PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Laboratory Manual
UE22EC351A Computer Communication Networks

By

Dr. Vamsi Krishna Tumuluru

Professor

Department of Electronics and Communications Engineering

Aug – Dec 2024

Contents

1. **Preliminary:** Introduction to Wireshark, GNS3 and Python
    - 1.1. Overview
        - 1.1.1. Concept of protocols layers
        - 1.1.2. Software tools
    - 1.2. Procedures
        - 1.2.1. Wireshark installation
        - 1.2.2. Familiarizing and running Wireshark
        - 1.2.3. GNS3 installation
        - 1.2.4. Simple network configuration using GNS3
        - 1.2.5. Python installation

List of experiments:

2. Lab 1: Introduction to Wireshark: Packet capture procedure, Filters, Analysis
    - 2.1. Objective
    - 2.2. Procedure
    - 2.3. Analyses
3. Lab 2: Analyze the downloading of embedded objects in a web-page using Wireshark
    - 3.1. Objective
    - 3.2. Procedure
    - 3.3. Analyses
4. Lab 3: Analyze the DNS query and response using Wireshark
    - 4.1. Objective
    - 4.2. Procedure
    - 4.3. Analyses
5. Lab 4: Socket programming and Wireshark analysis over LAN: UDP sockets and TCP sockets
    - 5.1. Objective
    - 5.2. Procedure
    - 5.3. Analyses
6. Lab 5: Analyze TCP connection and segmentation when downloading large file from a web-server using Wireshark
    - 6.1. Objective

6.2. Procedure

7. Lab 6: Design simple 1-hop and 2-hop networks and configure IPv4 addresses and RIP using GNS3
    7.1. Objective
    7.2. Procedure
    7.3. Analyses
8. Lab 7: Design a 1-hop network to demonstrate dynamic addressing using GNS3 and Wireshark
    8.1. Objective
    8.2. Procedure
    8.3. Analyses
9. Lab 8: Design a 2-hop network to demonstrate dynamic addressing using GNS3 and Wireshark
    9.1. Objective
    9.2. Procedure
    9.3. Analyses
10. Lab 9: Configure and analyze OSPF in a multihop network using GNS3
    10.1. Objective
    10.2. Procedure
    10.3. Analyses
11. Lab 10: Design a 2-hop network to demonstrate dynamic NAT configurations
    11.1. Objective
    11.2. Procedure
    11.3. Analyses
12. Lab 11: Design a switched LAN and analyze link layer addressing and ARP
    12.1. Objective
    12.2. Procedure
    12.3. Analyses

Chapter 1

Preliminary: Introduction to Wireshark, GNS3 and Python

The labs in this course were designed to help the students better understand the concepts that they learn in the course UE21EC351A Computer Communication Networks through hands-on experience. The experiments in this lab cover the important concepts and protocols related to the application layer, transport layer and network layer. Students may explore the link layer functionalities using open ended experiments.

This lab begins with an introduction to the concept of a protocol stack. Students are explained how the process of sending data from host to host across a communication network is parsed into various layers of the protocol stack. Students are also briefed about the role of IPv4 addresses and services provided by each layer.

Introduction to the various software used during the labs are also given.

1.1 Overview

1.1.1 Concept of protocol stack

Application developers have produced a variety of messages (e.g., email, messages containing webpage, etc.), each having its own representation and associated process or program for interpreting the messages and executing the tasks based on contents of the message (e.g., a web browser is a end-user process which runs the HTTP protocol for exchanging messages with a web server which stores webpages). Sending messages from host to host is a complex process as there is a jungle of network components (e.g., routers and switches) which have to be traversed before the message reaches the destination. A host may be running multiple application processes with each communicating with a corresponding process running on a different host on the other side of the internet (sometimes separated by thousands of miles). Further, there are millions of users which are also sending messages to their respective destination hosts via the common network components. Hence, there is a need to execute the above complex process as several subtasks where each subtask has its own functions. These subtasks are referred to as layers which are named as application layer (one which generates and formats the message), transport layer (one which multiplexes and demultiplexes the messages leaving and arriving at the hosts respectively), network layer (one which identifies the path of network components leading to the destination host), link layer (one which pushes the data onto the physical medium connecting the host with the first network component on the path to the destination) and the physical layer (one which carries the data as modulated carrier signals). At the sending host, an application message along with its header is encapsulated into a transport layer segment. The transport layer segment along with its header is encapsulated into an IP datagram. The IP datagram along with its header is encapsulated into a frame. The frame along with its header is converted into bits and the modulated carrier signal is transmitted at the physical layer. The reverse happens at the receiving host, where each layer starting with the link layer which removes its header after some pre-processing and passes the remaining payload to its upper layer. This process is repeated till the message is retrieved by the corresponding application layer process running at the receiving host.

1.1.2 Software tools

The various software tools used in this lab include Wireshark, GNS3 and Python.

Wireshark is a free network protocol analyzer that runs on Windows, Mac, and Linux /Unix computers. It's an ideal packet analyzer for our labs – it is stable, has a large user base and well documented support that includes a user-guide (available at http://www.wireshark.org/docs/wsug_html_chunked/), man pages (available at http://www.wireshark.org/docs/man-pages/), and a detailed FAQ page (available at http://www.wireshark.org/faq.html), rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers which use Ethernet, serial (PPP), 802.11 (Wi-Fi) wireless LANs, and many other link-layer technologies.

GNS3 is a very popular network emulator which can be used to virtually design computer networks. GNS3 is used by several companies such as AT&T, CISCO, Intel, etc. The configuration of the network components can also be exported to a real network. GNS3 offers a wide range of configuration commands for the network layer and link layer. A virtual network design in GNS3 can be integrated with a real network and data can be exchanged with destination hosts on the public internet via GNS3 network. Wireshark can be incorporated into GNS3 to analyze the packets exchanged between any pair of devices (e.g., routers and hosts).

Python is a powerful, simple and elegant programming tool. Various open source libraries are available online. Students will use Python 2.7 or above to write socket programs and demonstrate key application layer and transport layer concepts.

1.2 Procedures

1.2.1 Wireshark installation

To download and install the Wireshark go to http://www.wireshark.org/download.html and download and install the Wireshark binary for your computer. In order to run Wireshark, you'll need to have access to a computer that supports both Wireshark and the *libpcap* or *WinPCap* packet capture library. The *libpcap* software will be installed for you, if it is not installed within your operating system, when you install Wireshark. For Linux installation see the end of the chapter.

1.2.2 Familiarizing and running Wireshark

Upon running Wireshark the following window is displayed. The available network adapters can be observed in Fig. 1 under *Interface list*. The network adapter which is connected to the internet is chosen and the start button (displayed as green shark fin) is chosen. A web browser is opened and any url (e.g., http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html) is entered. Wireshark would start displaying various packets exchanged over the network adapter. Three sub-windows appear within the main window (top-down), namely, *packets listing window* which shows the various packets exchanged along with their labels, source and destination IP addresses (or host names), epoch time and size of the packet, *packet headers window* which shows the content of any packet selected in the packets listing window, such as protocol headers arranged according to the protocol stack and the application layer message (if any), and the *packet content window* which shows the raw data corresponding to the packet chosen. The various sub-windows can be observed from Fig. 2.

