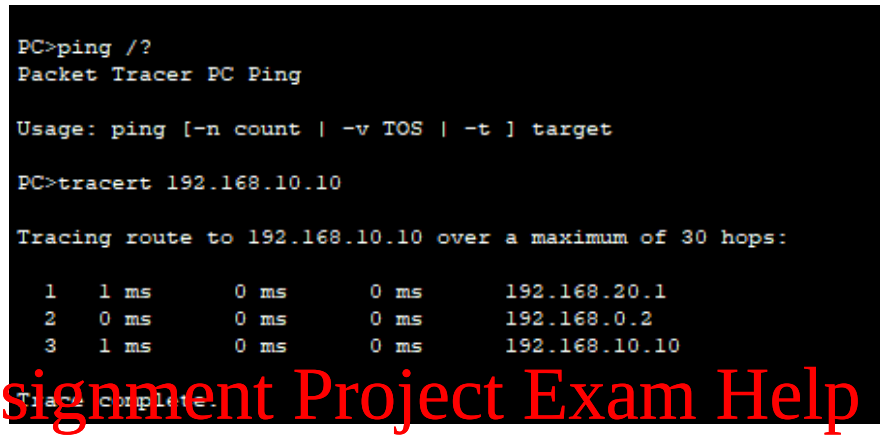


Figure 12: trace route command

Task 4: Configure FTP service

1) In the server config tab under services choose FTP, you need to set a username and a password (test, test), then choose the privilege of accessing the files (permit all). After that you add the user by clicking on the plus button. And of course, you need to enable the service by selecting on button at Service as shown in Figure 13.

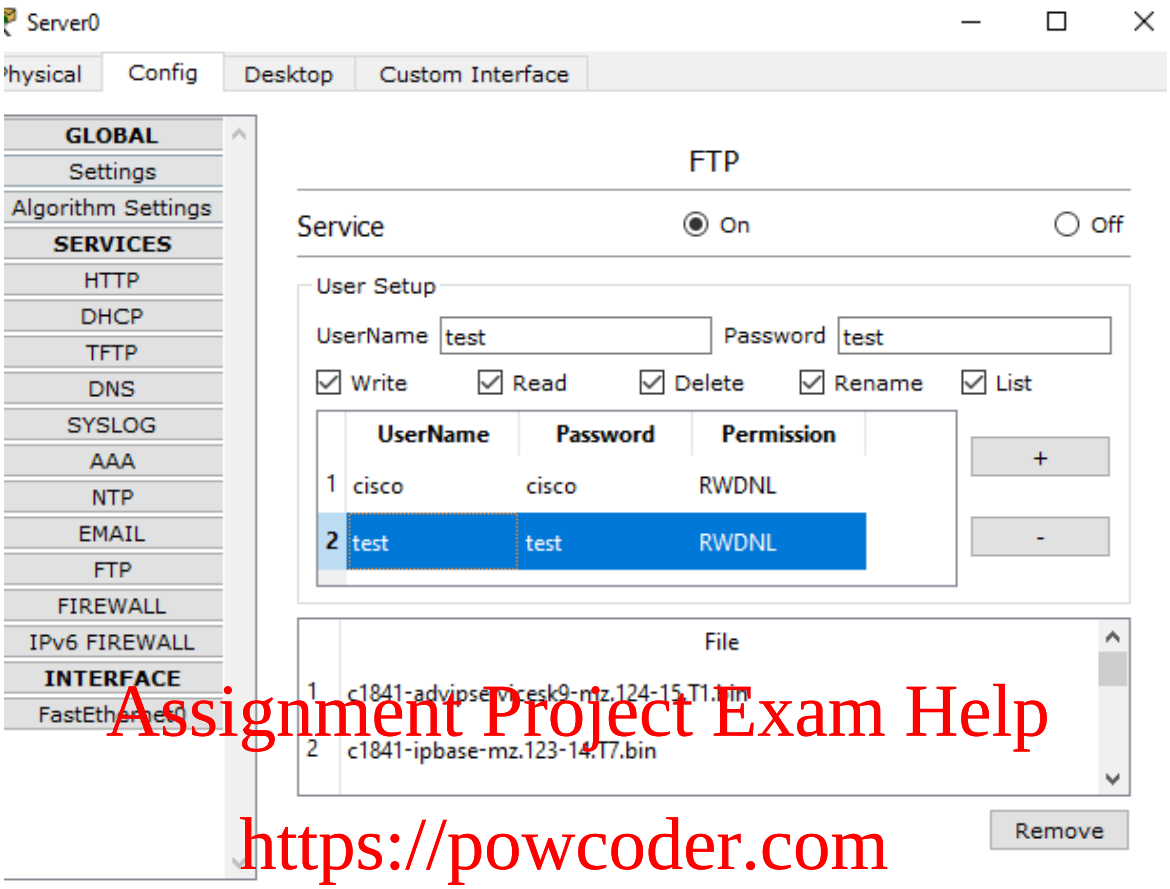


Figure 13: Configure FTP

Test the FTP service by PC. To access the files: open the Desktop in the PC then choose command prompt. Access the server by typing ftp 192.168.10.10. Username and password will be needed; use what you configured (test, test) then the files in the server are accessible to you. To list the files type *dir* as in Figure 14

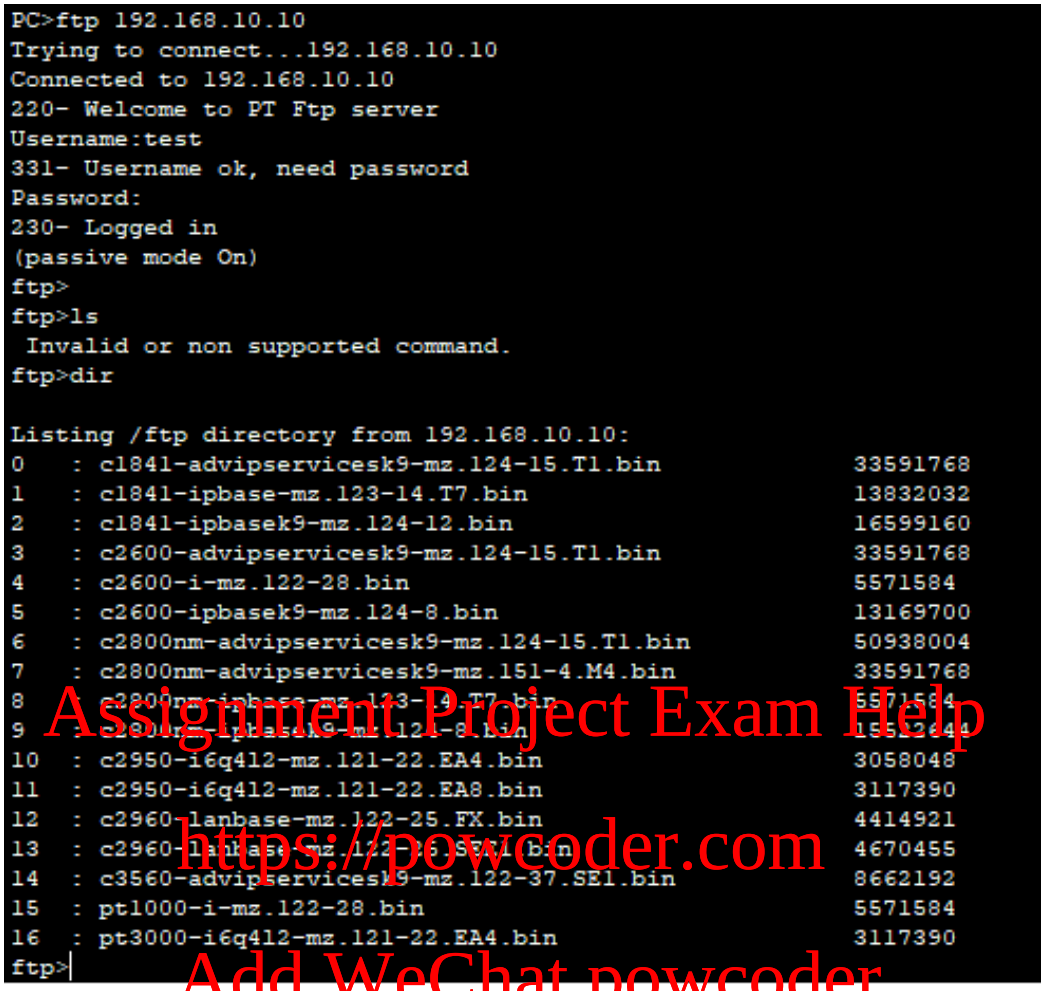


Figure 14: FTP access

Part 2: Using Wireshark to View Protocol Data Units

Objectives

In this part you will be able to explain the purpose of a protocol analyzer (Wireshark). Perform basic PDU capture, analysis using Wireshark on straightforward network data traffic. Test Wireshark features and options such as PDU capture and display filtering.

Introduction about Wireshark

Wireshark is a software protocol analyzer, or "packet sniffer" application, used for network troubleshooting, analysis, software and protocol development, and education. Before June 2006, Wireshark was known as Ethereal. A packet sniffer (also known as a network analyzer or protocol analyzer) is computer software that can intercept and log data traffic passing over a data network. As data streams travel back and forth over the network, the sniffer "captures" each protocol data unit (PDU) and can decode and analyze its content according to the appropriate

RFC or other specifications. Wireshark is programmed to recognize the structure of different network protocols. This enables it to display the encapsulation and individual fields of a PDU and interpret their meaning.

Installing and using Wireshark

Activity 1:

From the official Wireshark website (http://www.Wireshark.org) download and install Wireshark software.

Activity 2:

To capture PDUs, the computer on which Wireshark is installed must have a working connection to the network and Wireshark must be running before any data can be captured.

When Wireshark is started the screen in Figure *15* is displayed.

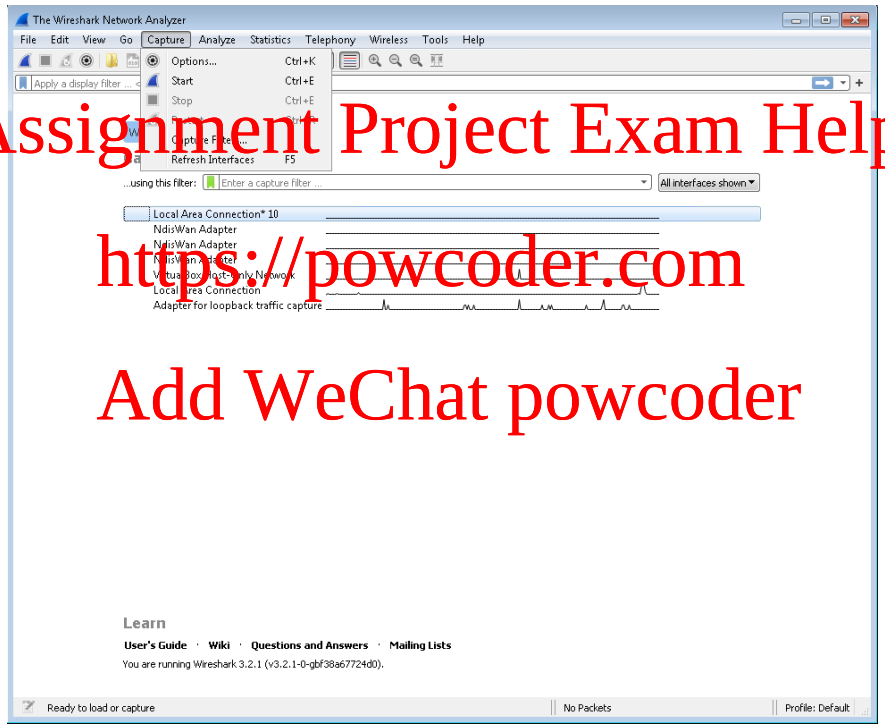


Figure 15: Wireshark home window

To start data capturing, it is first necessary to go to the Capture menu and select the Options choice. The Options dialog provides a range of settings and filters which determines which and how much data traffic is captured.

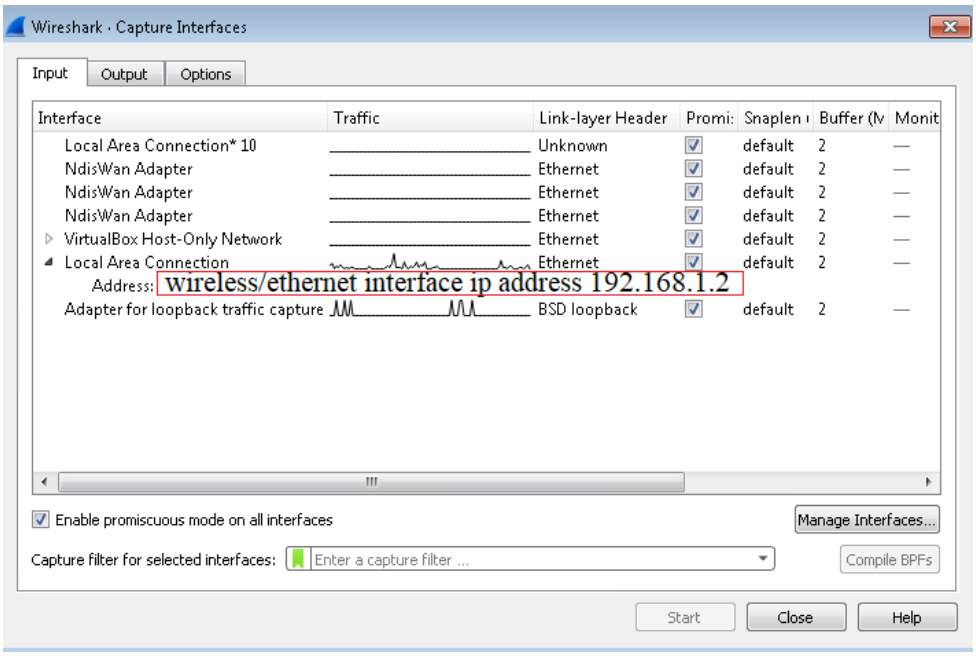


Figure 16: Interface capture

It is necessary to ensure that Wireshark is set to monitor the correct interface. From the Input tab, select the network adapter in use. Typically, for a computer this will be the connected Ethernet Adapter, and for a laptop this will be the connected wireless adapter. Then choose the adapter and start capturing by hitting start button as in Figure *16*.

Setting Wireshark to capture packets in promiscuous mode. If this feature is NOT checked, only PDUs destined for this computer will be captured. If this feature is checked, all PDUs destined for this computer AND all those detected by the computer NIC on the same network segment (i.e., those that "pass by" the NIC but are not destined for the computer) are captured.

Note: The capturing of these other PDUs depends on the intermediary device connecting the end device computers on this network. As you use different intermediary devices (hubs, switches, routers) throughout these courses, you will experience the different Wireshark results.

Setting Wireshark for network name resolution. This option allows you to control if Wireshark translates network addresses found in PDUs into names or not. Although this is a useful feature, the name resolution process may add extra PDUs to your captured data perhaps distorting the analysis.

There are also several other capture filtering and process settings available.

As data PDUs are captured, the Analyze menu can set capture filter and number which are indicated in the message box.

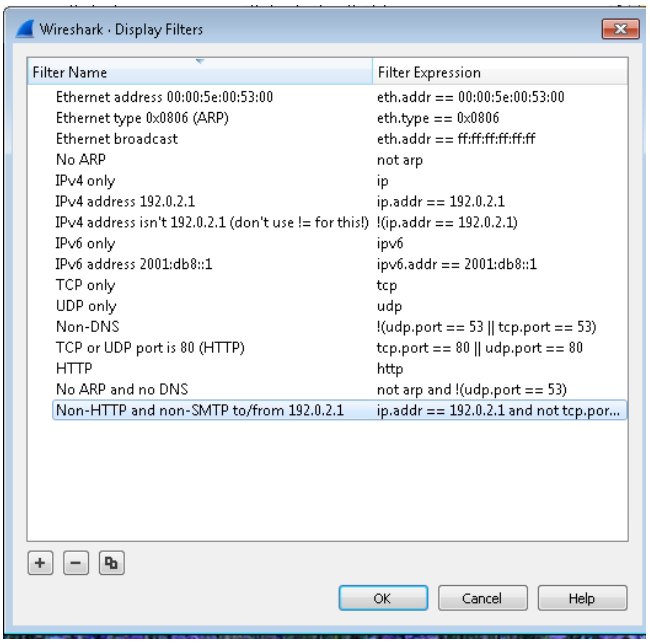


Figure 17: Display filter

The example in Figure 17 shows the capture of a HTTP process and then accessing a web page.

When the Stop button is clicked, the capture process is terminated, and the main screen is displayed.

Activity 3:

This main display window of Wireshark has three panes as in Figure *18*.

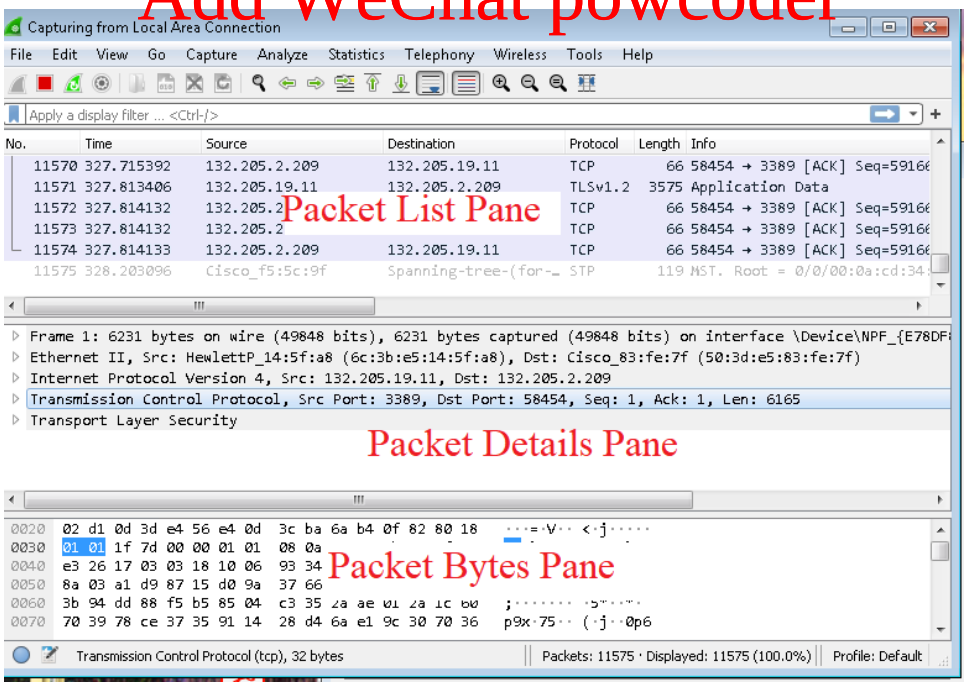


Figure 18: Wireshark panels

The PDU (or Packet) List Pane at the top of the diagram displays a summary of each packet captured. By clicking on packets in this pane, you control what is displayed in the other two panes.

The PDU (or Packet) Details Pane in the middle of the diagram displays the packet selected in the Packet List Pane in more detail.

The PDU (or Packet) Bytes Pane at the bottom of the diagram displays the actual data (in hexadecimal form representing the actual binary) from the packet selected in the Packet List Pane, and highlights the field selected in the Packet Details Pane as in Figure *19*.

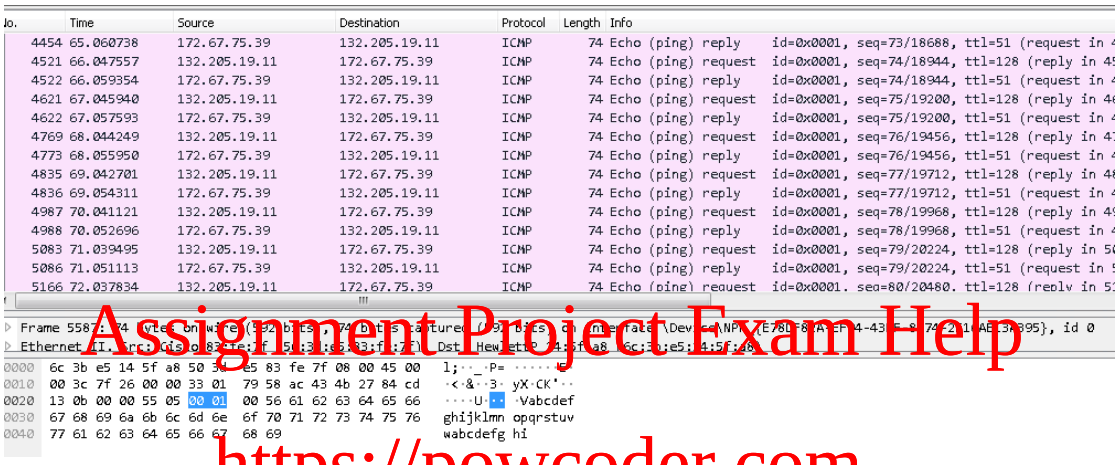


Figure 19: detail pan

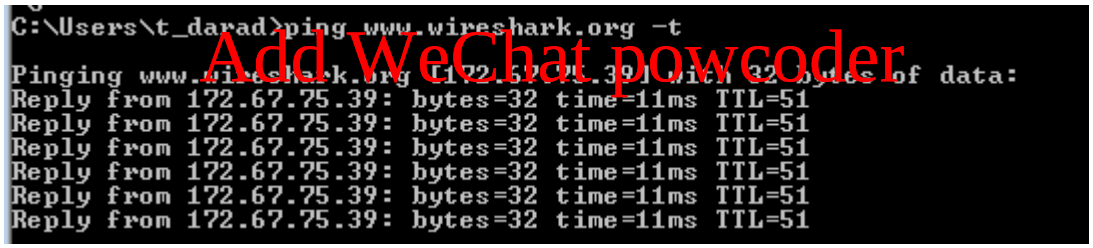


Figure 20: ping Wireshark website

Each line in the Packet List corresponds to one PDU or packet of the captured data. If you select a line in this pane, more details will be displayed in the "Packet Details" and "Packet Bytes" panes. The example above shows the PDUs captured when the ping utility was used and http://www.wireshark.org was accessed as in Figure *20*. Packet number 1 is selected in this pane.

The Packet Details pane shows the current packet (selected in the "Packet List" pane) in a more detailed form. This pane shows the protocols and protocol fields of the selected packet. The protocols and fields of the packet are displayed using a tree, which can be expanded and collapsed.

The Packet Bytes pane shows the data of the current packet (selected in the "Packet List" pane) in what is known as "hexdump" style. In this lab, this pane will not be examined in detail.

However, when a more in-depth analysis is required this displayed information is useful for examining the binary values and content of PDUs.

The information captured for the data PDUs can be saved in a file. This file can then be opened in Wireshark for future analysis without the need to re-capture the same data traffic again. The information displayed when a capture file is opened is the same as the original capture.

When closing a data capture screen or exiting Wireshark you are prompted to save the captured PDUs as in Figure *21*.

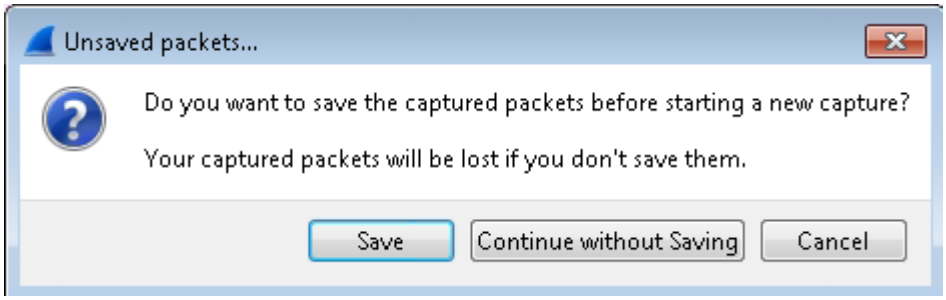


Figure 21: closing message

Clicking on Continue without Saving closes the file or exits Wireshark without saving the displayed captured data.

Task 1: Ping PDU Capture

Activity 1: Ping Wireshark website

Set the Capture Options as described above in the overview and start the capture process.

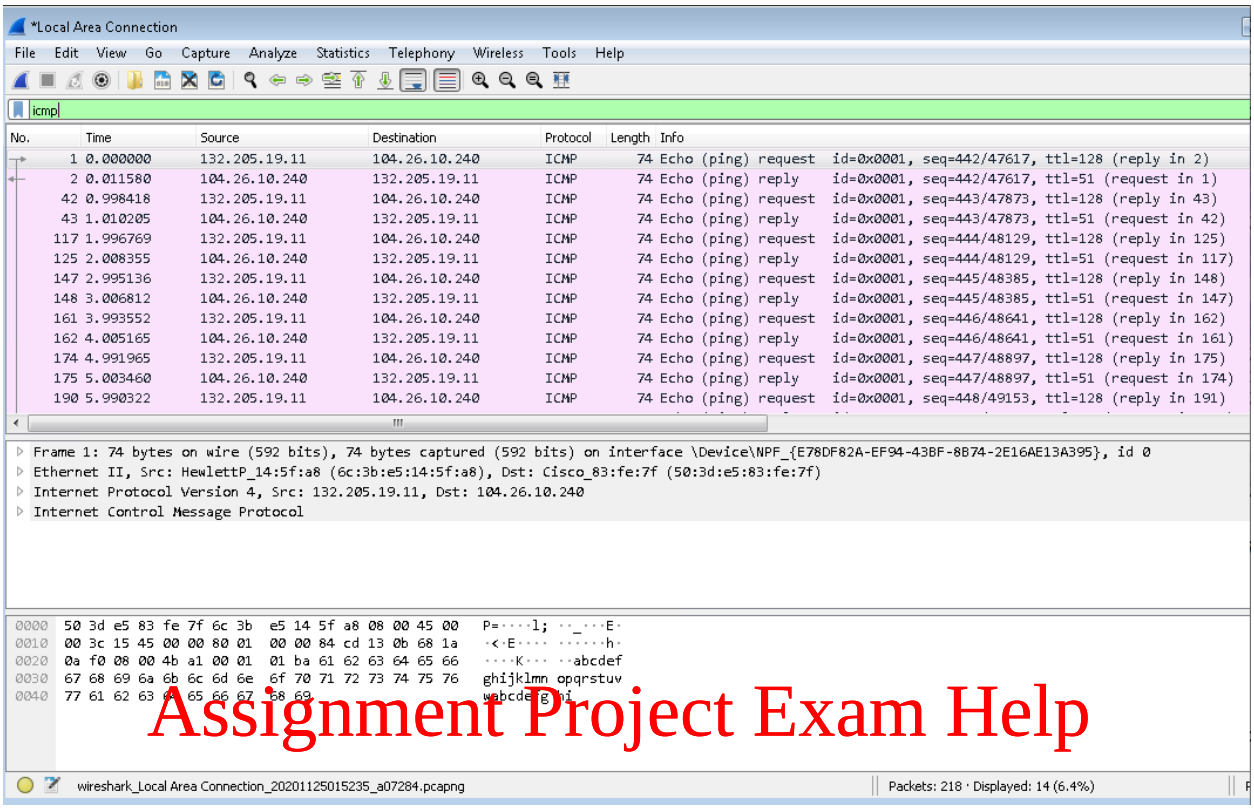
From the command line of the computer, ping the IP address of Wireshark website

command ***ping www.wireshark.org***

After receiving the successful replies to the ping in the command line window, stop the packet capture.

Activity 2: Examine the Packet List pane.

In the top par (filter) type icmp. The Packet List pane on Wireshark should now look something like this:



Look at the packets listed above; we are interested in packet numbers 1, 2, 42, 43 and 117.

Locate the equivalent packets on the packet list on your computer.

If you performed Step 1 above, match the messages displayed in the command line window when the ping was issued with the six packets captured by Wireshark.

From the Wireshark Packet List answer the following:

What protocol is used by ping? _____

What is the full protocol name? _____

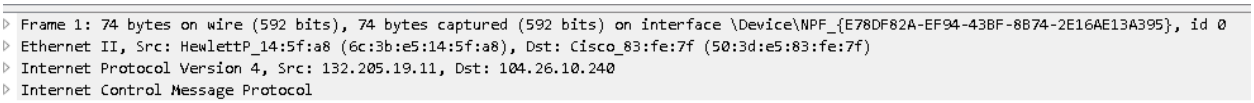
What are the names of the two ping messages? _____

_____

Are the listed source and destination IP addresses what you expected? Yes / No

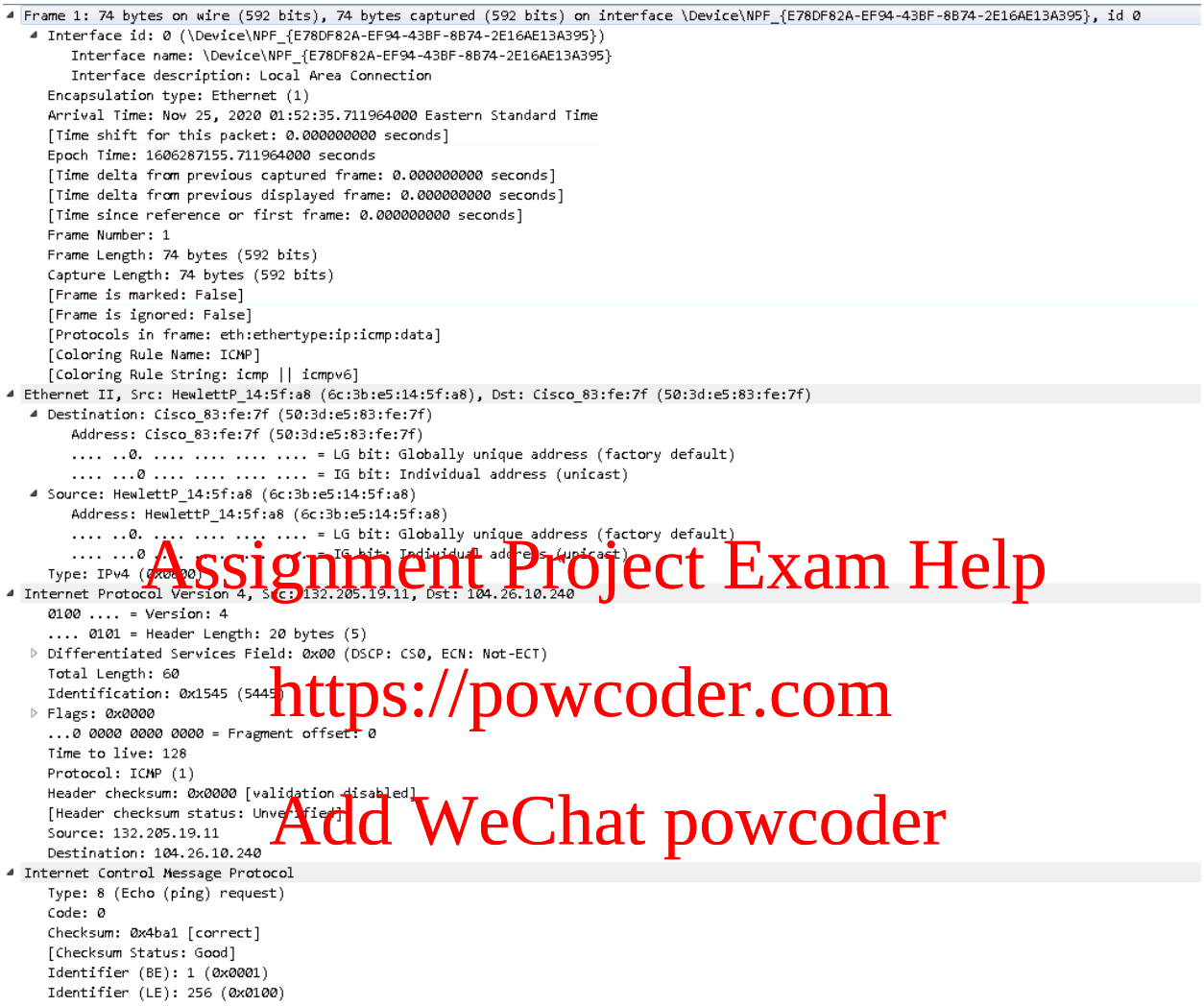
Why? _____

Activity 3: Select (highlight) the first echo request packet on the list with the mouse. The Packet Detail pane will now display something similar to the following:



Click on each of the four "+" to expand the information.

The packet Detail Pane will now be similar to the following:



As you can see, the details for each section and protocol can be expanded further. Spend some time scrolling through this information. At this stage of the course, you may not fully understand the information displayed but make a note of the information you do recognize.

Locate the two different types of 'Source" and "Destination". Why are there two types?

