SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution| Approved by AICTE| Affiliated to Anna University

Kuniamuthur, Coimbatore - 641008

DEPARTMENT OF

INFORMATION TECHNOLOGY

22IT502 – DATA COMMUNICATIONS AND COMPUTER NETWORKS LABORATORY

SEMESTER – V

NAME	: ....
REGISTER NO.	: ....
DEGREE & BRANCH	: ....
CLASS & SECTION	: ....

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution| Approved by AICTE| Affiliated to Anna University
Kuniamuthur, Coimbatore - 641008

DEPARTMENT OF INFORMATION TECHNOLOGY

22IT502 – DATA COMMUNICATIONS AND COMPUTER NETWORKS LABORATORY

CONTINUOUS ASSESSMENT RECORD

Submitted by

Name: …………………………………		**Reg. No**	**: …………………………………**
Class: …………………………………		**Branch**	**: …………………………………**

BONAFIDE CERTIFICATE

This is to certify that this bonafide record work done by Mr./Ms. .... (Register No...................................) during the academic year 2024-2025.

Submitted for the End Semester Practical Examination held on………………………

Faculty In-Charge **Head of the Department**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

VISION OF THE DEPARTMENT

To produce industry ready professionals with information technology acquaintance and human values to contribute to the society at large.

MISSION OF THE DEPARTMENT

- To develop and to promote student ability thereby to compete globally through excellence in education.
- To inculcate varied skill sets that meets industry standards and to practice moral values.
- To enrich high integrity to lead and to serve the society.

PROGRAMME EDUCATIONAL OBJECTIVES (PEO)

PEO 1: Graduates will have a profound knowledge in the various programming languages and possess globally competent skill sets by inculcating continuous up gradation of their technical skills and personality traits

PEO 2: Graduates will be able to analyze and find solutions to various applications and reconcile the dynamic trends in the field of Information Technology

PEO 3: Graduates will contribute to the society by their ethical behaviour and effective teamwork.

PEO 4: Graduates will excel with different skills like effective communication, leadership qualities, and provide smart solutions in business environment

PROGRAMME OUTCOMES (PO)

PO1 **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4 **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6 **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES (PSO)

Upon completion of the programme, graduates will have ability to:

- **PSO1** Graduates will demonstrate multidisciplinary knowledge for problem solving by creating solutions for product based and application-based software for the advancement of the society.

- **PSO2** Graduates attain advance knowledge in Information and Communication Technologies (ICT) thereby creating real time solutions for different projects by using modern tools prevailing in the current trends.

- **PSO3** Graduates will exhibit state of the art technologies by applying their knowledge in various programming skills to overcome the demand of sustainable development.

SYLLABUS

22IT502 DATA COMMUNICATIONS AND COMPUTER NETWORKS LABORATORY 0/0/3/1.5

1. A private organization appoints a system administrator and network administrator for troubleshooting and maintenance. System administrator focuses on servers and computer systems, while network administrators work more specifically with network-related tasks and equipment. Becoming a system administrator/network administrator will entail learning some specialized skills. Elucidate the roles and responsibilities & skill sets of a System administrator / Network administrator.

2. A newly constructed block in your college is planning to provide internet connection and also wants to make all the computers available in the block to be interconnected. The new block is of ‘m’ floors and overall ‘n’ rooms in each floor, also it has three buildings in its campus with the same capacity which are separated by ‘x’ meters.
   a. Identify the networking devices required for constructing the suitable network.
   b. List the features of the networking devices.
   c. Recommend and design the best suitable network type based on installation cost, performance, maintenance, installation time.

3. Build a network architecture for a Hypermarket with ‘m’ floors and ‘n’ systems. Choose and design the right Topology to build the architecture.

4. Consider a network with data frames transmitted from sender to receiver. Design a code for bit stuffing and un-stuffing mechanism considering the given input pattern.

5. In a network, assume that the sender sends ‘m’ frames each of ‘n’ bits. Develop a code to calculate the checksum value and to check whether the data frames are received in the receiver end.

6. Assume a network scenario where N is the sender’s window size = ‘m’ & damaged frame is ‘x’;
(a) At this situation, the sender sends frame 1 to ‘m’ before receiving the knowledge of frame 1. All the frames are numbered to deal with the most and duplicate frames. If the sender does not receive the receiver’s acknowledgement, then all the frames available in the current window ie., 1 to ‘m’ will be retransmitted.

(b) The sender will resend only the damaged frame ie., frame ‘x’.

Identify a suitable protocol for the given scenario (a) & (b) and write the code.

7. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

8. Recommend and suggest a suitable protocol to send, receive, and synchronize with Mail. Develop a code for implementing the above protocol.

9. Write a program to find the shortest path between vertices using Open Shortest Path First Routing algorithm.

10. Write a program for congestion control in a network using leaky bucket algorithm.

11. Study of Simulators and Emulators.

12. Simulate a LAN to Create a simple network and show the transfer of packets from one node to another node.

LIST OF EXPERIMENTS

Exp. No.	Date	Name of the Experiment	Page No.
1.		STUDY OF SYSTEM ADMINISTRATION ANDNETWORK ADMINISTRATION	
2.		DESIGN OF CAMPUS AREA NETWORK IN COLLEGE	
3.		DESIGN OF SUITABLE NETWORK TOPOLOGY FOR A MULTIFLOOR HYPERMARKET	
4.		BITSTUFFING AND UNSTUFFING IN DATA FRAME TRANSMISSION BETWEEN SENDER AND RECEIVER	
5.		ERROR DETECTION IN DATA FRAME TRANSMISSION BETWEEN SENDER AND RECEIVER USING CHECKSUM ERROR	
6.		SIMULATION OF SLIDING WINDOW PROTOCOL AND STOP AND WAIT PROTOCOL	
7.		TCP/IP SOCKET PROGRAMMING IN CLIENT SERVER MODEL FOR FILE TRANSFER APPLICATION	
8.		DEVELOPMENT OF SUITABLE PROTOCOL TO SEND, RECEIVE AND SYNCHRONIZE MAIL.	
9.		WRITE A PROGRAM TO FIND THE SHORTEST PATH BETWEEN VERTICES USING OPEN SHORTEST PATH FIRST ROUTING ALGORITHM.	
10.		WRITE A PROGRAM FOR CONGESTION CONTROL IN A NETWORK USING LEAKY BUCKET ALGORITHM.	
11.		STUDY OF NETWORK SIMULATORS AND EMULATORS	
12.		SIMULATION OF LAN TO CREATE A SIMPLE NETWORK	
ADDITIONAL EXPERIMENTS			
13.		SIMULATION OF TCP/IP SOCKET IN CLIENT SERVER MODEL FOR MAIL TRANSFER APPLICATION	
14.		SIMULATION OF TCP/IP SOCKET IN CLIENT SERVER MODEL FOR DNS APPLICATION	
		Signature of the Faculty	

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution| Approved by AICTE| Affiliated to Anna University
Kuniamuthur, Coimbatore - 641008

Department of Information Technology
(Accredited by NBA)

Rubrics for Evaluating Laboratory

Subject Code : 22IT502

Lab Name : Data Communications and Computer Networks Laboratory

Criteria	Range of Marks			
	Excellent	Good	Average	Below Average
Aim, Algorithm, and Description with sample data (20 Marks)	18-20	14-17	10-13	0-9
Coding/Designing (30 Marks)	27-30	21-26	15-20	0-14
Compilation and Debugging/ Configuration (30 Marks)	27-30	21-26	15-20	0-14
Results / Simulation (10 Marks)	18-20	14-17	10-13	0-9
Documentation & Viva (10 Marks)	9-10	7-8	5-6	0-4

OVERALL MARKS			
90-100	**70-89**	**50-69**	**0-49**
Excellent	**Good**	**Average**	**Below Average**

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution| Approved by AICTE| Affiliated to Anna University
Kuniamuthur, Coimbatore - 641008

Department of Information Technology
(Accredited by NBA)

Rubrics based Evaluation

Reg No: **Name of the Student:**
Name of the Lab: 22IT502 - Data Communications and Computer Networks Lab

Components	**Exp No and Date**													
	Exp 1	**Exp 2**	**Exp 3**	**Exp 4**	**Exp 5**	**Exp 6**	**Exp 7**	**Exp 8**	**Exp 9**	**Exp 10**	**Exp 11**	**Exp 12**	**Exp 13**	**Exp 14**
Aim, Algorithm, and Description with sample data (20 Marks)														
Coding/Designing (30 Marks)														
Compilation and Debugging/ Configuration (30 Marks)														
Results / Simulation (10 Marks)														
Documentation & Viva (10 Marks)														
Total														
Average														

Staff In Charge

PROGRAMME OUTCOMES

a) Graduates will demonstrate knowledge of mathematical, scientific and multidisciplinary approach for problem solving.

(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging)

b) Graduates will be able to apply their knowledge in various programming skills to create solutions for product based and application-based software.

(Criteria to be used for assessment Coding, Compilation and Debugging)

c) Graduates will possess the ability to create real time solutions for different projects by using modern tools prevailing in the current trends.

(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging, Execution and Results (Inclusion of Generalization like Subroutines, Modules)

e) Graduates attain advanced knowledge in the stream of Information Technology and basic knowledge in Electronics and Communication Engineering to develop and maintain the simple and complex information systems.

(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging, Execution and Results (Inclusion of Generalization like Subroutines, Modules

Lab In Charge

Ex No: 1

Date:

STUDY OF SYSTEM ADMINISTRATION AND NETWORK ADMINISTRATION

Aim:

A private organization appoints a system administrator and network administrator for troubleshooting and maintenance. System administrator focuses on servers and computer systems, while network administrators work more specifically with network-related tasks and equipment. Becoming a system administrator/network administrator will entail learning some specialized skills. Elucidate the roles and responsibilities & skill sets of a System administrator / Network administrator.

Theory:

System administrator:

System administrator, or sysadmin, is a person who is responsible for the upkeep, configuration, and reliable operation of computer systems; especially multi-user computers, such as servers. The system administrator seeks to ensure that the uptime, performance, resources, and security of the computers he or she manages meet the needs of the users, without exceeding the budget.

Skills:

- Entails a knowledge of operating systems and applications
- Problem solving Technique.
- To understand the behavior of software in order to deploy it and for troubleshooting problem.

Responsibilities of the System Administrator:

- User account management,
- Hardware management
- Perform file system backups, restores
- Install and configure new software and services

- Keep systems and services operating
- Monitor system and network, Troubleshoot problems
- Maintain documentation
- Audit security, Help users,
- Performance tuning.

Network administration:

A network administrator, sometimes called a systems administrator, is responsible for keeping an organization’s computer network up to date and running smoothly. Any company or organization that uses multiple computers or software platforms needs a network admin to coordinate the different systems. Network admins will especially be in high demand as companies and organizations invest in newer, faster technology and mobile networks. Growth is also expected in the healthcare industry as the use of information technology increases.