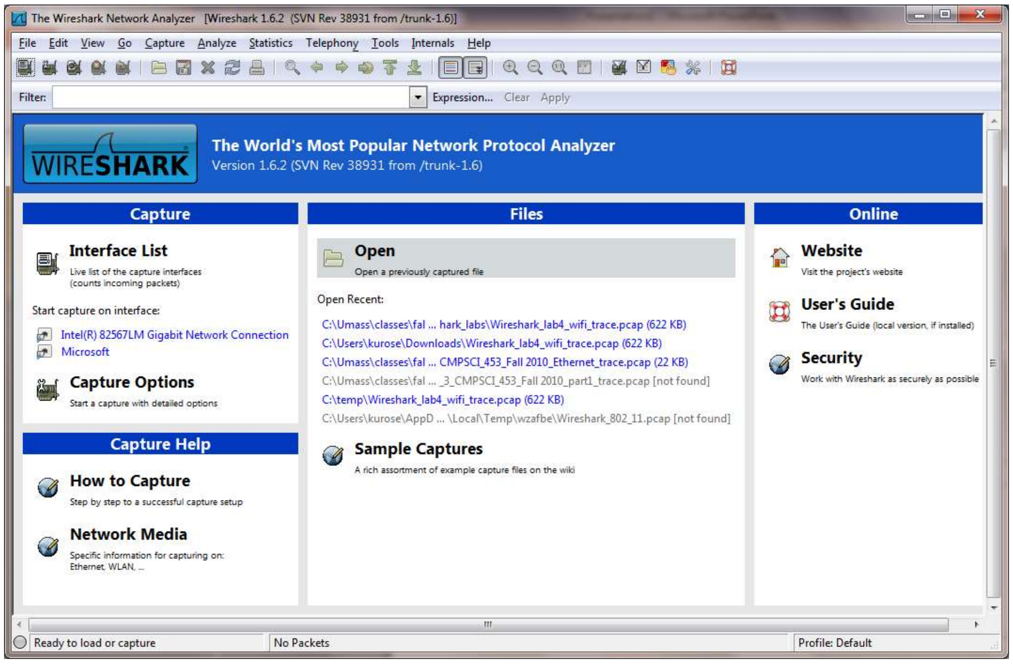
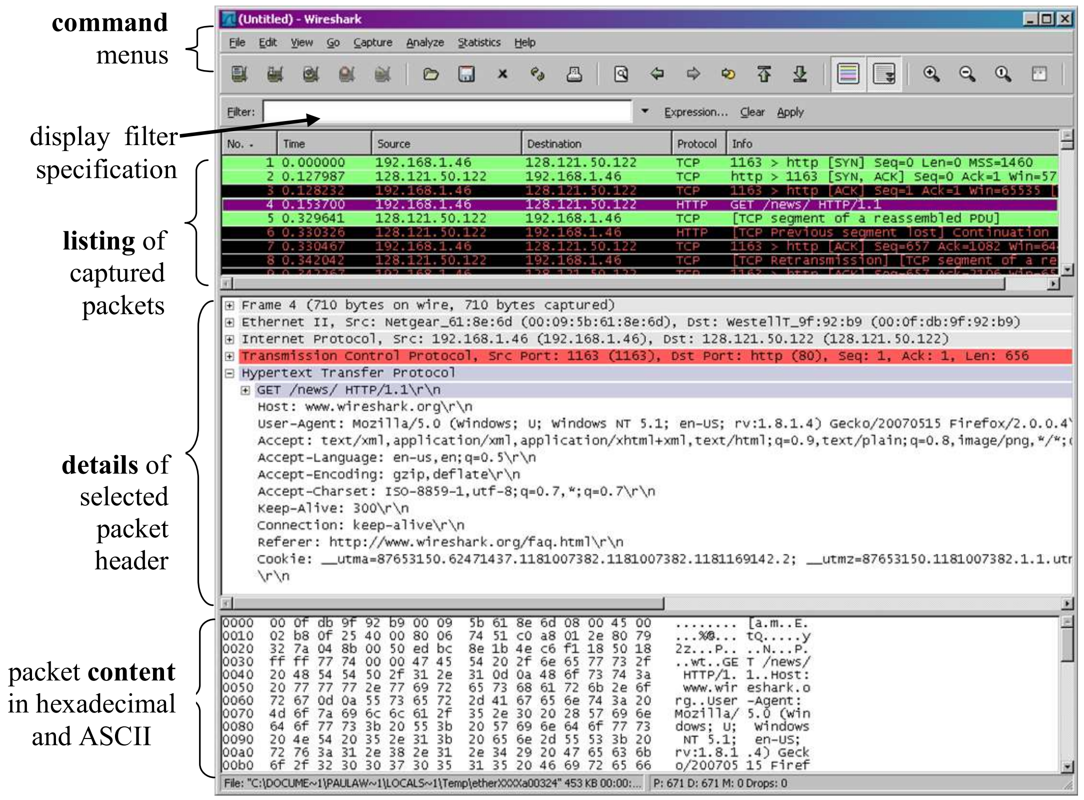

The Wireshark Network Analyzer [Wireshark 1.6.2 (SVN Rev 38931 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Version 1.6.2 (SVN Rev 38931 from /trunk-1.6)
Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start capture on interface:
Intel(R) 82567LM Gigabit Network Connection
Microsoft
Capture Options
Start a capture with detailed options
Capture Help
How to Capture
Step by step to a successful capture setup
Network Media
Specific information for capturing on: Ethernet, WLAN, ...
Files
Open
Open a previously captured file
Open Recent:
C:\Umass\classes\fal ... hark_labs\Wireshark_lab4_wifi_trace.pcap (622 KB)
C:\Users\kurose\Downloads\Wireshark_lab4_wifi_trace.pcap (622 KB)
C:\Umass\classes\fal ... CMPSCI_453_Fall 2010_Ethernet_trace.pcap (22 KB)
C:\Umass\classes\fal ... _3_CMPSCI_453_Fall 2010_part1_trace.pcap [not found]
C:\temp\Wireshark_lab4_wifi_trace.pcap (622 KB)
C:\Users\kurose\AppD ... \Local\Temp\wzafbe\Wireshark_802_11.pcap [not found]
Sample Captures
A rich assortment of example capture files on the wiki
Online
Website
Visit the project's website
User's Guide
The User's Guide (local version, if installed)
Security
Work with Wireshark as securely as possible
Ready to load or capture
No Packets
Profile: Default

Fig. 1: Initial Wireshark Screen

Packets can also be chosen according to the protocol type by entering the packet type in the search bar. For example, enter *http* into the search bar and only packets containing HTTP messages are displayed in the packets listing window. Selecting the HTTP GET message will display the HTTP message along with the HTTP header. Besides, even the transport layer header, datagram header (i.e., network layer header), link layer header can be observed as in Fig. 3.


command menus
display filter specification
listing of captured packets
details of selected packet header
packet content in hexadecimal and ASCII
(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