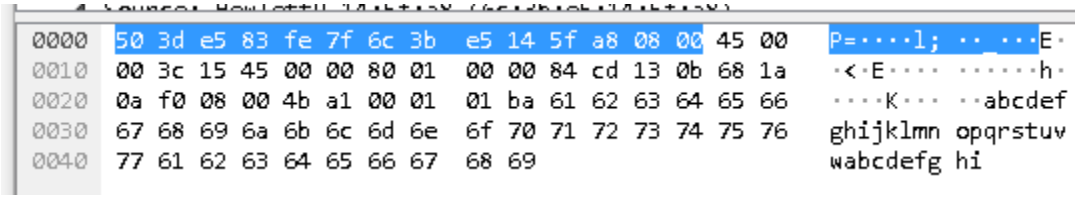
_____

What protocols are in the Ethernet frame?

_____

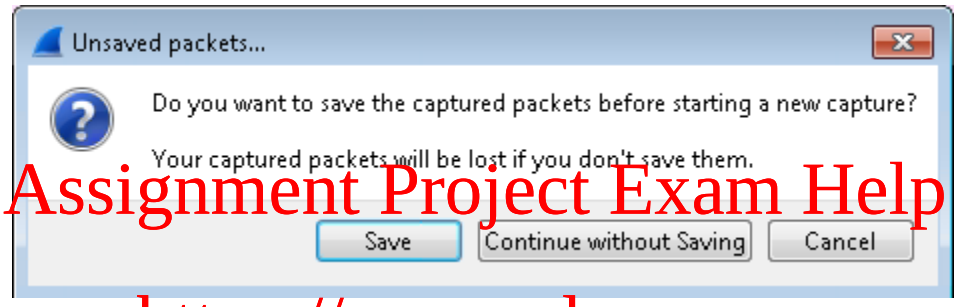
As you select a line in the Packets Detail pane all or part of the information in the Packet Bytes pane also becomes highlighted.

For example, if the second line (+ Ethernet II) is highlighted in the Details pane the Bytes pane now highlights the corresponding values.



This shows the binary values that represent that information in the PDU. At this stage of the course, it is not necessary to understand this information in detail.

Activity 4: Hit the red bottom in the shortcut menu to stop capture.
Click on Continue without Saving when this message box appears.



Experiment 02: Building network topology and enable RIP routing protocol

Introduction

In this experiment you will learn about CISCO IOS, and its basic commands. Connect routers, configure static and dynamic routes. Test network connectivity.

Background

Cisco IOS is designed as a modal operating system. The term modal describes a system where there are different modes of operation, each having its own domain of operation. The CLI uses a hierarchical structure for the modes. In order from top to bottom, the major modes are:

User executive mode (>)

Privileged executive mode (#)

Global configuration mode ((config)#)

Other specific configuration modes ((config-if)#)

Each mode is used to accomplish tasks and has a specific set of commands that are available when in that mode. For example, to configure a router interface, the user must enter interface configuration mode. All configurations that are entered in interface configuration mode apply only to that interface. Some commands are available to all users; others can be executed only after entering the mode in which that command is available. Each mode is distinguished with a distinctive prompt, and only commands that are appropriate for that mode are allowed. The hierarchal modal structure can be configured to provide security. Different authentication can be required for each hierarchal mode. This controls the level of access that network personnel can be granted.

As a security feature, the Cisco IOS software separates the EXEC sessions into two access modes. These two primary access modes are used within the Cisco CLI hierarchical structure. Each mode has similar commands. However, the privileged EXEC mode has a higher level of authority in what it allows to be executed.

User Executive Mode: The user executive mode, or user EXEC for short, has limited capabilities but is useful for some basic operations. The user EXEC mode is at the top of the modal hierarchical structure. This mode is the first entrance into the CLI of an IOS router. The user EXEC mode allows only a limited number of basic monitoring commands. This is often referred to as view-only mode. The user EXEC level does not allow the execution of any commands that might change the configuration of the device. By default, there is no authentication required to access the user EXEC mode from the console. It is a good practice to ensure that authentication is configured during the initial configuration. The user EXEC mode is identified by the CLI prompt that ends with the > symbol.

Privileged EXEC Mode: The execution of configuration and management commands requires that the network administrator use the privileged EXEC mode, or a specific mode further down

the hierarchy. The privileged EXEC mode can be identified by the prompt ending with the # symbol. By default, privileged EXEC does not require authentication. It is a good practice to ensure that authentication is configured.

Global configuration mode and all other more specific configuration modes can only be reached from the privileged EXEC mode. In a later section of this chapter, we will examine device configuration and some of the configuration modes.

The IOS CLI provides hot keys and shortcuts that make configuring, monitoring, and troubleshooting easier. The following shortcuts are most used:

Tab - Completes the remainder of the command or keyword

Ctrl-R - Redisplays a line

Ctrl-Z - Exits configuration mode and returns to the EXEC

Down Arrow - Allows user to scroll forward through former commands

Up Arrow - Allows user to scroll backward through former commands

Ctrl-Shift-6 – Allows the user to interrupt an IOS process such as ping or traceroute

Ctrl-C - Aborts the current command and exits the configuration mode

Abbreviated commands or keywords. Commands and keywords can be abbreviated to the minimum number of characters that identifies a unique selection. For example, the configure command can be abbreviated to conf because configure is the only command that begins with conf. An abbreviation of con will not work because more than one command begins with con.

Keywords can also be abbreviated. As another example, show interfaces can be abbreviated like this:

Router#show interfaces

Router#show int

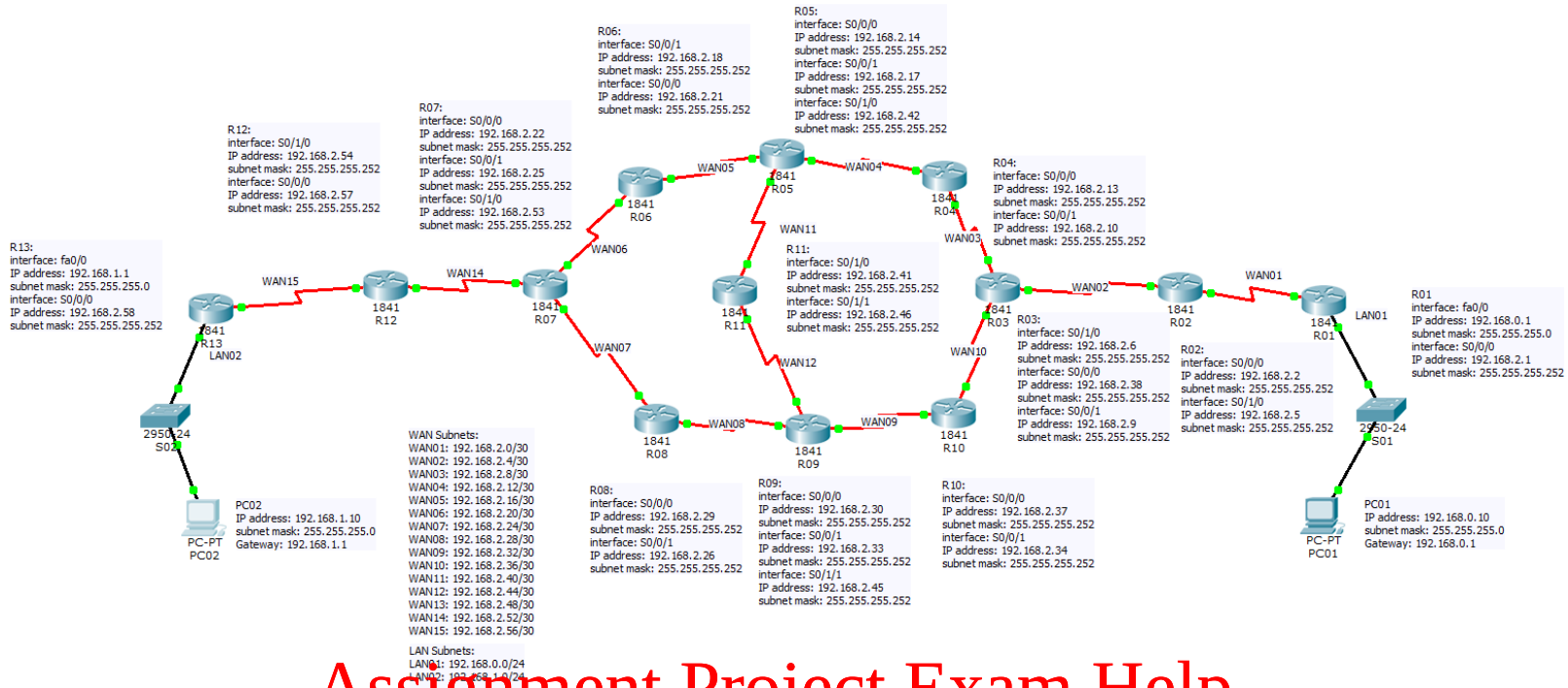
You can abbreviate both the command and the keywords, for example:

Router#sh int

Task 1: Building network topology and initial configuration

Activity1:

Build the following network topology and connect it. Do not forget to add the required hardwire in the devices, using device physical tab.



In the above figure we have 13 routers, two switches and two PCs. For serial interfaces links there are two types Terminal Equipment (DTE) and Data Communications Equipment (DCE). DCE generates clock pulses for synchronization and DTE receives clock pluses.

You need the IP plane for the WAN and LAN links. The following table summarizes all the needed IPs.

WAN Subnets:	LAN Subnets:
WAN01: 192.168.2.0/30 WAN02: 192.168.2.4/30 WAN03: 192.168.2.8/30 WAN04: 192.168.2.12/30 WAN05: 192.168.2.16/30 WAN06: 192.168.2.20/30 WAN07: 192.168.2.24/30 WAN08: 192.168.2.28/30 WAN09: 192.168.2.32/30 WAN10: 192.168.2.36/30 WAN11: 192.168.2.40/30 WAN12: 192.168.2.44/30 WAN13: 192.168.2.48/30 WAN14: 192.168.2.52/30 WAN15: 192.168.2.56/30	LAN01: 192.168.0.0/24 LAN02: 192.168.1.0/24
R01 interface: fa0/0	R02: interface: S0/0/0

IP address: 192.168.0.1 subnet mask: 255.255.255.0 interface: S0/0/0 IP address: 192.168.2.1 subnet mask: 255.255.255.252	IP address: 192.168.2.5 subnet mask: 255.255.255.252 interface: S0/1/0 IP address: 192.168.2.6 subnet mask: 255.255.255.252
R03: interface: S0/1/0 IP address: 192.168.2.7 subnet mask: 255.255.255.252 interface: S0/0/0 IP address: 192.168.2.38 subnet mask: 255.255.255.252 interface: S0/0/1 IP address: 192.168.2.9 subnet mask: 255.255.255.252	R04: interface: S0/0/0 IP address: 192.168.2.13 subnet mask: 255.255.255.252 interface: S0/0/1 IP address: 192.168.2.10 subnet mask: 255.255.255.252
R05: interface: S0/0/0 IP address: 192.168.2.14 subnet mask: 255.255.255.252 interface: S0/0/1 IP address: 192.168.2.17 subnet mask: 255.255.255.252 interface: S0/1/0 IP address: 192.168.2.42 subnet mask: 255.255.255.252	R06: interface: S0/0/18 IP address: 192.168.2.54 subnet mask: 255.255.255.252 interface: S0/0/0 IP address: 192.168.2.21 subnet mask: 255.255.255.252
R07: interface: S0/0/0 IP address: 192.168.2.22 subnet mask: 255.255.255.252 interface: S0/0/1 IP address: 192.168.2.25 subnet mask: 255.255.255.252 interface: S0/1/0 IP address: 192.168.2.53 subnet mask: 255.255.255.252	R08: interface: S0/0/0 IP address: 192.168.2.29 subnet mask: 255.255.255.252 interface: S0/0/1 IP address: 192.168.2.26 subnet mask: 255.255.255.252
R09: interface: S0/0/0 IP address: 192.168.2.30 subnet mask: 255.255.255.252 interface: S0/0/1 IP address: 192.168.2.33 subnet mask: 255.255.255.252 interface: S0/1/1 IP address: 192.168.2.45 subnet mask: 255.255.255.252	R10: interface: S0/0/0 IP address: 192.168.2.37 subnet mask: 255.255.255.252 interface: S0/0/1 IP address: 192.168.2.34 subnet mask: 255.255.255.252
R11:	R12:

interface: S0/1/0 IP address: 192.168.2.41 subnet mask: 255.255.255.252 interface: S0/1/1 IP address: 192.168.2.46 subnet mask: 255.255.255.252	interface: S0/1/0 IP address: 192.168.2.54 subnet mask: 255.255.255.252 interface: S0/0/0 IP address: 192.168.2.57 subnet mask: 255.255.255.252
R13: interface: fa0/0 IP address: 192.168.1.1 subnet mask: 255.255.255.0 interface: S0/0/0 IP address: 192.168.2.58 subnet mask: 255.255.255.252	PC01 IP address: 192.168.0.10 subnet mask: 255.255.255.0 Gateway: 192.168.0.1
PC02 IP address: 192.168.1.10 subnet mask: 255.255.255.0 Gateway: 192.168.1.1	

Activity 2:

Assign IP addresses for all network devices, and make sure to avoid any conflict by assigning same IP address more than one time. Also make sure that the subnetting is correct and you are configuring the clock rate for serial interfaces. The following table summarizes all the needed configurations.

Router> Router>enable Router#configure terminal Router(config)#hostname R01 R01(config)#no ip domain-lookup R01(config)#interface serial 0/0/0 R01(config-if)#no shutdown R01(config-if)#interface FastEthernet0/0 R01(config-if)# ip address 192.168.0.1 255.255.255.0 R01(config-if)# no shutdown R01(config-if)#exit R01(config)#exit R01#copy running-config startup-config Destination filename [startup-config]? Building configuration... [OK] R01#	Router> Router>enable Router#configure terminal Enter configuration commands, one per line. End with CNTL/Z. Router(config)#hostname R02 R02(config)#no ip domain-lookup R02(config)#interface serial 0/0/0 R02(config-if)#no shutdown %LINK-5-CHANGED: Interface Serial0/0/0, changed state to up %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0/0, changed state to up R02(config-if)#ip address 192.168.2.2 255.255.255.252 R02(config-if)#clock rate 9600 R02(config)#interface serial 0/1/0 R02(config-if)#no shutdown %LINK-5-CHANGED: Interface Serial0/1/0, changed state to down

	R02(config-if)#ip address 192.168.2.5 255.255.255.252 R02(config-if)#exit R02(config)#exit %SYS-5-CONFIG_I: Configured from console by console R02#copy running-config startup-config Destination filename [startup-config]? Building configuration... [OK] R02#
!R03 enable configure terminal hostname R03 no ip domain-lookup interface serial 0/0/0 no shutdown ip address 192.168.2.38 255.255.255.252 clock rate 9600 interface serial 0/1/0 no shutdown ip address 192.168.2.6 255.255.255.252 clock rate 9600 interface serial 0/0/1 no shutdown ip address 192.168.2.9 255.255.255.252 exit exit copy running-config startup-config	!R04 enable configure terminal hostname R04 no ip domain-lookup interface serial 0/0/0 no shutdown ip address 192.168.2.13 255.255.255.252 interface serial 0/0/1 no shutdown ip address 192.168.2.10 255.255.255.252 clock rate 9600 exit exit copy running-config startup-config
!R05 enable configure terminal hostname R05 no ip domain-lookup interface serial 0/0/0 no shutdown ip address 192.168.2.14 255.255.255.252 clock rate 9600 interface serial 0/1/0	!R06 enable configure terminal hostname R06 no ip domain-lookup interface serial 0/0/0 no shutdown ip address 192.168.2.21 255.255.255.252 interface serial 0/0/1

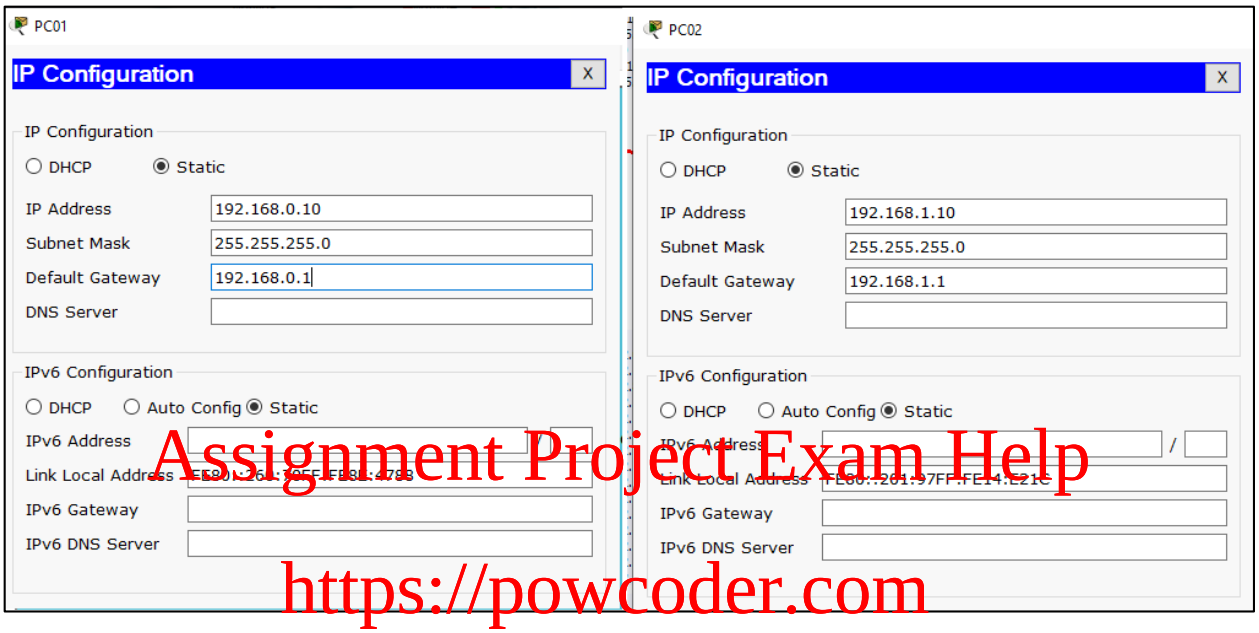
no shutdown ip address 192.168.2.42 255.255.255.252 clock rate 9600 interface serial 0/0/1 no shutdown ip address 192.168.2.17 255.255.255.252 exit exit copy running-config startup-config	no shutdown ip address 192.168.2.18 255.255.255.252 clock rate 9600 exit exit copy running-config startup-config
!R07 enable configure terminal hostname R07 no ip domain-lookup interface serial 0/0/0 no shutdown ip address 192.168.2.22 255.255.255.252 clock rate 9600 interface serial 0/1/0 no shutdown ip address 192.168.2.53 255.255.255.252 clock rate 9600 interface serial 0/0/1 no shutdown ip address 192.168.2.25 255.255.255.252 exit exit copy running-config startup-config	!R08 enable configure terminal hostname R08 no ip domain-lookup interface serial 0/0/0 no shutdown ip address 192.168.2.29 255.255.255.252 interface serial 0/0/1 no shutdown ip address 192.168.2.26 255.255.255.252 clock rate 9600 exit exit copy running-config startup-config
!R09 enable configure terminal hostname R09 no ip domain-lookup interface serial 0/0/0 no shutdown ip address 192.168.2.30 255.255.255.252 clock rate 9600 interface serial 0/1/1 no shutdown ip address 192.168.2.45 255.255.255.252	!R10 enable configure terminal hostname R10 no ip domain-lookup interface serial 0/0/0 no shutdown ip address 192.168.2.37 255.255.255.252 interface serial 0/0/1 no shutdown ip address 192.168.2.34 255.255.255.252

clock rate 9600 interface serial 0/0/1 no shutdown ip address 192.168.2.33 255.255.255.252 exit exit copy running-config startup-config	clock rate 9600 exit exit copy running-config startup-config
!R11 enable configure terminal hostname R11 no ip domain-lookup interface serial 0/1/0 no shutdown ip address 192.168.2.41 255.255.255.252 interface serial 0/1/1 no shutdown ip address 192.168.2.46 255.255.255.252 exit exit copy running-config startup-config	!R12 enable configure terminal hostname R12 no ip domain-lookup interface serial 0/1/0 no shutdown ip address 192.168.2.54 255.255.255.252 interface serial 0/0/0 no shutdown ip address 192.168.2.57 255.255.255.252 clock rate 9600 exit exit copy running-config startup-config
!R13 enable configure terminal hostname R13 no ip domain-lookup interface FastEthernet0/0 ip address 192.168.1.1 255.255.255.0 no shutdown interface Serial0/0/0 ip address 192.168.2.58 255.255.255.252 no shutdown exit exit copy running-config startup-config	

Note that from R03 you can just copy and paste the configurations and it will take place immediately.

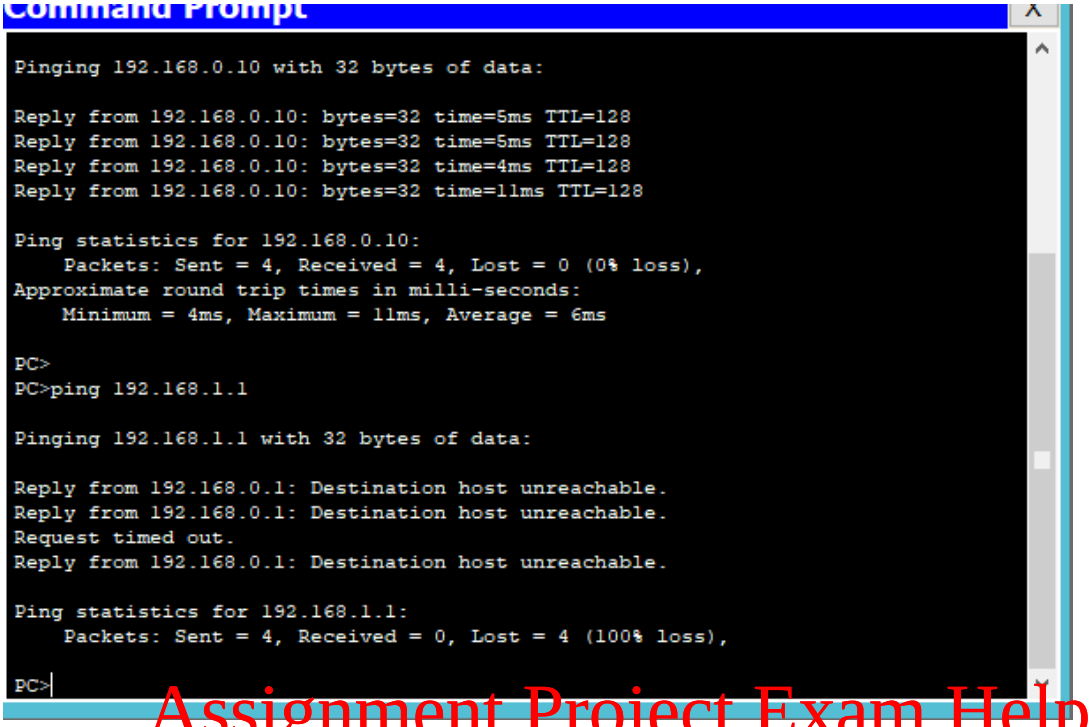
Activity 3:

Configure PC IP, set the IP address subnet mask and default gateway in PC as in the following figures.



Activity 4:

connectivity test using ping command, list the reachability between the devices. can you ping devices over any WAN links? Example from PC command prompt, try to ping GW and the other interface in the same router as in the following figure.



Why cannot you reach further distance IPs?

Now please **save the file**, because you will use it for next experiment. Make a new copy every time you work on it.

Task 2: Configure RIP routing protocol

Introduction

Routing is used for taking a packet from one device and sending it through the network to another device on a different network. Routers route traffic to all networks in your internetwork. To be able to route packets, a router must know, at a minimum, the following:

- Destination address
- Neighbor routers from which it can learn about remote networks
- Possible routes to all remote networks
- The best route to each remote network

How to maintain and verify routing information. If the destination network is directly connected to the router, then it already knows through which interface it can get to the network. If it is not directly connected, the router must learn how to get to the remote network with either static routing, which means that the administrator must hand-type all network locations into the routing table or use dynamic routing. Dynamic routing is the process of routing protocols running on the router communicating with neighbor routers. The routers then update each other about all the networks they know about. If a change occurs in the network, the dynamic routing protocols automatically inform all routers about the change.

Routing Information Protocol (RIP)

RIP is a distance-vector routing protocol. It sends the complete routing table out to all active interfaces every 30 seconds. RIP only uses hop count to determine the best way to a remote network but has a maximum allowable hop count of 15. RIP version 1 uses only classful routing, which means that all devices in the network must use the same subnet mask. This is because RIP version 1 does not send updates with subnet mask information included. To configure RIP routing, you need to turn on the protocol with the "router rip" command and tell the RIP routing which networks to advertise. Setting the routing protocol will allow you now to reach the networks.

Activity 1:

Use the following configuration to configure all routers. The configuration must disable auto routing summary and enable RIP version 2.

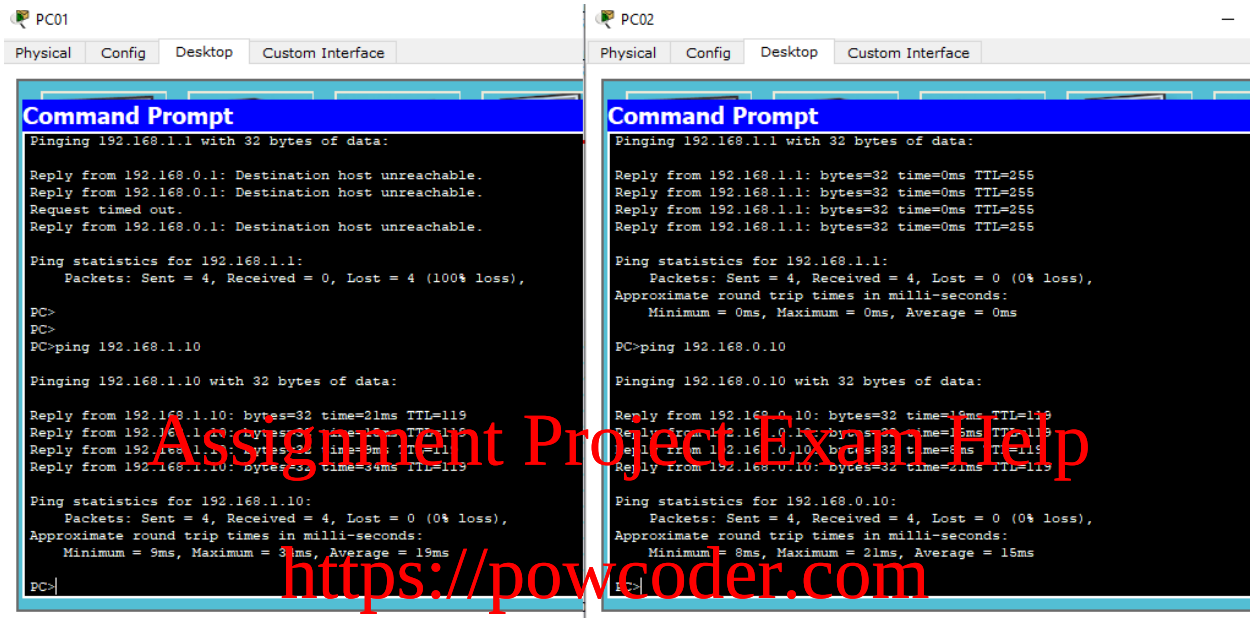
!R01 enable configure terminal router rip version 2 network 192.168.0.0 network 192.168.2.0 end write memory	!R02 enable configure terminal router rip version 2 network 192.168.2.0 network 192.168.2.4 end write memory
!R03 enable configure terminal router rip version 2 network 192.168.2.4 network 192.168.2.8 network 192.168.2.36 end write memory	!R04 enable configure terminal router rip version 2 network 192.168.2.8 network 192.168.2.12 end write memory
!R05 enable configure terminal router rip version 2 network 192.168.2.12 network 192.168.2.16 network 192.168.2.40 end	!R06 enable configure terminal router rip version 2 network 192.168.2.16 network 192.168.2.20 end write memory

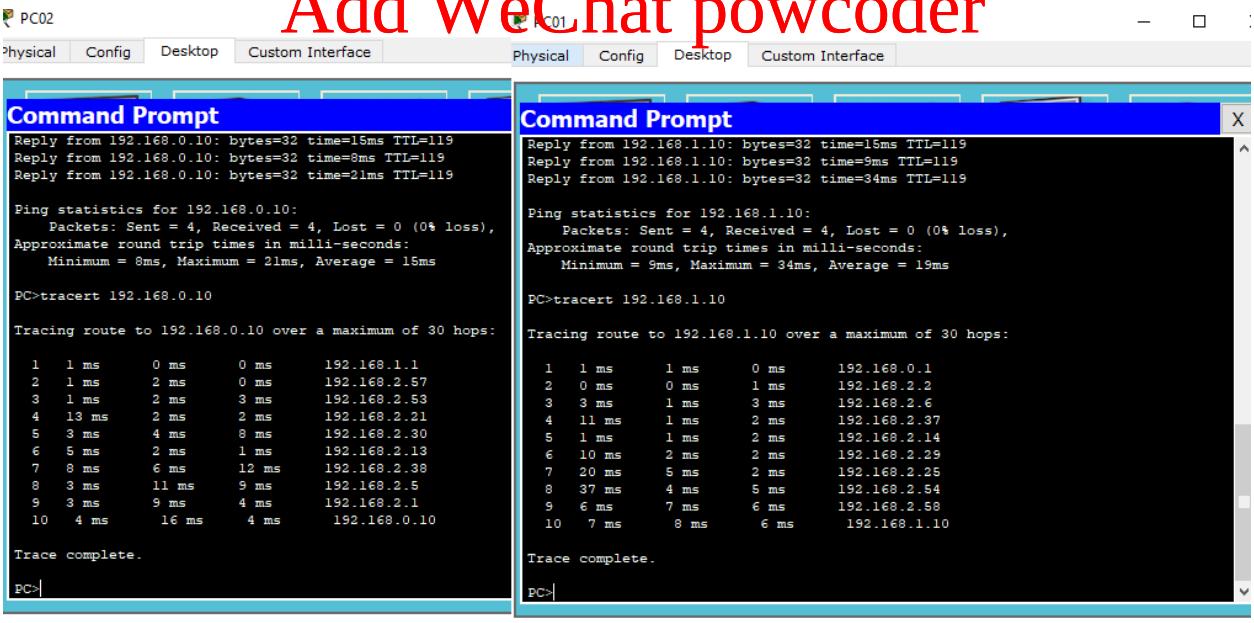
write memory	
!R07 enable configure terminal router rip version 2 network 192.168.2.20 network 192.168.2.24 network 192.168.2.52 end write memory	!R08 enable configure terminal router rip version 2 network 192.168.2.24 network 192.168.2.28 end write memory
!R09 enable configure terminal router rip version 2 network 192.168.2.28 network 192.168.2.32 network 192.168.2.44 end write memory	!R10 enable configure terminal router rip version 2 network 192.168.2.32 network 192.168.2.36 end write memory
!R11 enable configure terminal router rip version 2 network 192.168.2.40 network 192.168.2.44 end write memory	!R12 enable configure terminal router rip version 2 network 192.168.2.52 network 192.168.2.56 end write memory
!R13 enable configure terminal router rip version 2 network 192.168.2.56 network 192.168.1.0 end write memory	

Activity 2:

Try now to ping PC2 from PC1, can you reach it? And ping PC1 from PC2, Can you reach it? You are supposed to be able to reach any device in the network! Why?

Try to use *tracetrt* command and find the path between two hosts





Task 3: understand RIP routing protocol

Activity1:

Display the routing table and verify that you have valid routes and connectivity to all other routers in the network. Use the following command:

show ip route

Activity 2:

Make sure that you can successfully ping all of the other routers in your network by pinging any IP address in the router.

Activity 3:

Examine the routing table of your router and answer the following questions:

show ip route rip

show ip rip database

What are the routes between R1 and R13, how many hops are they away?

R1 via router/s ……and …… hops away.

By default, RIP calculates the routing metric based on ……..

By comparing the routing table with network topology, discuss the algorithm that RIP is using to find out the network topology and if it is finding the optimized routes in your network.

What information does any router have about other network networks? Is this information correct?

Activity 4:

Issue the following command to capture RIP routing updates.

R01(config)# logging console debugging

R01# debug ip rip events

Check the routing updates sent by your router through its interfaces:

What is/are the destination address/es of these updates? What does it mean?

How long does it take for the router to send two consecutive updates via interface S0/0/0?

Activity 5:

Save the current configuration of your router and exit.

R01(config)# no logging console debugging

R01# undebug all

Experiment 03: OSPF routing protocol

Introduction

Determining a router's best path involves the evaluation of multiple paths to the same destination network and selecting the optimum or "*shortest*" path to reach that network. Whenever multiple paths to reach the same network exist, each path uses a different exit interface on the router to reach that network. The best path is selected by a routing protocol based on the value or metric it uses to determine the distance to reach a network. Some routing protocols, such as RIP, use simple hop-count, which is the number of routers between a router and the destination network. Other routing protocols, such as OSPF, determine the shortest path by examining the bandwidth of the links, and using the links with the fastest bandwidth from a router to the destination network.