Responsibilities of the Network Administrator

As a network administrator, the tasks generally fall into the following areas:

- Designing and planning the network
- Setting up the network
- Maintaining the network
- Expanding the network

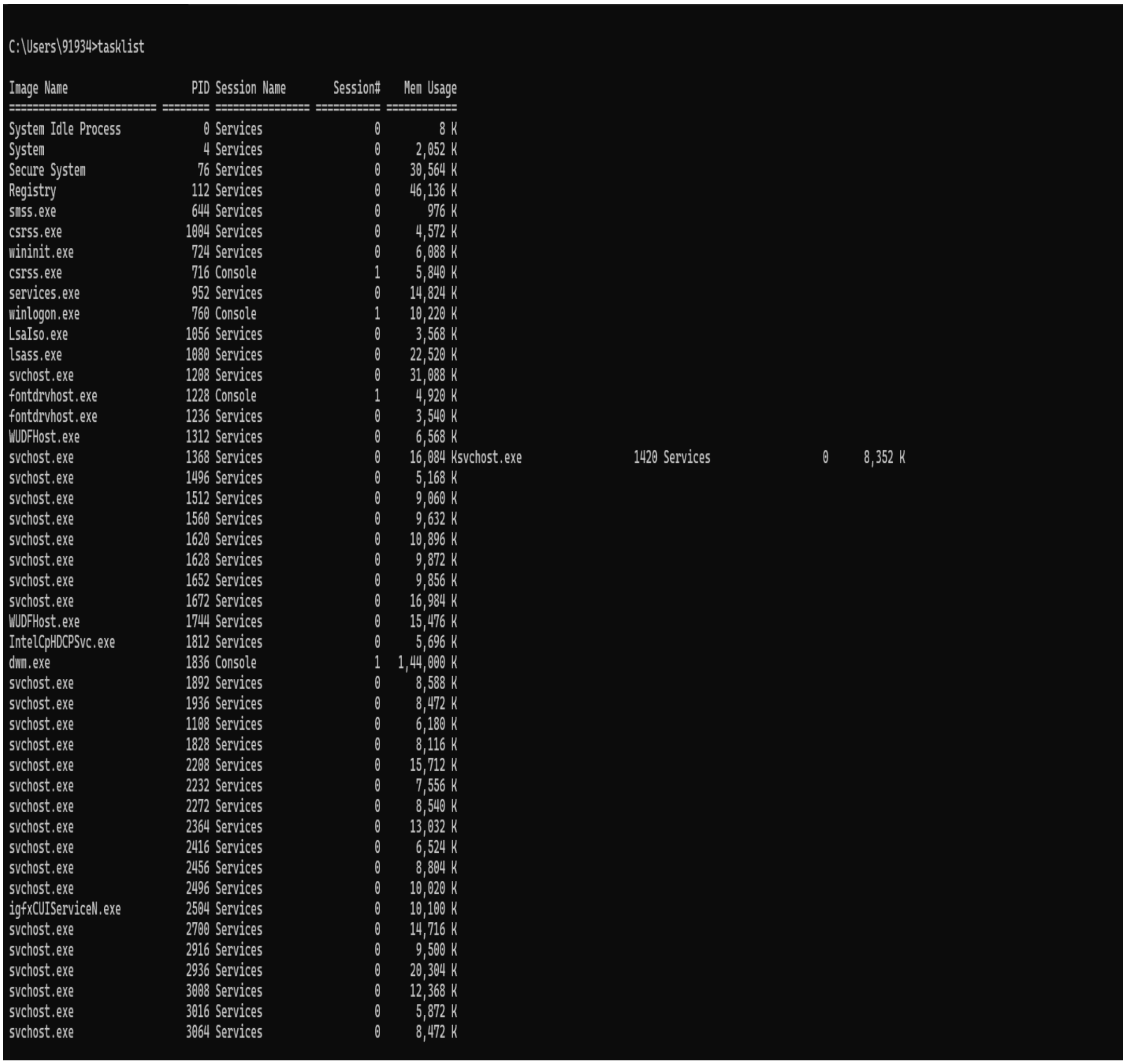
Procedure:

Run the network administrator commands and the system administrator commands in the command prompt.

SYSTEM COMMANDS

TASKLIST:

This command is used to Show List of Processes along with Their Name, Process ID and Memory Usage.

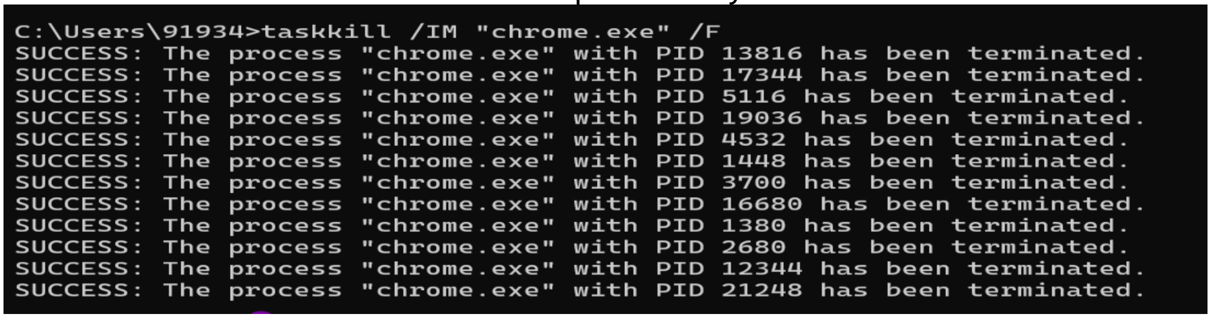


```
C:\Users\91934>tasklist

Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0          8 K
System                           4 Services                   0      2,052 K
Secure System                   76 Services                   0     30,564 K
Registry                       112 Services                   0     46,136 K
smss.exe                       644 Services                   0        976 K
csrss.exe                     1004 Services                   0      4,572 K
wininit.exe                    724 Services                   0      6,088 K
csrss.exe                      716 Console                    1      5,840 K
services.exe                   952 Services                   0     14,824 K
winlogon.exe                   760 Console                    1     10,220 K
LsaIso.exe                    1056 Services                   0      3,568 K
lsass.exe                     1080 Services                   0     22,520 K
svchost.exe                   1208 Services                   0     31,088 K
fontdrvhost.exe               1228 Console                    1      4,920 K
fontdrvhost.exe               1236 Services                   0      3,540 K
WUDFHost.exe                  1312 Services                   0      6,568 K
svchost.exe                   1368 Services                   0     16,084 Ksvchost.exe                   1420 Services                   0      8,352 K
svchost.exe                   1496 Services                   0      5,168 K
svchost.exe                   1512 Services                   0      9,060 K
svchost.exe                   1560 Services                   0      9,632 K
svchost.exe                   1620 Services                   0     10,896 K
svchost.exe                   1628 Services                   0      9,872 K
svchost.exe                   1652 Services                   0      9,856 K
svchost.exe                   1672 Services                   0     16,984 K
WUDFHost.exe                  1744 Services                   0     15,476 K
IntelCpHDCPSvc.exe            1812 Services                   0      5,696 K
dwm.exe                       1836 Console                    1   1,44,000 K
svchost.exe                   1892 Services                   0      8,588 K
svchost.exe                   1936 Services                   0      8,472 K
svchost.exe                   1108 Services                   0      6,180 K
svchost.exe                   1828 Services                   0      8,116 K
svchost.exe                   2208 Services                   0     15,712 K
svchost.exe                   2232 Services                   0      7,556 K
svchost.exe                   2272 Services                   0      8,540 K
svchost.exe                   2364 Services                   0     13,032 K
svchost.exe                   2416 Services                   0      6,524 K
svchost.exe                   2456 Services                   0      8,804 K
svchost.exe                   2496 Services                   0     10,020 K
igfxCUIServiceN.exe           2504 Services                   0     10,100 K
svchost.exe                   2700 Services                   0     14,716 K
svchost.exe                   2916 Services                   0      9,500 K
svchost.exe                   2936 Services                   0     20,304 K
svchost.exe                   3008 Services                   0     12,368 K
svchost.exe                   3016 Services                   0      5,872 K
svchost.exe                   3064 Services                   0      8,472 K
```

TASKKILL:

Eg. taskkill/IM “chrome.exe”/F

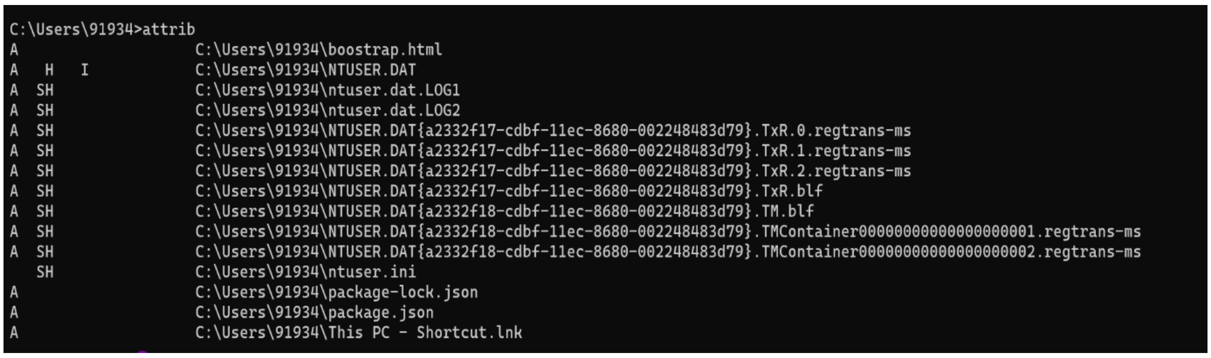
This command is used to kill the process by its name or PID name.



```
C:\Users\91934>taskkill /IM "chrome.exe" /F
SUCCESS: The process "chrome.exe" with PID 13816 has been terminated.
SUCCESS: The process "chrome.exe" with PID 17344 has been terminated.
SUCCESS: The process "chrome.exe" with PID 5116 has been terminated.
SUCCESS: The process "chrome.exe" with PID 19036 has been terminated.
SUCCESS: The process "chrome.exe" with PID 4532 has been terminated.
SUCCESS: The process "chrome.exe" with PID 1448 has been terminated.
SUCCESS: The process "chrome.exe" with PID 3700 has been terminated.
SUCCESS: The process "chrome.exe" with PID 16680 has been terminated.
SUCCESS: The process "chrome.exe" with PID 1380 has been terminated.
SUCCESS: The process "chrome.exe" with PID 2680 has been terminated.
SUCCESS: The process "chrome.exe" with PID 12344 has been terminated.
SUCCESS: The process "chrome.exe" with PID 21248 has been terminated.
```

ATTRIB:

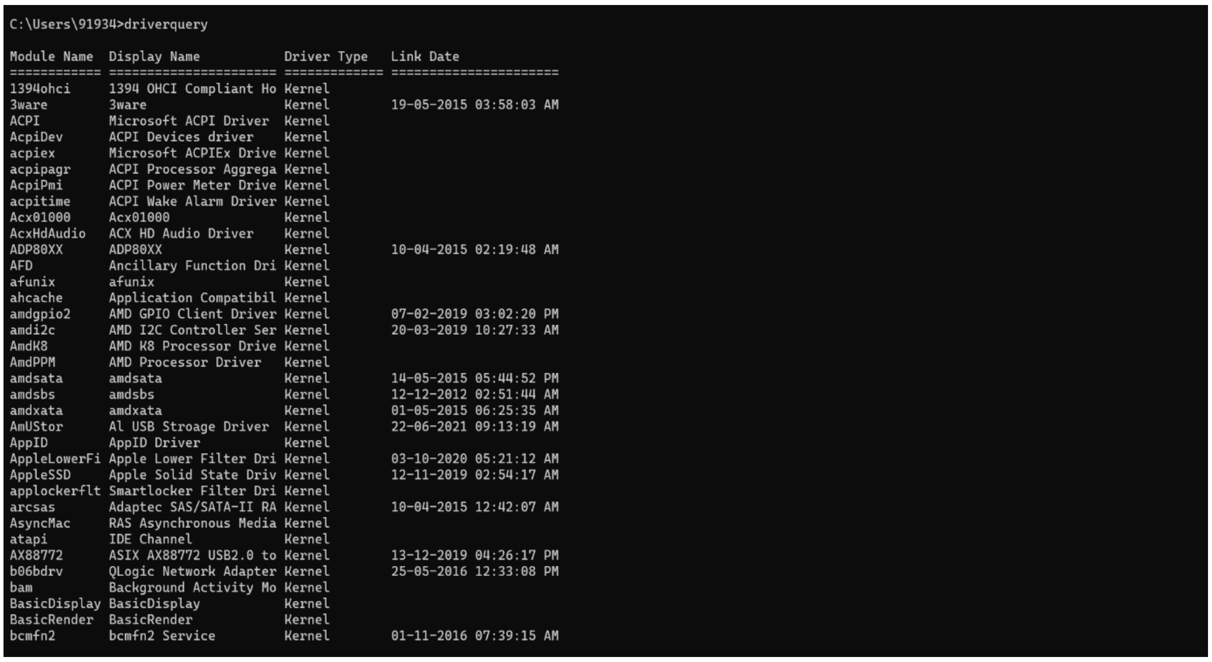
This command is used to remove and set file attributes (hidden, read-only, system and archive). It displays, sets or removes the read-only, hidden and archive file attributes assigned for a file or directory. It allows a user to change the file attribute directly using this command.



```
C:\Users\91934>attrib
A                    C:\Users\91934\boostrap.html
A   H   I            C:\Users\91934\NTUSER.DAT
A  SH                C:\Users\91934\ntuser.dat.LOG1
A  SH                C:\Users\91934\ntuser.dat.LOG2
A  SH                C:\Users\91934\NTUSER.DAT{a2332f17-cdbf-11ec-8680-002248483d79}.TxR.0.regtrans-ms
A  SH                C:\Users\91934\NTUSER.DAT{a2332f17-cdbf-11ec-8680-002248483d79}.TxR.1.regtrans-ms
A  SH                C:\Users\91934\NTUSER.DAT{a2332f17-cdbf-11ec-8680-002248483d79}.TxR.2.regtrans-ms
A  SH                C:\Users\91934\NTUSER.DAT{a2332f17-cdbf-11ec-8680-002248483d79}.TxR.blf
A  SH                C:\Users\91934\NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TM.blf
A  SH                C:\Users\91934\NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000001.regtrans-ms
A  SH                C:\Users\91934\NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000002.regtrans-ms
   SH                C:\Users\91934\ntuser.ini
A                    C:\Users\91934\package-lock.json
A                    C:\Users\91934\package.json
A                    C:\Users\91934\This PC - Shortcut.lnk
```

DRIVERQUERY:

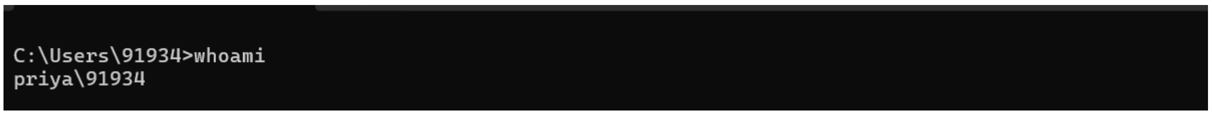
Used to Display the List of Drivers installed on the System with Given Name, Date and Time.



```
C:\Users\91934>driverquery

Module Name  Display Name           Driver Type   Link Date
============ ====================== ============= ======================
1394ohci     1394 OHCI Compliant Ho Kernel
3ware        3ware                  Kernel        19-05-2015 03:58:03 AM
ACPI         Microsoft ACPI Driver  Kernel
AcpiDev      ACPI Devices driver    Kernel
acpiex       Microsoft ACPIEx Drive Kernel
acpipagr     ACPI Processor Aggrega Kernel
AcpiPmi      ACPI Power Meter Drive Kernel
acpitime     ACPI Wake Alarm Driver Kernel
Acx01000     Acx01000               Kernel
AcxHdAudio   ACX HD Audio Driver    Kernel
ADP80XX      ADP80XX                Kernel        10-04-2015 02:19:48 AM
AFD          Ancillary Function Dri Kernel
afunix       afunix                 Kernel
ahcache      Application Compatibil Kernel
amdgpio2     AMD GPIO Client Driver Kernel        07-02-2019 03:02:20 PM
amdi2c       AMD I2C Controller Ser Kernel        20-03-2019 10:27:33 AM
AmdK8        AMD K8 Processor Drive Kernel
AmdPPM       AMD Processor Driver   Kernel
amdsata      amdsata                Kernel        14-05-2015 05:44:52 PM
amdsbs       amdsbs                 Kernel        12-12-2012 02:51:44 AM
amdxata      amdxata                Kernel        01-05-2015 06:25:35 AM
AmUStor      Al USB Stroage Driver  Kernel        22-06-2021 09:13:19 AM
AppID        AppID Driver           Kernel
AppleLowerFi Apple Lower Filter Dri Kernel        03-10-2020 05:21:12 AM
AppleSSD     Apple Solid State Driv Kernel        12-11-2019 02:54:17 AM
applockerflt Smartlocker Filter Dri Kernel
arcsas       Adaptec SAS/SATA-II RA Kernel        10-04-2015 12:42:07 AM
AsyncMac     RAS Asynchronous Media Kernel
atapi        IDE Channel            Kernel
AX88772      ASIX AX88772 USB2.0 to Kernel        13-12-2019 04:26:17 PM
b06bdrv      QLogic Network Adapter Kernel        25-05-2016 12:33:08 PM
bam          Background Activity Mo Kernel
BasicDisplay BasicDisplay           Kernel
BasicRender  BasicRender            Kernel
bcmfn2       bcmfn2 Service         Kernel        01-11-2016 07:39:15 AM
```

WHOAMI:

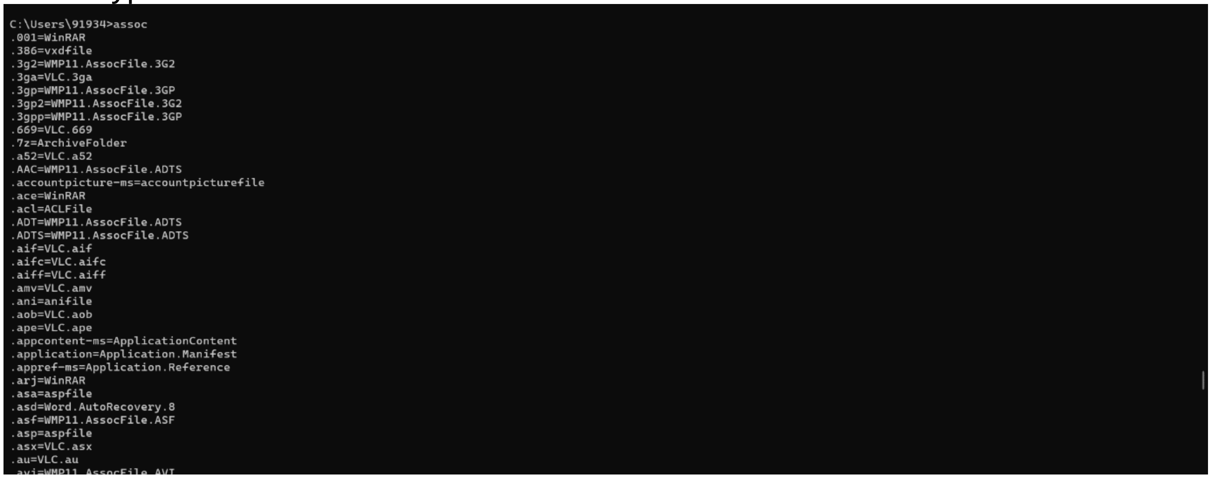
"whoami" command will help you to check the user details of logged in user and the group it belongs to.



```
C:\Users\91934>whoami
priya\91934
```

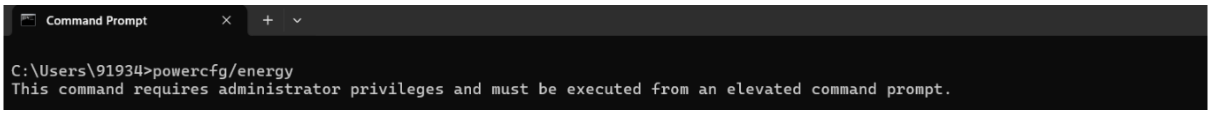
ASSOC:

assoc is a command that displays the program and/or functionality associated with a specific file type.



```
C:\Users\91934>assoc
.001=WinRAR
.386=vxdfile
.3g2=WMP11.AssocFile.3G2
.3ga=VLC.3ga
.3gp=WMP11.AssocFile.3GP
.3gp2=WMP11.AssocFile.3G2
.3gpp=WMP11.AssocFile.3GP
.669=VLC.669
.7z=ArchiveFolder
.a52=VLC.a52
.AAC=WMP11.AssocFile.ADTS
.accountpicture-ms=accountpicturefile
.ace=WinRAR
.acl=ACLFile
.ADT=WMP11.AssocFile.ADTS
.ADTS=WMP11.AssocFile.ADTS
.aif=VLC.aif
.aifc=VLC.aifc
.aiff=VLC.aiff
.amv=VLC.amv
.ani=anifile
.aob=VLC.aob
.ape=VLC.ape
.appcontent-ms=ApplicationContent
.application=Application.Manifest
.appref-ms=Application.Reference
.arj=WinRAR
.asa=aspfile
.asd=Word.AutoRecovery.8
.asf=WMP11.AssocFile.ASF
.asp=aspfile
.asx=VLC.asx
.au=VLC.au
.avi=WMP11.AssocFile.AVI
```

POWERCFG –ENERGY:

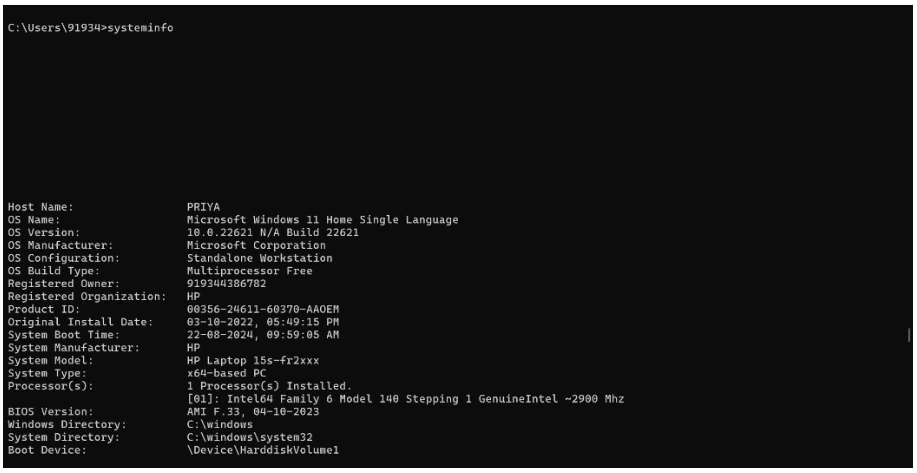


Command Prompt

```
C:\Users\91934>powercfg/energy
This command requires administrator privileges and must be executed from an elevated command prompt.
```

This command is used to check battery health and generate Energy Report in Windows.