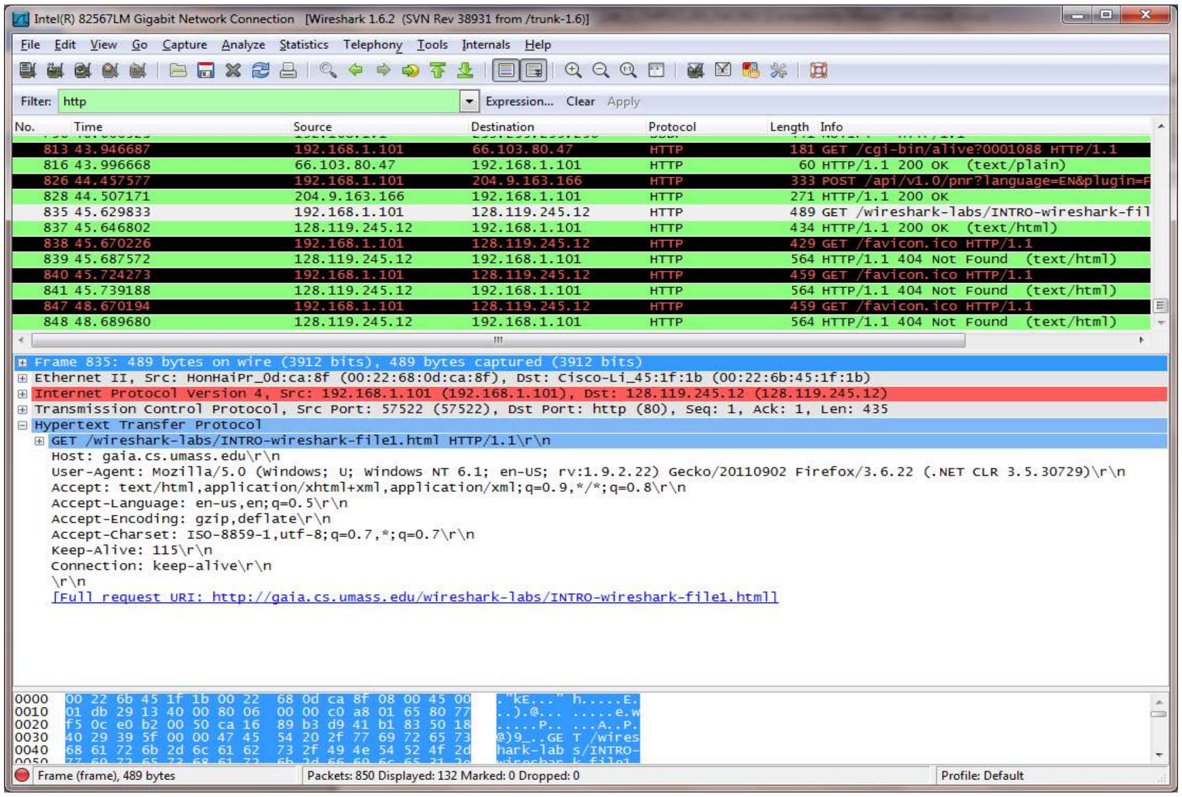
```
No. -  Time      Source          Destination     Protocol  Info
     1 0.000000  192.168.1.46    128.121.50.122  TCP       1163 > http [SYN] Seq=0 Len=0 MSS=1460
     2 0.127987  128.121.50.122  192.168.1.46    TCP       http > 1163 [SYN, ACK] Seq=0 Ack=1 Win=57
     3 0.128232  192.168.1.46    128.121.50.122  TCP       1163 > http [ACK] Seq=1 Ack=1 Win=65535 [
     4 0.153700  192.168.1.46    128.121.50.122  HTTP      GET /news/ HTTP/1.1
     5 0.329641  128.121.50.122  192.168.1.46    TCP       [TCP segment of a reassembled PDU]
     6 0.330326  128.121.50.122  192.168.1.46    HTTP      [TCP Previous segment lost] Continuation
     7 0.330467  192.168.1.46    128.121.50.122  TCP       1163 > http [ACK] Seq=657 Ack=1082 Win=64
     8 0.342042  128.121.50.122  192.168.1.46    TCP       [TCP Retransmission] [TCP segment of a re
```

```
⊞ Frame 4 (710 bytes on wire, 710 bytes captured)
⊞ Ethernet II, Src: Netgear_61:8e:6d (00:09:5b:61:8e:6d), Dst: WestellT_9f:92:b9 (00:0f:db:9f:92:b9)
⊞ Internet Protocol, Src: 192.168.1.46 (192.168.1.46), Dst: 128.121.50.122 (128.121.50.122)
⊞ Transmission Control Protocol, Src Port: 1163 (1163), Dst Port: http (80), Seq: 1, Ack: 1, Len: 656
⊟ Hypertext Transfer Protocol
  ⊞ GET /news/ HTTP/1.1\r\n
    Host: www.wireshark.org\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4
    Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Referer: http://www.wireshark.org/faq.html\r\n
    Cookie: __utma=87653150.62471437.1181007382.1181007382.1181169142.2; __utmz=87653150.1181007382.1.1.utr
    \r\n
```

```
0000  00 0f db 9f 92 b9 00 09  5b 61 8e 6d 08 00 45 00   ........ [a.m..E.
0010  02 b8 0f 25 40 00 80 06  74 51 c0 a8 01 2e 80 79   ...%@... tQ.....y
0020  32 7a 04 8b 00 50 ed bc  8e 1b 4e c6 f1 18 50 18   2z...P.. ..N...P.
0030  ff ff 77 74 00 00 47 45  54 20 2f 6e 65 77 73 2f   ..wt..GE T /news/
0040  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a    HTTP/1. 1..Host:
0050  20 77 77 77 2e 77 69 72  65 73 68 61 72 6b 2e 6f    www.wir eshark.o
0060  72 67 0d 0a 55 73 65 72  2d 41 67 65 6e 74 3a 20   rg..User -Agent:
0070  4d 6f 7a 69 6c 6c 61 2f  35 2e 30 20 28 57 69 6e   Mozilla/ 5.0 (Win
0080  64 6f 77 73 3b 20 55 3b  20 57 69 6e 64 6f 77 73   dows; U;  Windows
0090  20 4e 54 20 35 2e 31 3b  20 65 6e 2d 55 53 3b 20    NT 5.1;  en-US;
00a0  72 76 3a 31 2e 38 2e 31  2e 34 29 20 47 65 63 6b   rv:1.8.1 .4) Geck
00b0  6f 2f 32 30 30 37 30 35  31 35 20 46 69 72 65 66   o/200705 15 Firef
```

```
File: "C:\DOCUME~1\PAULAW~1\LOCALS~1\Temp\etherXXXXa00324" 453 KB 00:00:...   P: 671 D: 671 M: 0 Drops: 0
```

Fig. 2: Wireshark Graphical User Interface, during packet capture and analysis


Intel(R) 82567LM Gigabit Network Connection [Wireshark 1.6.2 (SVN Rev 38931 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http
Expression... Clear Apply

```
No.  Time       Source          Destination     Protocol  Length Info
813 43.946687   192.168.1.101   66.103.80.47    HTTP      181 GET /cgi-bin/alive?0001088 HTTP/1.1
816 43.996668   66.103.80.47    192.168.1.101   HTTP       60 HTTP/1.1 200 OK  (text/plain)
826 44.457577   192.168.1.101   204.9.163.166   HTTP      333 POST /api/v1.0/pnr?language=EN&plugin=F
828 44.507171   204.9.163.166   192.168.1.101   HTTP      271 HTTP/1.1 200 OK
835 45.629833   192.168.1.101   128.119.245.12  HTTP      489 GET /wireshark-labs/INTRO-wireshark-fil
837 45.646802   128.119.245.12  192.168.1.101   HTTP      434 HTTP/1.1 200 OK  (text/html)
838 45.670226   192.168.1.101   128.119.245.12  HTTP      429 GET /favicon.ico HTTP/1.1
839 45.687572   128.119.245.12  192.168.1.101   HTTP      564 HTTP/1.1 404 Not Found  (text/html)
840 45.724273   192.168.1.101   128.119.245.12  HTTP      459 GET /favicon.ico HTTP/1.1
841 45.739188   128.119.245.12  192.168.1.101   HTTP      564 HTTP/1.1 404 Not Found  (text/html)
847 48.670194   192.168.1.101   128.119.245.12  HTTP      459 GET /favicon.ico HTTP/1.1
848 48.689680   128.119.245.12  192.168.1.101   HTTP      564 HTTP/1.1 404 Not Found  (text/html)
```

```
⊞ Frame 835: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
⊞ Ethernet II, Src: HonHaiPr_0d:ca:8f (00:22:68:0d:ca:8f), Dst: Cisco-Li_45:1f:1b (00:22:6b:45:1f:1b)
⊞ Internet Protocol Version 4, Src: 192.168.1.101 (192.168.1.101), Dst: 128.119.245.12 (128.119.245.12)
⊞ Transmission Control Protocol, Src Port: 57522 (57522), Dst Port: http (80), Seq: 1, Ack: 1, Len: 435
⊟ Hypertext Transfer Protocol
  ⊞ GET /wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.22) Gecko/20110902 Firefox/3.6.22 (.NET CLR 3.5.30729)\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 115\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html]
```

```
0000  00 22 6b 45 1f 1b 00 22  68 0d ca 8f 08 00 45 00   ."kE..." h.....E.
0010  01 db 29 13 40 00 80 06  00 00 c0 a8 01 65 80 77   ..).@... .....e.w
0020  f5 0c e0 b2 00 50 ca 16  89 b3 d9 41 b1 83 50 18   .....P.. ...A..P.
0030  40 29 39 5f 00 00 47 45  54 20 2f 77 69 72 65 73   @)9_..GE T /wires
0040  68 61 72 6b 2d 6c 61 62  73 2f 49 4e 54 52 4f 2d   hark-lab s/INTRO-
```

```
Frame (frame), 489 bytes   Packets: 850 Displayed: 132 Marked: 0 Dropped: 0   Profile: Default
```

Fig. 3: Wireshark window upon entering http in search bar

1.2.3 GNS3 installation

GNS3 1.1.3 is downloaded from www.gns3.com and installed by selecting the exe file. Select GNS3, WinPCAP, Wireshark, Dynamips, VPCS during the installation (in the choose components window) and follow the instructions.

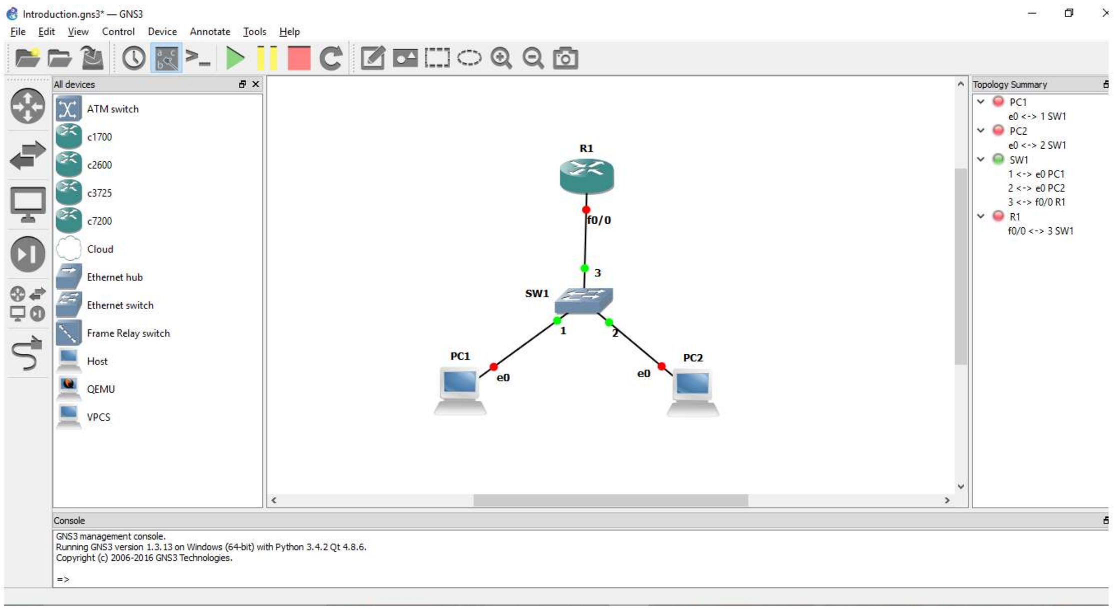
Download the ios images for CISCO routers c7200 and c3725 from the internet by searching in google **or** in the link https://mega.nz/folder/nJR3BTjJ#N5wZsncqDkdKyFQLELU1wQ or from the google drive link https://drive.google.com/drive/folders/18Aqx2n6XlNTc-ajxp7ThigRCMc1ac4u5?usp=sharing.