Dynamic routing protocols typically use their own rules and metrics to build and update routing tables. A metric is the quantitative value used to measure the distance to a given route. The best path to a network is the path with the lowest metric. For example, a router will prefer a path that is 5 hops away over a path that is 10 hops away.

The primary objective of the routing protocol is to determine the best path for each route to include in the routing table. The routing algorithm generates a value, or a metric, for each path through the network. Metrics can be based on either a single characteristic or several characteristics of a path. Some routing protocols can base route selection on multiple metrics, combining them into a single metric. The smaller the value of the metric, the better the path.

Comparing Hop Count and Bandwidth Metrics

Two metrics that are used by some dynamic routing protocols are:

- Hop count: Hop count is the number of routers that a packet must travel through before reaching its destination. Each router is equal to one hop. A hop count of four indicates that a packet must pass through four routers to reach its destination. If multiple paths are available to a destination, the routing protocol, such as RIP, picks the path with the least number of hops.
- Bandwidth: Bandwidth is the data capacity of a link, sometimes referred to as the speed of the link. For example, Cisco's implementation of the OSPF routing protocol uses bandwidth as its metric. The best path to a network is determined by the path with an accumulation of links that have the highest bandwidth values, or the fastest links.

Note: Speed is technically not an accurate description of bandwidth because all bits travel at the same speed over the same physical medium. Bandwidth is more accurately defined as the number of bits that can be transmitted over a link per second.

When hop count is used as the metric, the resulting path may sometimes be suboptimal.

Open Shortest Path First (OSPF)

OSPF was developed by the internet community to answer the need to a highly functional and non-proprietary Internal Gateway Protocol (IGP) for TCP/IP. OSPF is a link-state routing protocol, which is different from the Bellman-Ford vector-based algorithm used in RIP.

OSPF is a link-state protocol. A link can be thought like an interface on the router. The link state is a description of the interface and its relationship with its neighbor routers. An interface description can include the IP address of interface, subnet mask, peer IP address, bandwidth, etc. All these links state information is kept in link state database. OSPF uses the Dijkstra algorithm to calculate the shortest path to all known destinations. This is a link state algorithm. The algorithm places each router as the root of a tree and calculates the shortest path to each destination based on the cumulative cost required to reach that destination through the tree. In an OSPF network, each router has its own view of the network topology. Link or interface cost is a general indication of the overhead required to reach the peer router through a certain interface. In OSPF, the interface cost is inversely proportional to the bandwidth of that interface. A higher bandwidth indicates a lower cost, therefore, less overhead to reach the destination.

OSPF uses multicast flooding to exchange link-state updates among all routers. Any change in network topology is flooded to all routers in the network. Areas are introduced to put a boundary on link-state updates. Link-state flooding and calculation of Dijkstra algorithm is limited to changes within an area. All routers within an area have the same link-state database. A router with all of its interfaces within the same area is called an Internal Router (IR). A router with interfaces in multiple areas is called, Area Border Router (ABR).

Configuring OSPF on Routers

Configuring OSPF includes the following two steps in global configuration mode:

- Enabling an OSPF process using the router ospf <process-id> command.
- Assigning areas to the interfaces using the network <network or IP address> < Wildcard-mask> <area-id> command.

The OSPF process-id is a numeric value local to the router. It does not have to match process-ids on other routers.

Task1: Configure OSPF

Activity1:

Using the network topology template packet tracer file, apply the following configurations on the routers.

!R01 enable configure terminal router ospf 1	!R02 enable configure terminal router ospf 1

network 192.168.0.0 0.0.0.255 area 0 network 192.168.2.0 0.0.0.3 area 0 end write memory	 network 192.168.2.0 0.0.0.3 area 0 network 192.168.2.4 0.0.0.3 area 0 end write memory
!R03 enable configure terminal router ospf 1 network 192.168.2.4 0.0.0.3 area 0 network 192.168.2.8 0.0.0.3 area 0 network 192.168.2.36 0.0.0.3 area 0 end write memory	!R04 enable configure terminal router ospf 1 network 192.168.2.8 0.0.0.3 area 0 network 192.168.2.12 0.0.0.3 area 0 end write memory
!R05 enable configure terminal router ospf 1 network 192.168.2.12 0.0.0.3 area 0 network 192.168.2.16 0.0.0.3 area 0 network 192.168.2.40 0.0.0.3 area 0 end write memory	!R06 enable configure terminal router ospf 1 network 192.168.2.16 0.0.0.3 area 0 network 192.168.2.20 0.0.0.3 area 0 end write memory
!R07 enable configure terminal router ospf 1 network 192.168.2.20 0.0.0.3 area 0 network 192.168.2.24 0.0.0.3 area 0 network 192.168.2.52 0.0.0.3 area 0 end write memory	!R08 enable configure terminal router ospf 1 network 192.168.2.24 0.0.0.3 area 0 network 192.168.2.28 0.0.0.3 area 0 end write memory
!R09 enable configure terminal router ospf 1	!R10 enable configure terminal router ospf 1

network 192.168.2.28 0.0.0.3 area 0 network 192.168.2.32 0.0.0.3 area 0 network 192.168.2.44 0.0.0.3 area 0 end write memory	network 192.168.2.32 0.0.0.3 area 0 network 192.168.2.36 0.0.0.3 area 0 end write memory
!R11 enable configure terminal router ospf 1 network 192.168.2.40 0.0.0.3 area 0 network 192.168.2.44 0.0.0.3 area 0 end write memory	!R12 enable configure terminal router ospf 1 network 192.168.2.52 0.0.0.3 area 0 network 192.168.2.56 0.0.0.3 area 0 end write memory
!R13 enable configure terminal router ospf 1 network 192.168.2.56 0.0.0.3 area 0 network 192.168.1.0 0.0.0.255 area 0 end write memory	

Activity 2:

Verify routing table in a random two routers as shown in the following figures.

```
R13#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

O    192.168.0.0/24 [110/513] via 192.168.2.57, 00:17:42, Serial0/0/0
C    192.168.1.0/24 is directly connected, FastEthernet0/0
     192.168.2.0/30 is subnetted, 14 subnets
O       192.168.2.0 [110/512] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.4 [110/448] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.8 [110/384] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.12 [110/320] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.16 [110/256] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.20 [110/192] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.24 [110/192] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.28 [110/256] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.32 [110/320] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.36 [110/384] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.40 [110/320] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.44 [110/320] via 192.168.2.57, 00:17:42, Serial0/0/0
O       192.168.2.52 [110/128] via 192.168.2.57, 00:17:42, Serial0/0/0
C       192.168.2.56 is directly connected, Serial0/0/0
```

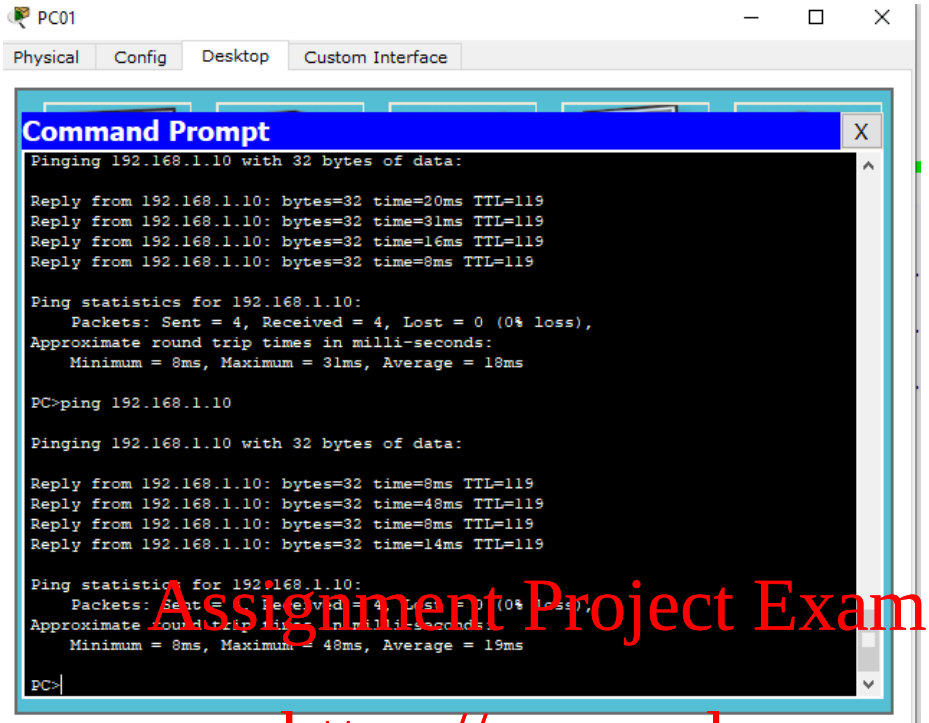
```
R01#show  ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.0.0/24 is directly connected, FastEthernet0/0
O    192.168.1.0/24 [110/513] via 192.168.2.2, 00:18:30, Serial0/0/0
     192.168.2.0/30 is subnetted, 14 subnets
C       192.168.2.0 is directly connected, Serial0/0/0
O       192.168.2.4 [110/128] via 192.168.2.2, 00:21:06, Serial0/0/0
O       192.168.2.8 [110/192] via 192.168.2.2, 00:20:43, Serial0/0/0
O       192.168.2.12 [110/256] via 192.168.2.2, 00:20:33, Serial0/0/0
O       192.168.2.16 [110/320] via 192.168.2.2, 00:20:22, Serial0/0/0
O       192.168.2.20 [110/384] via 192.168.2.2, 00:20:09, Serial0/0/0
O       192.168.2.24 [110/384] via 192.168.2.2, 00:19:29, Serial0/0/0
O       192.168.2.28 [110/320] via 192.168.2.2, 00:19:29, Serial0/0/0
O       192.168.2.32 [110/256] via 192.168.2.2, 00:19:29, Serial0/0/0
O       192.168.2.36 [110/192] via 192.168.2.2, 00:20:43, Serial0/0/0
O       192.168.2.40 [110/320] via 192.168.2.2, 00:20:22, Serial0/0/0
O       192.168.2.44 [110/320] via 192.168.2.2, 00:19:29, Serial0/0/0
O       192.168.2.52 [110/448] via 192.168.2.2, 00:19:59, Serial0/0/0
O       192.168.2.56 [110/512] via 192.168.2.2, 00:18:54, Serial0/0/0
```

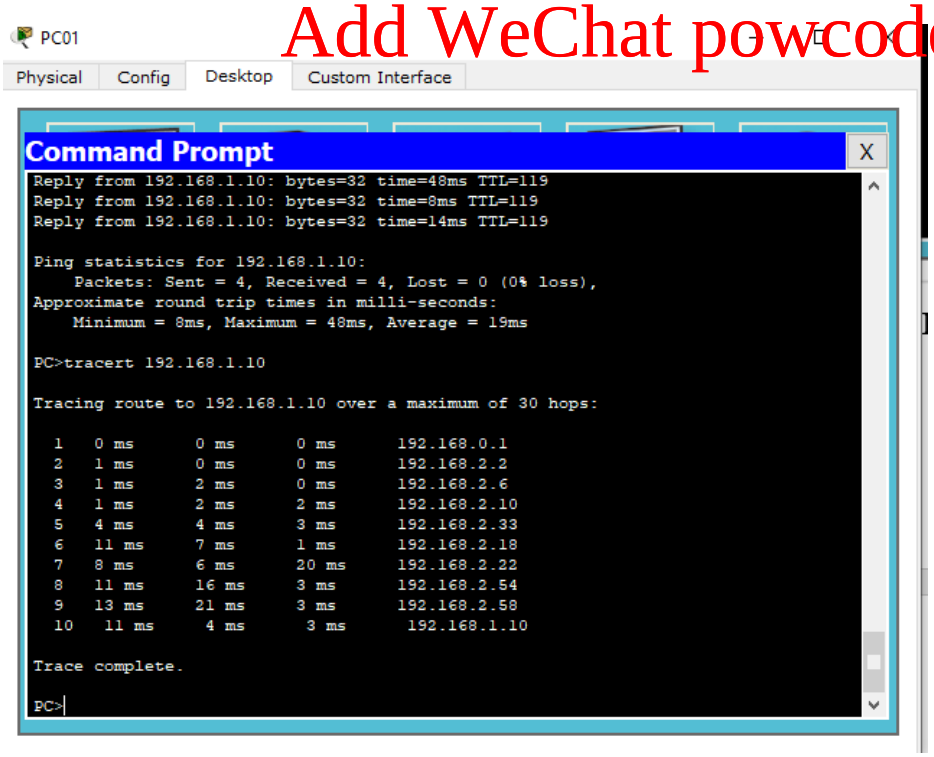
Activity 3:

Verifying the connectivity by using ping tool between PC1 and PC2.



Activity 4:

Use tracert command in PCs, show the paths between the PCs which routers choose.



Task 2: change OSPF routing table

Activity 1:

Issue show ip route command.

```
R05#show ip rou
R05#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

O    192.168.0.0/24 [110/257] via 192.168.2.13, 03:00:51, Serial0/0/0
O    192.168.1.0/24 [110/257] via 192.168.2.18, 02:58:57, Serial0/0/1
     192.168.2.0/30 is subnetted, 14 subnets
O       192.168.2.0 [110/256] via 192.168.2.13, 03:00:51, Serial0/0/0
O       192.168.2.4 [110/192] via 192.168.2.13, 03:00:51, Serial0/0/0
O       192.168.2.8 [110/128] via 192.168.2.13, 03:00:51, Serial0/0/0
C       192.168.2.12 is directly connected, Serial0/0/0
C       192.168.2.16 is directly connected, Serial0/0/1
O       192.168.2.20 [110/128] via 192.168.2.18, 03:00:40, Serial0/0/1
O       192.168.2.24 [110/192] via 192.168.2.18, 03:00:20, Serial0/0/1
O       192.168.2.28 [110/192] via 192.168.2.41, 02:59:40, Serial0/1/0
O       192.168.2.32 [110/192] via 192.168.2.41, 02:59:40, Serial0/1/0
O       192.168.2.36 [110/192] via 192.168.2.13, 03:00:51, Serial0/0/0
C       192.168.2.40 is directly connected, Serial0/1/0
O       192.168.2.44 [110/128] via 192.168.2.41, 02:59:40, Serial0/1/0
O       192.168.2.52 [110/192] via 192.168.2.18, 03:00:20, Serial0/0/1
O       192.168.2.56 [110/256] via 192.168.2.18, 02:59:20, Serial0/0/1
R05#
```


Routing table shows the destination network and the preferred interface to direct the traffic to according to the cost.

Activity 2:

What are the metrics that OSPF depend on?

What do the values between square brackets [110/257] mean? explain the records in the router table.

Which route entry has higher preference in the table.

Activity 3:

Compare the routing cost and entry between OSPF and RIP from the previous experiment.

Task3: change bandwidth in routers

Activity 1:

check interfaces assigned IPs, and routers interface bandwidth using the two commands:

Show interface serial 0/0/0

Show ip interface brief

R05#show interfaces serial 0/0/0
Serial0/0/0 is up, line protocol is up (connected)
  Hardware is HD64570
  Internet address is 192.168.2.14/30
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
  reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set,
keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
  Conversations 0/0/256 (active/max active/max total)
  Reserved Conversations 0/0 (allocated/max allocated)
  Available Bandwidth 1158 kilobits/sec
  5 minute input rate 61 bits/sec, 0 packets/sec
  5 minute output rate 65 bits/sec, 0 packets/sec
  1176 packets input, 83256 bytes, 0 no buffer
  Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
  0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
  1169 packets output, 83772 bytes, 0 underruns
	R05#show ip interface brief
	Interface IP-Address OK? Method Status Protocol
	
	FastEthernet0/0 unassigned YES unset administratively down down
	
	FastEthernet0/1 unassigned YES unset administratively down down
	
	Serial0/0/0 192.168.2.14 YES manual up up
	
	Serial0/0/1 192.168.2.17 YES manual up up
	
	Serial0/1/0 192.168.2.42 YES manual up up
	
	Serial0/1/1 unassigned YES unset administratively down down
	
	Vlan1 unassigned YES unset administratively down down
	R05#

0 output errors, 0 collisions, 2 interface resets 0 output buffer failures, 0 output buffers swapped out 0 carrier transitions DCD=up DSR=up DTR=up RTS=up CTS=up	

Activity 2:

What do the abbreviations **MTU**, **BW**, and **DLY** mean?

Set or verify that all serial interfaces are “up” on physical and data link layers.

Verify the Queuing strategy. It is …….

Verify the encapsulation type. It is ……...

Activity 3:

Now if you change the interfaces bandwidth under routers interfaces, does it impact the routing table in OSPF and RIP? Apply the following change and check the path between PCs using tracert command.

!R05 enable configure terminal interface serial 0/0/1 bandwidth 200 end write	!R10 enable configure terminal interface serial 0/0/1 bandwidth 200 end write

What is the path now between the PCs? does it changes in OSPF?

```
PC>tracert 192.168.1.10

Tracing route to 192.168.1.10 over a maximum of 30 hops:

  1    1 ms      0 ms      0 ms      192.168.0.1
  2    0 ms      0 ms      0 ms      192.168.2.2
  3    4 ms      3 ms      1 ms      192.168.2.6
  4    1 ms      2 ms      1 ms      192.168.2.10
  5    2 ms      3 ms      2 ms      192.168.2.14
  6    3 ms      2 ms      11 ms     192.168.2.41
  7    10 ms     1 ms      3 ms      192.168.2.33
  8    5 ms      3 ms      5 ms      192.168.2.29
  9    4 ms      4 ms      5 ms      192.168.2.22
  10    3 ms      14 ms     7 ms      192.168.2.54
  11    14 ms     4 ms      7 ms      192.168.2.58
  12    15 ms     6 ms      11 ms     192.168.1.10

Trace complete.

PC>
```

Activity 4:

Apply the same change in the RIP routing configuration. Does the path change?? Explain why.

Task4: verifying interfaces types and checking routing update

Activity 1:

Determine the Serial cable types attached to the routers by executing the following command:

show controllers serial0/0/0

What are the cable types connected to all routers? list them in a table.

Activity 2:

Check the Ethernet interfaces on your router.

Show interfaces fastEthernet0/0.

Set or verify that the Ethernet interfaces are “up” on physical and data link layers.

Verify the Ethernet interface types. It is …….

Verify the Ethernet interface speed. It is ……..

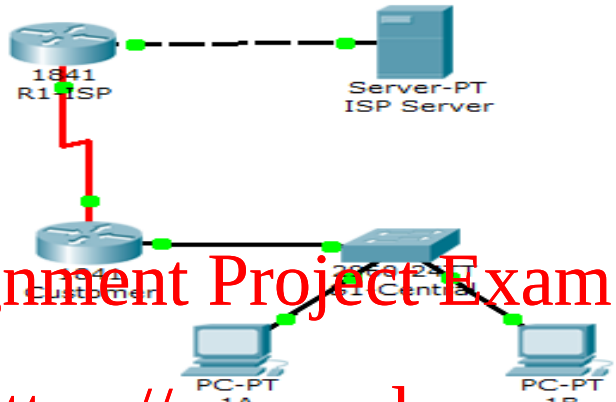
Verify the Queuing strategy. It is ……

Activity 3:
Issue the following command

show ip route ospf

What is the output from this command?

Task 5: Configuring DHCP in Router



Router Name	GE0	Serial 0 IP address	Interface Type	Loopback address
Customer	192.168.1.1/24	209.165.200.225/27	DCE	N/A
ISP	10.0.0.1/24	209.165.200.226/27	DTE	209.165.200.1/27

Activity 1: Cable and routers configuration

Based on the topology diagram, connect the PC, switch, and routers using the appropriate cabling.

1. Configure each router with the following parameters: hostname and enable secret password.
2. Configure the router interfaces with the appropriate IP address and mask. Make sure that the interfaces are in usable condition and can ping a directly connected interface or host.
3. Configure the ISP router with a loopback address to be used to test the customer router. The loopback address represents a distant network. ISP(config)#interface loopback 0 ISP(config-if)#ip address 209.165.200.1 255.255.255.224

Activity 2: Configure a default route on the customer router

1. On the customer router, configure a default route pointing toward the ISP. All packets destined for networks that are *not* in the customer routing table are forwarded to the ISP router, which has a much larger routing table and connections to other Internet providers. Notice how this default route uses the neighbor router IP address as the last number.
   Customer(config)#ip route 0.0.0.0 0.0.0.0 209.165.200.226
2. Why is a default route *not* used on the ISP? A default route on the ISP router would be a bad configuration if it pointed toward a customer site. Any routes not found in the ISP routing table would be automatically sent to the customer router. Of course, the customer router would not know what to do with the packet and would send the packet to the default route of the customer router, which is the ISP. A routing loop would occur.

Activity 3: Configure and test the DHCP pool

On the customer router, configure a DHCP pool for the internal clients.

Customer(config)#ip dhcp excluded-address 192.168.1.1
Customer(config)#ip dhcp pool INTERNAL
Customer(dhcp-config)#network 192.168.1.0 255.255.255.0
Customer(dhcp-config)#default-router 192.168.1.1

On the customer host PC, click Desktop > IP Configuration > choose DHCP option. If necessary, open a command prompt and issue the ipconfig /release and ipconfig /renew commands.

What IP address is issued to the PC? _____

What is the MAC address of the host PC? _____

From the host PC, ping the default gateway (the router Ethernet interface). Does the ping succeed? _____ Troubleshoot as necessary and do not proceed until the ping is successful.

Activity 4: Display DHCP binding on the customer router

To see the IP address and host hardware (MAC) address combination assigned by the DHCP server in the router, issue the show ip dhcp binding command on the customer router.

Customer#show ip dhcp binding
IP address Client-ID/ Lease expiration Type
Hardware address
192.168.1.2 0100.0bdb.04a5.cd May 26 2007 11:19 AM Automatic

Do the IP address and Hardware address displayed match those recorded for the host PC in Step 3? _____

Task 6: Configure NAT/PAT

Continue using the same network setup:

Activity 1: Configure NAT

1. On the customer router, use the access-list command to identify the addresses that need to be translated. The network number is stated, but instead of a normal mask that usually comes next, a wildcard mask is used (0.0.0.255).
   Customer(config)#access-list 1 permit 192.168.1.0 0.0.0.255
2. On the customer router, define where NAT looks for the IP addresses it needs to translate (source list 1 refers to access list 1 that you just created). Also define which interface IP address to use as the real address for each packet that comes through the GE interface destined for the Serial interface. The overload parameter at the end of the command shown below means that the serial port IP address is used and that port numbers are used to track the packets. Approximately 4,000 addresses can realistically be translated using this method, even though it is technically possible to translate even more.
   Customer(config)#ip nat inside source list 1 interface serial 0/0/0 overload
3. Apply NAT to the inside and outside interfaces.

Customer(config)#interface serial 0/0

Customer(config-if)#ip nat outside

Customer(config-if)#exit

Customer(config)#interface fast 0/0

Customer(config)#ip nat inside

Customer(config)#end

Activity 2: Test NAT/PAT

1. From the host PC command prompt, ping the ISP router loopback address.

ping 209.165.200.1

2. Was the ping successful? If not, perform appropriate troubleshooting.
3. On the customer router, issue this command to verify the NAT translation:

Customer#show ip nat translation

Pro Inside global Inside local Outside local Outside global

icmp 209.165.200.225:512 192.168.1.2:512 209.165.200.1:512 209.165.200.1:512

List the following IP addresses: What is the inside global IP address shown?

_____

What is the inside local IP address shown?

_____

What is the outside local IP address shown?

_____

What is the outside global IP address shown?

Task7: Read OMNeT++ tutorial part as preparation for experiment 04

OMNeT++ Simulator tutorial

Introduction

OMNeT++ is an extensible C++ component-based simulation framework primarily for building network simulators that includes wired and wireless communication networks [3]. OMNeT++ offers an Eclipse-based IDE and GUI runtime environment. OMNeT++ is distributed under the Academic Public License [6]. OMNeT model consists of modules that communicate through message passing. Simple modules are build using simulation class library. Simple modules are grouped into compound modules as shown in Figure 22. Boxes represent simple (grey background) and compound modules, arrows represent connections between modules using their gates (ports).

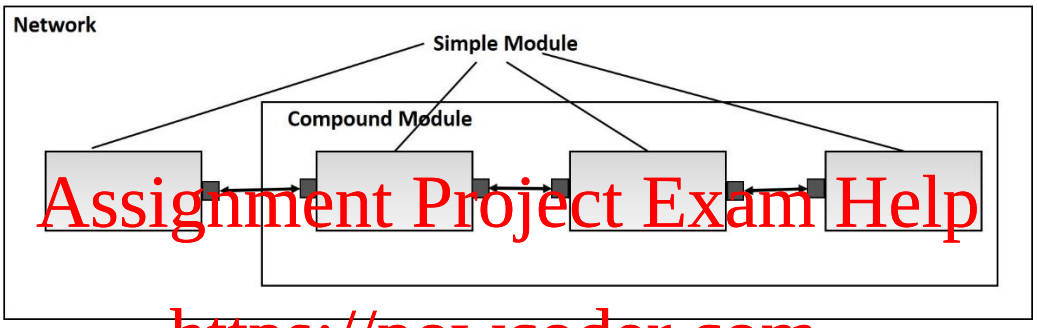


Figure 22. Simple and Compound Modules

Task 01: OMNeT++ Installation

OMNeT++ is an opensource simulation that can be installed on different Operating Systems (OSs) and platforms such as Windows 10, Linux and Mac OS. This manual guides you to install OMNeT++ at Windows 10 OS, using the following steps:

Download OMNeT++ 5.6.2 from the following url: https://omnetpp.org/download/ according to your OS as shown in Figure 23. Note that we download the version for Windows 10. Once the download is completed, you have to extract the zip downloaded file.

- It is recommended to read the instruction guide file named **InstallGuide.pdf** that is located the **doc** folder (shown in figure 24), which is located inside **omnetpp-5.6.2** folder where you extracted the downloaded file of OMNeT++.
- In windows installation there is a bug during the building process, so to fix this bug open file **configure.user** (shown in figure 24) in the **omnetpp-5.6.2** folder, and change PREFER_CLANG=yes to PREFER_CLANG=no and save the change.

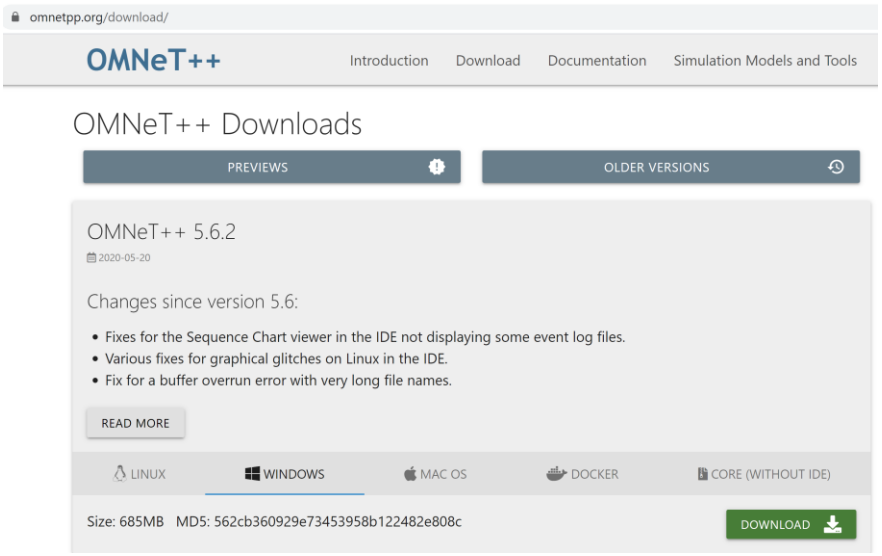


Figure 23. Download OMNeT++

- Run the installation by double clicking the execution file named **mingwenv** (shown in figure 24) in the **omnetpp-5.6.2** folder. Note that the installation process may take time to be done.

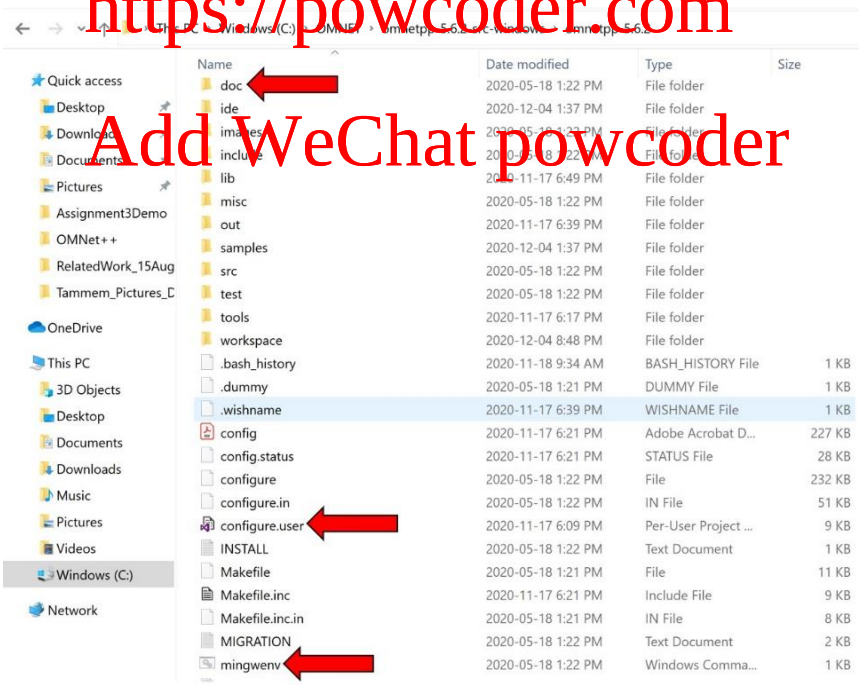


Figure 24. Installing OMNeT++

- Once the installation is completed, a prompt message asks you to type "./configure" and then "make" to build the simulation libraries will be shown. Follow that, type *./configure* and press enter.

- Once the configuration is done, type *make* and press enter. Note that the make process may take time.
- Once the make process is completed, a prompt message tells you that Now you can type “omnetpp” to start the IDE. So, type omnetpp and press enter to start the IDE. Note that you can create a shortcut for the omnetpp on your Desktop. To do so, go to the folder **ide** inside **omnetpp-5.6.2** folder, right click on the file omnetpp.exe choice send to Desktop (see Figure 25). You can click on the shortcut icon where you created it to launch the OMNeT++ IDE.

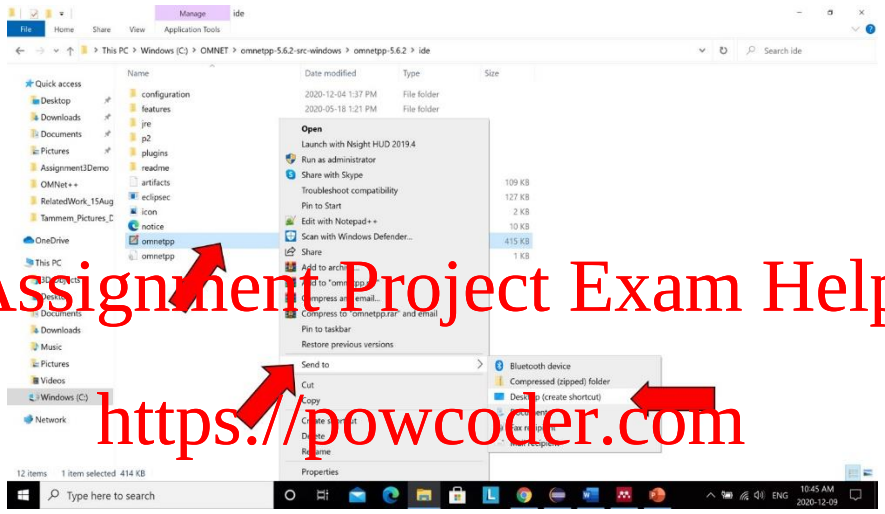


Figure 25. omnetpp shortcut

- Once the IDE is launched, you will get the welcome screen as shown in Figure 26.