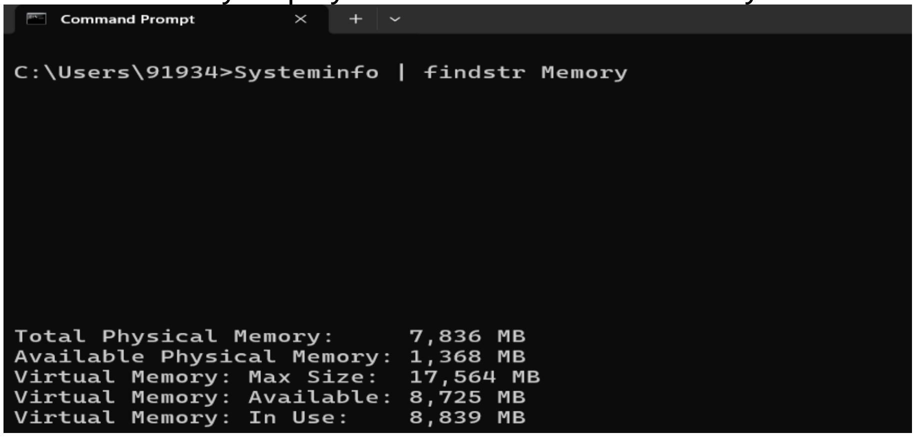
SYSTEMINFO



```
C:\Users\91934>systeminfo

Host Name:                 PRIYA
OS Name:                   Microsoft Windows 11 Home Single Language
OS Version:                10.0.22621 N/A Build 22621
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          919344386782
Registered Organization:   HP
Product ID:                00356-24611-60370-AAOEM
Original Install Date:     03-10-2022, 05:49:15 PM
System Boot Time:          22-08-2024, 09:59:05 AM
System Manufacturer:       HP
System Model:              HP Laptop 15s-fr2xxx
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 140 Stepping 1 GenuineIntel ~2900 Mhz
BIOS Version:              AMI F.33, 04-10-2023
Windows Directory:         C:\windows
System Directory:          C:\windows\system32
Boot Device:               \Device\HarddiskVolume1
```

SYSTEMINFO | FINDSTR MEMORY:

This command easily displays the total amount of memory which is on your system.

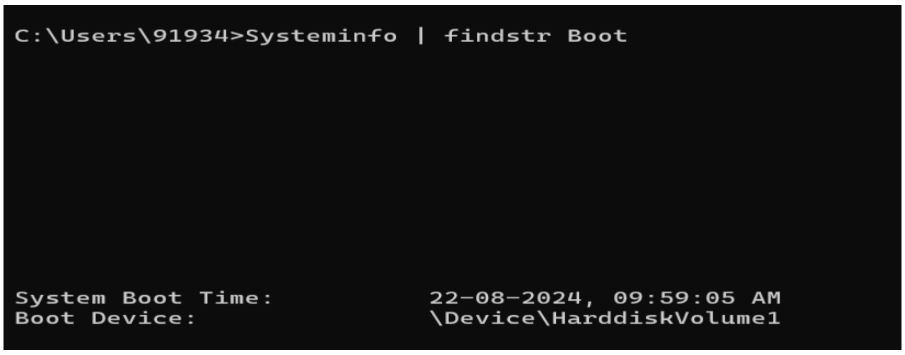


Command Prompt

```
C:\Users\91934>Systeminfo | findstr Memory

Total Physical Memory:      7,836 MB
Available Physical Memory: 1,368 MB
Virtual Memory: Max Size:  17,564 MB
Virtual Memory: Available: 8,725 MB
Virtual Memory: In Use:     8,839 MB
```

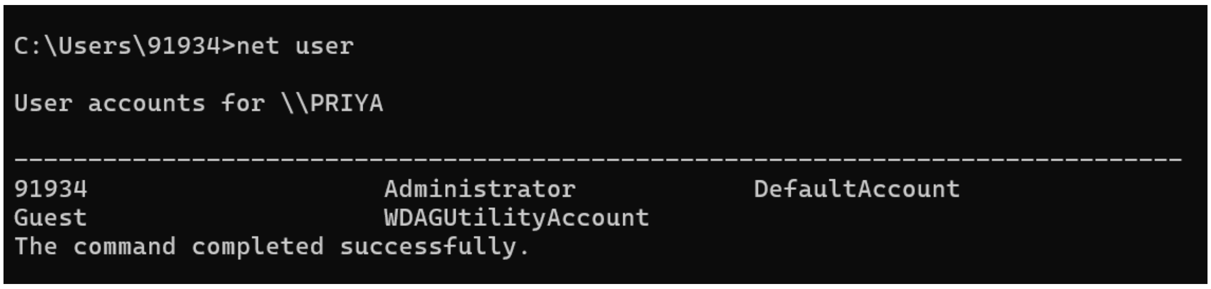
SYSTEMINFO \FINDSTR BOOT:

Systeminfo "System Boot Time" gives the time of the last reboot, not the last cold startup as I believe it used to do.



```
C:\Users\91934>Systeminfo | findstr Boot







System Boot Time:              22-08-2024, 09:59:05 AM
Boot Device:                   \Device\HarddiskVolume1
```

NET USER:

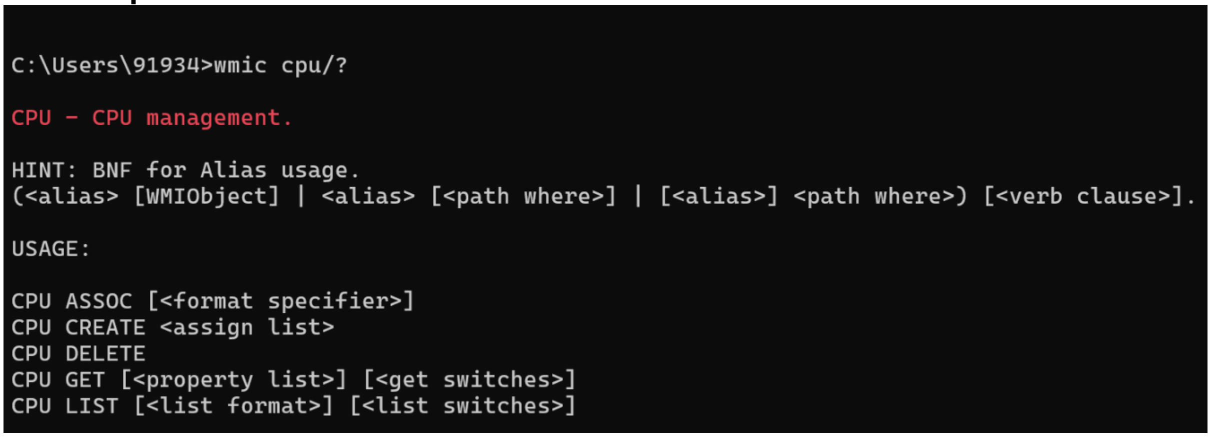


```
C:\Users\91934>net user

User accounts for \\PRIYA

-------------------------------------------------------------------------------
91934                    Administrator            DefaultAccount
Guest                    WDAGUtilityAccount
The command completed successfully.
```

The net user command is used to add, remove, and make changes to the **user** accounts on a computer.

WMIC CPU:

Eg. wmic cpu/?



```
C:\Users\91934>wmic cpu/?

CPU - CPU management.

HINT: BNF for Alias usage.
(<alias> [WMIObject] | <alias> [<path where>] | [<alias>] <path where>) [<verb clause>].

USAGE:

CPU ASSOC [<format specifier>]
CPU CREATE <assign list>
CPU DELETE
CPU GET [<property list>] [<get switches>]
CPU LIST [<list format>] [<list switches>]
```

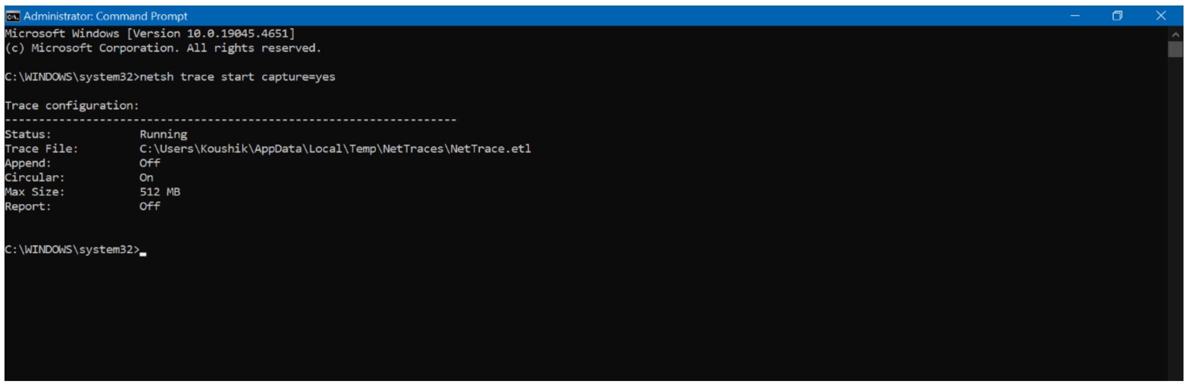
The Windows Management Instrumentation Command line (WMIC) is a software utility that allows users to perform Windows Management Instrumentation (WMI) operations with a command prompt.

NETWORK COMMANDS

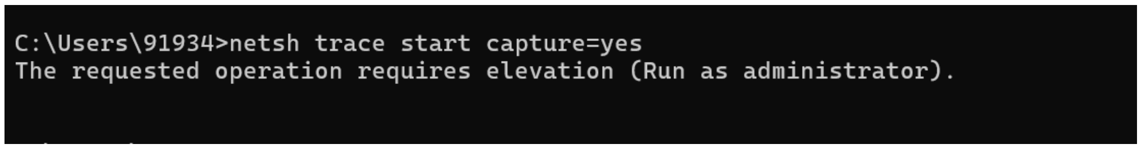
NETSH

i) NETSH TRACE START CAPTURE=YES

capture =yes (ensures network trace is captured) persistent =yes (specifies whether the tracing session continues across reboots, and is on until netsh trace stop is issued)



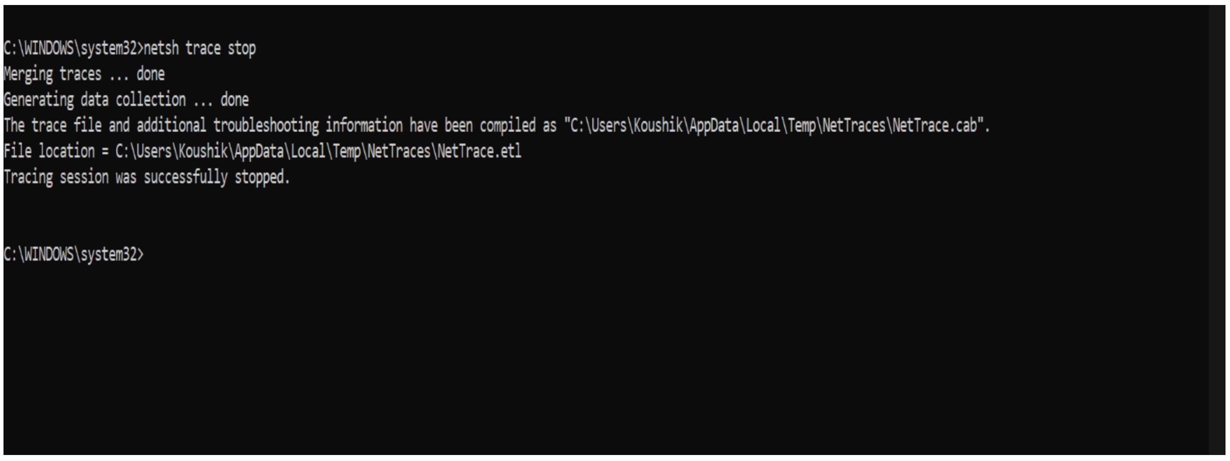
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>netsh trace start capture=yes