Once either .image or .bin file is downloaded, install them by following the instructions as given in https://www.computernetworkingnotes.com/ccna-study-guide/how-to-add-install-or-import-ios-in-gns3.html **or** Start GNS3 and select the Edit from the Menu and then select Preferences. Choose IOS routers. Add a new router IOS by locating it in the PC.

After adding the router, click on the Ok button. For Linux installation see the end of the chapter.

1.2.4 Simple network configuration using GNS3

Draw the following simple network by dragging and dropping the components from the left pane as shown in Fig. 4. Right click on the router and select configure. Select slots, choose slot 2 and add NM-4T to add 4 serial ports to the router (will be used in later experiments).



PC1
e0 <-> 1 SW1
PC2
e0 <-> 2 SW1
SW1
1 <-> e0 PC1
2 <-> e0 PC2
3 <-> f0/0 R1
R1
f0/0 <-> 3 SW1

```
GNS3 management console.
Running GNS3 version 1.3.13 on Windows (64-bit) with Python 3.4.2 Qt 4.8.6.
Copyright (c) 2006-2016 GNS3 Technologies.

=>
```

Fig. 4: Simple network in GNS3

Turn on all devices by clicking on the play button. Right click on the router and select Idle_PC, thereafter, choose any value with an asterisk to set the resources for the virtual router. Type the following commands for configuring the router (c3725 router).

```
R1#configure terminal
R1(config)#interface f0/1
R1(config-if)#ip address 10.0.0.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
```

Type the command PC1> ip 10.0.0.2/24 10.0.0.1 in PC1 by double clicking it. Similarly, type PC1> ip 10.0.0.3/24 10.0.0.1 in PC2 by double clicking it. Next, type the following command in PC1 to display the ping statistics. Observe whether ping was successful. Ping is used to check if a destination device (clients/router) are available.

```
PC1> ping 10.0.0.3
```

Right click on any port and select Start capture. Repeat ping operation and observe the packets exchanged.

1.2.5 Python installation

Python installation can be done by downloading Python 2.7 or above from www.python.org. After installing python, open Python IDLE from which you can choose the editor for writing python programs with the .py.

Alternatively, you can install Sypder 3 or above from https://www.anaconda.com/download/

Linux installation

1. Open Ubuntu software center and install Spyder 3
2. To install GNS 3:
    a. Open terminal and type the following
    b. sudo add-apt-repository ppa:gns3/ppa
    c. sudo apt-get update
    d. sudo apt-get install dynamips gns3
3. To import router C3725 open https://www.sysnettechsolutions.com/en/gns3/gns3-supported-ios-images-download/. Download c3725-adventerprisek 9-mz.124-15.T14. After extracting the file, open GNS3 and add ios image by right clicking on the router images.
4. To enable Wireshark capture type the following in the terminal
    a. sudo dpkg-reconfiugure wireshark-common
    b. select yes
    c. sudo chmod 777 /usr/bin/dumpcap

Chapter 2

Lab 1: Introduction to Wireshark: Packet capture procedure, Filters, Analysis

2.1 Objective

To explore aspects such as packet capture procedure, filters and analysis that can be done using Wireshark. We will analyze delay and throughput when the packets are transmitted by the hosts. In this regard, we will invoke the ping and tracert commands.

2.2 Procedure

Do the following:

1. Start up the Wireshark packet sniffer, as described in the Preliminary chapter (but don't yet begin packet capture).
2. Wait a bit more than one minute, and then begin Wireshark packet capture.
3. Open the terminal window and perform the following steps
    a. Type the command "ipconfig /all" (without the quotation marks) and IPv4 address (that is the IP address assigned to the Ethernet interface of your PC).
    b. Type "ping 8.8.8.8" (this is Google's DNS server; more on DNS later)
    c. Type "tracert 8.8.8.8" (this performs several ping operations with different TTL values till the destination 8.8.8.8 is reached)
4. Stop Wireshark and type "icmp" (without the quotation marks) into the display filter and press the enter key
5. You should find the ICMP packets Echo (ping) request and Echo (ping) reply. The former packet is sent from your terminal while the later packet is received by your terminal.
6. Now analyze the throughput and delay during your experiment.

2.3 Analyses

1. How many Echo (ping) request and Echo (ping) reply packets were observed?
2. What were the sizes of Echo (ping) request and Echo (ping) reply packets?
3. Inspect the Echo (ping) reply and find out the response time (aka round trip time).
4. What is the type and code observed in the Echo (ping) request and Echo (ping) reply packets?

5. When you performed the “tracert 8.8.8.8” command, how many intermediate nodes identified TTL expiry? Write their IP addresses. What response did they send in when they observed TTL expiry? Specify the type and code in the responses.

6. Calculate the throughput experienced by your computer across all the ICMP packets. To calculate throughput you must add all the packet lengths and divide the sum by the difference of the time of receiving the last Echo (ping) reply and the time of sending the first Echo (ping) request.

Provide screenshots of the packet-listing window displaying the ICMP packets. Try the command “ip.src == 8.8.8.8” or “ip.addr == 8.8.8.8” in the display filter to know the packets sent from the IP address 8.8.8.8

Chapter 3

Lab 2: Analyze the downloading of embedded objects in a web-page using Wireshark

3.1 Objective

To see how a webpage having embedded objects (image files) are downloaded. Understand the format of the HTTP messages exchanged between the client and the server. We will analyze the throughput and delay across a persistent connection.

3.2 Procedure

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above. Start up the Wireshark packet sniffer.
2. Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html
3. Your browser should display a short HTML file with two images. These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. Your browser will use the persistent TCP connection to retrieve the Publisher's logo from "gaia.cs.umas.edu". However, for the book logo, your browser makes a secure connection (https) to "kurose.cslash.net" and downloads it. Notice that for unsecure connections (http) the server port is 80 whereas for secure connections (https) the server port is 443.
4. Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.
5. You can filter "ip.src == 178.79.137.164" to see the packets (TCP in protocol column) which delivered the book logo.

3.3 Analyses

1. How many HTTP GET request messages were sent by your browser (excluding the object favicon.ico)? To which Internet addresses were these GET requests sent?
2. Fill the following table:

Object	**Source IP Address**	**Destination IP Address**	**Source Port**	**Destination Port**	**Transport Layer Protocol**
HTTP-wireshark-					

file4.html					
pearson.png					
8E_cover_small.jpg					

3. How many hosts were responding to your browser? What are the host names?
4. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.
5. What is the total delay incurred in downloading the webpage along with the embedded objects?
6. What was the throughput when downloading the publisher's logo?
7. Now inspect the HTTP GET message corresponding to the object "HTTP-wireshark-file4.html". What is the server name? How do we know it is persistent connection?
8. Now inspect the HTTP response message corresponding to the object "HTTP-wireshark-file4.html". What is the time at which the object was retrieved by the server? When did this version of the object become available on the server?
9. Now inspect the HTTP messages corresponding to the object "pearson.png". What is name of the server to which the GET message was sent? What is size of the object returned by the server? How could you tell?
10. What were the status and the code returned by the server when the HTTP GET message was sent by the browser to fetch the object "8E_cover_small.jpg"?

Chapter 4

Lab 3: Analyze the DNS query and response using Wireshark

4.1 Objective

Your experiment will be conducted in four parts. First, you will query for the IP address of the given host name. Second, you will query for the canonical host name of the given host name (it may be the mnemonic host name). Third, you will query for the authoritative DNS servers' host name of the given host name. And finally, you will query for the mail server's canonical host name using the given host name.