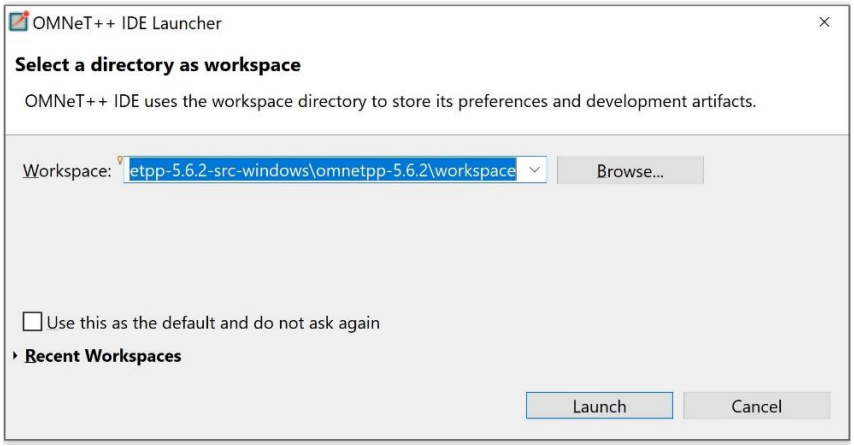


Figure 26. OMNeT++ Workspace

- OMNeT++ IDE uses a workspace directory to save your created projects (see Figure 26).

- Once IDE is launched, you will get a welcome screen as shown in Figure 27.

Figure 27. Welcome Screen of OMNeT++ IDE

- Close the welcome screen with two check boxes, checked by default, that ask you to Install INET Framework and Import OMNeT++ programming examples. We recommend postponing these steps for now by unchecking the two check boxes as shown in Figure 28. We will introduce and install INET Framework later in this manual.

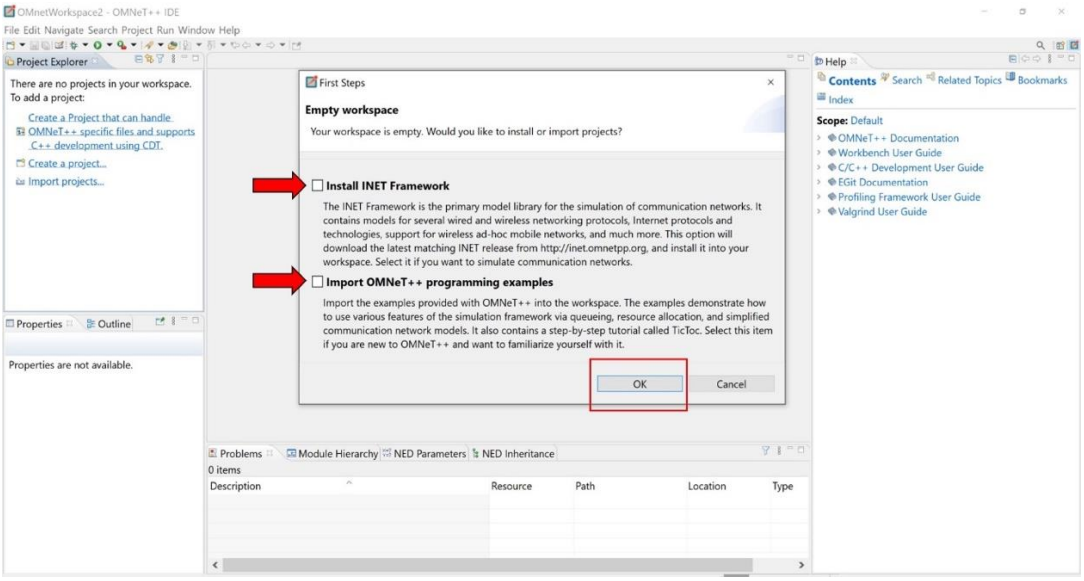
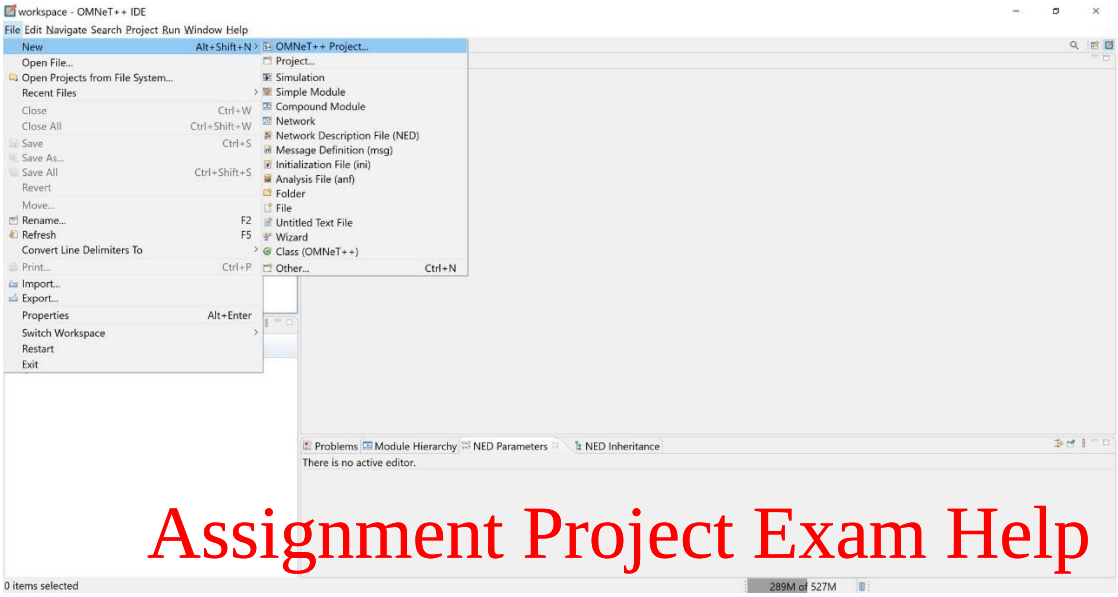


Figure 28. Empty Workspace

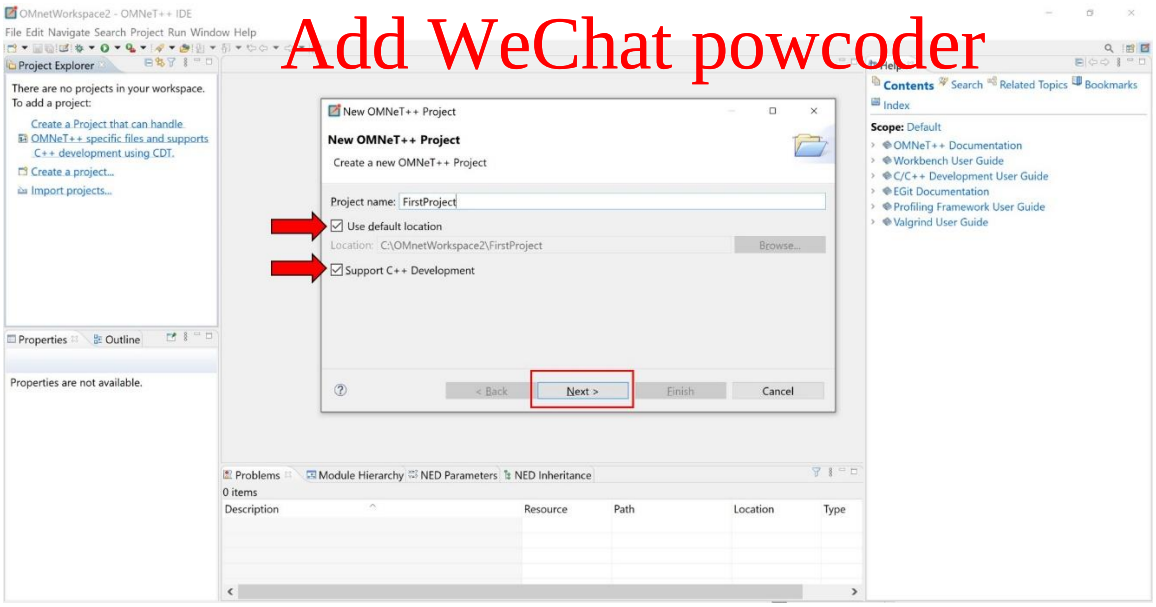
Task 02: Creating New Project

To create a new OMNeT++ project, do the following steps:

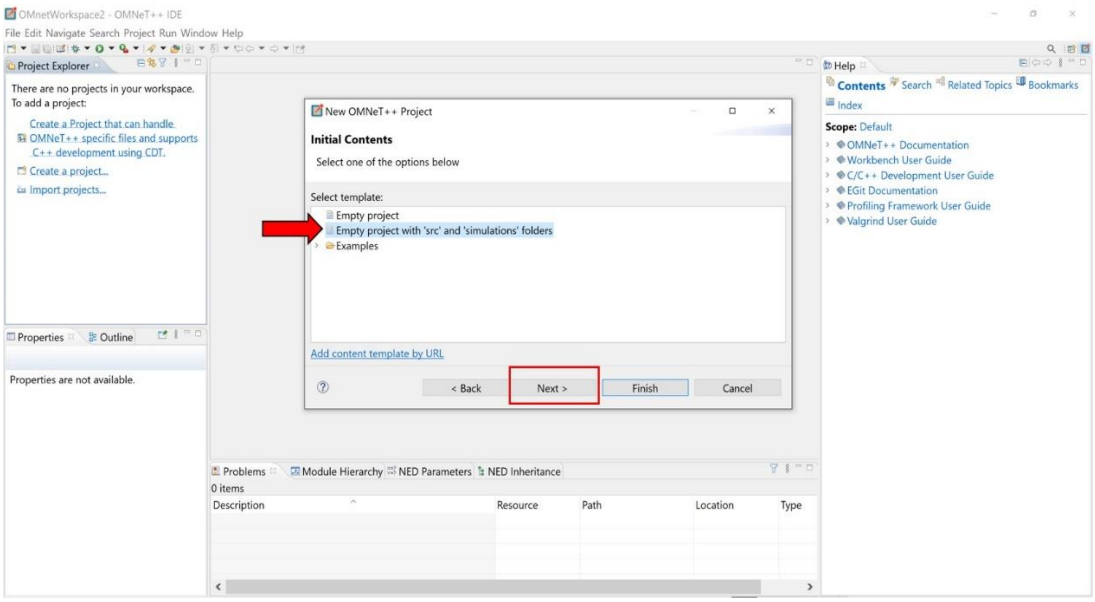
- Click on File menu of IDE and select New and then select OMNeT++ project.



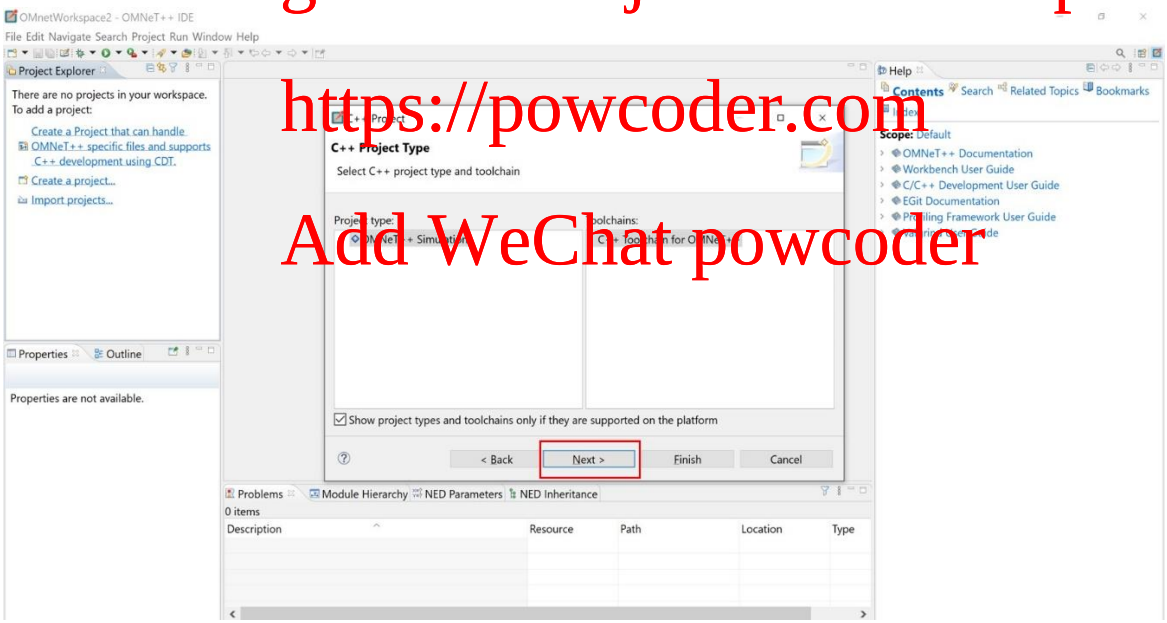
- A new project window will appear, type the name of your project. Let us name the first project as 'FirstProject'. Keep the two check boxes 'use default location' and 'Support C++ Development' checked and then click Next.



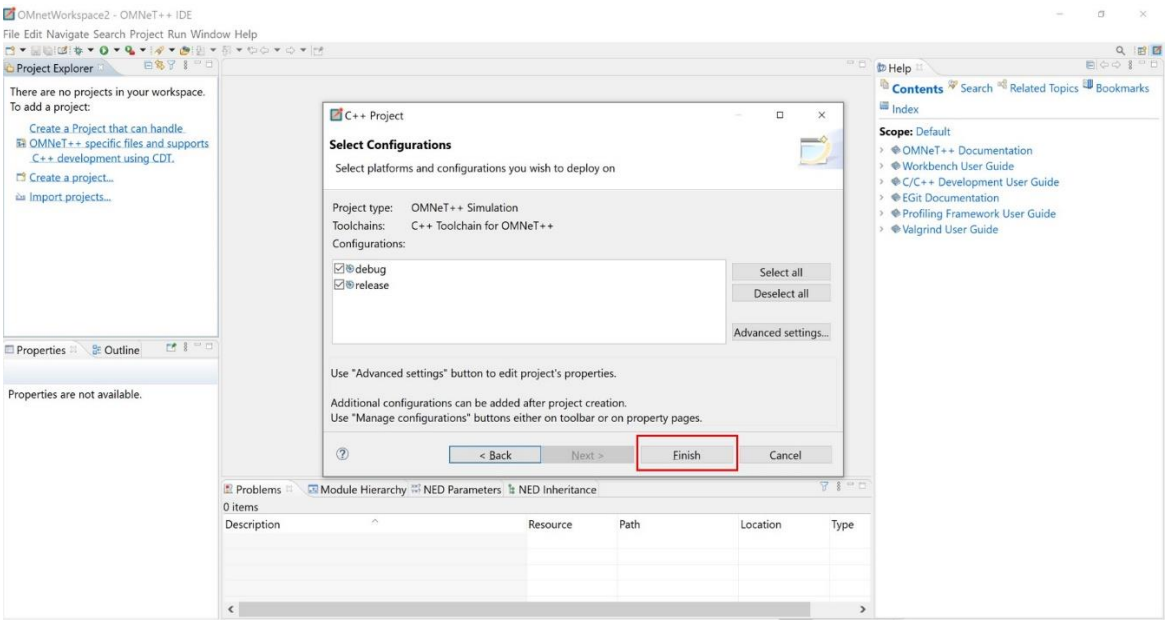
- The Initial Contents window will appear. In the Select template box, select the second option which 'Empty project with 'src' and 'simulations' folders and then click Next.



- C++ Project Type window will appear, do not change anything, and click Next.



- Select Configurations window will appear, do not change anything, and click Finish.



- The project will be created, have a look to the Project Explorer tab. Double click on the package.ned file and just have a look to the IDE window and options. We will learn more about NED language later on this manual.

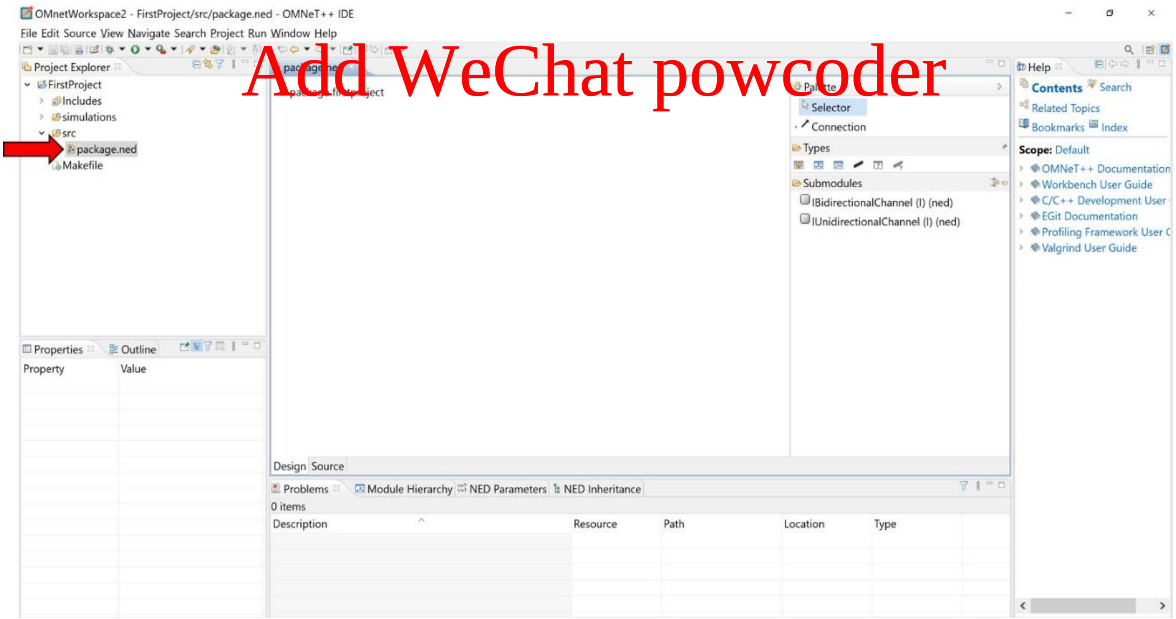


Figure 29. Steps for creating new project

Task 03: OMNeT++ files structures

Learning the main components to build networking topology using OMNeT++. For any project in OMNet++ you need following files:

1) Network Description (NED) topology descriptor. It can be edited either graphically or in text mode, and the user can switch between the two modes at any time, using the tabs at the bottom of the editor window.
2) Message definitions .msg files that define messages types (for special messages design)
3) INI file, to configure simulation parameters for execution.
4) Parameters input can be assigned in NED file or configuration files INI files. It can be one of following types: 1) String, 2) Numeric, 3) Boolean, and 4) XML data tree file.

Please read more about that and summary what you understand about them with example.

Task 04: INET Framework

OMNeT++ new projects are based on existing libraries that provide basic network functionalities. Voice and multimedia network applications are implemented in the INET framework [7]. INET is an open-source framework that can be used in OMNeT++ simulation. It provides models, protocols, agents for students and researchers to build and test communication networks.

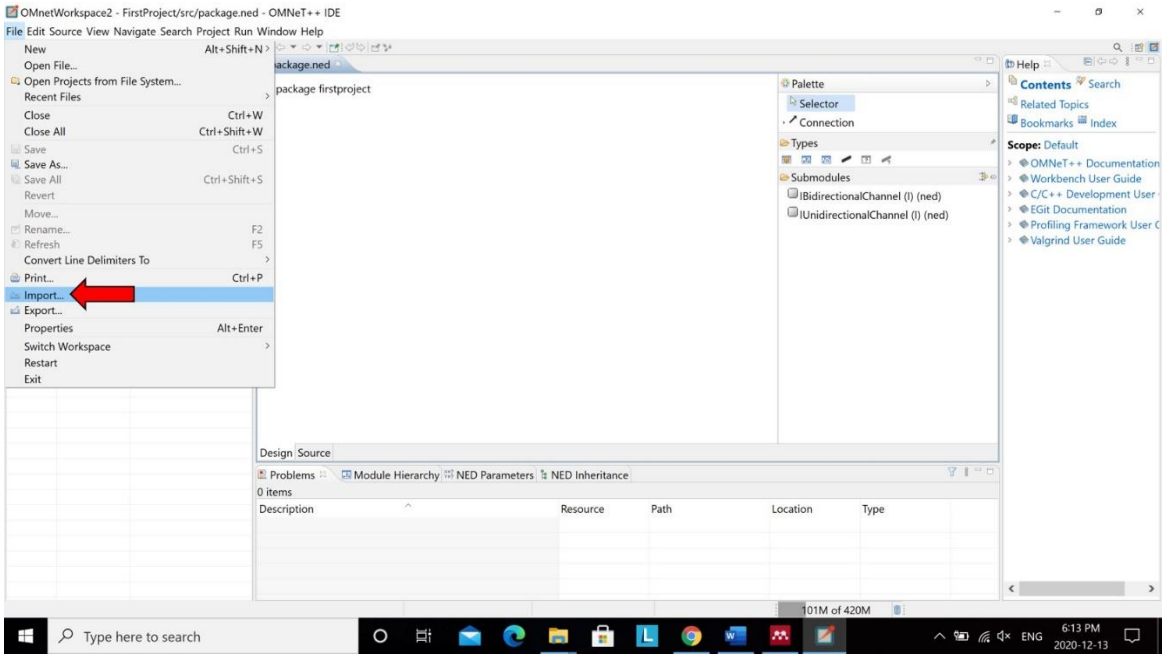
Activity 01: Add INET to OMNeT++

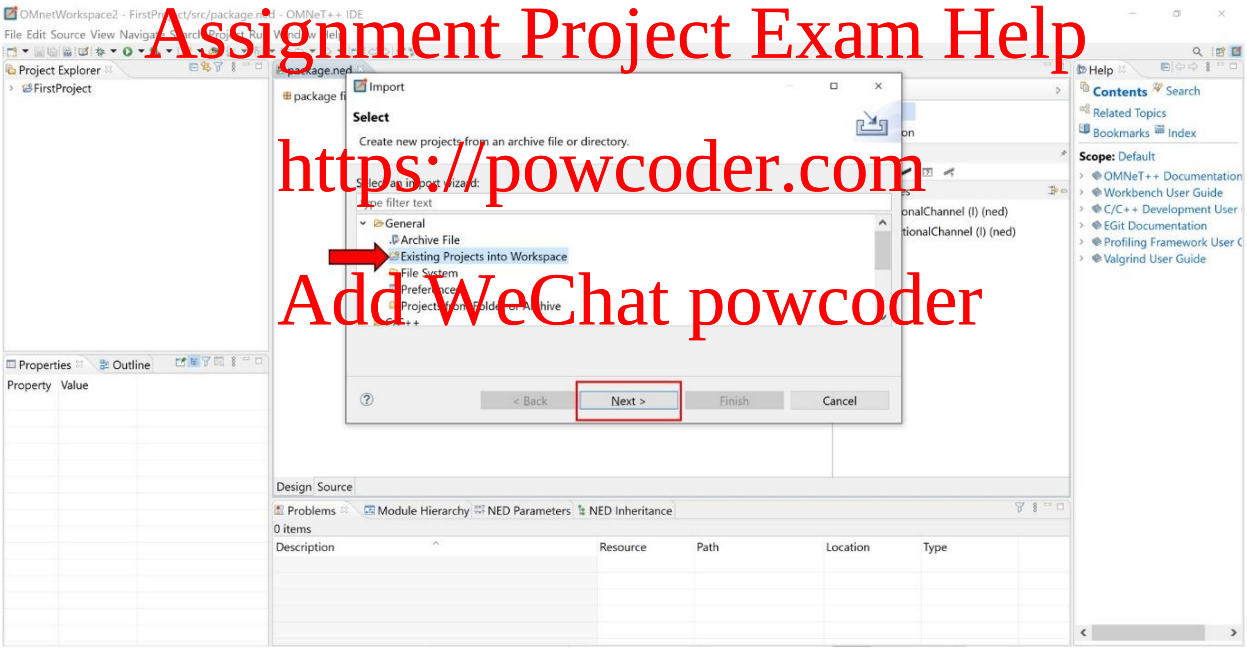
To add INET to OMNet++ do the following steps: (figure 30)

- Download INET framework from the following URL, try to download a stable version for OMNeT++ 5.4.1 and later.

https://inet.omnetpp.org/Download.html

- Unzip the downloaded INTE zip file.
- Start the OMNeT++ IDE, go to file and select import, import wizard window will appear, expand General and select Existing as in listed Figures below.
- Directory selection window will appear, select the option Select root directory and then click on Browse button, and select the INET directory you get after unzipping it. Finally click Finish button.
- Look at the Project explorer tab, you will find inet project. Try to expand and explore it.





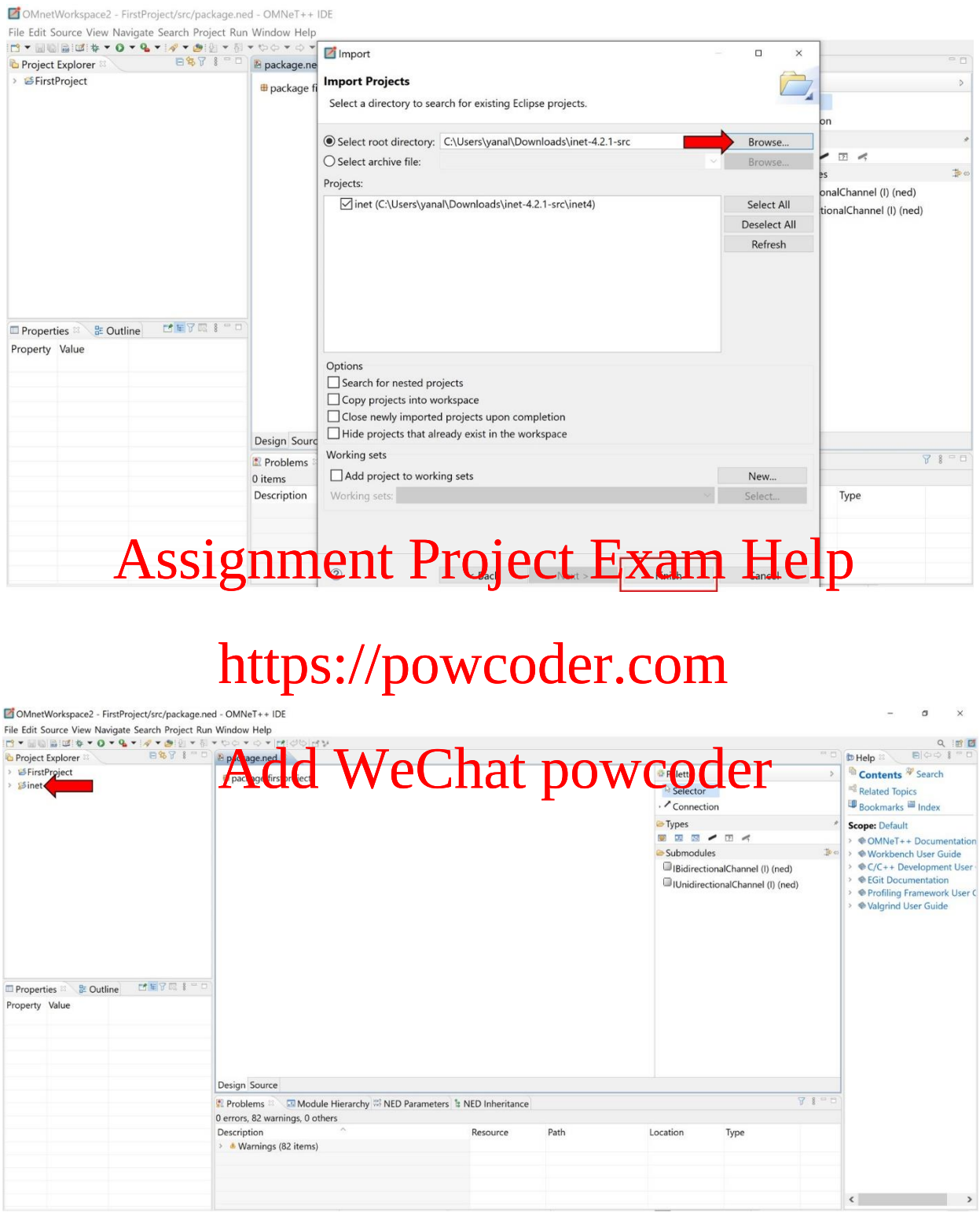
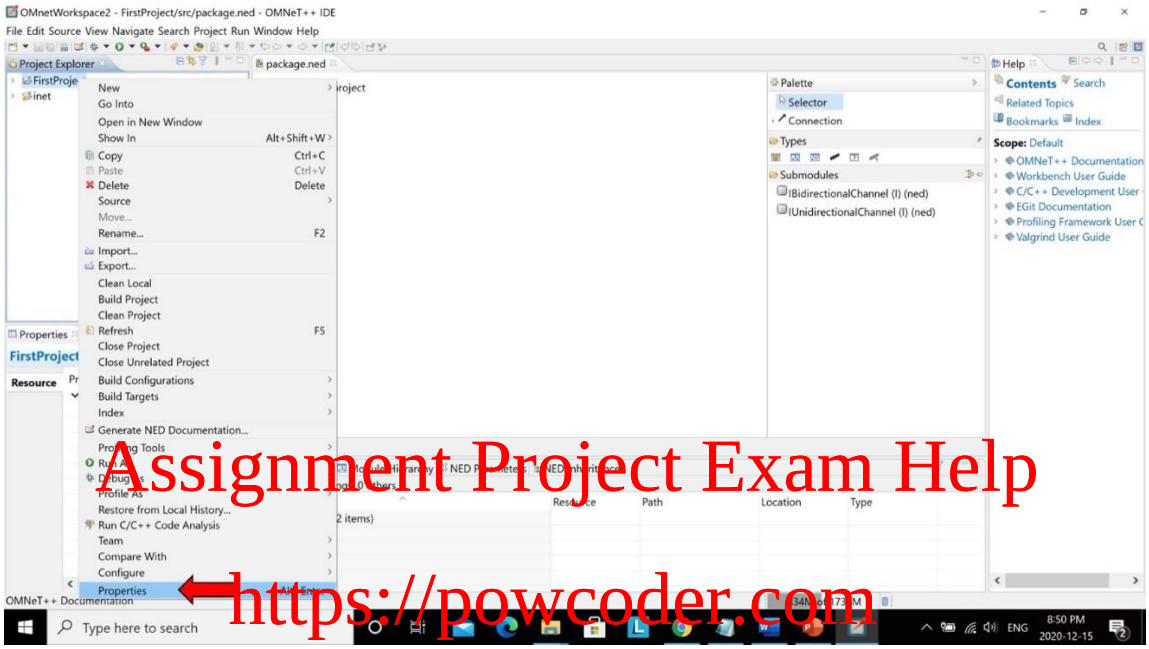


Figure 30. Adding INET to OMNeT++

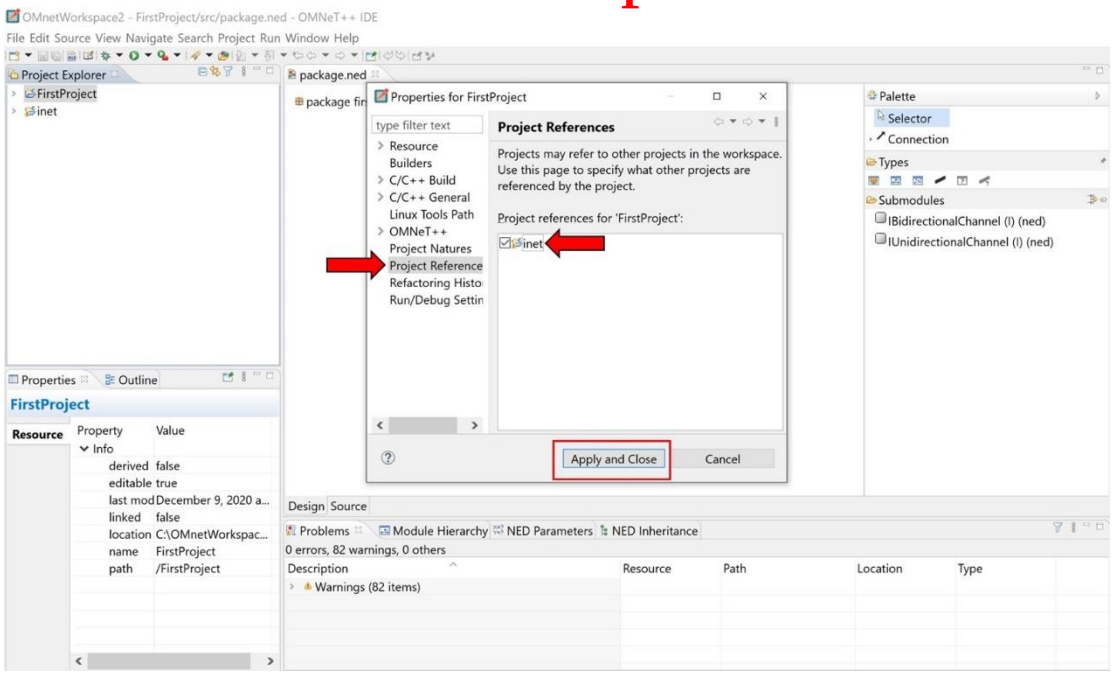
Activity 02: Reference INET to existing project

To reference INET to an existing project do the following steps:

- Right click on the project that you want to reference INET for it and then select properties.