Trace configuration:
-------------------------------------------------------------------
Status:             Running
Trace File:         C:\Users\Koushik\AppData\Local\Temp\NetTraces\NetTrace.etl
Append:             Off
Circular:           On
Max Size:           512 MB
Report:             Off


C:\WINDOWS\system32>_
```



```
C:\Users\91934>netsh trace start capture=yes
The requested operation requires elevation (Run as administrator).
```

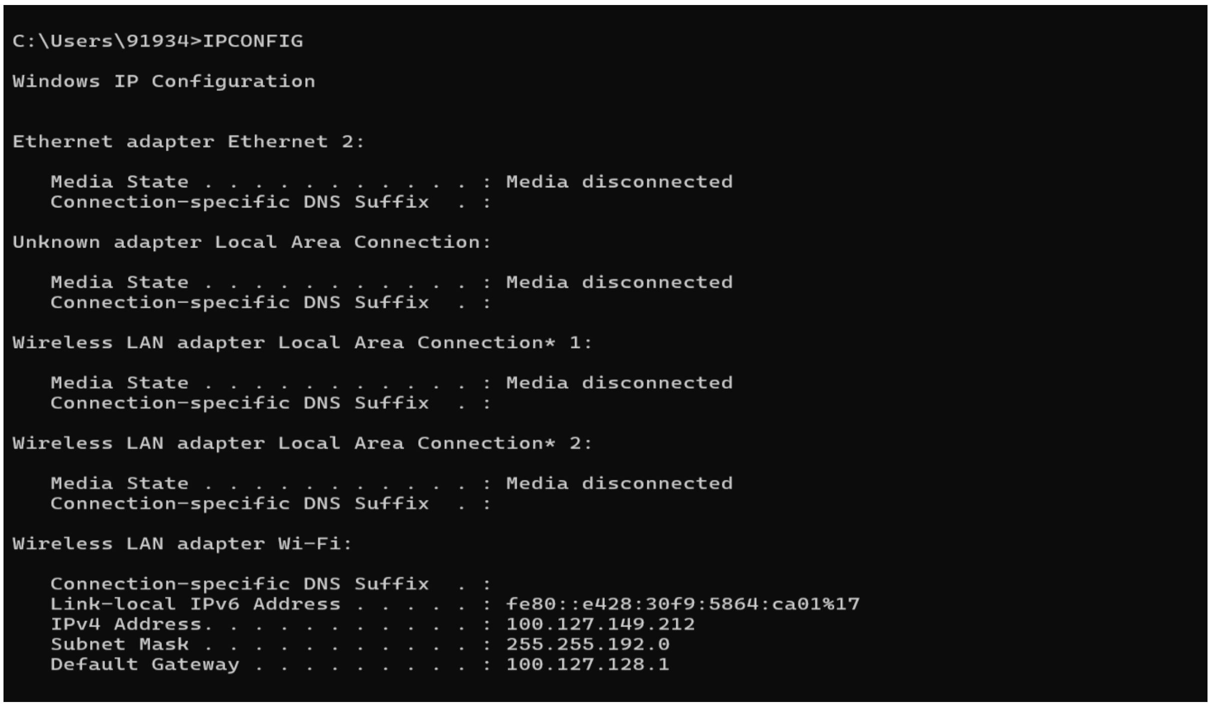
ii) NETSH TRACE STOP

This command line tool has a trace feature. To run it, open an elevated command prompt and type netsh. Then the netsh prompt appears. To start the capture type “trace start <parameters>”, please find more details about the parameters and some examples below. To stop the capture, type “trace stop”.



```
C:\WINDOWS\system32>netsh trace stop
Merging traces ... done
Generating data collection ... done
The trace file and additional troubleshooting information have been compiled as "C:\Users\Koushik\AppData\Local\Temp\NetTraces\NetTrace.cab".
File location = C:\Users\Koushik\AppData\Local\Temp\NetTraces\NetTrace.etl
Tracing session was successfully stopped.


C:\WINDOWS\system32>
```

IPCONFIG:

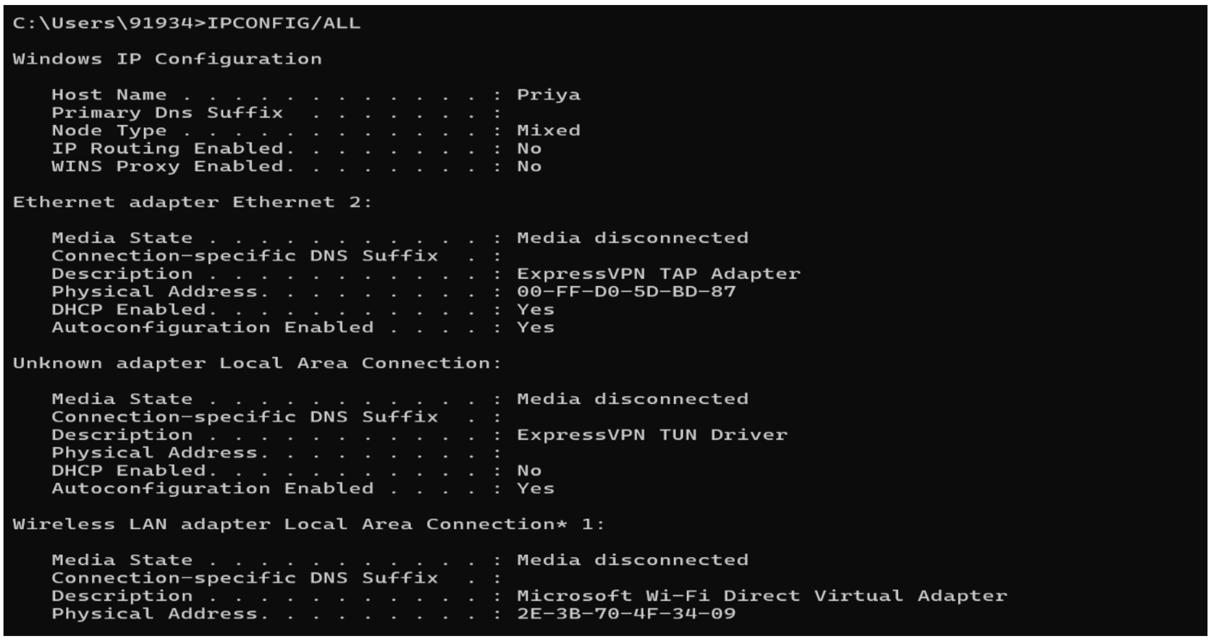


```
C:\Users\91934>IPCONFIG

Windows IP Configuration


Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::e428:30f9:5864:ca01%17
   IPv4 Address. . . . . . . . . . . : 100.127.149.212
   Subnet Mask . . . . . . . . . . . : 255.255.192.0
   Default Gateway . . . . . . . . . : 100.127.128.1
```

This command allows you to get the IP address information of a Windows computer. It also allows some control over your network adapters, IP addresses (DHCP-assigned specifically), even your DNS cache. Ipconfig replaced the older winipcfg utility.

IPCONFIG /ALL:

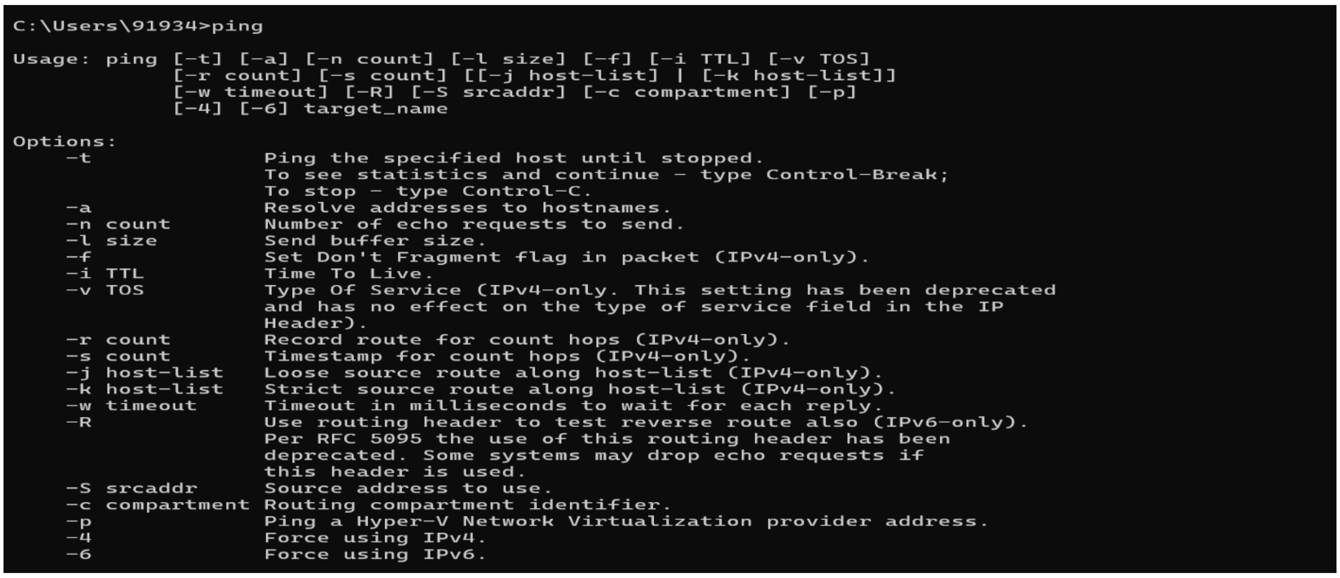
By Using the ‘ipconfig /all’ command, we can see an increased amount of information namely each NIC’s DHCP configuration and the DNS servers.