4.2 Procedure

1. <u>Windows:</u> Open command prompt and type *ipconfig /all* to determine the local DNS IP address and your host IP address. <u>Ubuntu:</u> In terminal, type *nmcli dev show enp2s0*
2. <u>Windows:</u> To view the DNS records stored in your system, type *ipconfig /displaydns*. And to clear all DNS records, type *ipconfig /flushdns*
3. Open and start Wireshark. Issue commands to query DNS records from command prompt. Save the Wireshark files after the DNS response for packet analysis. Repeat this step for each of the four types of queries.
4. For querying IP address of *ieeexplore.ieee.org*, type *nslookup –type=A ieeexplore.ieee.org* in command prompt.
5. For querying canonical hostname of *ieeexplore.ieee.org*, type *nslookup –type=CNAME www.ieee.org* in command prompt.
6. For querying the names of authoritative DNS servers of *www.pes.edu*, type *nslookup –type=NS www.pes.edu* in command prompt.
7. For querying the mail server alias host names of *mail.google.com*, type *nslookup –type=MX mail.google.com* in command prompt.

4.3 Analyses

1. What was the transport protocol used for DNS queries? Give source and destination port numbers for each query in the experiment.
2. Under the Type A query, what is the length of the DNS query message? (Hint: use UDP length, UDP header size and DNS header size)
3. What is the IP address returned for the Type A query? What was the TTL value? How many answers were returned?

4. What is the canonical name returned for the Type CNAME query? What was the TTL value? What was the additional information observed in the response?
5. What is the authoritative DNS servers' name returned for the Type NS query? What was the TTL value? Was any additional records observed in the response?
6. What is the canonical host name returned for the Type MX query? What was the TTL value? What was the name of the authoritative DNS server observed in the response?
7. Given that the DNS query and response use the same header format, how can you tell the difference between a DNS query and response?
8. Did the client request for recursive query? How can you tell?
9. Did the DNS server support recursive query? How can you tell?
10. What is the throughput under Type NS combining the DNS query and its response?

Chapter 5

Lab 4: Write and analyze socket programs using Python

5.1 Objective

Demonstrate socket programming using Python and analyze the headers of the UDP and TCP segments in Wireshark. The client sends a lowercase text to the server. The server converts the sentence into uppercase and returns it to the client.

- In part 1, show the exchange of text messages between a client and a server using UDP sockets.
- In part 2, show the exchange of text messages between a client and a server using TCP sockets.

5.2 Procedure

1. Students will write the UDP client program in one computer and the UDP server program in another computer. The IP address and the port number used in the server program must also be provided in the UDP client program

UDP client program

```
from socket import *
serv_addr = "IP address of server"
serv_port = 8000
client_sock = socket(AF_INET, SOCK_DGRAM)
msg = input("Enter the text message: ")
client_sock.sendto(msg.encode(), (serv_addr, serv_port))
mod_msg, s = client_sock.recvfrom(2048)
print("From Server: ", mod_msg.decode())
```

UDP server program

```
from socket import *
serv_addr = "IP address of server"
serv_port = 8000
serv_sock = socket(AF_INET, SOCK_DGRAM)
serv_sock.bind(serv_addr, serv_port)
print("The server is ready to receive")
while 1:
        msg, client_addr  = serv_sock.recvfrom(2048)
        print("Got message from", client_addr)
        mod_msg = msg.upper()
        serv_sock.sendto(mod_msg, client_addr)
```

2. Run the UDP server program. Run Wireshark on the Ethernet connection. Run the UDP client program. Save the packet capture file for analysis.

3. Students will write the TCP client program in one computer and the TCP server program in another computer. The IP address and the port number used in the server program must also be provided in the TCP client program

   TCP client program

   ```
   from socket import *
   serv_addr = "IP address of server"
   serv_port = 8000
   client_sock = socket(AF_INET, SOCK_STREAM)
   client_sock.connect(serv_addr, serv_port)
   msg = input("Enter the text message: ")
   client_sock.send(msg.encode())
   mod_msg  = client_sock.recv(2048)
   print("From Server: ", mod_msg.decode())
   client_sock.close()
   ```

   TCP server program

   ```
   from socket import *
   serv_addr = "IP address of server"
   serv_port = 8000
   serv_sock = socket(AF_INET, SOCK_STREAM)
   serv_sock.bind(serv_addr, serv_port)
   serv_sock.listen(1)
   print("The server is ready to receive")
   while 1:
           conn_sock, client_addr  = serv_sock.accept()
           print("Got connection from", client_addr)
           msg  = conn_sock.recv(2048)
           mod_msg = msg.upper()
           conn_sock.send(mod_msg)
           conn_sock.close()
   ```

4. Run the TCP server program. Run Wireshark on the Ethernet connection. Run the TCP client program. Save the packet capture file.

5.3 Analysis

1. What is the client port number in the UDP segment? What is the value of the Length field?
2. What is the value of the checksum in the UDP segment sent by the client? What is the value of the checksum in the UDP segment sent by the server?
3. What are the TCP segments observed for opening the TCP connection? What are the flags set in these segments?
4. What is the source port number in the TCP segment sent by the client?
5. What is the value of the Header Length field?
6. What are the TCP segments observed for closing the TCP connection? What are the flags set in these segments?
7. What are the flags set in the remaining TCP segments which carry the data?
8. What is the round trip time observed by the client during the TCP connection?

Chapter 6

Lab 5: Analyze TCP connection and segmentation when downloading large file from a web-server using Wireshark

6.1 Objective

We download a large object from a web server and analyze the following: TCP connection establishment, message segmentation, usage of sequence numbers and acknowledgement numbers and TCP connection closing. We want to see how the TCP segments are reassembled at the client to display the whole webpage.

6.2 Procedure

1 Start up your web browser, and make sure your browser's cache is cleared.

2 Start up the Wireshark packet sniffer

3 Enter the following URL into your browser http://gaia.cs.umass.edu/wiresharklabs/alice.txt.

4 Your browser should display the rather lengthy Alice's Adventures In Wonderland

5 Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

6 Note the source port number (e.g., xxxx) used by the host to send the HTTP GET message. Then enter "tcp.port == xxxx" in the display-filter-specification window, so that the entire TCP session is displayed.

6.3 Analyses

Answer the following questions:

1. Fill the following table:

Frame Type	Frame No	Source Port No	Destination Port No
SYN			
SYNACK			
ACK			
HTTP Request			
First segment of object			
Last segment of			

object			
FIN			
ACK for FIN			

2. What are the sizes of the TCP header in the SYN and SYNACK segments? How does it compare to the TCP header size of the data-carrying TCP segments?
3. Identify the TCP segment which carried the HTTP GET message. Write the corresponding relative sequence number and the actual sequence number. [Note: Actual segments can be viewed by right clicking on sequence number, selecting protocol preferences and un-ticking the relative sequence numbers]
4. What are the actual sequence number and acknowledgement numbers of the segments used in the connection establishment?
5. What are the actual sequence number and acknowledgement numbers of the segments used in the connection closing?
6. How much time did it take to fully download the requested object? What is the throughput of the TCP session?
7. Plot the Round Trip Time Graph. Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the gaia.cs.umass.edu server. Then select: Statistics→TCP Stream Graph→Round Trip Time Graph.

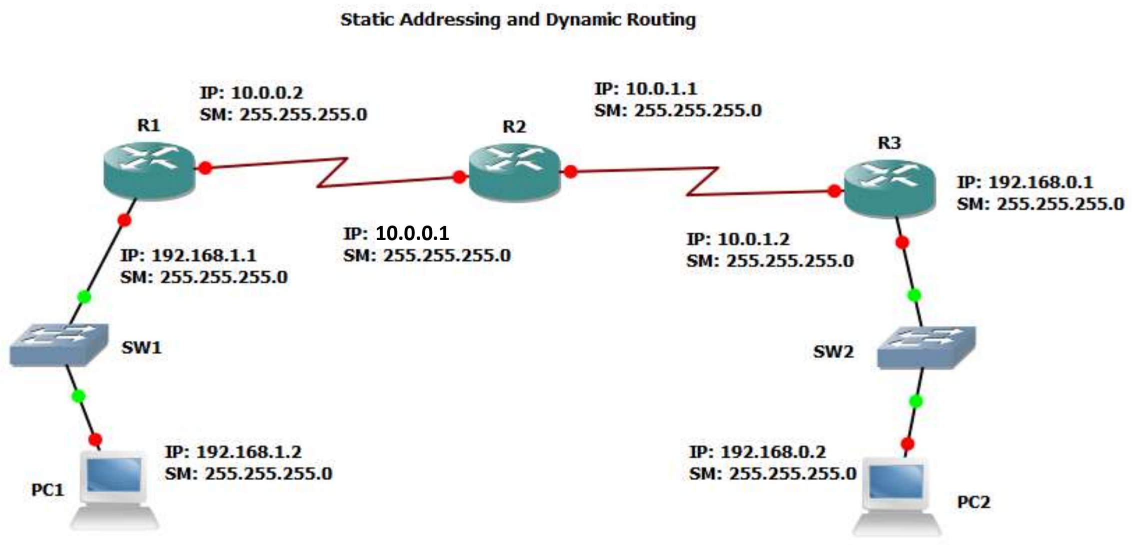
Provide screenshot of the RTT graph. Provide screenshot of the packet listing window showing SYN, SYNACK and ACK segments. Provide screenshot of the packet listing window showing the segments meant for closing the TCP connection.