- Properties window will show up, select OMNET++ and then select Project Reference, check the box inet and click on Apply and close button.



- Have a look and explore the Submodules and Connections that have been added as a result of referencing INET.

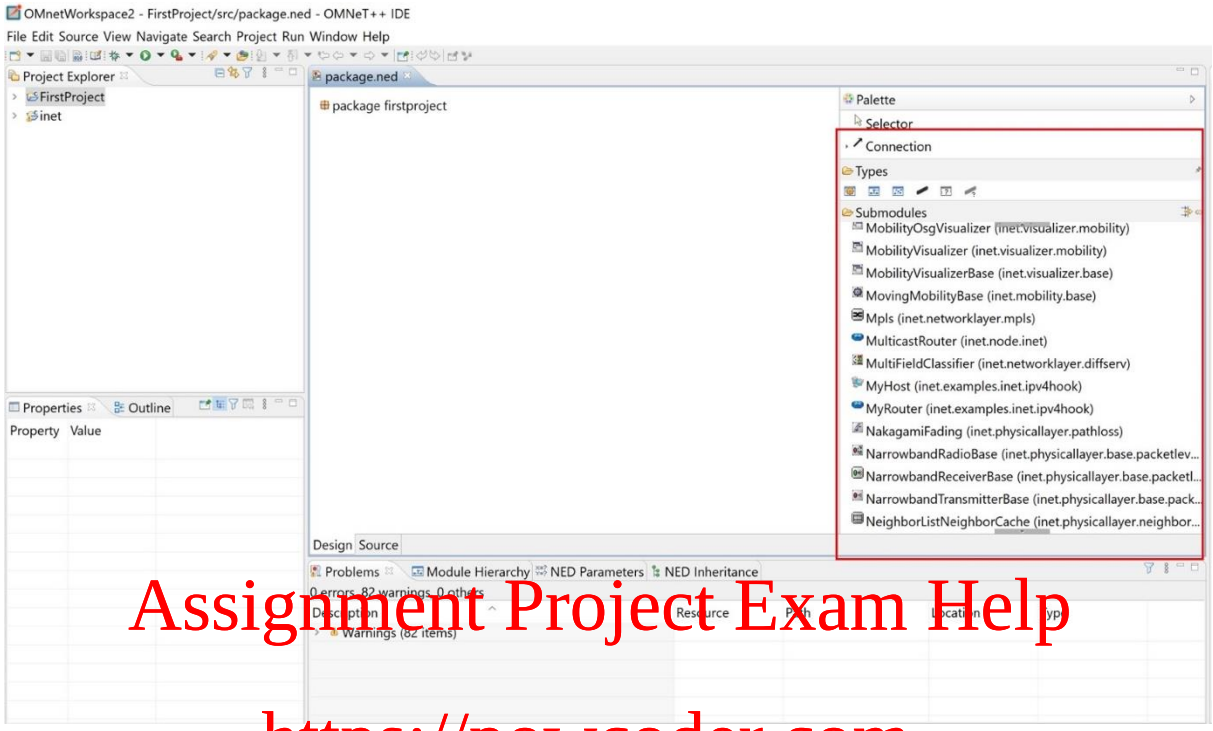
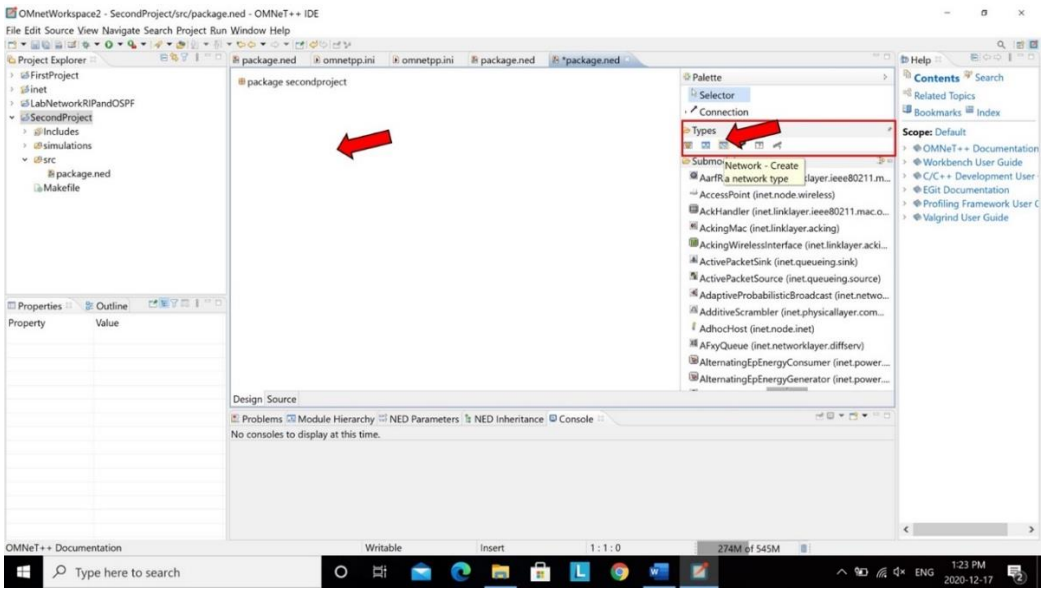


Figure 31. Reference INET

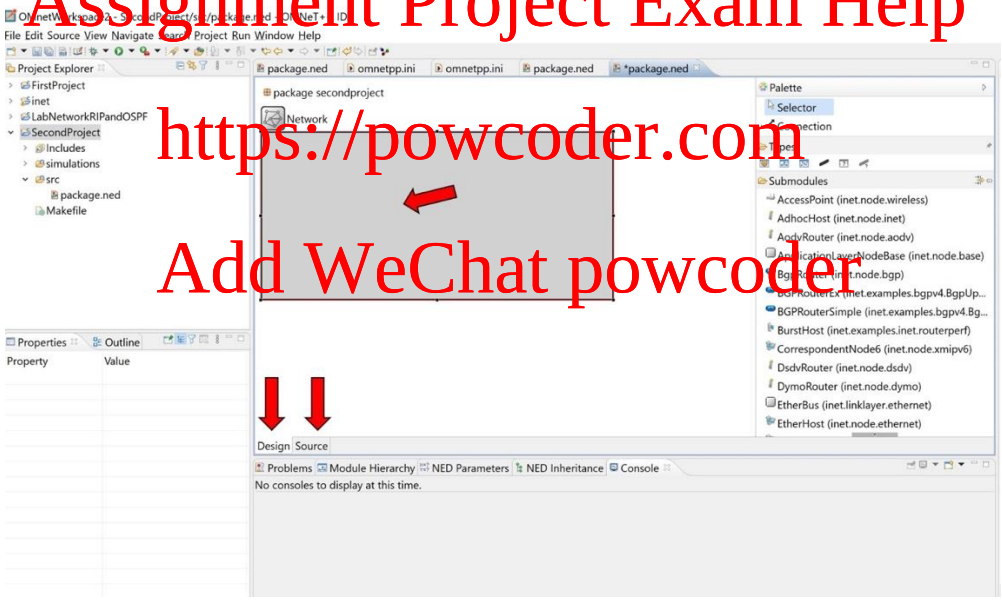
Task 05: UDP simple application example

To create a simple network where sender communicate with the receiver using intermediate router using simple UDP application. To do this project, do the following steps:

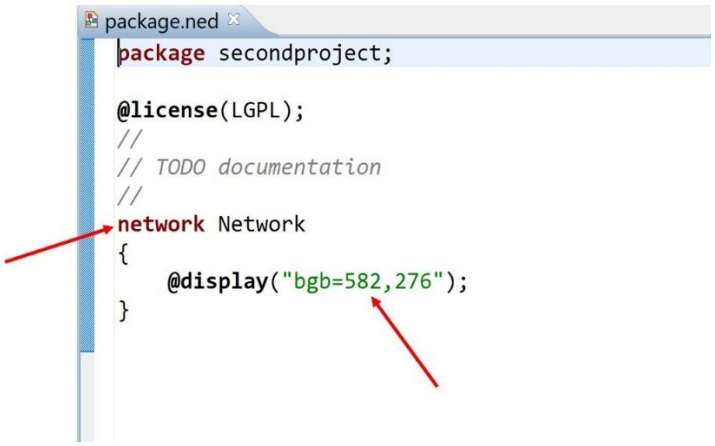
- Create a new project call it ‘SecondProject’
- Add INET reference to SecondProject
- Go to Types menu list figures below, and click on **Network** icon, notice that the cursor shape will be changed to arrow and plus sign



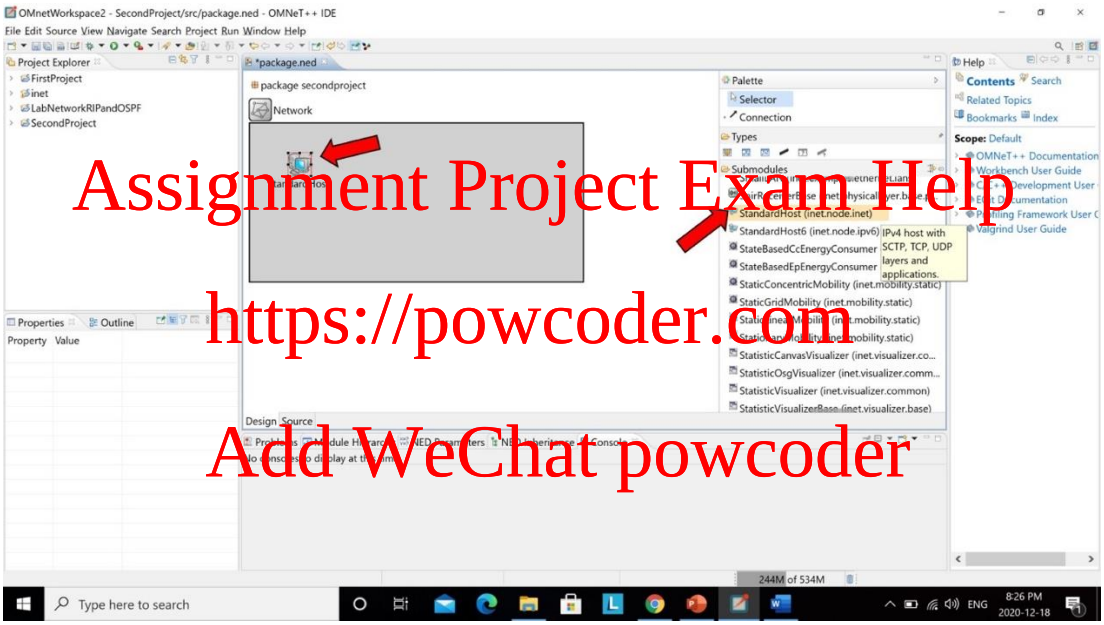
- Go to design area and click there, you will see a grey box which represents the network element that you selected.



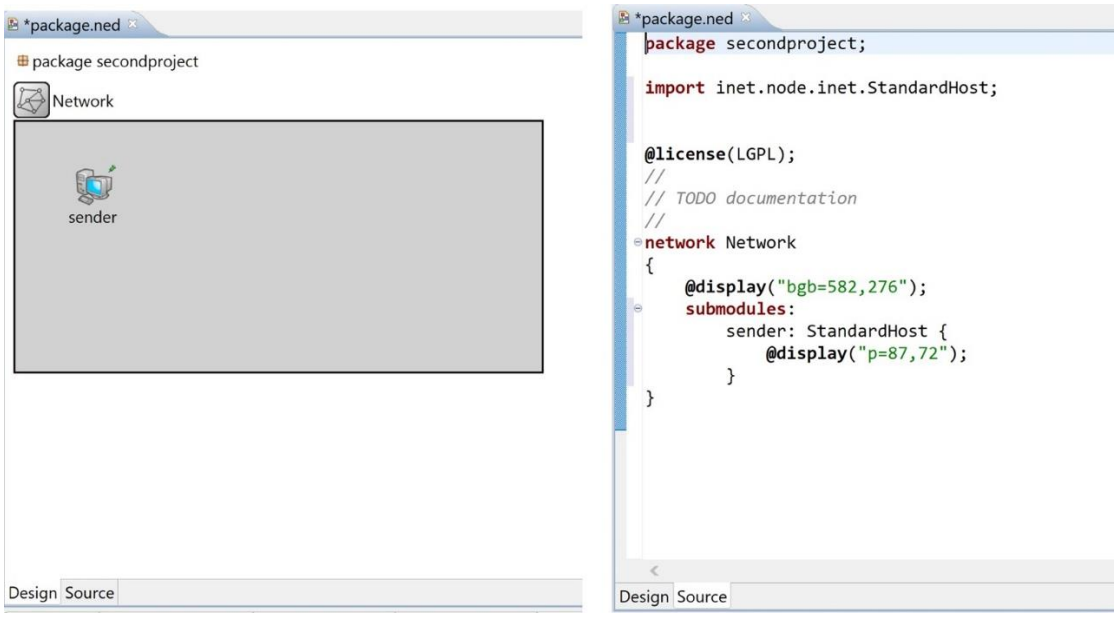
- Open the source tab. You will see NED code.
    - So far, we create a network instance called ‘network’ from the class Network.
    - @display is a tag command that refers to the position of the network element. Note that if you change the position of the network, the values of the display tag will be changed accordingly.



- Add **standardHost** instance to network.

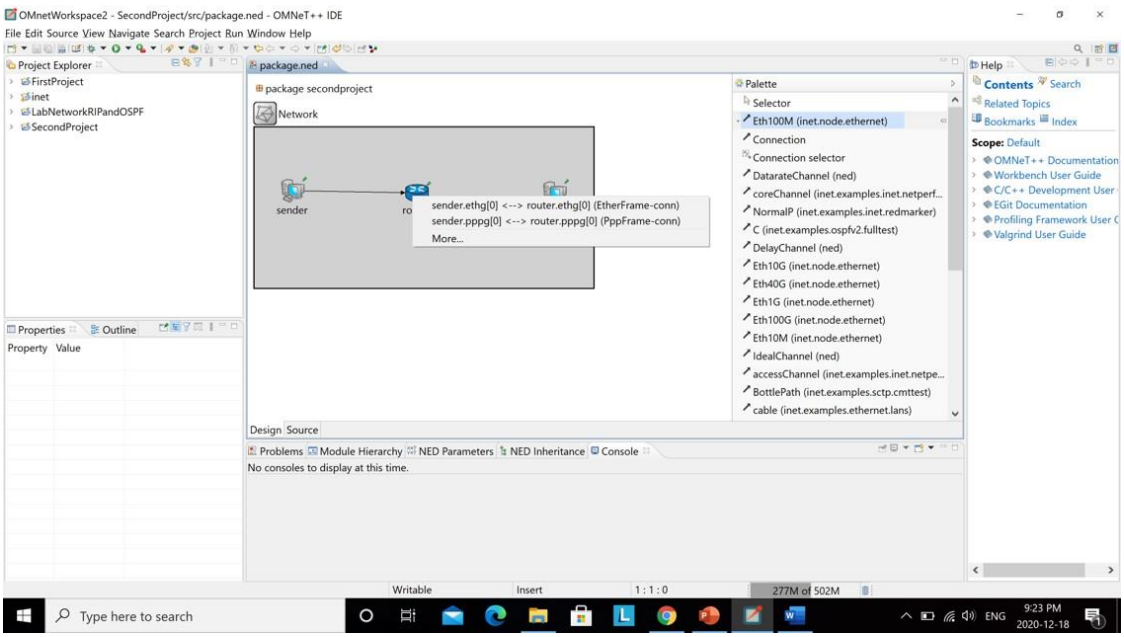


- Rename **standardHost** to sender. You can do that by double click on the label of the standard host and write the new name or do it from the ned source code.

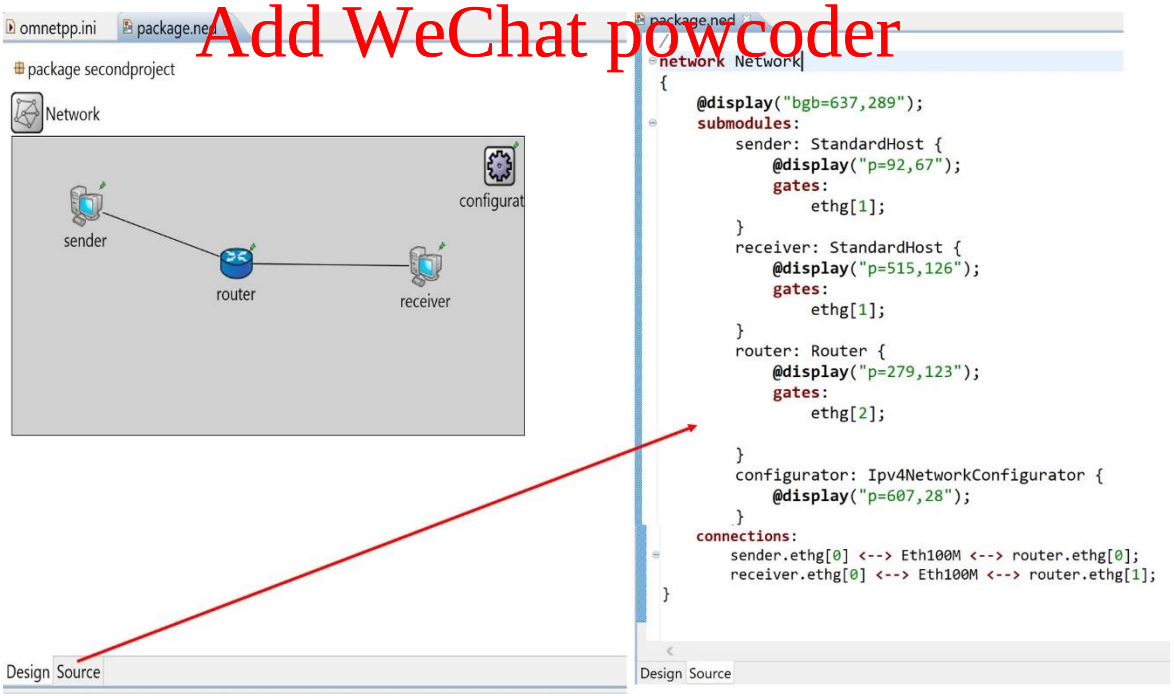


- Add another **standardHost** and name receiver.
- Add **Router** instance and name its router. Note that all the instance elements under the submodules of the network class.

- Connect sender and receiver with the router. To do that, expand the Connection and select **Eth100M** instance, and then click on the sender and then click on the router. You will get different connection options, select the option *sender.ethg[0]< -- > router.ethg[0] (EtherFrame-conn)*. Do the same thing for the receiver. have a look to the ned source code. Note that the connection instance is under the Connections category of the network class.

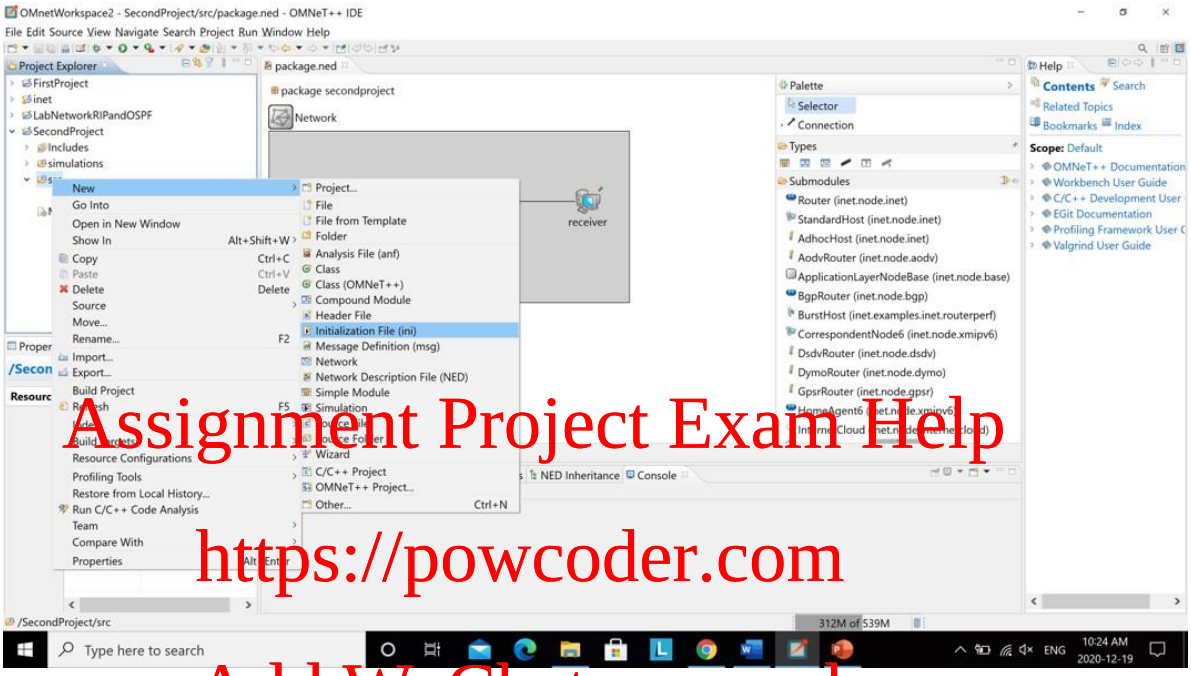


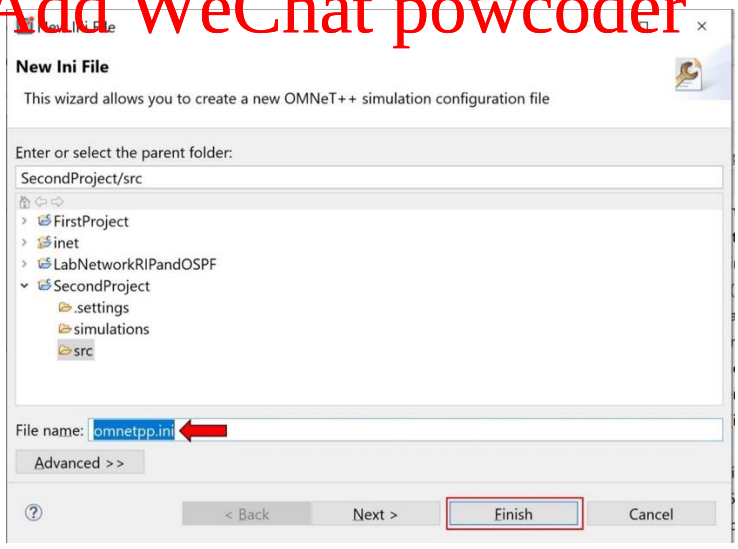
- We need to connect sender and receiver with the router on specific interfaces. So, you have to create one interface **ethg[1]** for the sender and receiver, and two interfaces **ethg[2]** for the router. Note that you create a vector of ethg with specific size. To access a specific interface, use the index starting from zero. For example, the first interface of the router is referred to by ethg[0] and the second interface is referred to by ethg[1]. Connect the sender ethg[0] with the first ethg[0] of the router, and connect the receiver ethg[0] with the second ethg[1] of the router.

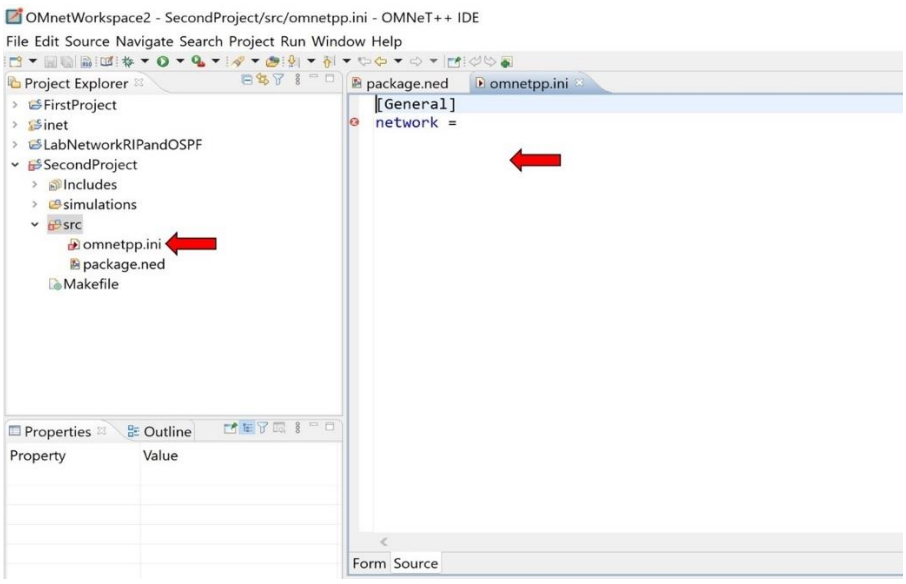


- With the above steps, you completed only the design of the network topology.

- Now it is time to implement the functionality of the network. We use the initialization file **omnetpp.ini** file to write functionality code of the network.
- To create omnet.ini file, right click on the src folder under your project folder ‘SecondProject’ and select New, and then Initialization File (ini). A New ini File will appear, write **omnetpp.ini** in the File name box, finally click on Finish button. Note that a new file named omnetpp.ini will be created in src folder.







- Write the code shown in the figure below inside omnetpp.ini.

omnetpp.ini

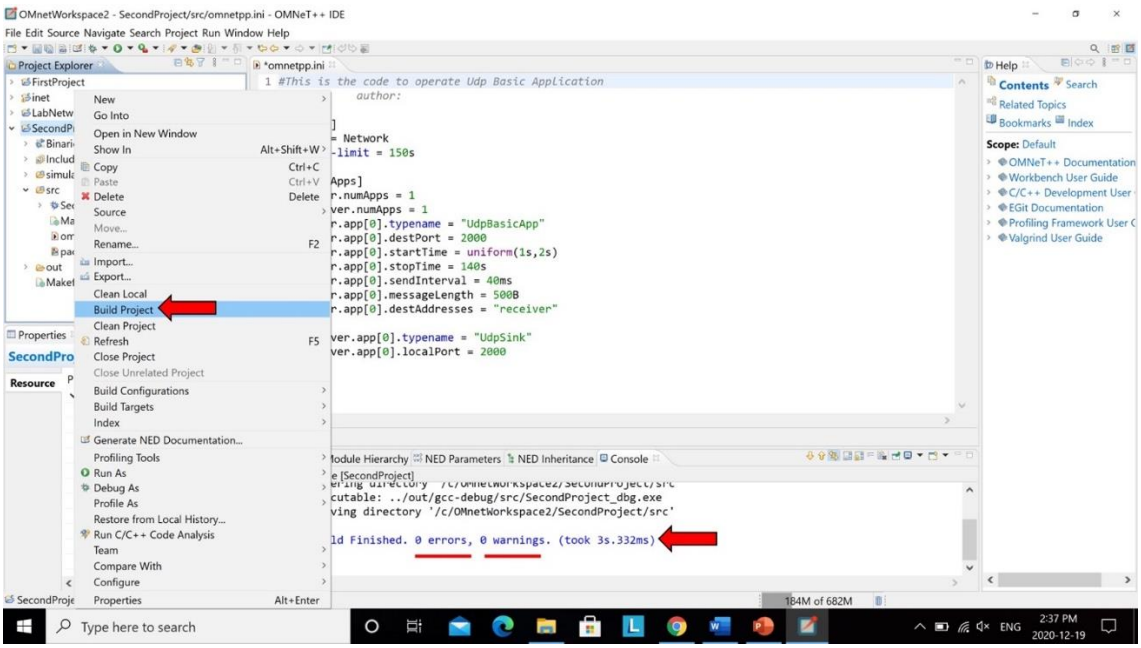
```
1  #This is the code to operate Udp Basic Application
2  #date:      author:
3
4  [General]
5  network = Network
6  sim-time-limit = 150s
7
8  [Config Apps]
9  **.sender.numApps = 1
10 **.receiver.numApps = 1
11 **.sender.app[0].typename = "UdpBasicApp"
12 **.sender.app[0].destPort = 2000
13 **.sender.app[0].startTime = uniform(1s,2s)
14 **.sender.app[0].stopTime = 140s
15 **.sender.app[0].sendInterval = 40ms
16 **.sender.app[0].messageLength = 500B
17 **.sender.app[0].destAddresses = "receiver"
18
19 **.receiver.app[0].typename = "UdpSink"
20 **.receiver.app[0].localPort = 2000
21
22
23
```

Form Source

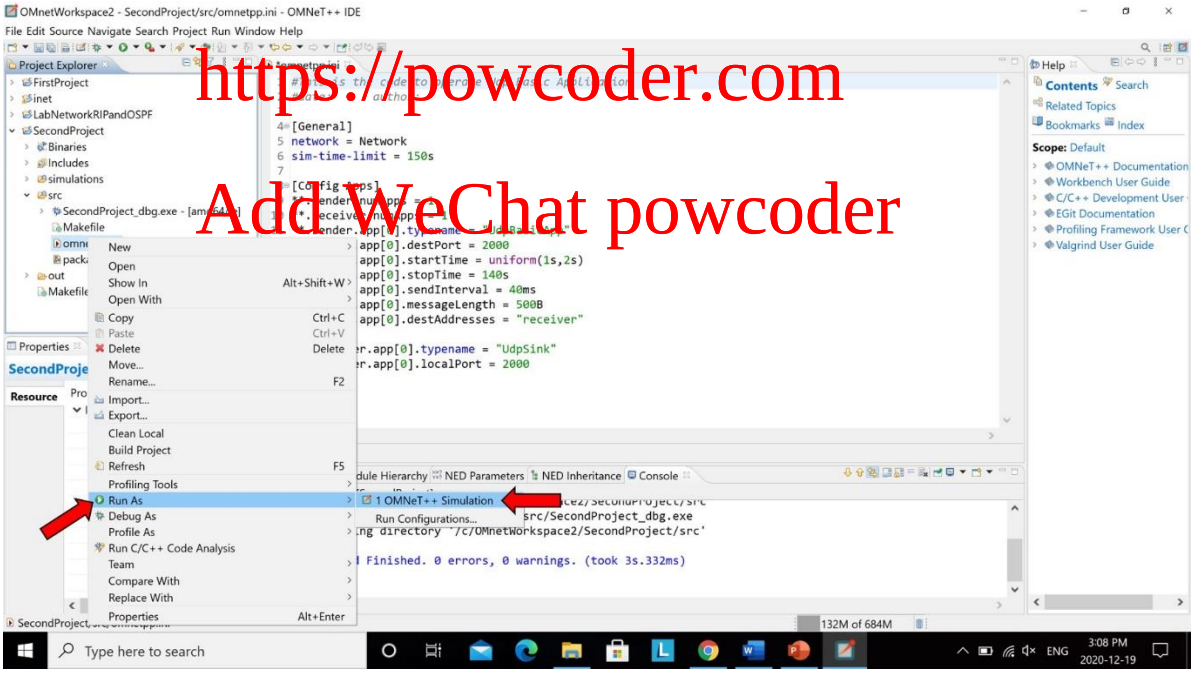
- Below is the description of the code
    - Lines 1 and 2 are comments, you can comment a line of code using the operator #
    - lines 4 and 5 is the General configuration information of the network instance

- line 6 is the period of the simulation time
- lines from 8 up to 20 present the configuration of applications on the sender and receiver
- line 9 refers to the number of applications to run on the sender. In this activity we run one application on the sender. Note that two ** before the sender means the current package (project).
- Line 10 refers to the number of applications to run on the receiver. In this activity we run one application at the receiver.
- Line 11 refers to the application type that will run on the sender. In this activity the type is "**UdpBasicApp**". Note that the applications that run on the sender or reciver are stored in **app** vector. Therefore, to access a specific application use the index of the application starting from zero index (**app[0]**).
- Line 12 refers to the port number to run application on the sender.
- Line 13 refers to the start time of the application.
- Line 14 refers to the stop time of the application.
- Line 15 refers to the send interval of the application message (packet).
- Line 16 refers to the message size of the application that will be sent.
- Line 17 refers to the destination address of the messages from the sender, and the destination is the receiver.
- Line 19 refers to the application type that will run on the receiver. The application type is 'UdpSink'.
- Line 20 refers to the port number to run the application 'UdpSink' at the receiver.