```
C:\Users\91934>IPCONFIG/ALL

Windows IP Configuration

   Host Name . . . . . . . . . . . . : Priya
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Mixed
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : ExpressVPN TAP Adapter
   Physical Address. . . . . . . . . : 00-FF-D0-5D-BD-87
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : ExpressVPN TUN Driver
   Physical Address. . . . . . . . . :
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
   Physical Address. . . . . . . . . : 2E-3B-70-4F-34-09
```

PING:

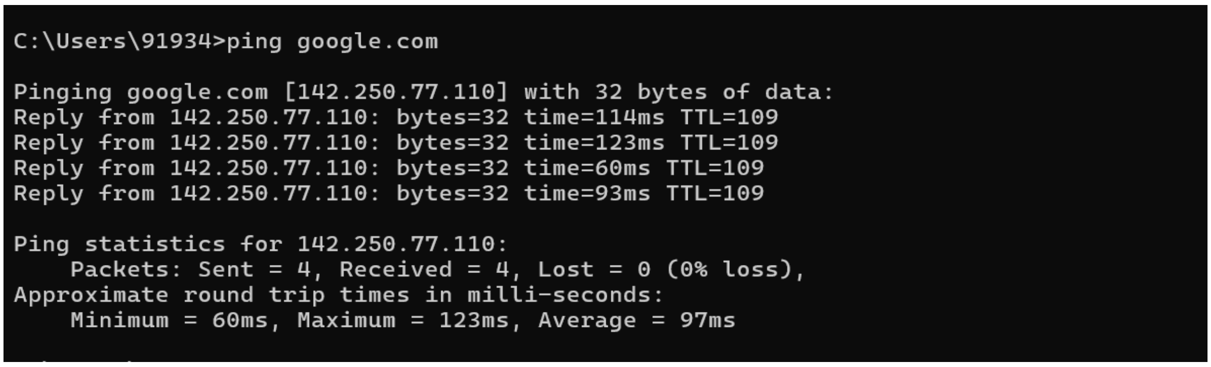
ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution. Used without parameters, this command displays Help content. You can also use this command to test both the computer name and the IP address of the computer



```
C:\Users\91934>ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been deprecated
                   and has no effect on the type of service field in the IP
                   Header).
    -r count       Record route for count hops (IPv4-only).
    -s count       Timestamp for count hops (IPv4-only).
    -j host-list   Loose source route along host-list (IPv4-only).
    -k host-list   Strict source route along host-list (IPv4-only).
    -w timeout     Timeout in milliseconds to wait for each reply.
    -R             Use routing header to test reverse route also (IPv6-only).
                   Per RFC 5095 the use of this routing header has been
                   deprecated. Some systems may drop echo requests if
                   this header is used.
    -S srcaddr     Source address to use.
    -c compartment Routing compartment identifier.
    -p             Ping a Hyper-V Network Virtualization provider address.
    -4             Force using IPv4.
    -6             Force using IPv6.
```

PING "WEBSITE-NAME" --------------***Note: "Website-name"- Give any website name:***

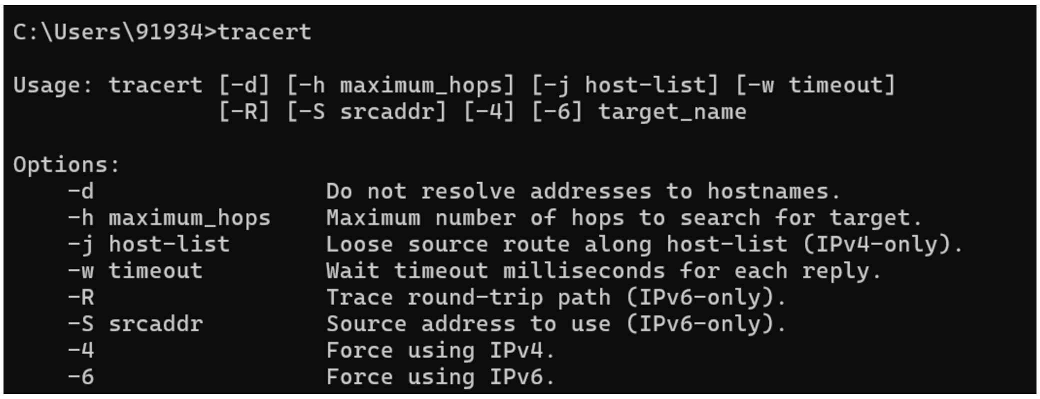
Eg. Ping google.com



```
C:\Users\91934>ping google.com

Pinging google.com [142.250.77.110] with 32 bytes of data:
Reply from 142.250.77.110: bytes=32 time=114ms TTL=109
Reply from 142.250.77.110: bytes=32 time=123ms TTL=109
Reply from 142.250.77.110: bytes=32 time=60ms TTL=109
Reply from 142.250.77.110: bytes=32 time=93ms TTL=109

Ping statistics for 142.250.77.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 60ms, Maximum = 123ms, Average = 97ms
```

TRACERT:

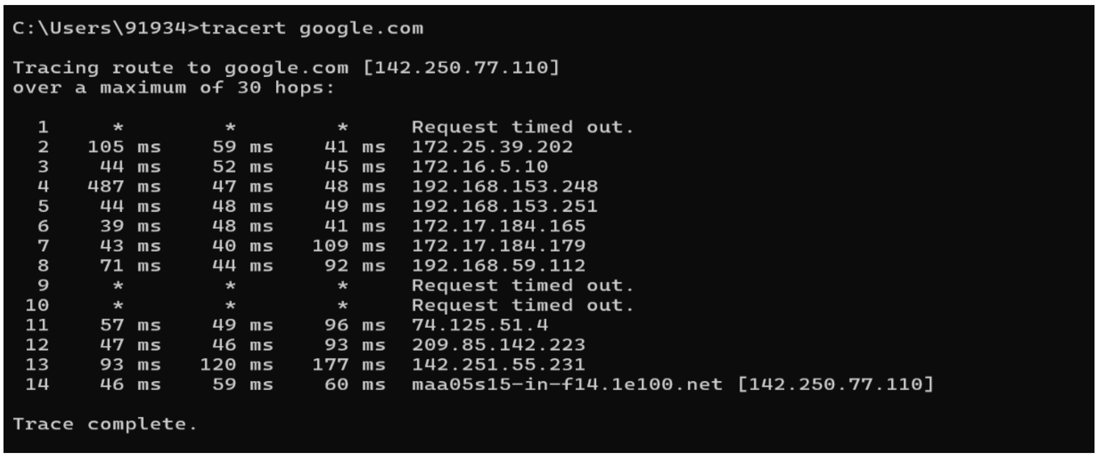
The tracert command (spelled traceroute in Unix/Linux implementations) is one of the key diagnostic tools for TCP/IP. It displays a list of all the routers that a packet must go through to get from the computer where tracert is run to any other computer on the Internet.



```
C:\Users\91934>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
               [-R] [-S srcaddr] [-4] [-6] target_name

Options:
    -d                 Do not resolve addresses to hostnames.
    -h maximum_hops    Maximum number of hops to search for target.
    -j host-list       Loose source route along host-list (IPv4-only).
    -w timeout         Wait timeout milliseconds for each reply.
    -R                 Trace round-trip path (IPv6-only).
    -S srcaddr         Source address to use (IPv6-only).
    -4                 Force using IPv4.
    -6                 Force using IPv6.
```

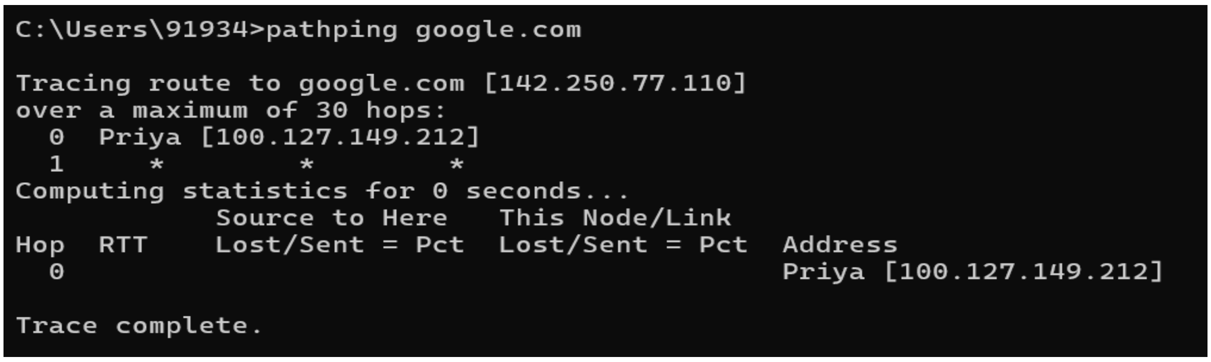
tracert "WEBSITE-NAME"

Eg. tracert google.com



```
C:\Users\91934>tracert google.com

Tracing route to google.com [142.250.77.110]
over a maximum of 30 hops:

  1     *        *        *     Request timed out.
  2   105 ms    59 ms    41 ms  172.25.39.202
  3    44 ms    52 ms    45 ms  172.16.5.10
  4   487 ms    47 ms    48 ms  192.168.153.248
  5    44 ms    48 ms    49 ms  192.168.153.251
  6    39 ms    48 ms    41 ms  172.17.184.165
  7    43 ms    40 ms   109 ms  172.17.184.179
  8    71 ms    44 ms    92 ms  192.168.59.112
  9     *        *        *     Request timed out.
 10     *        *        *     Request timed out.
 11    57 ms    49 ms    96 ms  74.125.51.4
 12    47 ms    46 ms    93 ms  209.85.142.223
 13    93 ms   120 ms   177 ms  142.251.55.231
 14    46 ms    59 ms    60 ms  maa05s15-in-f14.1e100.net [142.250.77.110]

Trace complete.
```

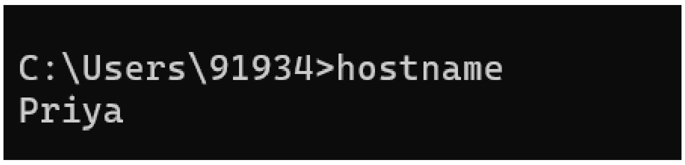
PATHPING (PING AND TRACERT)



```
C:\Users\91934>pathping google.com

Tracing route to google.com [142.250.77.110]
over a maximum of 30 hops:
  0  Priya [100.127.149.212]
  1     *        *        *
Computing statistics for 0 seconds...
            Source to Here   This Node/Link
Hop  RTT    Lost/Sent = Pct  Lost/Sent = Pct  Address
  0                                           Priya [100.127.149.212]

Trace complete.
```

HOSTNAME:

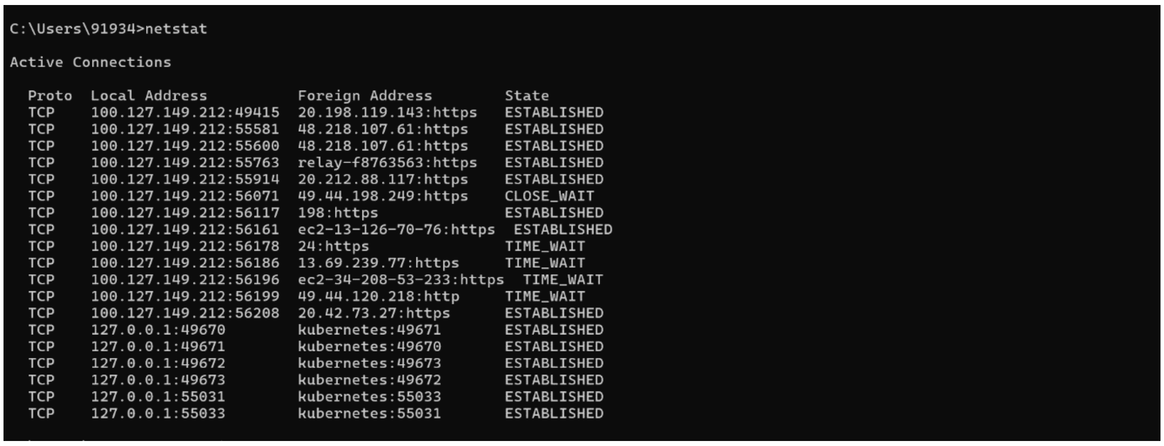
This command is used to display the IP address of the remote machine.