Chapter 7

Lab 6: Design simple 1-hop and 2-hop networks and configure IPv4 addresses and RIP using GNS3

7.1 Objective

Design a simple 1-hop and 2-hop networks to demonstrate static addressing and dynamic routing using GNS3. For dynamic routing we use the simple RIP protocol.



7.2 Procedure

1. Configure the router interfaces as shown in the Section 1.2.4. For serial interface choose the labels as s1/0, s1/1 and so on. Example for R1 given below.

```
R1# configure terminal
R1(config)# interface s1/0
R1(config-if)# ip address 10.0.0.2 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# exit

R1(config)# interface f0/0
R1(config-if)# ip address 192.168.1.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# exit
```

2. For dynamic routing write the sample code based on the subnets directly connected to a router as follows. Example for R2

```
R2(config)# router rip
R2(config-router)# version 2
R2(config-router)# network 10.0.0.0
R2(config-router)# network 10.0.1.0
```

```
R2(config-router)# end
```

3. For PC assign IP address as shown in Section 1.2.4. You can add as many switches to SW1 and SW2. You can also add as many PCs as you want. Example for PC1

   ```
   PC1> 192.168.1.2/24 192.168.1.1
   ```

To make a 1 hop network simply reduce the number of active routers in the network to 1. Make only one router interface connected to the switch active. In this manner, you can vary the number of hops (i.e., routers) that your packet passes through. **The above diagram allows up to 3-hops.**

7.3 Analyses

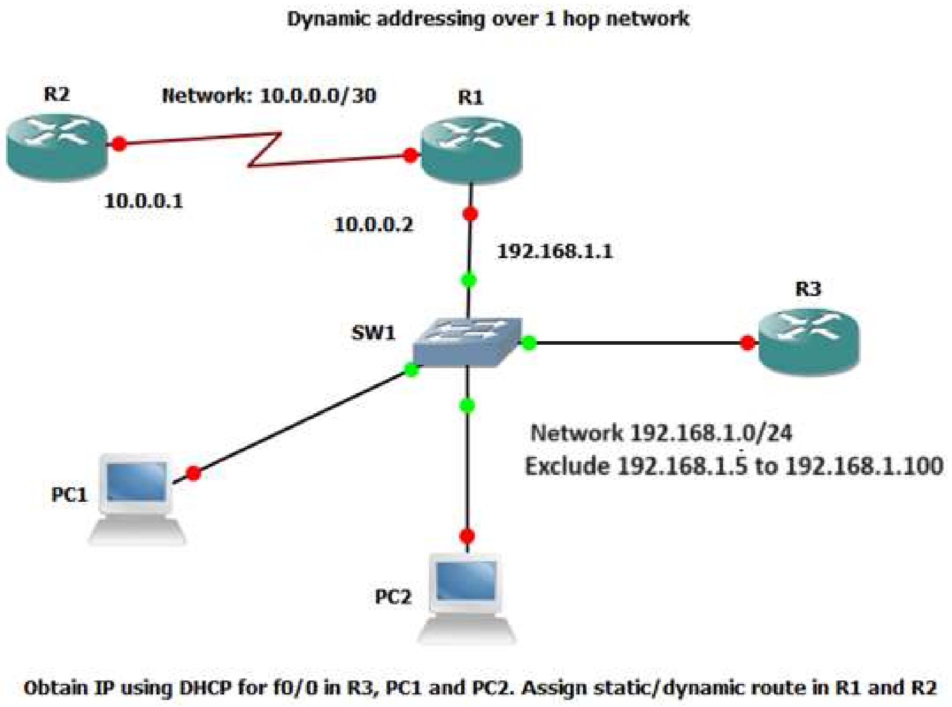
1. Suppose subnet masks such as 255.255.255.252, 255.255.255.248, 255.255.255.240, 255.255.255.224 or 255.255.255.192 are used in the first subnet. What are the possible range of IP addresses that can be assigned to the Fast Ethernet interface of R1 and PC1? Tabulate your result. Test a few cases.
2. Show the IP addresses of the active interfaces of R1, R2 and R3 in the router's terminal
3. Show the routing tables in R1, R2 and R3 in the router's terminal
4. Run Wireshark on the serial interface of R1 and show the screenshot of packet capture.
5. Show the ping operation by pinging PC2 from PC1. Show packet capture and write port numbers, IP addresses of each Echo request and reply. Explain ping statistics.
6. Show ping operation over TCP by pinging PC2 from PC1. Show packet capture and write port numbers, IP addresses of each Echo request and Echo reply.

Chapter 8

Lab 7: Design a 1-hop network to demonstrate dynamic addressing using GNS3 and Wireshark

8.1 Objective

Design a simple 1-hop network to demonstrate dynamic addressing and dynamic routing using GNS3. Configure a DHCP server in R1 with address pool 192.168.1.0 /24 and configure R3 as a DHCP client. Assign addresses using DHCP to the interfaces in PC1, PC2 and R3.


Dynamic addressing over 1 hop network
R2
Network: 10.0.0.0/30
R1
10.0.0.1
10.0.0.2
192.168.1.1
R3
SW1
Network 192.168.1.0/24
Exclude 192.168.1.5 to 192.168.1.100
PC1
PC2
Obtain IP using DHCP for f0/0 in R3, PC1 and PC2. Assign static/dynamic route in R1 and R2

8.2 Procedure

1. Configure the router interfaces of R1 and R2 as shown in the Section 1.2.4.

2. For dynamic routing configure as shown in Section 8.2.

3. For configuring DHCP server in PC1:

```
R1(config)# ip dhcp pool POOL_A
R1(dhcp-config)# ip dhcp excluded-address 192.168.1.5 192.168.1.100
R1(config)# ip dhcp pool POOL_A
R1(dhcp-config)# dns-server 192.168.1.1
R1(dhcp-config)# default-router 192.168.1.1
R1(dhcp-config)# lease 3
R1(dhcp-config)# network 192.168.1.0 /24
```

```
R1(dhcp-config)# import all
```

4. To configure R3 as a DHCP client and obtain IP address for f0/0

```
R3(config)# interface f0/0
R3(config-if)#ip dhcp client client-id ascii My_name
R3(config-if)#ip address dhcp
```

5. For PC assign IP address using DHCP

```
PC1> ip dhcp -d
```

8.3 Analyses

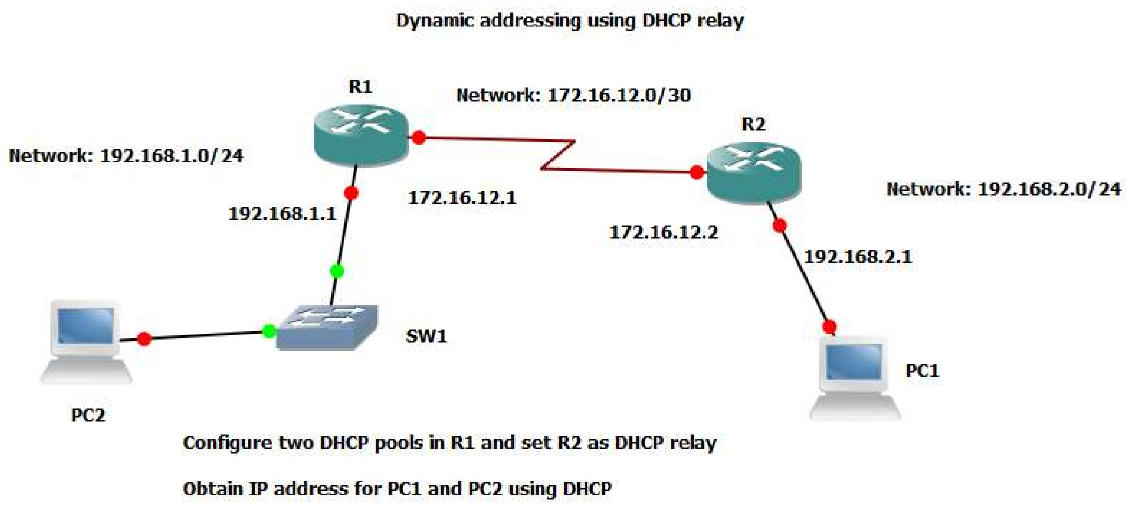
1. Show the IP addresses assigned via DHCP in the router R1's terminal window
2. Show the routing tables in R1 and R2 in the router terminal windows
3. Show DHCP server statistics in the router R1's terminal window
4. Show DHCP server's pool information in the router R1's terminal window
5. Run Wireshark on PC1 and analyze the packets exchanged between PC1 and the DHCP server when obtaining IP address. Write port numbers, IP address and MAC address for each packet observed.
6. Run Wireshark on PC1 to show the packets exchange when PC2 is pinged from PC1. Show packet capture and write port numbers, IP addresses of each Echo request and reply. Explain ping statistics.