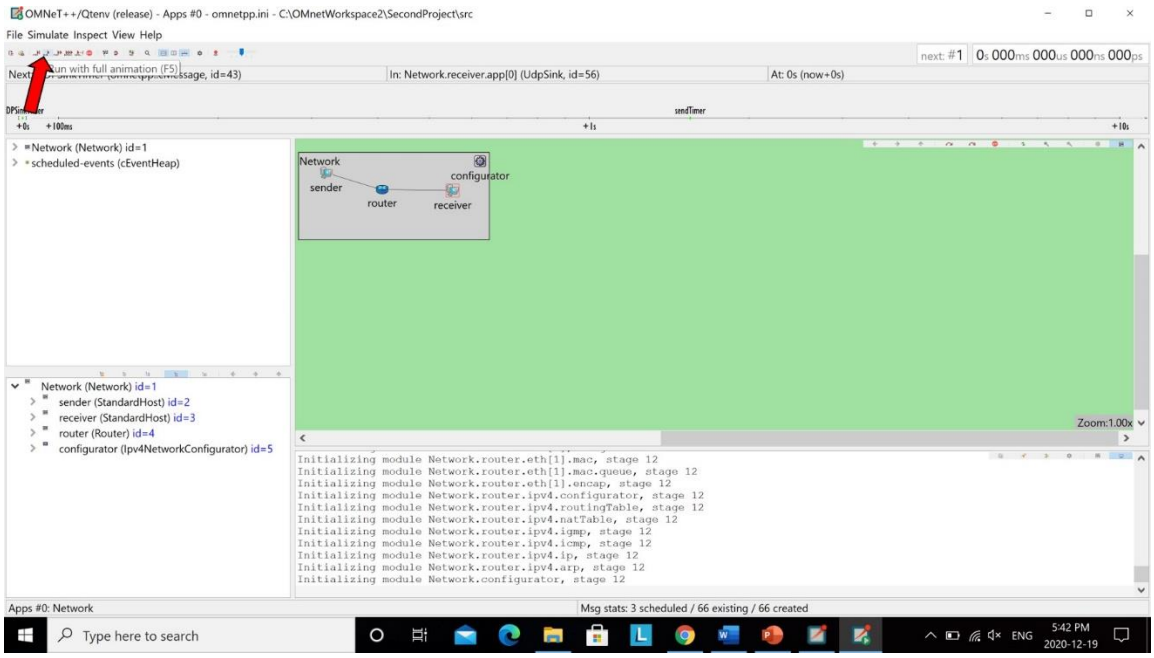
- Now it is time to run your project. First you have to build the project to make sure it free from errors. To build the project, right click on the project and select Build Project. Note that the build process may take long time.



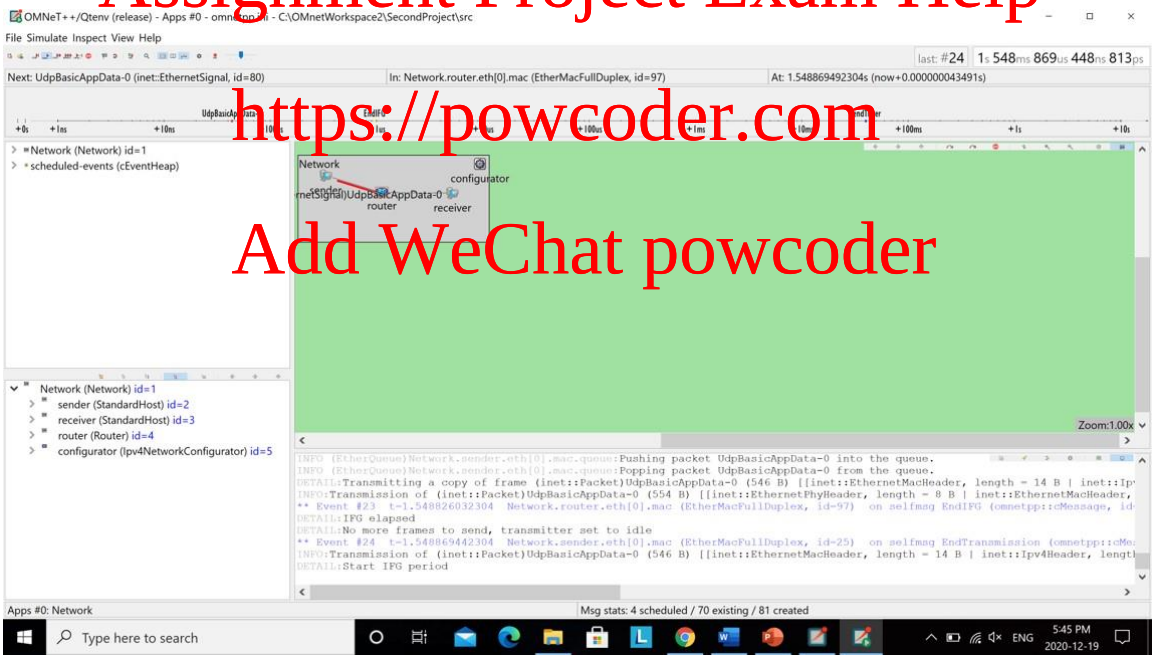
- If you get 0 errors, you are ready to run your project. Right click on the **omnet.ini** file and select Run as OMNeT++ simulation.



- The simulation run window will apeear, click on the run icon. See the behavior of the simulation, network, elements and applications. Do not forget to glance the logs.



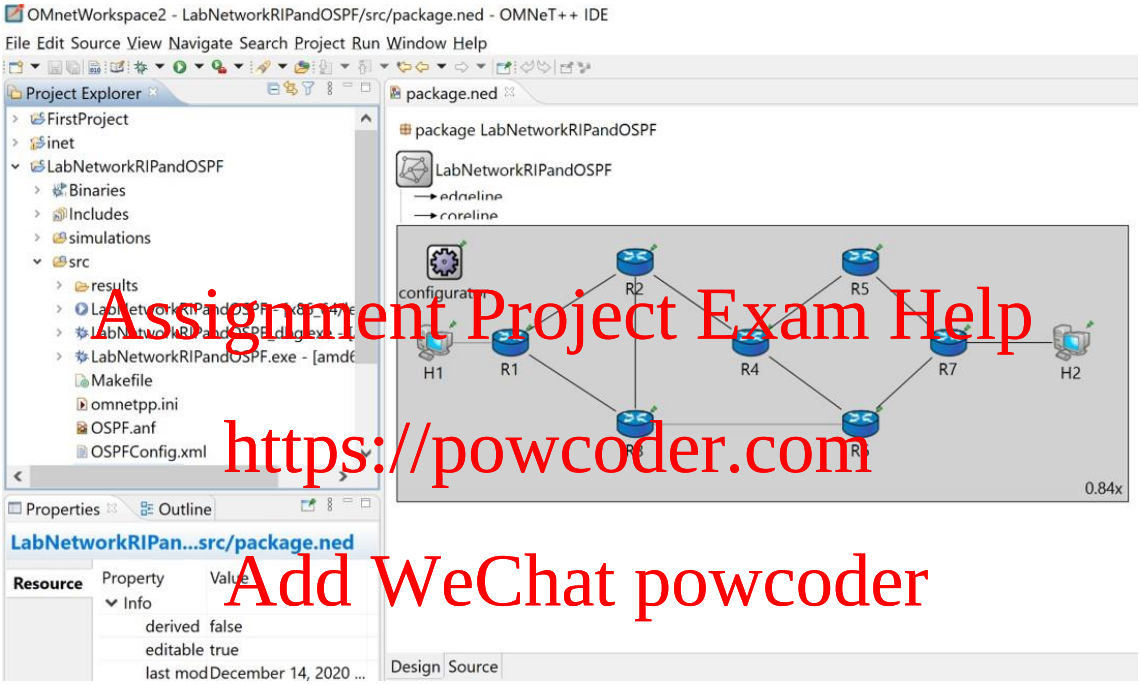
- You can stop running the simulation by clicking on the stop icon.



Task 06: Routing protocols and queuing quality

In this part you will learn more about connections between routers as well as enabling the routing protocols such as RIP and OSPF. Moreover, you will learn how to enable queuing at the routers and know how to measure the quality of the networking applications. To do this activity, do the following steps:

- Create a new project named ‘LabNetworkRIPandOSPF’
- Build the network topology as shown in the figure below. Pay attention to the NED code in **package.ned** file.



- Lines 12-14 defines parameters **edgeDatarate** and **coreDatarate** to define the properties of the edge and core lines that are defined as channels (lines 17-30) for the communication between hosts and routers.
- Lines 18-23 define a communication channel called ‘**edgeline’** with the following properties
    - Delay of 2 ms
    - Datarate equal to edgeDatarate. The value of the edgeDatarate will be assigned in the initilaization file (omnet.ini)
    - thruputDisplayFormat = "b B U"

package.ned

```
package LabNetworkRIPandOSPF;

import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.common.misc.ThruputMeteringChannel;
import inet.node.ethernet.Eth100M;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;

@license(LGPL);
network LabNetworkRIPandOSPF
{
    parameters:
        double edgeDatarate @unit(bps);
        double coreDatarate @unit(bps);
        @display("bgb=953,360");

    types:
        channel edgeline extends ThruputMeteringChannel
        {
            delay = 2ms;
            datarate = edgeDatarate;
            thruputDisplayFormat = "b B U";
        }

        channel coreline extends ThruputMeteringChannel
        {
            delay = 2ms;
            datarate = coreDatarate;
            thruputDisplayFormat = "b B U";
        }

    submodules:
        R1: Router {
            @display("p=148,153");
            gates:
                ethg[1];
        }
        R2: Router {
            @display("p=311,47");
        }
        R3: Router {
            @display("p=311,259");
        }
        R4: Router {
            @display("p=461,153");
        }
        R5: Router {
            @display("p=605,47");
        }
        R6: Router {
            @display("p=605,259");
        }
        R7: Router {
            @display("p=719,153");
            gates:
                ethg[1];
        }
        H1: StandardHost {
            @display("p=49,153");
            gates:
                ethg[1];
        }
        H2: StandardHost {
            @display("p=879.2438,151.92375");
            gates:
                ethg[1];
        }
        configurator: Ipv4NetworkConfigurator {
            @display("p=61.00875,46.65375");
        }

    connections:
        //Between hosts and routers
        H1.ethg[0] <--> Eth100M <--> R1.ethg[0];
        H2.ethg[0] <--> Eth100M <--> R7.ethg[0];

        //Between routers
        R1.pppg++ <--> edgeline <--> R2.pppg++;
        R1.pppg++ <--> edgeline <--> R3.pppg++;
        R2.pppg++ <--> coreline <--> R3.pppg++;
        R2.pppg++ <--> coreline <--> R4.pppg++;
        R3.pppg++ <--> coreline <--> R6.pppg++;
        R4.pppg++ <--> edgeline <--> R5.pppg++;
        R4.pppg++ <--> edgeline <--> R6.pppg++;
        R5.pppg++ <--> coreline <--> R7.pppg++;
        R6.pppg++ <--> edgeline <--> R7.pppg++;

}
```


- Lines 25-30 define a communication channel called '**coreline**' with the following properties
    - Delay of 2 ms
    - Datarate equal to coreDatarate. The value of the coreDatarate will be assigned in the initilaization file (omnet.ini)
    - thruputDisplayFormat = "b B U"
- Note that some routers are connected through edgeline, and others are connected through coreline
- Create omnetpp.ini file and write the code.

package.ned omnetpp.ini

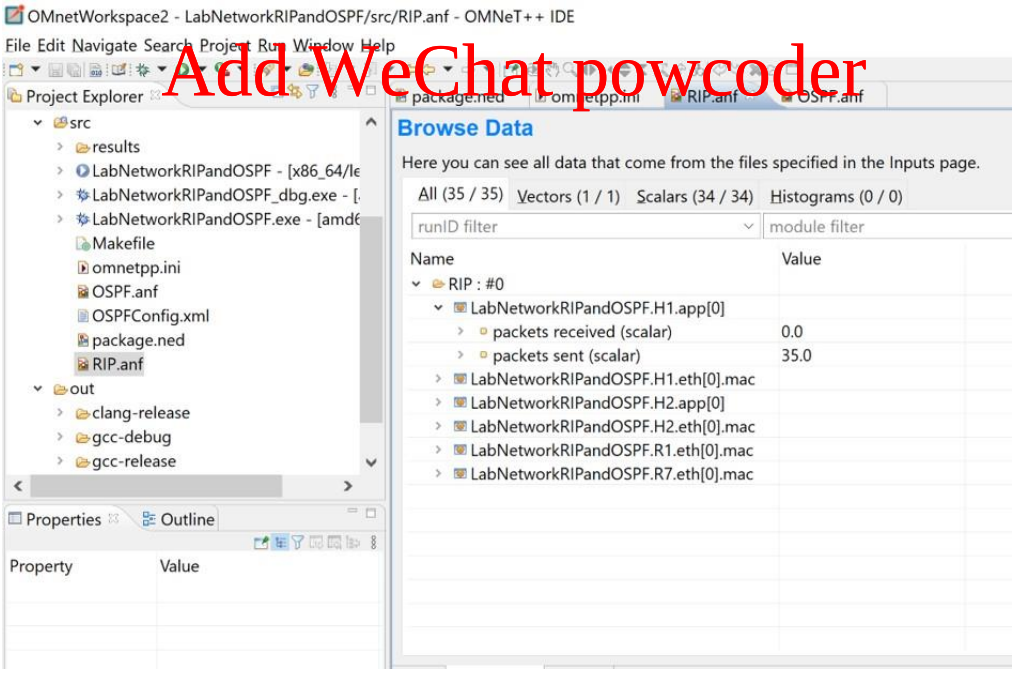
```
1  [General]
2  network = LabNetworkRIPandOSPF
3
4  sim-time-limit = 150s
5
6  **.result-recording-modes =
7  **.scalar-recording = true
8
9  # default queues
10 **.queue.typename = "EtherQosQueue"
11 **.queue.dataQueue.typename = "DropTailQueue"
12 **.queue.packetCapacity = 100
13 **.queue.dataQueue.packetCapacity = 100
14
15
16 [Config Apps]
17
18 **.H?.numApps = 1
19
20 **.H1.app[0].typename = "UdpBasicApp"
21 **.H1.app[0].destPort = 2000
22 **.H1.app[0].startTime = uniform(1s,2s)
23 **.H1.app[0].stopTime = 140s
24 **.H1.app[0].sendInterval = 40ms
25 **.H1.app[0].messageLength = 500B
26 **.H1.app[0].destAddresses = "H2"
27
28 **.H2.app[0].typename = "UdpSink" #"UdpEchoApp"
29 **.H2.app[0].localPort = 2000
30
31 [Config Exp]
32 **.edgeDatarate = 32kbps
33 **.coreDatarate = 16kbps
34
35 **.ppp[*].ppp.queue.typename = "DropTailQueue"
36 **.eth[*].mac.queue.typename = "EtherQosQueue"
37 **.eth[*].mac.queue.dataQueue.typename = "DropTailQueue"
38 **.queue.packetCapacity = 100
39
40 # statistics
41 **.H?.app[*].sentPk.result-recording-modes = count
42 **.H?.app[*].rcvdPk.result-recording-modes = count
43 **.H?.app[*].dropPk.result-recording-modes = vector
44 **.H?.app[*].endToEndDelay.result-recording-modes = vector # for computing median
45
46 **.R?.ppp[*].**Queue.rcvdPk.result-recording-modes = count
47 **.R?.ppp[*].**Queue.dropPk.result-recording-modes = count
48 **.R?.ppp[*].**Queue.queueLength.result-recording-modes = timeavg
49 **.R?.ppp[*].**Queue.queueingTime.result-recording-modes = vector # for computing median
50
51 **.R?.ppp[*].**Queue.*.scalar-recording = true
52 **.app[*].sentPk*.scalar-recording = true
53 **.app[*].rcvdPk*.scalar-recording = true
54 **.app[*].endToEndDelay*.scalar-recording = true
55 **.afQueue.*.scalar-recording = true
56
57
58 [Config OSPF]   # with OSPF
59 extends = Apps, Exp
60
61 #Disable RIP on all routers
62 **.R*.hasRip = false
63
64 # Enable OSPF on all routers
65 **.R*.hasOspf = true
66 **.R*.ospf.ospfConfig = xmldoc("OSPFConfig.xml")
67
68
69 [Config RIP]    # with RIP
70 extends = Apps, Exp
71
72 #Disable OSPF on all routers
73 **.R*.hasOspf = false
74
75 # Enable RIP on all routers
76 **.R*.hasRip = true
```

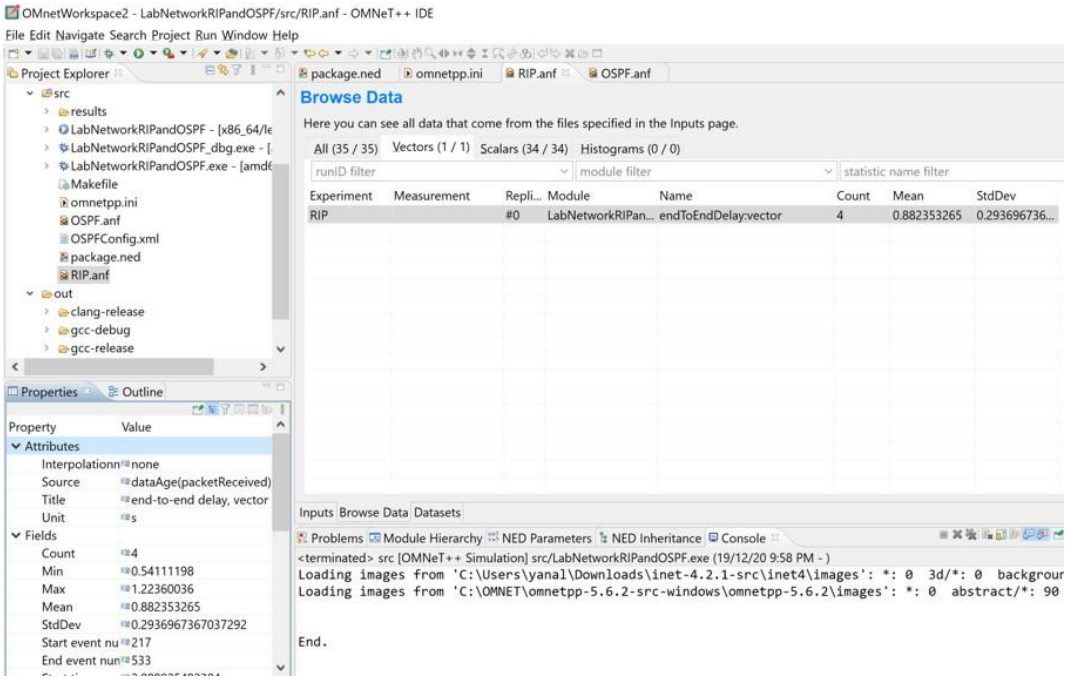

- Lines 6-7 (omnetpp.ini) to enable recording the measurement of the network
- Line 10 defines the default type of the queue at the routers that is ‘EtherQosQueue’
- Line 11 defines the default data queue at router that is ‘DropTailQueue’
- Line 12 defines the packet capacity of the queue that is equal to 100 packets

- Line 13 defines the packet capacity of the data queue that is equal to 100 packets
- Line 18 refers that all hosts run only one application. H? command refers to any instance with the name ‘H’ regards of the prefix after. Note that host H1 runs application of type ‘UdpBasicApp’, and H2 runs application of type ‘UdpSink’
- Lines 31-55 config the experiment as follows
    - Line 32 sets the value of the edgeDatarate that is equal to 32kbps
    - Line 33 sets the value of the coreDatarate that is equal to 16kbps
    - Line 35 sets the type of the queue as ‘DropTailQueue’
    - Line 36 sets the type of the queue at the ethernet mac address as ‘EtherQosQueue’
    - Line 37 sets the type of the data queue at the ethernet mac address as ‘DropTailQueue’
    - Line 38 sets the packet capacity of the queue with a value equals to 100 packets
    - Lines 41-55 to configure the statistics at hosts and routers
    - Line 41 count number of the send packets of all applications for each host
    - Line 42 count number of the received packets of all applications for each host
    - Line 43 compute number of the drop packets of each application (stored as vector) at each host.
    - Line 44 end to end delay of each application (stored as vector) at each host.
    - Line 46 count number of the received packets for each queue at any router
    - Line 47 count number of the dropped packets for each queue at any router
    - Line 48 compute the average of the queue length at any router
    - Line 49 compute the queuing time at any router
    - Line 51 enables recording the measurements per each queue at routers
    - Line 52 enables to record the send packets per each application
    - Line 53 enables to record the received packets per each application
    - Line 54 enables to record the end-to-end delay per each application
    - Line 55 enables the measurements for the afQueue
- Lines 58-66 to configure the OSPF routing protocol as follows:

- Line 59 to inherit the above applications and configuration of the above experiment
- Line 62 disables the RIP routing protocol
- Line 65 enables the OSPF routing protocol
- Line 66 reads the configuration for the OSPF from the xml file named 'OSPFConfig.xml' that is located in src folder

- Lines 69-76 to configure the RIP routing protocol as follows:
  - Line 70 to inherit the above applications and configuration of the above experiment
  - Line 73 disables the OSPF routing protocol
  - Line 76 enables the RIP routing protocol
- Build the project
- If the project is built with zero errors, run the omnetpp.ini file
- Try to select RIP in one run. See the behavior of the network
- Open the file named 'RIP.anf' to read the measurements values as shown in the two figures below.





- Try to select OSPF in another run. See the behavior of the network.
- Open the file named 'OSPF.anf' to read the measurements values.

Activity 01: parameters passing XML for OSPF

OSPF configuration started by enabling OSPF on all routers in the network, and then we configure OSPF through xml file as shown in the next code in the ini file.

```
1 **.R*.hasOspf = true
2 **.R*.ospf.ospfConfig = xmldoc("OSPFConfig.xml")
```

In line 1, OSPF is enabled in all routers in the network (R*)
In line 2, the OSPF configuration is read from the **OSPFConfig.xml** file. This xml file consists of two parts, the first part creates an area with id "0.0.0.0" and masks all the hosts and all links of the network as follows:

```
<Area id="0.0.0.0">
  <AddressRange address="H1" mask="H1" />
  <AddressRange address="H2" mask="H2" />
  <AddressRange address="R1>R2" mask="R1>R2" />
  <AddressRange address="R2>R1" mask="R2>R1" />
  <AddressRange address="R1>R3" mask="R1>R3" />
  <AddressRange address="R3>R1" mask="R3>R1" />
  <AddressRange address="R2>R3" mask="R2>R3" />
  <AddressRange address="R3>R2" mask="R3>R2" />
```

```
    <AddressRange address="R2>R4" mask="R2>R4" />
    <AddressRange address="R4>R2" mask="R4>R2" />
    <AddressRange address="R4>R5" mask="R4>R5" />
    <AddressRange address="R5>R4" mask="R5>R4" />
    <AddressRange address="R4>R6" mask="R4>R6" />
    <AddressRange address="R6>R4" mask="R6>R4" />
    <AddressRange address="R5>R7" mask="R5>R7" />
    <AddressRange address="R7>R5" mask="R7>R5" />
    <AddressRange address="R6>R7" mask="R6>R7" />
    <AddressRange address="R7>R6" mask="R7>R6" />
  </Area>
```

The second part of the xml configuration defines the costs and area id of broadcasting interfaces of all ethernet and Point to Point interfaces of all routers in the network as follows:

```
<Router name="**" RFC1583Compatible="true">
 <BroadcastInterface ifName='eth[*]' areaID='0.0.0.0' interfaceOutputCost='0' />
 <PointToPointInterface ifName='ppp[*]' areaID='0.0.0.0' interfaceOutputCost='0' />
</Router>
```

Experiment 04: Enable Queuing QoS

Introduction

In this experiment you will use weighted Queuing at routers to enhance the network performance for time sensitive application. Performance and quality of network service can be measured before and after applying quality of service. You must compare between them.

Background

Network performance refers to measures of service quality of a telecommunication product as seen by the customer. The following measures are often considered important:

- Bandwidth commonly measured in bits/second is the maximum rate that information can be transferred
- Throughput, the actual rate that information is transferred
- Latency, the delay between the sender and the receiver decoding it, this is mainly a function of the signals travel time, and processing time at any nodes the information traverses end to end delay.
- Jitter is variation in delay.
- Error rate the number of corrupted bits expressed as a percentage or fraction of the total sent drooping ratio

However, throughput, latency, the type of information transmitted, and the way that information is applied all affect the perceived speed of a connection. You have covered multiple types of queues in the course like:

1) Priority Queuing (PQ)

Priority Queuing allows you to prioritize traffic in a network. You can have a maximum of four traffic priorities. Packet filters can be identified to classify traffic and put them in each relevant queue. The queue with the highest priority will be served first until it is empty, then the lower priority queues are served in sequence. PQ gives a high priority queue absolute preference over lower priority queues; important traffic in the highest priority queue has precedence over traffic in the lower priority queues. Packets are classified based on user-specified criteria like source/destination address and TCP/UDP port number. Each packet will be placed in one of the four priority output queues, high, medium, normal and low based on the assigned priority. Packets that are not classified by priority get into the normal queue. Figure 35 illustrates this process:

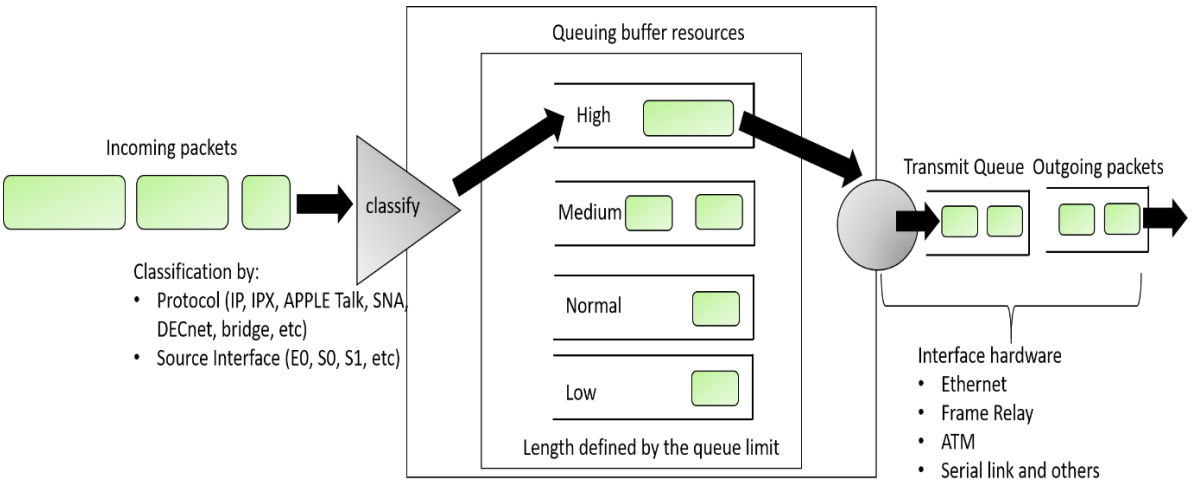


Figure 32. Priority Queuing

Priority queues on an interface are scanned for packets in descending order of priority. The packets in the highest priority queue are chosen first for transmission, then the packets in the lower priority queue and so on. This procedure is repeated every time a packet is to be sent. The maximum length of a queue is defined by the length limit. When a queue is longer than the queue limit, all additional packets are dropped.

Packet Classification for Priority Queuing

A priority list identifies the rules that describe how packets should be assigned to one of the four priority queues. A default priority queue or queue size limits can also be identified. Protocol type, incoming interface, packet size, fragments and access list can classify packets.

Why Use Priority Queuing? PQ provides absolute preferential treatment to high priority traffic, ensuring that mission-critical traffic receives priority treatment. In addition, PQ provides a faster response time than other queuing methods. Although you can enable priority output queuing for any interface, it is best used for low bandwidth, congested serial interfaces.

Restrictions on using PQ! When choosing to use PQ, consider that because lower priority traffic is often denied bandwidth in favor of higher priority traffic, use of PQ could, in the worst case, result in lower priority traffic never being sent.

2) Weighted Round Robin scheduling as Custom Queuing

Custom Queuing allows you to define a certain number of bytes to be transferred from a queue each time the queue is serviced. Queues are serviced on a round robin basis. This allows you to share bandwidth among applications.

How Custom Queue Works? Custom Queue handles traffic by specifying the number of packets or bytes being served for each queue. Its services queues by cycling through them in round robin fashion, sending the portion of bandwidth allocated to each queue before moving to the next queue. The byte count or bandwidth assigned to each queue can be seen as weight. If a queue is empty, router will send packets from the next queue which contains packets. When CQ is enabled on an interface, the system maintains 17 output queues for that interface. You can specify queues 1 through 16. Associated with each output queue is a configurable byte count, which specifies how many bytes of data the system should deliver from the current queue before it moves on to the next queue.

Queue number 0 is the system queue; it is emptied before any of the queues numbered 1 through 16 are processed. The system queue serves high priority packets, such as signaling packets. Other traffic cannot be configured to use this queue. For queue numbers 1 through 16, the system cycles through the queues sequentially (in a round-robin fashion), transmitting the configured byte count from each queue in each cycle and moving on to the next one. When a particular queue is being processed, packets are sent until the number of bytes sent exceeds the queue byte count or the queue is empty. Bandwidth used by a particular queue can be indirectly specified only in terms of byte count and queue length.

Figure 36 shows how CQ works

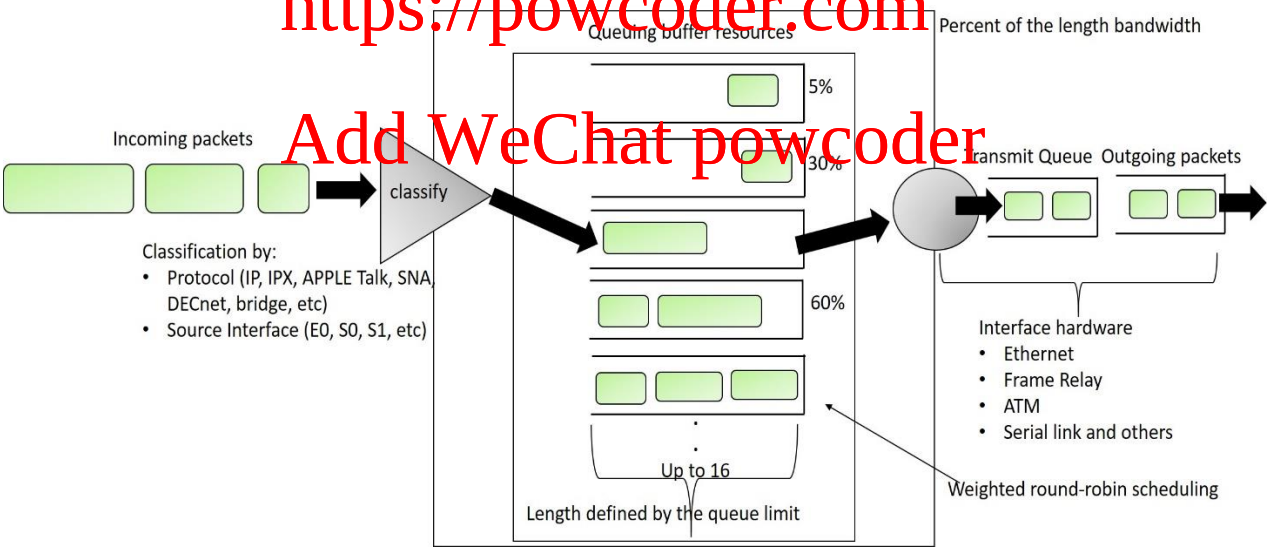


Figure 33. Custom Queuing

CQ ensures that no application or specified group of applications achieves more than a predetermined proportion of overall capacity when the line is under stress. Like PQ, CQ is statically configured and does not automatically adapt to changing network conditions.

What is Byte Count? The router sends packets from a particular queue until the byte count is exceeded. Once the byte count value is exceeded, the packet that is currently being sent will be completely sent. Therefore, if you set the byte count to 100 bytes and the packet size of your protocol is 1024 bytes, then every time this queue is serviced, 1024 bytes will be sent, not 100 bytes. For example, suppose one protocol has 500-byte packets, another has 300-byte packets, and a third has 100-byte packets. If you want to split the bandwidth evenly across all three protocols, you might choose to specify byte counts of 200, 200, and 200 for each queue. However, this configuration does not result in a 33/33/33 ratio. When the router services the first queue, it sends a single 500-byte packet; when it services the second queue, it sends a 300-byte packet; and when it services the third queue, it sends two 100-byte packets. The effective ratio is 50/30/20.

Thus, setting the byte count too low can result in an unintended bandwidth allocation. However, very large byte counts will produce a "jerky" distribution. That is, if you assign 10 KB, 10 KB, and 10 KB to three queues in the example given, each protocol is serviced promptly when its queue is the one being serviced, but it may be a long time before the queue is serviced again. A better solution is to specify 500-byte, 600-byte, and 500-byte counts for the queue. This configuration results in a ratio of 31/38/31, which may be acceptable. In order to service queues in a timely manner and ensure that the configured bandwidth allocation is as close as possible to the required bandwidth allocation, you must determine the byte count based on the packet size of each protocol; otherwise, your percentages may not match what you configure.

3) **Differentiated Services (DiffServ, or DS)** is a multiple service model that satisfying different quality of service (QoS) requirements.