```
C:\Users\91934>hostname
Priya
```

NETSTAT:

This command displays active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols). Used without parameters, this command displays active TCP connections.

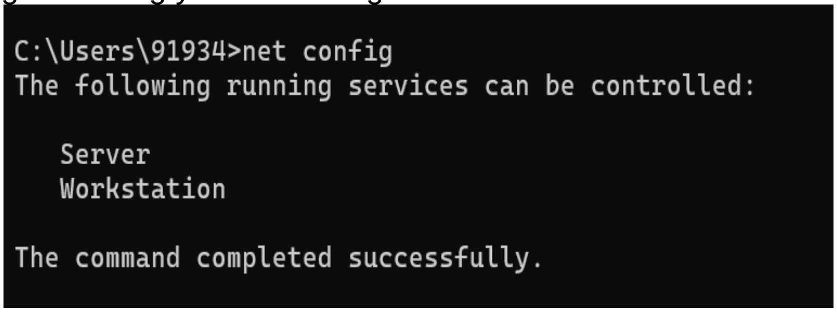


```
C:\Users\91934>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    100.127.149.212:49415  20.198.119.143:https   ESTABLISHED
  TCP    100.127.149.212:55581  48.218.107.61:https    ESTABLISHED
  TCP    100.127.149.212:55600  48.218.107.61:https    ESTABLISHED
  TCP    100.127.149.212:55763  relay-f8763563:https   ESTABLISHED
  TCP    100.127.149.212:55914  20.212.88.117:https    ESTABLISHED
  TCP    100.127.149.212:56071  49.44.198.249:https    CLOSE_WAIT
  TCP    100.127.149.212:56117  198:https              ESTABLISHED
  TCP    100.127.149.212:56161  ec2-13-126-70-76:https  ESTABLISHED
  TCP    100.127.149.212:56178  24:https               TIME_WAIT
  TCP    100.127.149.212:56186  13.69.239.77:https     TIME_WAIT
  TCP    100.127.149.212:56196  ec2-34-208-53-233:https  TIME_WAIT
  TCP    100.127.149.212:56199  49.44.120.218:http     TIME_WAIT
  TCP    100.127.149.212:56208  20.42.73.27:https      ESTABLISHED
  TCP    127.0.0.1:49670        kubernetes:49671       ESTABLISHED
  TCP    127.0.0.1:49671        kubernetes:49670       ESTABLISHED
  TCP    127.0.0.1:49672        kubernetes:49673       ESTABLISHED
  TCP    127.0.0.1:49673        kubernetes:49672       ESTABLISHED
  TCP    127.0.0.1:55031        kubernetes:55033       ESTABLISHED
  TCP    127.0.0.1:55033        kubernetes:55031       ESTABLISHED
```

NET CONFIG:

Using net config command we can configure server and workstation services on Windows computer. For server service, we can configure few settings using this command.

Using net config you can change auto disconnect time and hidden attributes.



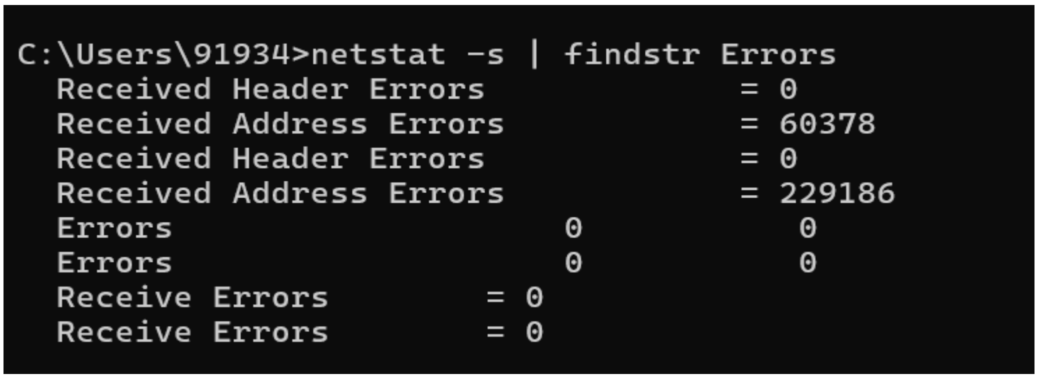
```
C:\Users\91934>net config
The following running services can be controlled:

   Server
   Workstation

The command completed successfully.
```

NETSTAT –S | FINDSTR ERRORS:

netstat is a command-line network tool that is a handy troubleshooting command. Its cross-platform utility means you can use it on Linux, macOS, or Windows.

netstat can be very handy in the following.

- Display incoming and outgoing network connections
- Display routing tables
- Display number of network interfaces
- Display network protocol statistic

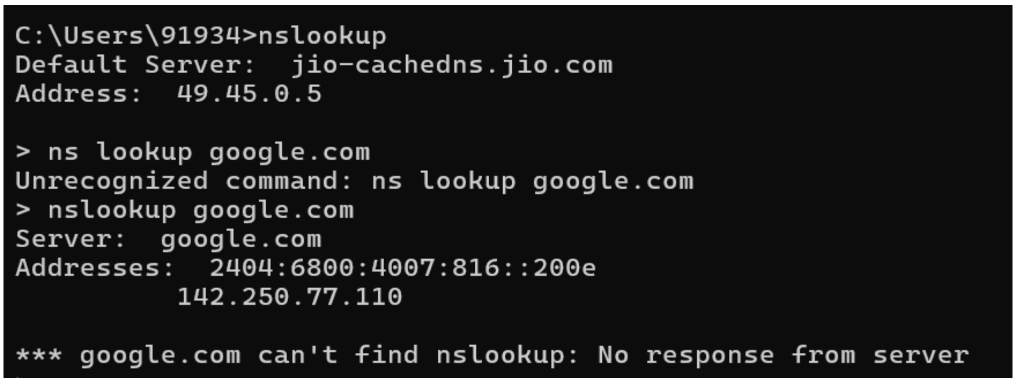


```
C:\Users\91934>netstat -s | findstr Errors
  Received Header Errors            = 0
  Received Address Errors           = 60378
  Received Header Errors            = 0
  Received Address Errors           = 229186
  Errors                       0           0
  Errors                       0           0
  Receive Errors         = 0
  Receive Errors         = 0
```

NSLOOKUP:

- nslookup
- nslookup **"WEBSITE-NAME"**
  Eg. nslookup google.com

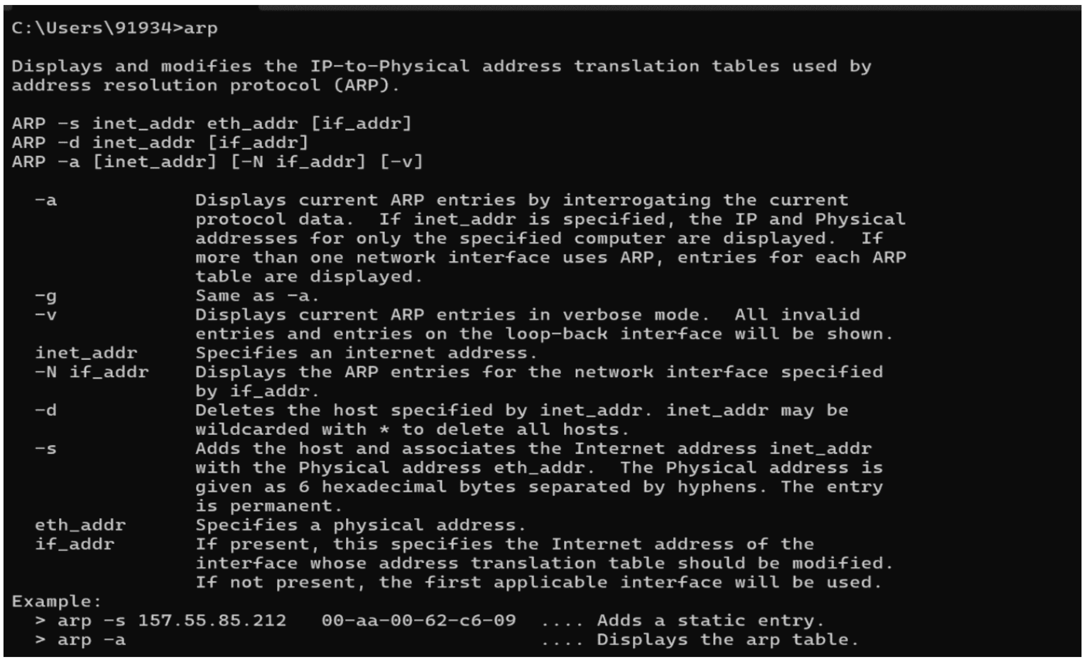
Nslookup (from name server lookup) is a network administration command-line tool for querying the Domain Name System (DNS) to obtain the mapping between domain name and IP address, or other DNS records.



```
C:\Users\91934>nslookup
Default Server:  jio-cachedns.jio.com
Address:  49.45.0.5

> ns lookup google.com
Unrecognized command: ns lookup google.com
> nslookup google.com
Server:  google.com
Addresses:  2404:6800:4007:816::200e
          142.250.77.110

*** google.com can't find nslookup: No response from server
```

ARP:

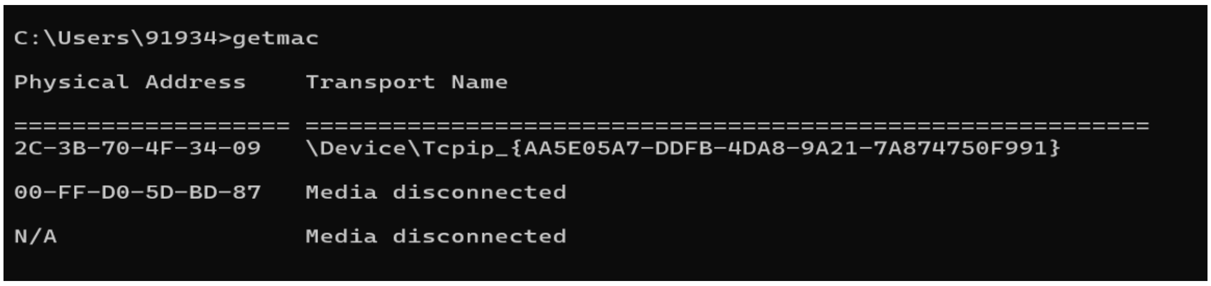
arp command manipulates the System's ARP cache. It also allows a complete dump of the ARP cache. ARP stands for Address Resolution Protocol. The primary function of this protocol is to resolve the IP address of a system to its mac address, and hence it works between level 2(Data link layer) and level 3(Network layer).



```
C:\Users\91934>arp

Displays and modifies the IP-to-Physical address translation tables used by
address resolution protocol (ARP).