Chapter 9

Lab 8: Design a 2-hop network to demonstrate dynamic addressing using GNS3 and Wireshark

9.1 Objective

Design a 2-hop network to demonstrate dynamic addressing and dynamic routing using GNS3. Configure a DHCP server in R1 with two pools 192.168.1.0 /24 and 192.168.2.0 /24. Exclude the addresses 192.168.1.1–192.168.1.50 and 192.168.2.1–192.168.2.100 from the address pools. Configure R2 as a DHCP relay. Assign addresses using DHCP to the interfaces in PC1 and PC2.


Dynamic addressing using DHCP relay
R1
Network: 172.16.12.0/30
R2
Network: 192.168.1.0/24
172.16.12.1
192.168.1.1
172.16.12.2
Network: 192.168.2.0/24
192.168.2.1
SW1
PC1
PC2
Configure two DHCP pools in R1 and set R2 as DHCP relay
Obtain IP address for PC1 and PC2 using DHCP

9.2 Procedure

1. Configure the router interfaces of R1 and R2 as shown in the Section 1.2.4.

2. For dynamic routing configure as shown in Section 7.2.

3. For configuring DHCP server in R1 follow the steps in Section 8.2.

4. Configuring R2 as DHCP relay

```
R2(config)# interface f1/0
R2(config-if)# ip helper-address 172.16.12.1
```

5. For PC assign IP address using DHCP
```
PC1> ip dhcp -d
```

9.3 Analyses

1. Show the IP addresses assigned via DHCP

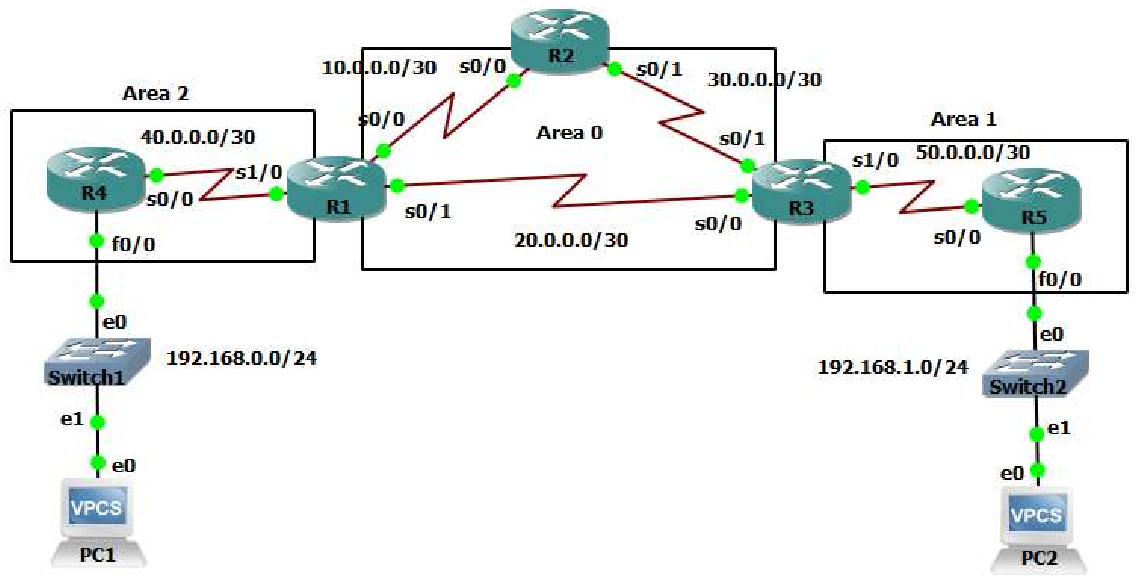
2. Show the routing tables in R1 and R2.
3. Show DHCP server statistics.
4. Show DHCP server's pool information.
5. Analyze the packets exchanged between PC1 and the DHCP server when obtaining IP address. Write port numbers, IP address and MAC address for each packet observed.
6. Show the ping operation by pinging PC2 from PC1. Show packet capture and write port numbers, IP addresses of each Echo request and reply. Explain ping statistics.

Chapter 10

Lab 9: Configure and analyze OSPF in a multihop network using GNS3

10.1 Objective

To learn about OSPF and configure a multihop network which exchanges link state advertisements (LSAs) according to the OSPF protocol. Each OSPF router has a unique router ID (also an IP address). The multihop network is divided into areas which designated as Area 0, Area 1, etc. The areas exchange the LSAs internally and are interconnected by Area 0 for exchanging across areas. The LSAs are sent using multicast IP address 224.0.0.5 (all OSPF routers)



10.2 Procedure

1. Configure the static IP addresses to the routers as per the above figure. Refer to Section 1.2.4.

```
R1# configure terminal
R1(config)# interface s0/0
R1(config-if)# ip address 10.0.0.1 255.255.255.252
R1(config-if)# no shutdown
R1(config-if)#exit
R1(config)# interface s0/1
R1(config-if)# ip address 20.0.0.1 255.255.255.252
R1(config-if)# no shutdown
R1(config-if)#exit
R1(config)# interface s1/0
R1(config-if)# ip address 40.0.0.1 255.255.255.252
```

```
R1(config-if)# no shutdown
R1(config-if)#end
R1#

R2# configure terminal
R2(config)# interface s0/0
R2(config-if)# ip address 10.0.0.2 255.255.255.252
R2(config-if)# no shutdown
R2(config-if)#exit
R2(config)# interface s0/1
R2(config-if)# ip address 30.0.0.1 255.255.255.252
R2(config-if)# no shutdown
R2(config-if)#end
R2#

R3# configure terminal
R3(config)# interface s0/0
R3(config-if)# ip address 20.0.0.2 255.255.255.252
R3(config-if)# no shutdown
R3(config-if)#exit
R3(config)# interface s0/1
R3(config-if)# ip address 30.0.0.2 255.255.255.252
R3(config-if)# no shutdown
R3(config-if)#exit
R3(config)# interface s1/0
R3(config-if)# ip address 50.0.0.1 255.255.255.252
R3(config-if)# no shutdown
R3(config-if)#end
R3#

R4# configure terminal
R4(config)# interface s0/0
R4(config-if)# ip address 40.0.0.2 255.255.255.252
R4(config-if)# no shutdown
R4(config-if)#exit
R4(config)# interface f0/0
R4(config-if)# ip address 192.168.0.1 255.255.255.0
R4(config-if)# no shutdown
R4(config-if)#end
R4#

R5# configure terminal
R5(config)# interface s0/0
R5(config-if)# ip address 50.0.0.2 255.255.255.252
R5(config-if)# no shutdown
R5(config-if)#exit
R5(config)# interface f0/0
R5(config-if)# ip address 192.168.1.1 255.255.255.0
R5(config-if)# no shutdown
```

```
R5(config-if)#end
R5#
```

2. To configure OSPF router you need the network addresses (A.B.C.D) and the corresponding wildcard address E.G.F.H (here E=255-I,…,H=255-L where I.J.K.L is the subnet mask of A.B.C.D) of the interfaces of that router. An example of configuring router R1is given below

```
R1(config)# router ospf 1
R1(config-router)# network 10.0.0.0 0.0.0.3 area 0
R1(config-router)# network 20.0.0.0 0.0.0.3 area 0
R1(config-router)# network 40.0.0.0 0.0.0.3 area 2
R1(config-router)# end
```

3. For PC assign IP address as given in Section 1.2.4 . As an example PC1 is configured as

```
PC1> 192.168.0.2/24 192.168.0.1
```

4. To configure loopback adapter in R1 with an arbitrary address (e.g., 111.111.111.111)

```
R1(config)# int loopback 25
R1(config)# ip address 111.111.111.111 255.255.255.255
R1(config)# no shutdown
R1(config)# end
```