DS tries to deliver a specific kind of service according to QoS that is specified by each packet. Distinguishing the QoS can be done using different ways, for example, using the 6-bit differentiated services code point (DSCP) located in IP packets or source and destination addresses. The network uses the QoS specification to classify, mark, traffic and to perform intelligent queueing. Differentiated Services is used for critical applications and for ensuring end-to-end QoS. DS is appropriate for aggregate flows because it does a relatively multiple level of traffic classification.

DS Field Definition

A replacement header field, called the DS field, is defined by Differentiated Services. Six bits of the DS field are used as the DSCP to select the Per-Hop Behavior (PHB) at each interface.

Per-Hop Behaviors (Default PHB)

The default PHB essentially specifies that a packet marked with a DSCP value of 000000 (recommended) receives the traditional best-effort service from a DS-compliant node. If a packet arrives at a DS-compliant node, and the DSCP value is not mapped to any other PHB, the packet will get mapped to the default PHB.

Class-Selector PHB

To keep backward-compatibility with any IP precedence scheme that are currently used in network, DS has defined a DSCP value in the form xxx000, where x is either 0 or 1. These DSCP values are called Class-Selector Code Points. For example, packets with a DSCP value of 11000 (the equivalent of the IP Precedence-based value of 110) have better forwarding treatment (for scheduling, queueing, etc), as compared to packets with a DSCP value of 100000 (the equivalent of the IP Precedence-based value of 100).

Assured Forwarding AF PHB

An AF PHB defines a method that can give different forwarding assurances. For example, network traffic can be divided into the following classes:

- Gold: Traffic in this category is allocated 50 percent of the available bandwidth.
- Silver: Traffic in this category is allocated 30 percent of the available bandwidth.
- Bronze: Traffic in this category is allocated 20 percent of the available bandwidth.

AF PHB defines four AF classes: AF1, AF2, AF3, and AF4. Each class is assigned a specific amount of buffer space and interface bandwidth, according to Service Level Agreement (SLA) with the service provider or policy map. Within each AF class, you can specify three drop precedence (dp) values: 1, 2, and 3.

4) DropTail queuing?

Drop tail is of passive queue management type. It stores the packet until the buffer gets full, and when the buffer gets full, i.e., there is no more space in the queue, it starts dropping every new coming packet. It is based on best-effort traffic schedule based on FIFO queue strategy with limited size.

Task 01: Create network with best effort performance and measure the performance

Activity 01: build the network topology

Crate new OMNet++ project and add INET framework to it. Create network topology as in figure 37 below.

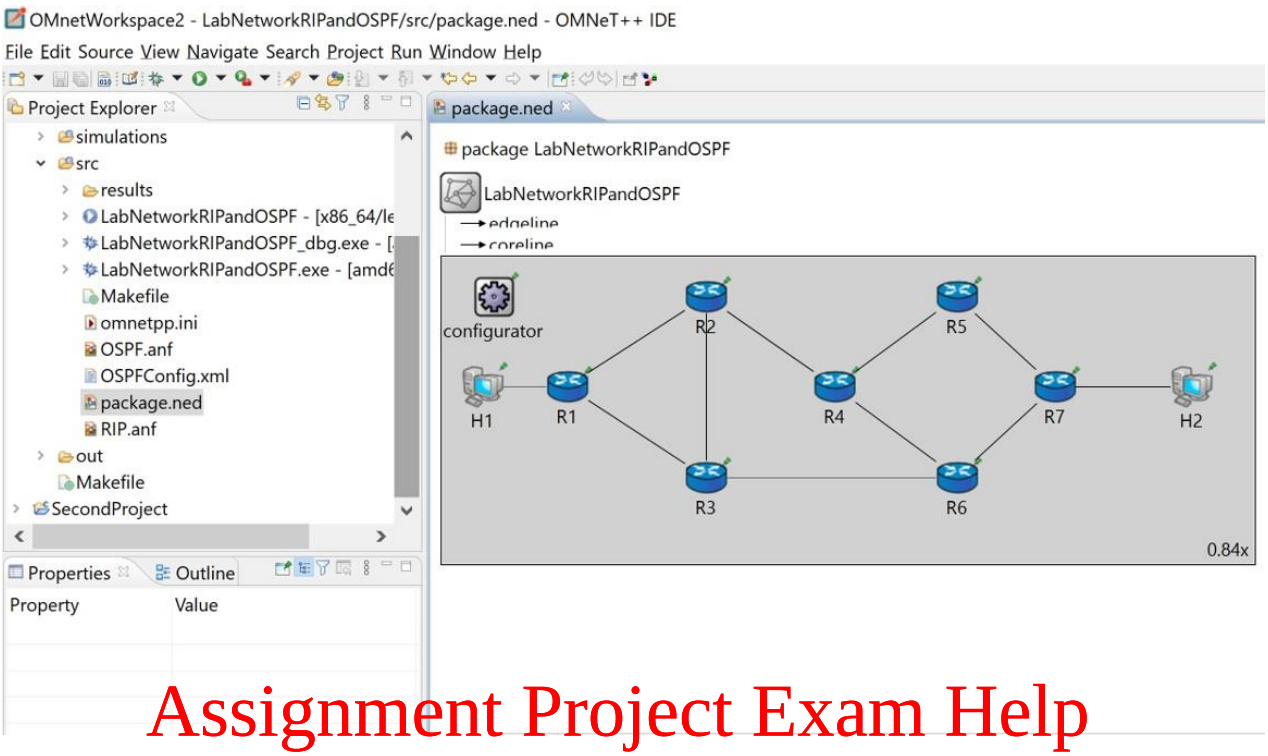


Figure 34. Network Topology

You need the following code in the ned source file


```
package LabNetworkQueueing;

import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.common.misc.ThruputMeteringChannel;
import inet.node.ethernet.Eth100M;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;

@license(LGPL);
network LabNetworkQueueing
{
    parameters:
        double edgeDatarate @unit(bps);
        double coreDatarate @unit(bps);
        @display("bgb=953,360");

    types:
        channel edgeline extends ThruputMeteringChannel
        {
            delay = 2ms;
            datarate = edgeDatarate;
            thruputDisplayFormat = "b B U";
        }

        channel coreline extends ThruputMeteringChannel
        {
            delay = 2ms;
```

```
            datarate = coreDatarate;
            thruputDisplayFormat = "b B U";
        }

    submodules:
        R1: Router {
            @display("p=148,153");
            gates:
                ethg[1];
        }
        R2: Router {
            @display("p=311,47");
        }
        R3: Router {
            @display("p=311,259");
        }
        R4: Router {
            @display("p=461,153");
        }
        R5: Router {
            @display("p=605,47");
        }
        R6: Router {
            @display("p=605,259");
        }
        R7: Router {
            @display("p=710,153");
            gates:
                ethg[1];
        }
        H1: StandardHost {
            @display("p=49,153");
            gates:
                ethg[1];
        }
        H2: StandardHost {
            @display("p=828,153");
            gates:
                ethg[1];
        }
        configurator: Ipv4NetworkConfigurator {
            @display("p=42,51");
        }

    connections:
        //Between hosts and routers
        H1.ethg[0] <--> Eth100M <--> R1.ethg[0];
        H2.ethg[0] <--> Eth100M <--> R7.ethg[0];

        //Between routers
        R1.pppg++ <--> edgeline <--> R2.pppg++;
        R1.pppg++ <--> edgeline <--> R3.pppg++;
        R2.pppg++ <--> coreline <--> R3.pppg++;
        R2.pppg++ <--> coreline <--> R4.pppg++;
        R3.pppg++ <--> coreline <--> R6.pppg++;
```

```
        R4.pppg++ <--> edgeline <--> R5.pppg++;
        R4.pppg++ <--> edgeline <--> R6.pppg++;
        R5.pppg++ <--> coreline <--> R7.pppg++;
        R6.pppg++ <--> edgeline <--> R7.pppg++;

}
```

Activity 02: configure general parameters

for statistical and result gathering you must enable `result-recording-modes`. Queuing and OSPF parameters are set too as following code.

```
[General]
network = LabNetworkQueueing

sim-time-limit = 1250s

**.result-recording-modes = all
**.scalar-recording = true

# default queues
**.queue.typename = "EtherQosQueue"
**.queue.dataQueue.typename = "DropTailQueue"
**.queue.packetCapacity = 100
**.queue.dataQueue.packetCapacity = 100

# Enable OSPF on all routers
**.R*.hasOspf = true

**.R*.ospf.ospfConfig = xmldoc("OSPFConfig.xml")
```


Activity 03: configure applications

run the following three applications on the two hosts H1 and H2

1. UdpBasicBurst (to simulate a voice streaming)
2. UdpBasicApp (to simulate a video streaming)
3. TcpBasicClientApp (to simulate a regular web streaming)

```
[Config Apps]

**.H?.numApps = 3

# first app: voice streaming
**.H1.app[0].typename = "UdpBasicBurst"
**.H1.app[0].destAddresses = "H2"
**.H1.app[0].chooseDestAddrMode = "once"
**.H1.app[0].destPort = 1000
**.H1.app[0].startTime = uniform(1s,2s)
**.H1.app[0].stopTime = 1200s
```

```
**.H1.app[0].messageLength = 172B # 160B voice + 12B Rtp header
**.H1.app[0].burstDuration = exponential(0.352s)
**.H1.app[0].sleepDuration = exponential(0.650s)
**.H1.app[0].sendInterval = 20ms

**.H2.app[0].typename = "UdpBasicBurst"
**.H2.app[0].localPort = 1000
**.H2.app[0].delayLimit = 0ms
**.H2.app[0].destAddresses = ""
**.H2.app[0].chooseDestAddrMode = "once"
**.H2.app[0].destPort = 0
**.H2.app[0].messageLength = 0B
**.H2.app[0].burstDuration = 0s
**.H2.app[0].sleepDuration = 0s
**.H2.app[0].sendInterval = 0ms

#second app: video
**.H1.app[1].typename = "UdpBasicApp"
**.H1.app[1].destPort = 2000
**.H1.app[1].startTime = uniform(1s,2s)
**.H1.app[1].stopTime = 1200s
**.H1.app[1].sendInterval = 40ms
**.H1.app[1].messageLength = 1000B
**.H1.app[1].destAddresses = "H2"

**.H2.app[1].typename = "UdpSink" #"UdpEchoApp"
**.H2.app[1].localPort = 2000

#third app: tcp traffic
**.H1.app[2].typename = "TcpBasicClientApp"
**.H1.app[2].connectAddress = "H2"
**.H1.app[2].connectPort = 3000
**.H1.app[2].numRequestsPerSession = 1 # HTTP 1.0
**.H1.app[2].requestLength = 50B
**.H1.app[2].replyLength = 100B
**.H1.app[2].thinkTime = 40ms
**.H1.app[2].idleInterval = 10s
**.H1.app[2].stopTime = 1200s

**.H2.app[2].typename = "TcpGenericServerApp"
**.H2.app[2].localPort = 3000
```


Activity 04: configure network parameters and statistical measurements

For the network, there are two different links configuration and capturing network performance (Sent packets, Received packets, Dropped packets, End to end delay). You can note that we used traffic classifier and markers TC1 at two locations R1 and R7 explain why??

Using the following code:

```
[Config Exp]
**.edgeDatarate = 32kbps
**.coreDatarate = 16kbps

**.R1.eth[*].ingressTC.typename = "TC1"
**.R7.eth[*].ingressTC.typename = "TC1"

**.ingressTC.numClasses = 3
**.ingressTC.classifier.filters = xmldoc("filters.xml",
"//experiment[@id='default']")
**.ingressTC.marker.dscps = "AF11 AF21 AF31 AF41 BE"

# statistics
**.H?.app[*].sentPk.result-recording-modes = count
**.H?.app[*].rcvdPk.result-recording-modes = count
**.H?.app[*].dropPk.result-recording-modes = vector
**.H?.app[*].endToEndDelay.result-recording-modes = vector # for computing median

**.R?.ppp[*].**Queue.rcvdPk.result-recording-modes = count
**.R?.ppp[*].**Queue.dropPk.result-recording-modes = count
**.R?.ppp[*].**Queue.queueLength.result-recording-modes = timeavg
**.R?.ppp[*].**Queue.queueingTime.result-recording-modes = vector # for computing
median
**.R?.ppp[*].**Queue.*.scalar-recording = true

**.app[*].sentPk*.scalar-recording = true
**.app[*].rcvdPk*.scalar-recording = true
**.app[*].endToEndDelay*.scalar-recording = true

**.afQueue.*.scalar-recording = true
```

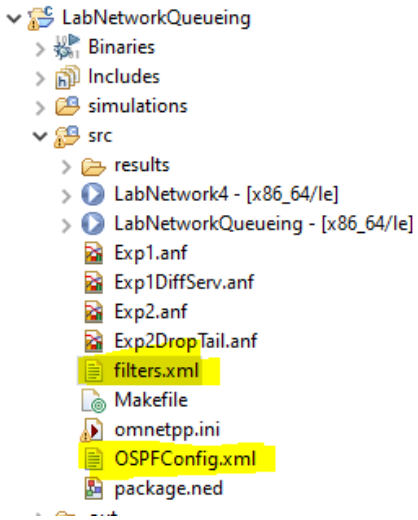

Activity 05: Configure two types of queuing

Configure best effort and weighted round robin queue to measure (Received packets, Dropped packets, Queue length, Queuing time) in both cases as in the following code:

```
[Config Exp1DiffServ]      # with Diffserv Queue
extends = Apps, Exp
**.R?.ppp[*].ppp.queue.typename = "DSQueue1"  #Diffserv Queue
**.R?.ppp[*].ppp.queue.packetCapacity = 100  #-1
**.R?.ppp[*].ppp.queue.*.packetCapacity = 100
**.R?.ppp[*].ppp.queue.wrr.weights = "10 40 5 1 1"

[Config Exp2DropTail]      # with DropTailQueue
extends = Apps, Exp
**.ppp[*].ppp.queue.typename = "DropTailQueue"
**.eth[*].mac.queue.typename = "EtherQosQueue"
**.eth[*].mac.queue.dataQueue.typename = "DropTailQueue"
**.queue.packetCapacity = 100
```

Activity 06: add xml files for OSPF configuration and traffic classification



filters.xml:

```
<filters>
  <experiment id="default">
    <filter srcAddress="H1" destPort="1000" gate="0"/>
    <filter srcAddress="H1" destPort="2000" gate="1"/>
    <filter srcAddress="H1" destPort="3000" gate="2"/>
    <filter srcAddress="H2" gate="0"/>
  </experiment>
</filters>
```


OSPFConfig.xml

```
<?xml version="1.0"?>
<OSPFASConfig>

  <Area id="0.0.0.0">
    <AddressRange address="H1" mask="H1" />
    <AddressRange address="H2" mask="H2" />

    <AddressRange address="R1>R2" mask="R1>R2" />
    <AddressRange address="R2>R1" mask="R2>R1" />

    <AddressRange address="R1>R3" mask="R1>R3" />
    <AddressRange address="R3>R1" mask="R3>R1" />

    <AddressRange address="R2>R3" mask="R2>R3" />
    <AddressRange address="R3>R2" mask="R3>R2" />

    <AddressRange address="R2>R4" mask="R2>R4" />
    <AddressRange address="R4>R2" mask="R4>R2" />

    <AddressRange address="R3>R6" mask="R3>R6" />
    <AddressRange address="R6>R3" mask="R6>R3" />
```

```
    <AddressRange address="R4>R5" mask="R4>R5" />
    <AddressRange address="R5>R4" mask="R5>R4" />

    <AddressRange address="R4>R6" mask="R4>R6" />
    <AddressRange address="R6>R4" mask="R6>R4" />

    <AddressRange address="R5>R7" mask="R5>R7" />
    <AddressRange address="R7>R5" mask="R7>R5" />

    <AddressRange address="R6>R7" mask="R6>R7" />
    <AddressRange address="R7>R6" mask="R7>R6" />
  </Area>

  <Router name="**" RFC1583Compatible="true">
     <BroadcastInterface ifName='eth[*]' areaID='0.0.0.0' interfaceOutputCost='0'
/>
     <PointToPointInterface ifName='ppp[*]' areaID='0.0.0.0'
interfaceOutputCost='0' />
  </Router>

</OSPFASConfig>
```


TASK 2: run the experiments under two configurations mode

You need to run the experiment in two cases:

Case 1, with best effort drop tail queue and measure the network performance

Case 2, with WRR queue and measure the network performance

Task 03: Displaying and Exporting Simulation Results Data

After closing the simulation of the experiments, a few files will be created in the results folder under the src package of the project (sca, vec). Double clicking one of these files will create an anf file directly in the src folder as shown in Figure 38. When you open the anf file you will have the window shown in Figure 39 with all the data collected in the Data section separated as they were collected whether as scalar data in the sca file, or as vector data in the vec file.

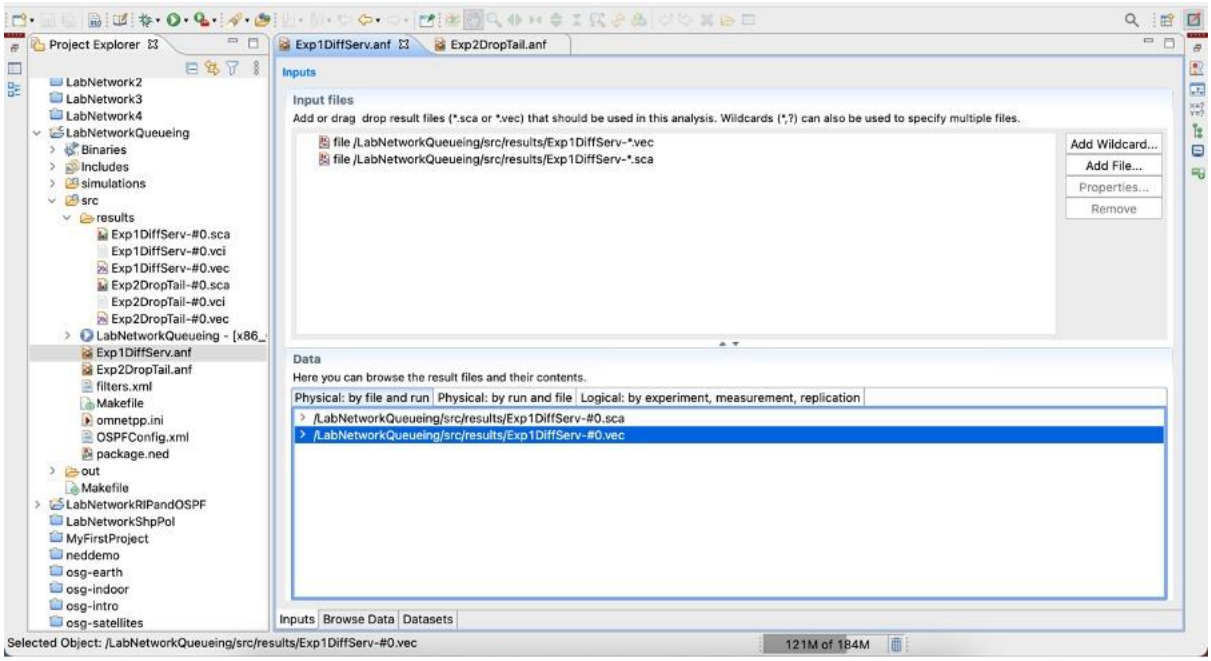


Figure 35: Simulation Results Files

For example, let us open the vec file for Exp1 the DiffServ. After expanding the main vector called "Exp1DiffServ: #0", you will have a list with all vectors collected during the simulation as shown in Figure 39.

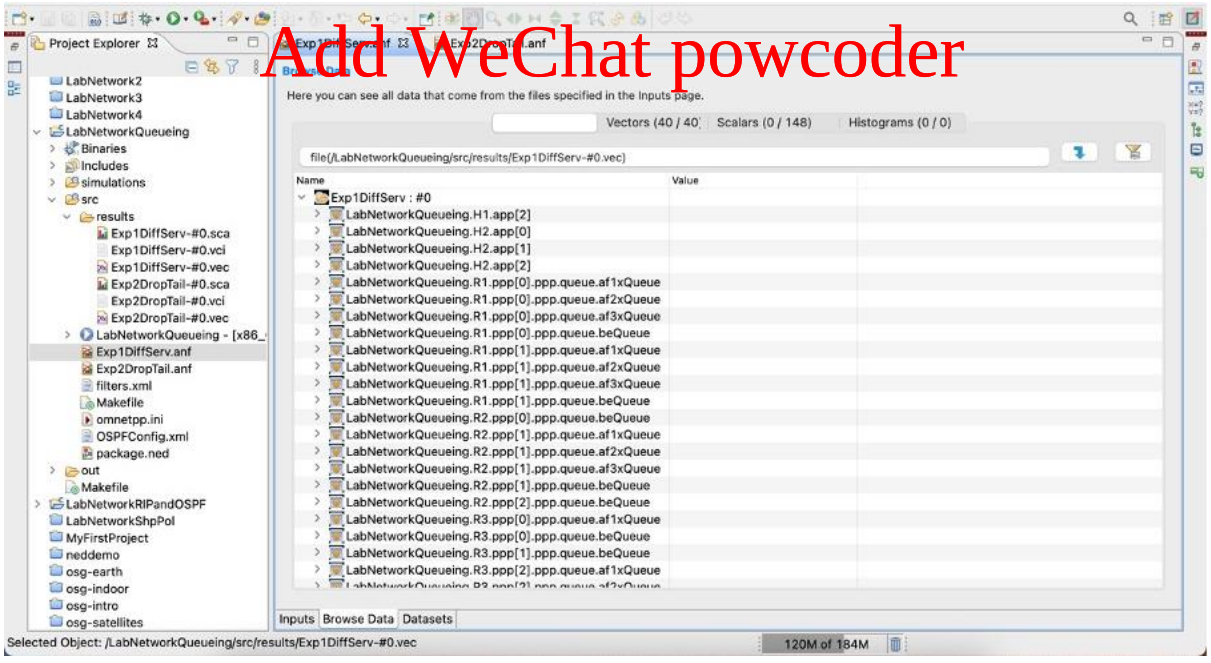


Figure 36: Exp1DiffServ vector data results

For example, Let us expand the vector of Host 2, app0 that is called "LabNetworkQueueing.H2.app[0]" and then expand the vector inside called "endToEndDelay" as shown in Figure 40. Now you can see metrics that are calculated for 742 packets (the Count value) received at Host 2 for application 0.

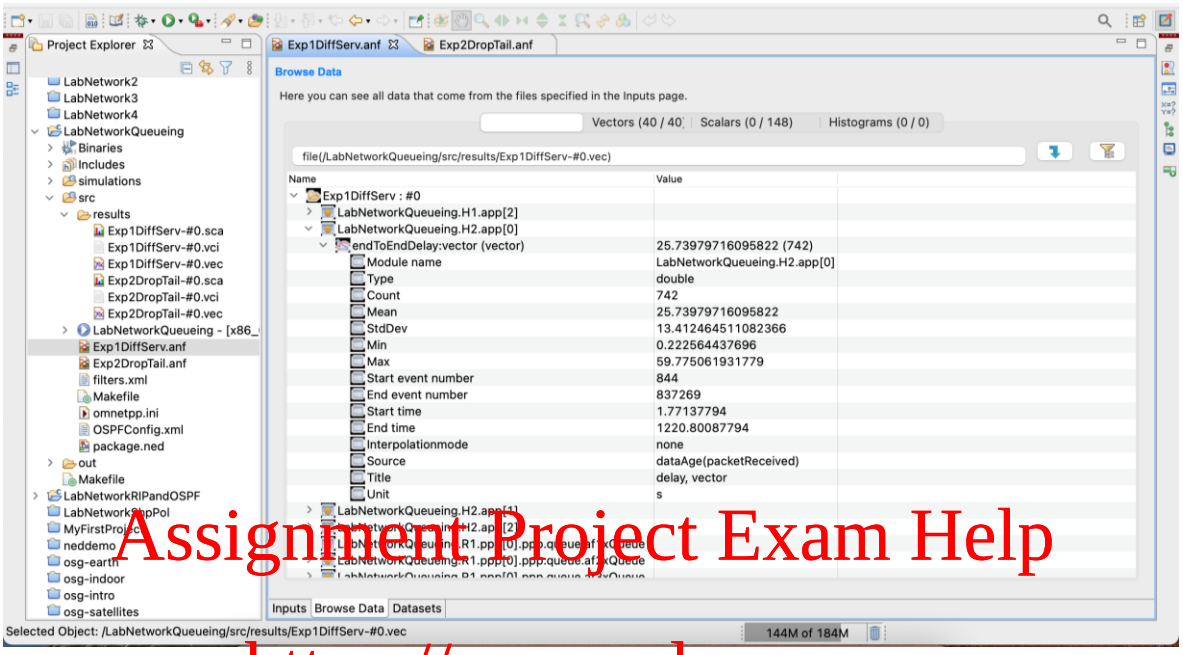


Figure 37: End to End Delay vector data of application 0

To see the actual 742 values collected, you need to export the vector into a spreadsheet. To do this, right click the vector name "endToEndDelay" and click on Export Data and then choose CSV for Spreadsheets as shown in Figure 41.

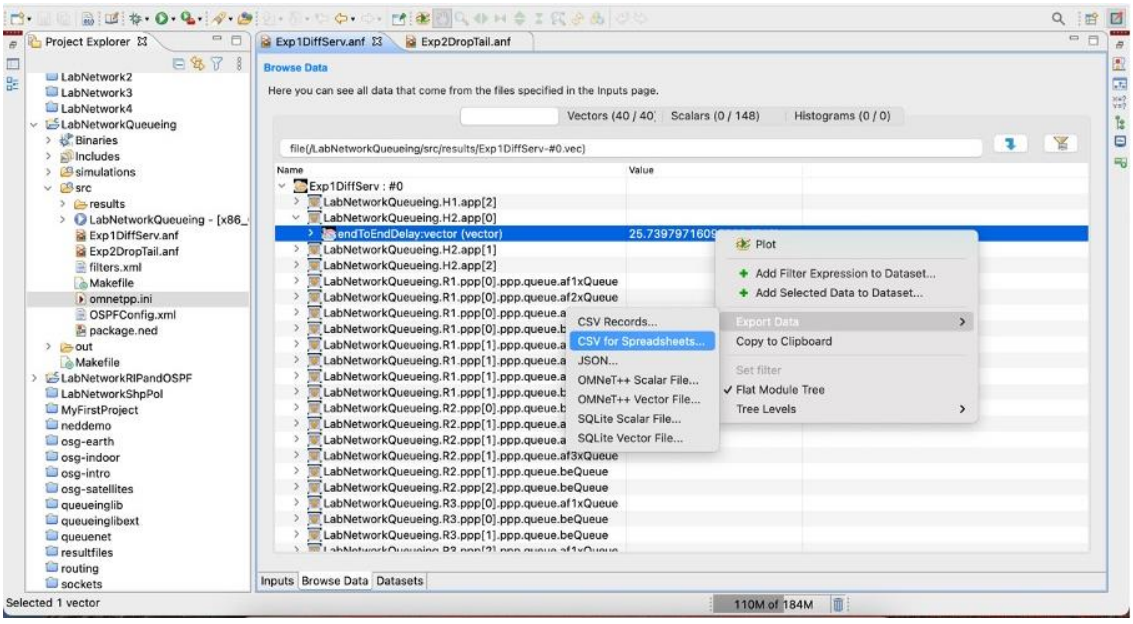


Figure 38: Exporting the vector to CSV spreadsheet

From the “CSV-S Export” window shown in Figure 42, Browse for the directory you want to save your sheet file in and give it a name as shown in Figure 43, and then click Finish.

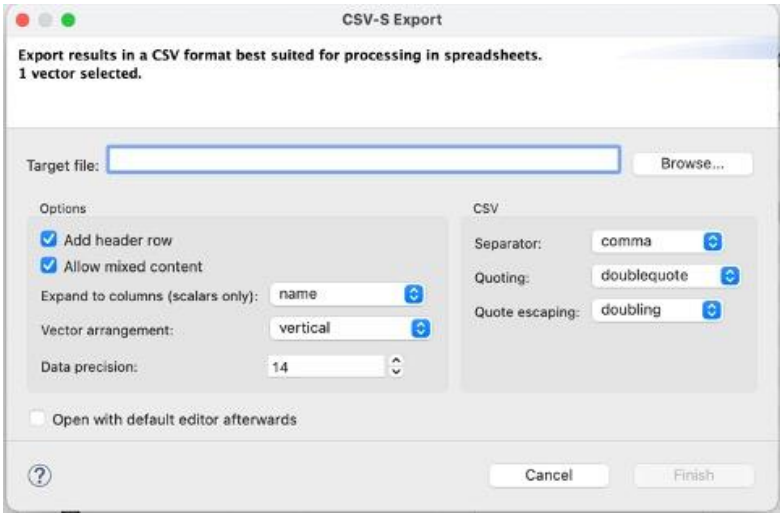


Figure 39: CSV Export Window

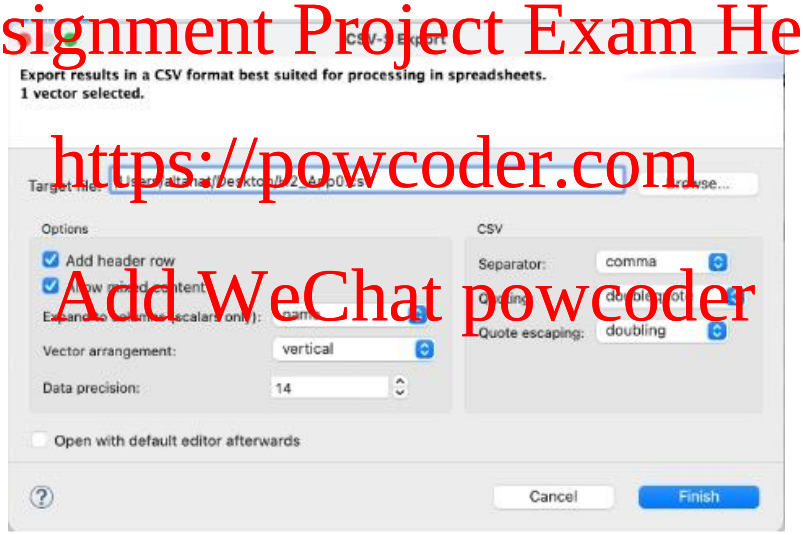
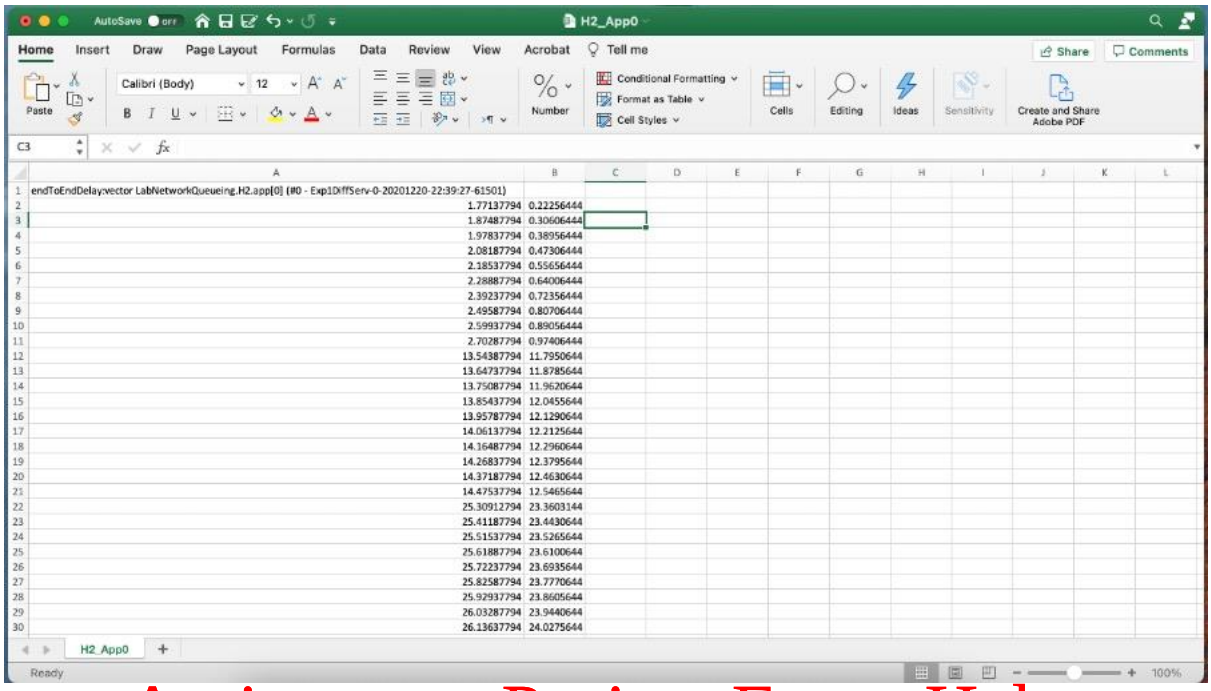


Figure 40: Export CSV to file H2_app0.csv

Now, you can open the generated file in Excel and see all the data, the 742 records as shown in Figure 44.



endToEndDelay:vector LabNetworkQueueing.H2.app[0] (#0 - Exp1DiffServ-0-20201220-22:39:27-61501)		
1	1.77137794	0.22256444
2	1.87487794	0.30606444
3	1.97837794	0.38956444
4	2.08187794	0.47306444
5	2.18537794	0.55656444
6	2.28887794	0.64006444
7	2.39237794	0.72356444
8	2.49587794	0.80706444
9	2.59937794	0.89056444
10	2.70287794	0.97406444
11	13.54387794	11.7950644
12	13.64737794	11.8785644
13	13.75087794	11.9620644
14	13.85437794	12.0455644
15	13.95787794	12.1290644
16	14.06137794	12.2125644
17	14.16487794	12.2960644
18	14.26837794	12.3795644
19	14.37187794	12.4630644
20	14.47537794	12.5465644
21	25.30912794	23.3603144
22	25.41187794	23.4430644
23	25.51537794	23.5265644
24	25.61887794	23.6100644
25	25.72237794	23.6935644
26	25.82587794	23.7770644
27	25.92937794	23.8605644
28	26.03287794	23.9440644
29	26.13637794	24.0275644
30		

Figure 41: Generated CSV spreadsheet file

Now you can calculate any metric you want from the excel sheet file. For example, Figure 45 is a graph for all end to end delays. The x-axis is the first column in sheet, and the y-axis is the second column.

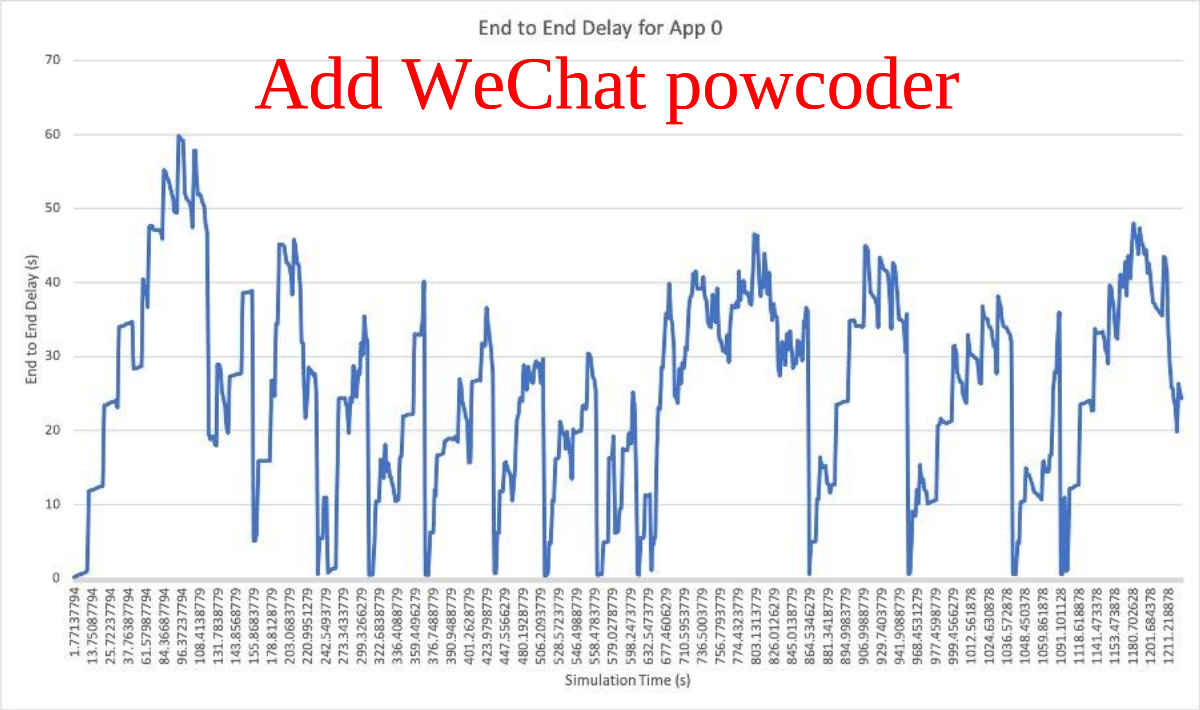


Figure 42: End to end delays for all 742 packets of application 0 received at Host 2

Experiment 05: Traffic shaping and policing

Introduction

In this experiment you will learn about traffic shaping and policing. In addition, you will get more practice for router queuing while applying shaping and policing to enhance QoS.

Background

Traffic shaping and policing are two networking QoS techniques that are used to regulate the traffic between customer and ISP to avoid SLA violation between them. Traffic policing is about taking actions for the packets by passing those conforms to a specific rate and dropping those packets that violates the specified rate. Usually, traffic policing is applied on the ingress bound traffic of ISP. Traffic shaping tries to make the traffic adhere to a specific rate by slowing the packets sending through delaying packets in a buffer and sending them once the space becomes available. Usually, traffic shaping is applied on the egress of the router at the customer's side. For example, if the ISP applies a policing that ensures a customer's traffic should not exceed 10 Mbps, in this case the customer can applies traffic shaping to make sure traffic conforms to 10Mbps. Otherwise, any traffic above 10 Mbps will be dropped by ISP.

Let us go a step further with technical terms. Let us assume an organization requests 25 Mbps internet service through fiber media from a specific ISP. How does the ISP ensure that the organisation will get only for what they pay for 25 Mbps even through the physical fiber connection to the organisation can support more transmission rate of 100 Mbps? Here the ISP can apply traffic policing to control that. The ISP will sign a contract with the organization that any traffic exceeds 25 Mbps will be dropped. There are few technical terms.

- 25 Mbps represents **Committed Information Rate (CIR)** which is the average speed that ISP guarantees for the customer
- 100 Mbps represents **Access Rate (AR)** which is the maximum rate that the physical medium, fiber in the above example, can transmit
- **Committed Burst (Bc)** represents the number of packets (usually in bytes) that can be sent as a group without violating CIR.
- **Excess Burst (Be)** represents the ability to store up tokens that you did not use during a time interval.

Task 01: Create network with best effort performance and measure the performance

Activity 01: build the network topology

Crate new OMNet++ project and add INET framework to it. Create network topology as in below figure 46.

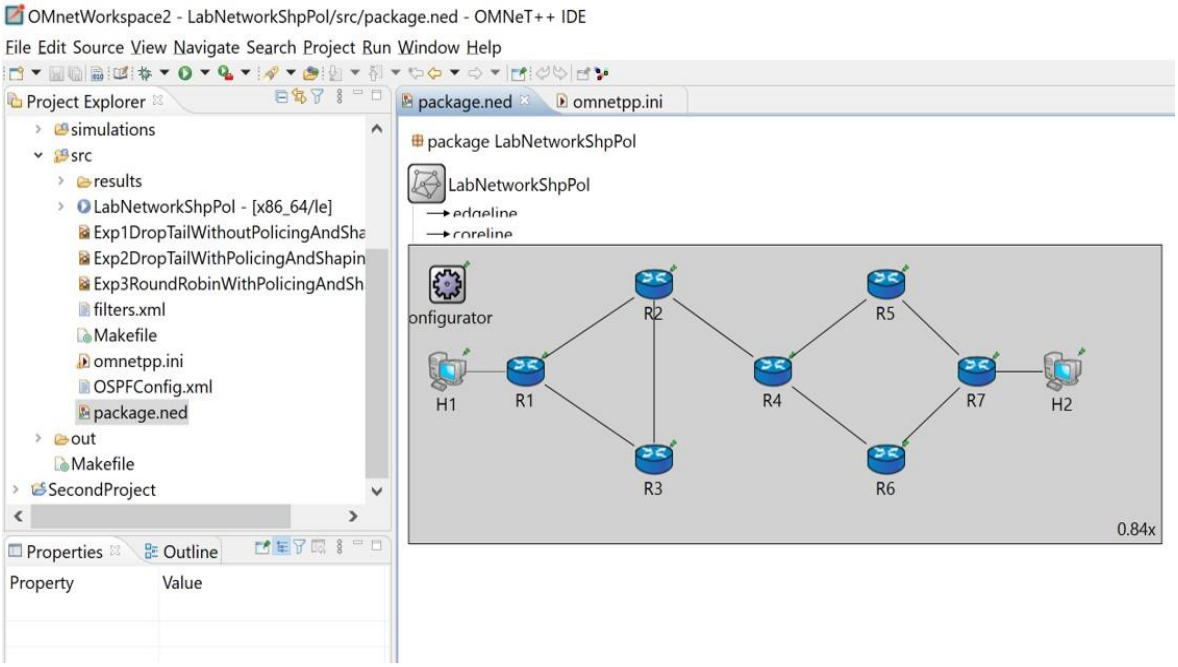


Figure 43. Network Topology

You need the following code.


```
package LabNetworkShpPol;

import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.common.misc.ThruputMeteringChannel;
import inet.node.ethernet.Eth100M;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;

@license(LGPL);
network LabNetworkShpPol
{
    parameters:
        double edgeDatarate @unit(bps);
        double coreDatarate @unit(bps);
        @display("bgb=953,360");

    types:
        channel edgeline extends ThruputMeteringChannel
        {
            delay = 2ms;
            datarate = edgeDatarate;
            thruputDisplayFormat = "b B U";
        }

        channel coreline extends ThruputMeteringChannel
        {
            delay = 2ms;
```

```
            datarate = coreDatarate;
            thruputDisplayFormat = "b B U";
        }

    submodules:
        R1: Router {
            @display("p=148,153");
            gates:
                ethg[1];
        }
        R2: Router {
            @display("p=311,47");
        }
        R3: Router {
            @display("p=311,259");
        }
        R4: Router {
            @display("p=461,153");
        }
        R5: Router {
            @display("p=605,47");
        }
        R6: Router {
            @display("p=605,259");
        }
        R7: Router {
            @display("p=710,153");
            gates:
                ethg[1];
        }
        H1: StandardHost {
            @display("p=49,153");
            gates:
                ethg[1];
        }
        H2: StandardHost {
            @display("p=828,153");
            gates:
                ethg[1];
        }
        configurator: Ipv4NetworkConfigurator {
            @display("p=47.85,46.65375");
        }

    connections:
        //Between hosts and routers
        H1.ethg[0] <--> Eth100M <--> R1.ethg[0];
        H2.ethg[0] <--> Eth100M <--> R7.ethg[0];

        //Between routers
        R1.pppg++ <--> edgeline <--> R2.pppg++;
        R1.pppg++ <--> edgeline <--> R3.pppg++;
        R2.pppg++ <--> coreline <--> R3.pppg++;
        R2.pppg++ <--> coreline <--> R4.pppg++;
        R4.pppg++ <--> edgeline <--> R5.pppg++;
```

```
        R4.pppg++ <--> edgeline <--> R6.pppg++;
        R5.pppg++ <--> coreline <--> R7.pppg++;
        R6.pppg++ <--> edgeline <--> R7.pppg++;

}
```

Activity 02: configure general parameters

for statistical and result gathering you must enable `result-recording-modes`. Queuing and OSPF parameters are set too as following code.

```
[General]
network = LabNetworkQueueing

sim-time-limit = 1250s

**.result-recording-modes = all
**.scalar-recording = true

# default queues
**.queue.typename = "EtherQosQueue"
**.queue.dataQueue.typename = "DropTailQueue"
**.queue.packetCapacity = 100
**.queue.dataQueue.packetCapacity = 100

# Enable OSPF on all routers
**.R*.hasOspf = true

**.R*.ospf.ospfConfig = xmldoc("OSPFConfig.xml")
```


Activity 03: configure applications

run the following three applications on the two hosts H1 and H2

1. UdpBasicBurst (to simulate a voice streaming)
2. UdpBasicApp (to simulate a video streaming)
3. TcpBasicClientApp (to simulate a regular web streaming)

```
[Config Apps]

**.H?.numApps = 3

# first app: voice streaming
**.H1.app[0].typename = "UdpBasicBurst"
**.H1.app[0].destAddresses = "H2"
**.H1.app[0].chooseDestAddrMode = "once"
**.H1.app[0].destPort = 1000
```

```
**.H1.app[0].startTime = uniform(1s,2s)
**.H1.app[0].stopTime = 1200s
**.H1.app[0].messageLength = 172B # 160B voice + 12B Rtp header
**.H1.app[0].burstDuration = exponential(0.352s)
**.H1.app[0].sleepDuration = exponential(0.650s)
**.H1.app[0].sendInterval = 20ms

**.H2.app[0].typename = "UdpBasicBurst"
**.H2.app[0].localPort = 1000
**.H2.app[0].delayLimit = 0ms
**.H2.app[0].destAddresses = ""
**.H2.app[0].chooseDestAddrMode = "once"
**.H2.app[0].destPort = 0
**.H2.app[0].messageLength = 0B
**.H2.app[0].burstDuration = 0s
**.H2.app[0].sleepDuration = 0s
**.H2.app[0].sendInterval = 0ms

#second app: video
**.H1.app[1].typename = "UdpBasicApp"
**.H1.app[1].destPort = 2000
**.H1.app[1].startTime = uniform(1s,2s)
**.H1.app[1].stopTime = 1200s
**.H1.app[1].sendInterval = 40ms
**.H1.app[1].messageLength = 500B
**.H1.app[1].destAddresses = "H2"

**.H2.app[1].typename = "UdpSink" # "UdpEchoApp"
**.H2.app[1].localPort = 2000

#third app: tcp traffic
**.H1.app[2].typename = "TcpBasicClientApp"
**.H1.app[2].connectAddress = "H2"
**.H1.app[2].connectPort = 3000
**.H1.app[2].numRequestsPerSession = 1 # HTTP 1.0
**.H1.app[2].requestLength = 50B
**.H1.app[2].replyLength = 100B
**.H1.app[2].thinkTime = 40ms
**.H1.app[2].idleInterval = 10s
**.H1.app[2].stopTime = 1200s

**.H2.app[2].typename = "TcpGenericServerApp"
**.H2.app[2].localPort = 3000
```


Activity 04: configure network parameters and statistical measurements

For the network there are two different links configuration and capturing network performance (Sent packets, Received packets, Dropped packets, End to end delay). You can note that we used traffic classifier and markers TC1 at two locations R1 and R7 explain why??

Using the following code:

```
[Config Exp]
**.edgeDatarate = 32kbps
**.coreDatarate = 16kbps

#Traffic Classifier and Marker
**.R1.eth[*].ingressTC.typename = "TC1"
**.R7.eth[*].ingressTC.typename = "TC1"

**.ingressTC.numClasses = 3
**.ingressTC.classifier.filters = xmldoc("filters.xml",
"//experiment[@id='default']")
**.ingressTC.marker.dscps = "AF11 AF21 AF31 AF41 BE"

# statistics
**.H?.app[*].sentPk.result-recording-modes = count
**.H?.app[*].rcvdPk.result-recording-modes = count
**.H?.app[*].dropPk.result-recording-modes = vector
**.H?.app[*].endToEndDelay.result-recording-modes = vector # for computing median

**.R?.ppp[*].**Queue.rcvdPk.result-recording-modes = count
**.R?.ppp[*].**Queue.dropPk.result-recording-modes = count
**.R?.ppp[*].**Queue.queueLength.result-recording-modes = timeavg
**.R?.ppp[*].**Queue.queueingTime.result-recording-modes = vector # for computing
median
**.R?.ppp[*].**Queue.*.scalar-recording = true

**.app[*].sentPk*.scalar-recording = true
**.app[*].rcvdPk*.scalar-recording = true
**.app[*].endToEndDelay*.scalar-recording = true

**.afQueue.*.scalar-recording = true
```


Task 02: Configure shaping and policing

Activity 01: configure shaping

Configure shaping on router R2 as in the code below.

```
[Config PolicingAndShaping]
#Shaping on Router 2
**.R2.ppp[2].egressTC.typename = "TrafficConditioner"
#**.R2.ppp[2].ppp.queue.interfaceTableModule = "^.^.^.interfaceTable"

**.R2.ppp[2].egressTC.efMeter.cir = "70%"
**.R2.ppp[2].egressTC.efMeter.cbs = 50KiB
**.R2.ppp[2].egressTC.defaultMeter.cir = "30%"
**.R2.ppp[2].egressTC.defaultMeter.cbs = 2KiB
**.R2.ppp[2].egressTC.defaultMeter.ebs = 4KiB
```

Activity 02: configure policing

Configure policing on router R4 as in the code below.

```
#Policing on Router 4
**.R4.ppp[0].ingressTC.typename = "TrafficConditioner"
#**.R4.ppp[0].ppp.queue.interfaceTableModule = "^.^.^.interfaceTable"
**.R4.ppp[0].ingressTC.efMeter.cir = "50%"
**.R4.ppp[0].ingressTC.efMeter.cbs = 40KiB
**.R4.ppp[0].ingressTC.defaultMeter.cir = "25%"
**.R4.ppp[0].ingressTC.defaultMeter.cbs = 2KiB
**.R4.ppp[0].ingressTC.defaultMeter.ebs = 4KiB
```

Activity 03: configure queuing with and without policing and shaping

Configure best effort queuing without policing and shaping as in the code below.

```
[Config Exp1DropTailWithoutPolicingAndShaping]      #Best Effort without Policing
and Shaping
#description = "Drop Tail Queueing Without Traffic Conditioning"
extends = Apps, Exp
**.ppp[*].ppp.queue.typename = "DropTailQueue"
**.eth[*].mac.queue.typename = "EtherQosQueue"
**.eth[*].mac.queue.dataQueue.typename = "DropTailQueue"

**.queue.packetCapacity = 100
```

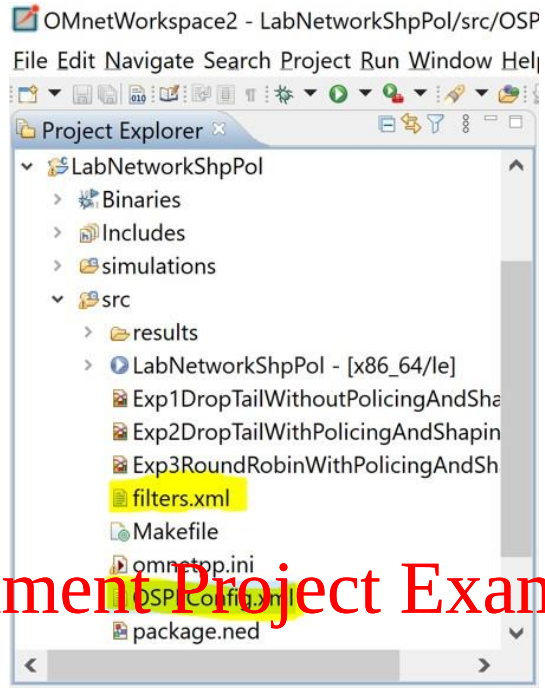

Configure best effort queuing with policing and shaping as in the code below.


```
[Config Exp2DropTailWithPolicingAndShaping]  #Best Effort with Policing and
Shaping
#description = "Drop Tail Queueing With Traffic Policing and Shaping"
extends = Apps, Exp, PolicingAndShaping
**.ppp[*].ppp.queue.typename = "DropTailQueue"
**.eth[*].mac.queue.typename = "EtherQosQueue"
**.eth[*].mac.queue.dataQueue.typename = "DropTailQueue"
**.queue.packetCapacity = 100
```

Configure round robin queuing with policing and shaping as the below code

```
[Config Exp3RoundRobinWithPolicingAndShaping] # Round Robin with Policing and
Shaping
#description = "Diffserv Queueing With Traffic Policing and Shaping"
extends = Apps, Exp, PolicingAndShaping
**.R?.ppp[*].ppp.queue.typename = "DSQueue1"  #Diffserv Queue
**.R?.ppp[*].ppp.queue.packetCapacity = 100  #-1
**.R?.ppp[*].ppp.queue.*.packetCapacity = 100
**.R?.ppp[*].ppp.queue.wrr.weights = "10 40 5 1 1"
```

Activity 04: add xml files for OSPF configuration and traffic classification as below figure



filters.xml:

```
<filters>
  <experiment id="default">
    <filter srcAddress="H1" destPort="1000" gate="0"/>
    <filter srcAddress="H1" destPort="2000" gate="1"/>
    <filter srcAddress="H1" destPort="3000" gate="2"/>
    <filter srcAddress="H2" gate="0"/>
  </experiment>
</filters>
```

OSPFConfig.xml

```
<?xml version="1.0"?>
<OSPFASConfig>

  <Area id="0.0.0.0">
    <AddressRange address="H1" mask="H1" />
    <AddressRange address="H2" mask="H2" />

    <AddressRange address="R1>R2" mask="R1>R2" />
    <AddressRange address="R2>R1" mask="R2>R1" />

    <AddressRange address="R1>R3" mask="R1>R3" />
    <AddressRange address="R3>R1" mask="R3>R1" />

    <AddressRange address="R2>R3" mask="R2>R3" />
```

```
    <AddressRange address="R3>R2" mask="R3>R2" />

    <AddressRange address="R2>R4" mask="R2>R4" />
    <AddressRange address="R4>R2" mask="R4>R2" />

    <AddressRange address="R4>R5" mask="R4>R5" />
    <AddressRange address="R5>R4" mask="R5>R4" />

    <AddressRange address="R4>R6" mask="R4>R6" />
    <AddressRange address="R6>R4" mask="R6>R4" />

    <AddressRange address="R5>R7" mask="R5>R7" />
    <AddressRange address="R7>R5" mask="R7>R5" />

    <AddressRange address="R6>R7" mask="R6>R7" />
    <AddressRange address="R7>R6" mask="R7>R6" />
  </Area>

  <Router name="**" RFC1583Compatible="true">
     <BroadcastInterface ifName='eth[*]' areaID='0.0.0.0' interfaceOutputCost='0'
/>
     <PointToPointInterface ifName='ppp[*]' areaID='0.0.0.0'
interfaceOutputCost='0' />
  </Router>

</OSPFASConfig>
```


TASK 03: run the experiments under three configurations mode

You need to run the experiment in three cases:

Case 1, with best effort queue without policing and shaping and measure the network performance

Case 2, with best effort queue with policing and shaping and measure the network performance

Case 3, with best round robin queue with policing and shaping and measure the network performance

Task 04: Displaying and Exporting Simulation Results Data

After closing the simulation of the experiments, a few files will be created in the results folder under the src package of the project (sca, vec). Double clicking one of these files will create an anf file directly in the src folder as shown in Figure 47. When you open the anf file you will have the window shown in Figure 48 with all the data collected in the Data section separated as they were collected whether as scalar data in the sca file, or as vector data in the vec file.

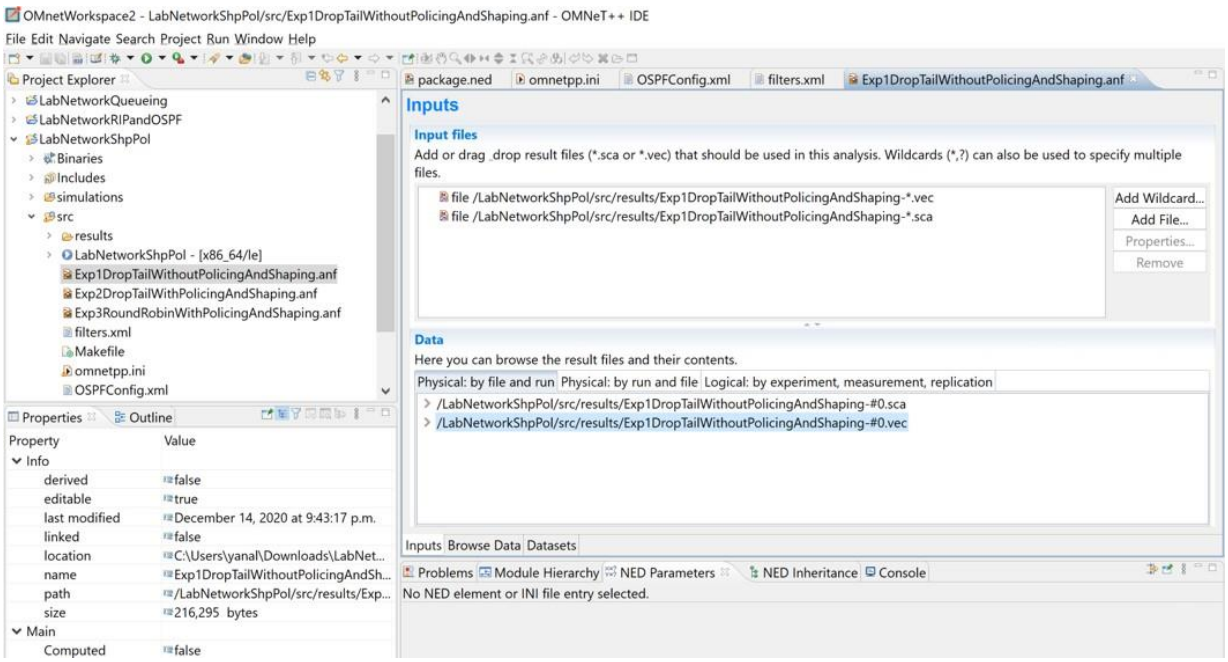


Figure 44. Simulation Results Files

For example, let us open the vec file for Exp3 the round robin with policing and shaping. After expanding the main vector called "Exp3RoundRobinWithPolicingAndShaping: #0", you will have a list with all vectors collected during the simulation as shown in Figure 48.

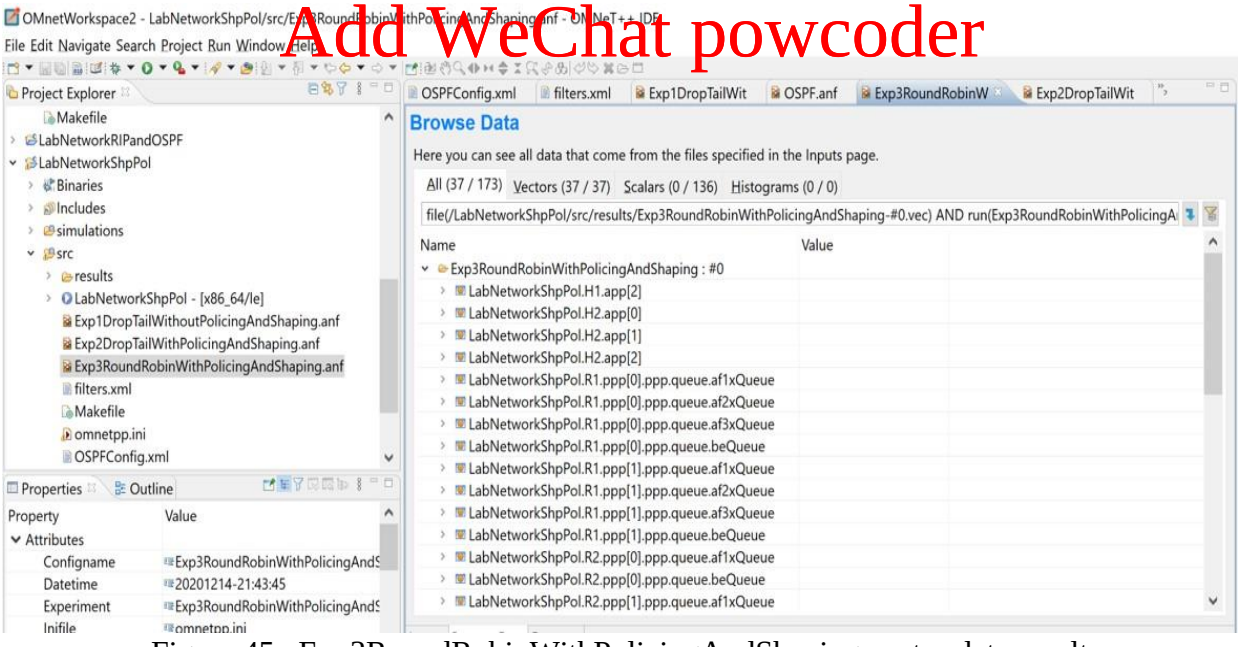


Figure 45. Exp3RoundRobinWithPolicingAndShaping vector data results

For example, Let us expand the vector of Host 2, app1 that is called LabNetworkShpPol.H2.app[1] and then expand the vector inside called “endToEndDelay” as shown in Figure 49. Now you can see metrics that are calculated for 795 packets (the Count value) received at Host 2 for application 1.

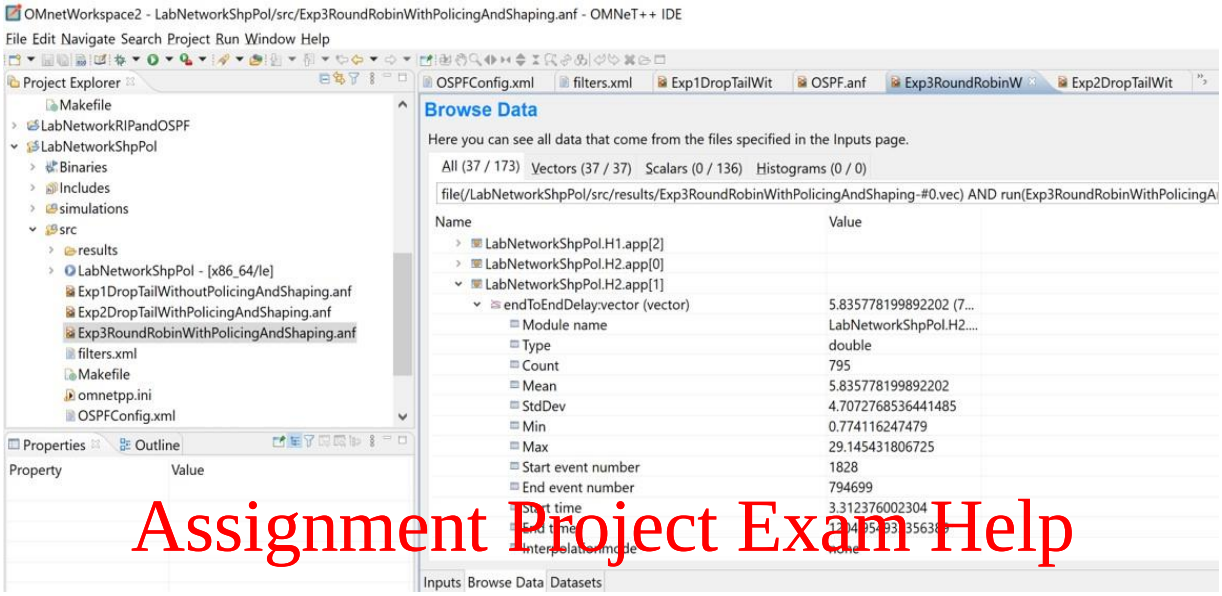


Figure 46. End to End Delay vector data of application 1

References

[1] Cisco network academy, Network Essentials free course from Cisco Network academy https://www.netacad.com/courses/networking/networking-essentials

[2] Wireshark documentation https://www.wireshark.org/docs/

[3] OMNeT++ documentation https://omnetpp.org/documentation/

[4] Microsoft administration commands https://docs.microsoft.com/en-us/windows-server/administration/windows-commands

[5] Old manual and ECE Network lab http://www.ece.concordia.ca/~netlab.

[6] OMNeT License. https://omnetpp.org/intro/license

[7] INET Framework. https://inet.omnetpp.org/Introduction.

Appendix

Abbreviation

Term	Description
VLAN	Short for ***virtual LAN***, a network of computers that behave as if they are connected to the same wire even though they may be physically located on different segments of a LAN. VLANs are configured through software rather than hardware, which make them extremely flexible. One of the biggest advantages of VLANs is that when a computer is physically moved to another location, it can stay on the same VLAN without any hardware reconfiguration.
IOS	Cisco Internetwork Operating System
CLI	Cisco Command-line Interface
OSPF	Open Shortest Path First (OSPF) protocol, defined in RFC 2328, is an Interior Gateway Protocol used to distribute routing information within a single Autonomous System
QoS	Quality of Service
FTP	Three Letter Acronym for ***F****ile* ***T****ransfer* ***P****rotocol, the* protocol for exchanging files over the Internet.
TFTP	A file transfer protocol notable for its simplicity. It is generally used for automated transfer of configuration or boot files between machines in a local environment. Compared to FTP, TFTP is extremely limited, providing no authentication, and is rarely used interactively by a user.
SNMP	"Internet-standard protocol for managing devices on IP networks." Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks, and more."[1] It is used mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention.
NTP	Network Time Protocol, a means of synchronizing clocks over a computer network

DHCP	A network protocol that is used to configure network devices so that they can communicate on an IP network.
PT	Cisco packet tracer
PDU	Protocol data unit
ICMP	Internet Control Message Protocol