10.3 Analyses

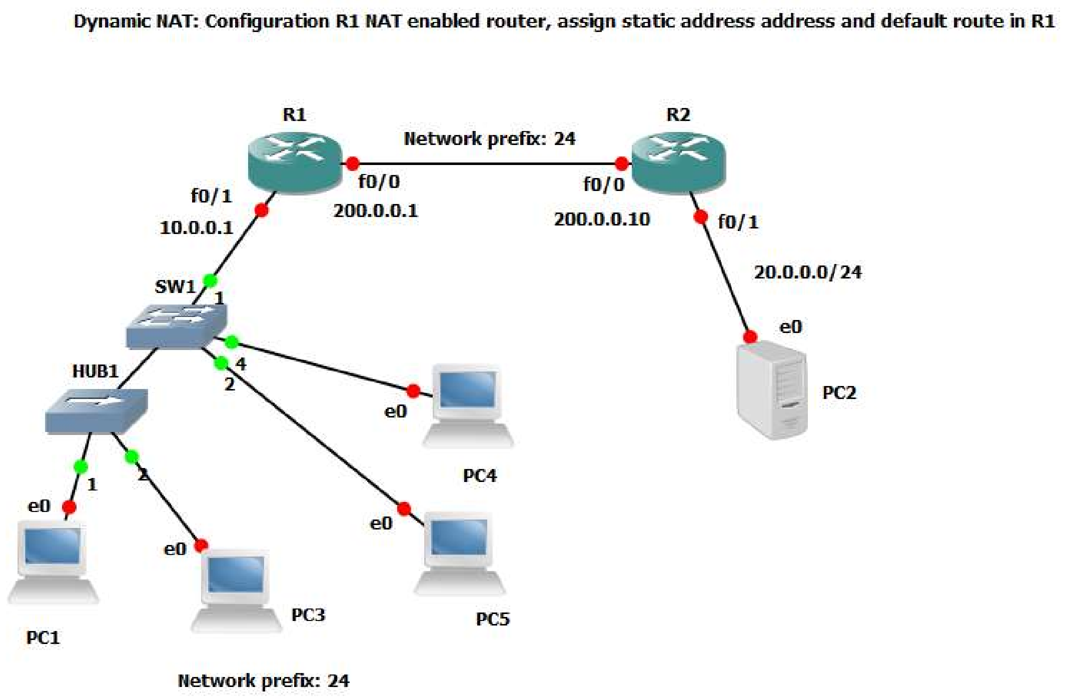
1. Provide the screenshots of the IP addresses assigned to the interfaces
2. Verify the Router ID assigned to each router in the network (e.g., R1# show ip protocols). Try to give a new IP address (your choice) to the loopback interface of the routers and repeat the task. [Note: You must run "reload" in the router, switch-off and restart the router to verify]
3. Provide the screenshots of the routers neighbours (e.g., R1# show ip ospf neighbor).
4. Verify the forwarding table in each router (e.g., R1# show ip route)
5. Verify the ping operation by pinging PC2 from PC1. Show packet capture and write port numbers, IP addresses of each Echo request and reply. Explain ping statistics.
6. Provide screenshot of the packet listing window and the packet content window in Wireshark corresponding to any one OSPF LSA.

Chapter 11

Lab 10: Dynamic NAT configuration using one NAT enabled router

11.1 Objective

Design a NATed network to demonstrate using GNS3 and explain packet capture. Add a server (PC icon changed) to router R2 on network 20.0.0.0/24. Connect network 10.0.0.0/24 as a private network to NAT enabled router R1. Ping the server from any PC in the private network. Let public address pool include 200.0.0.2 to 200.0.0.5.



11.2 Procedure

1. Configure the router interfaces of R1 and R2 as shown in the Section 1.2.4. While declaring interface f0/1 in R1, configure it as the interface for the private network. While declaring interface f0/0 in R1, configure it as the interface for the public network.

2. For dynamic routing configure as shown in Section 7.2.

3. For configuring NAT in R1 follow the steps below.

```
R1(config)#ip nat pool my_pub_ips 200.0.0.2 200.0.0.5 netmask 255.255.255.0
R1(config)#access-list 1 permit 10.0.0.0 0.0.0.255
R1(config)#ip nat inside source list 1 pool my_pub_ips
```

```
R1(config)#ip nat log translations syslog
```

4. For PC assign IP address as given in Section 1.2.4

11.3 Analyses

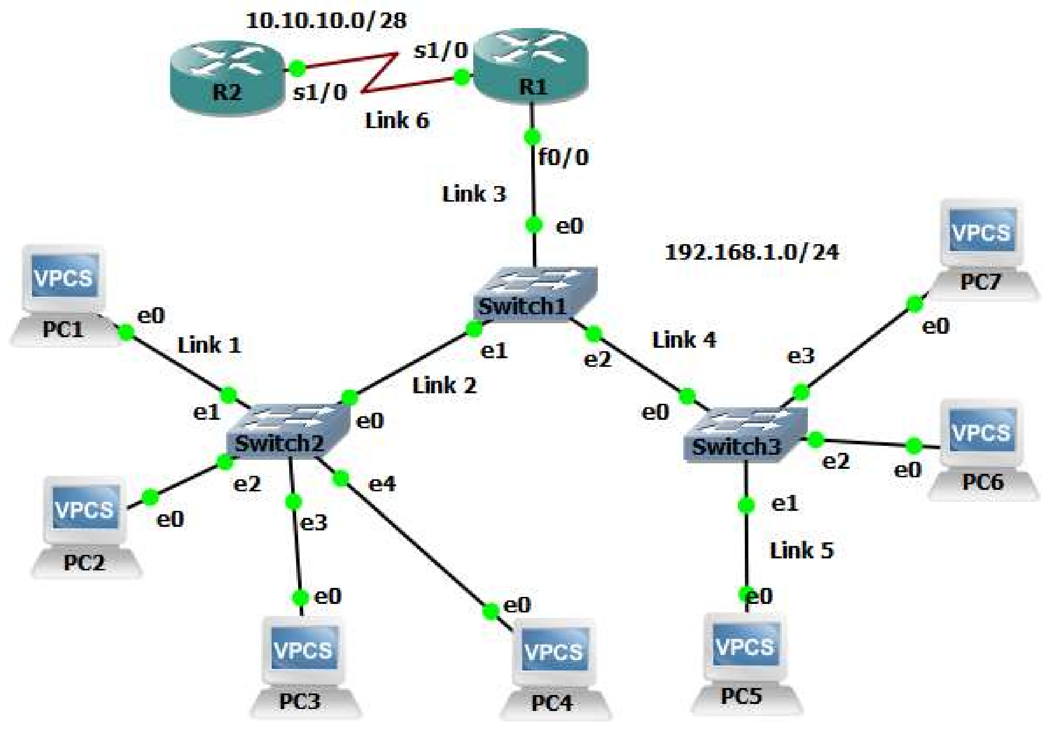
1. Show the NAT translations
2. Show routing tables in R1 and R2 (you can use static or dynamic routing).
3. Show the ping operation by pinging PC2 from PC1. Show packet capture and write port numbers, IP addresses of each Echo request and reply. Explain ping statistics.
4. Do the opposite of step 3 and explain what happened.

Chapter 12

Lab 11: Design a switched LAN and analyze link layer addressing and ARP

12.1 Objective

Design a switched LAN as shown in the figure below. The ARP and link layer header are inspected in ICMP packets. Analyze the packets in Wireshark.



12.2 Procedure

1. Configure the fast Ethernet interface in router R1 as shown in the Section 1.2.4.
2. Configure the PCs with the gateway corresponding to the IP address of the LAN
3. Introduce router R2 into the network (e.g., Link 6 connects s0/0 in R1 is connected with s0/0 in R2). Let the new subnet be 10.10.10.0/16. Configure RIP in R1 and R2.
4. Run Wireshark on Link 1, Link 2, Link 3, Link 4 and Link 5.

12.3 Analyses

1. Perform ping operation from PC1to PC2. What was the first packet sent by PC1 on Link 1? Provide the details of the packet along with its screenshot. Was this packet observed on other links?

2. What was the second packet observed on Link 1? Provide the details of this packet along with its screenshot. Was this packet observed on other links?
3. What were the links on which the ICMP packets were observed?
4. Perform ping operation from PC1to PC5. On which links did you observe the ARP packets? On which links did you observe the ICMP packets?
5. Ping the interface s0/0 in R2 from PC2. On which links were the ARP packets observed? Fill the following table in the sequence of the packets observed. Add rows as required.

Packet type	Link ID	Source MAC address	Destination MAC address

6. Show ARP tables in the hosts and routers