ARP -s inet_addr eth_addr [if_addr]
ARP -d inet_addr [if_addr]
ARP -a [inet_addr] [-N if_addr] [-v]

  -a            Displays current ARP entries by interrogating the current
                protocol data.  If inet_addr is specified, the IP and Physical
                addresses for only the specified computer are displayed.  If
                more than one network interface uses ARP, entries for each ARP
                table are displayed.
  -g            Same as -a.
  -v            Displays current ARP entries in verbose mode.  All invalid
                entries and entries on the loop-back interface will be shown.
  inet_addr     Specifies an internet address.
  -N if_addr    Displays the ARP entries for the network interface specified
                by if_addr.
  -d            Deletes the host specified by inet_addr. inet_addr may be
                wildcarded with * to delete all hosts.
  -s            Adds the host and associates the Internet address inet_addr
                with the Physical address eth_addr.  The Physical address is
                given as 6 hexadecimal bytes separated by hyphens. The entry
                is permanent.
  eth_addr      Specifies a physical address.
  if_addr       If present, this specifies the Internet address of the
                interface whose address translation table should be modified.
                If not present, the first applicable interface will be used.
Example:
  > arp -s 157.55.85.212   00-aa-00-62-c6-09  .... Adds a static entry.
  > arp -a                                    .... Displays the arp table.
```

GETMAC:

Getmac is a Windows command used to display the Media Access Control (MAC) addresses for each network adapter in the computer. These activities will show you how to use the getmac command to display MAC addresses.



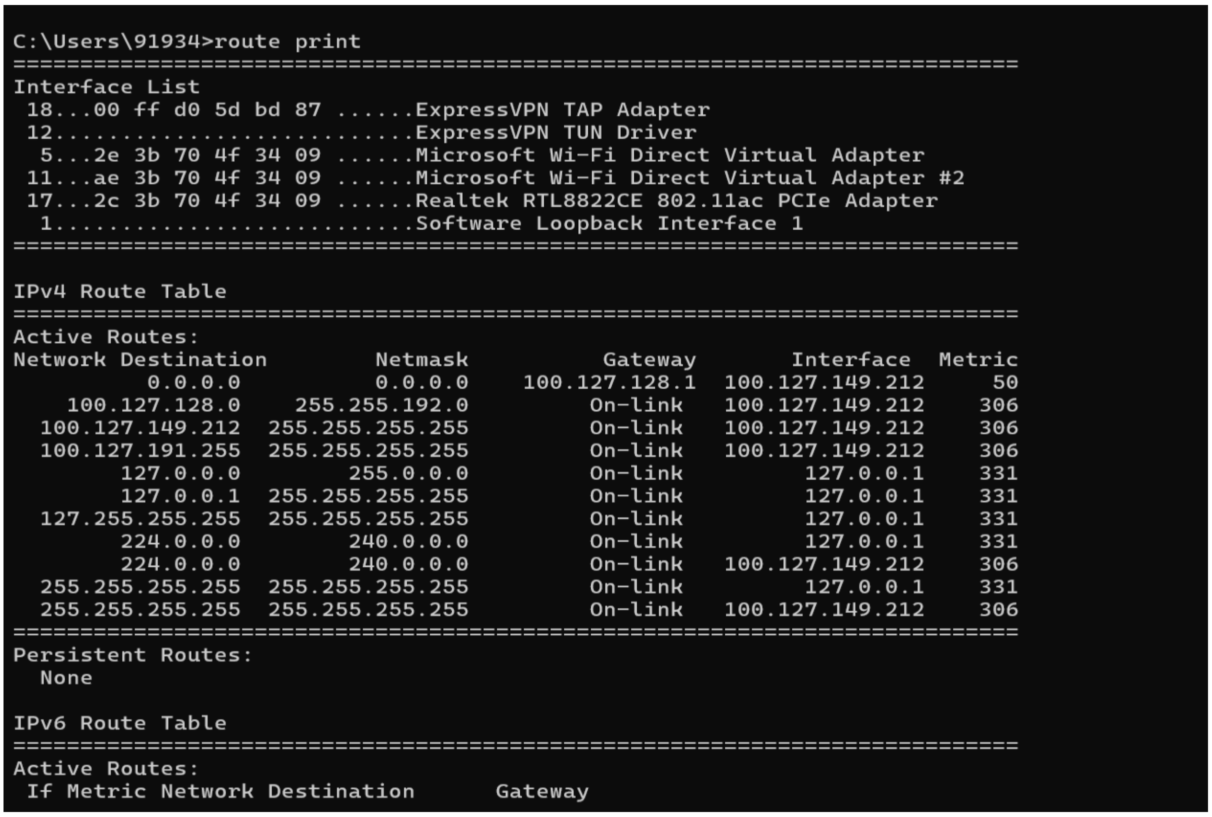
```
C:\Users\91934>getmac

Physical Address    Transport Name

=================== ==========================================================
2C-3B-70-4F-34-09   \Device\Tcpip_{AA5E05A7-DDFB-4DA8-9A21-7A874750F991}

00-FF-D0-5D-BD-87   Media disconnected

N/A                 Media disconnected
```

ROUTE PRINT:

The ‘route print’ Command from an Administrative Command Prompt in Windows 7 provides a variety of useful information. Observing the output of the Command indicates there are 5 Major Sections. The Sections include:

Interface List IPv4 Route Table

IPv4 Persistent Routes IPv6 Route Table IPv6 Persistent Routes



```
C:\Users\91934>route print
===========================================================================
Interface List
 18...00 ff d0 5d bd 87 ......ExpressVPN TAP Adapter
 12...........................ExpressVPN TUN Driver
  5...2e 3b 70 4f 34 09 ......Microsoft Wi-Fi Direct Virtual Adapter
 11...ae 3b 70 4f 34 09 ......Microsoft Wi-Fi Direct Virtual Adapter #2
 17...2c 3b 70 4f 34 09 ......Realtek RTL8822CE 802.11ac PCIe Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    100.127.128.1  100.127.149.212     50
    100.127.128.0    255.255.192.0         On-link   100.127.149.212    306
  100.127.149.212  255.255.255.255         On-link   100.127.149.212    306
  100.127.191.255  255.255.255.255         On-link   100.127.149.212    306
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link   100.127.149.212    306
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link   100.127.149.212    306
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
```

Result:

Thus the roles and responsibilities & skill sets of a System administrator / Network administrator were tested through commands in command prompt and studied.

Ex No: 2

Date:23/09/24

DESIGN OF CAMPUS AREA NETWORK WITHIN A COLLEGE CAMPUS.

Aim:

A newly constructed block in your college is planning to provide internet connection and also wants to make all the computers available in the block to be interconnected. The new block is of 'm' floors and overall 'n' rooms in each floor, also it has three buildings in its campus with the same capacity which are separated by 'x' meters.

a. Identify the networking devices required for constructing the suitable network.
b. List the features of the networking devices.

Recommend and design the best suitable network type based on installation cost, performance, maintenance, and installation time.

Theory:

To provide an effective and interconnected network for the newly constructed block in a college, it is essential to identify the appropriate networking devices and network type. Here's a step-by-step guide to designing a suitable network for the scenario described:

a. Networking Devices Required:

- **Routers**:
  Routers connect different networks, allowing communication between them. They are crucial for internet connectivity and routing data between the local network and the internet.
- **Switches**:
  Switches connect multiple devices within the same network, enabling them to communicate efficiently. They manage data traffic and ensure that data is sent only to its intended destination within the network.

- **Access Points (APs)**:
  Access points provide wireless connectivity to devices. They extend the wired network by creating a wireless local area network (WLAN), enabling mobile devices to connect to the internet and the local network.
- **Network Cables (Ethernet cables, Fiber optic cables)**:
  These cables physically connect devices to the network. Ethernet cables are commonly used within buildings, while fiber optic cables are ideal for connecting different buildings over longer distances due to their higher bandwidth and lower latency.

b. Features of the Networking Devices

- **Routers:**
  - Dynamic routing capabilities
  - Support for multiple network protocols (e.g., IPv4, IPv6)
  - Quality of Service (QoS) features for traffic management
  - Built-in firewall and VPN support
- **Switches:**
  - High port density for connecting many devices
  - Support for VLANs (Virtual LANs) for network segmentation
  - PoE (Power over Ethernet) support for powering devices like IP cameras
  - Layer 2 and Layer 3 switching capabilities
- **Access Points:**
  - Support for the latest Wi-Fi standards (e.g., Wi-Fi 6)
  - Seamless roaming and load balancing features
  - Multiple SSIDs for different user groups
  - Security features like WPA3 encryption
- **Network Cables:**
  - Ethernet cables: Cat5e, Cat6, or Cat6a for different speed and distance requirements
  - Fiber optic cables: Single-mode or multi-mode for high-speed long-distance connections

c. **Recommended Network Type**

Network Type: Campus Area Network (CAN)

Reasons for Recommendation:

- **Installation Cost**: Moderate cost as it utilizes existing Ethernet and fiber optic cabling infrastructure efficiently.
- **Performance**: High performance with the use of fiber optics for building interconnectivity and gigabit Ethernet within buildings.
- **Maintenance**: Relatively low maintenance due to centralized management capabilities in modern networking equipment.
- **Installation Time**: Moderate installation time, leveraging structured cabling systems and modular network components.

Design Overview:

Wired Backbone:

- Use fiber optic cables to connect the three buildings, ensuring high-speed data transfer and minimal latency.

In-Building Network:

- Deploy gigabit Ethernet switches on each floor to connect rooms and provide reliable wired connections.
- Use VLANs to segment the network logically for different departments or functions.

Wireless Network:

- Install access points throughout the floors to provide wireless connectivity, ensuring coverage in all rooms and common areas.

Internet Connectivity:

- Use a high-performance router to connect the entire block to the internet, with redundancy options to prevent downtime.

This design ensures that the college's new block has a robust, high-performance network that supports current needs and future expansion.

Procedure:

For a campus with three blocks, namely, UG block, Admin block and newly added PG block, with following setup,

UG Block – 136 hosts (68 per floor)

Admin Block – 96 hosts (48 per floor)

PG Block – 52 hosts (26 per floor)

and assuming the original IP address assigned for the entire network is 192.168.1.0, the following procedure can be implemented, employing ***subnetting.***

UG BLOCK:

136 hosts required.

Hence 256 host addresses can be provided to cover required 136.

$256 = 2^8$

No. of host addresses available = $2^8 – 2 = 256 – 2 = 254$

No. of bits required for host address = 8

No. of bits required for network address = 32-8 = 24

Subnet mask representation: 192.168.1.0/24

Subnet IP address : 192.168.1.0

Subnet mask : 255.255.255.0 (11111111 11111111 11111111 00000000)

Range of available IP addresses:

First IP address: 192.168.1.1

Last IP address: 192.168.1.254

ADMIN BLOCK:

96 hosts required.

Hence 128 host addresses can be provided to cover required 96.

$128 = 2^7$

No. of host addresses available = $2^7 – 2 = 128 – 2 = 126$

No. of bits required for host address = 7

No. of bits required for network address = 32-7 = 25

Subnet mask representation: 192.168.2.0/25

Subnet IP address : 192.168.2.0

Subnet mask : 255.255.255.128 (11111111 11111111 11111111 10000000)

Range of available IP addresses:

First IP address: 192.168.2.1

Last IP address: 192.168.2.126

NEW PG BLOCK

52 hosts required.

Hence 64 host addresses can be provided to cover required 52.

$64 = 2^6$

No. of host addresses available = $2^6 – 2 = 64 – 2 = 62$

No. of bits required for host address = 6

No. of bits required for network address = 32-6 = 26

Subnet mask representation: 192.168.2.0/26

Subnet IP address : 192.168.2.128

Subnet mask : 255.255.255.192 (11111111 11111111 11111111 11000000)

Range of available IP addresses:

First IP address: 192.168.2.129

Last IP address: 192.168.2.190

ROUTER CONFIGURATION

Router 0:

Fastethernet 0/0 :

IP:192.168.1.100, Subnet mask : 255.255.255.0

Fastethernet 0/1 :

IP:192.168.2.100, Subnet mask : 255.255.255.128

Se 2/0 :

IP:192.168.2.225, Subnet mask : 255.255.255.252

Router 1:

Fastethernet 0/0 :

IP:192.168.2.190, Subnet mask : 255.255.255.192

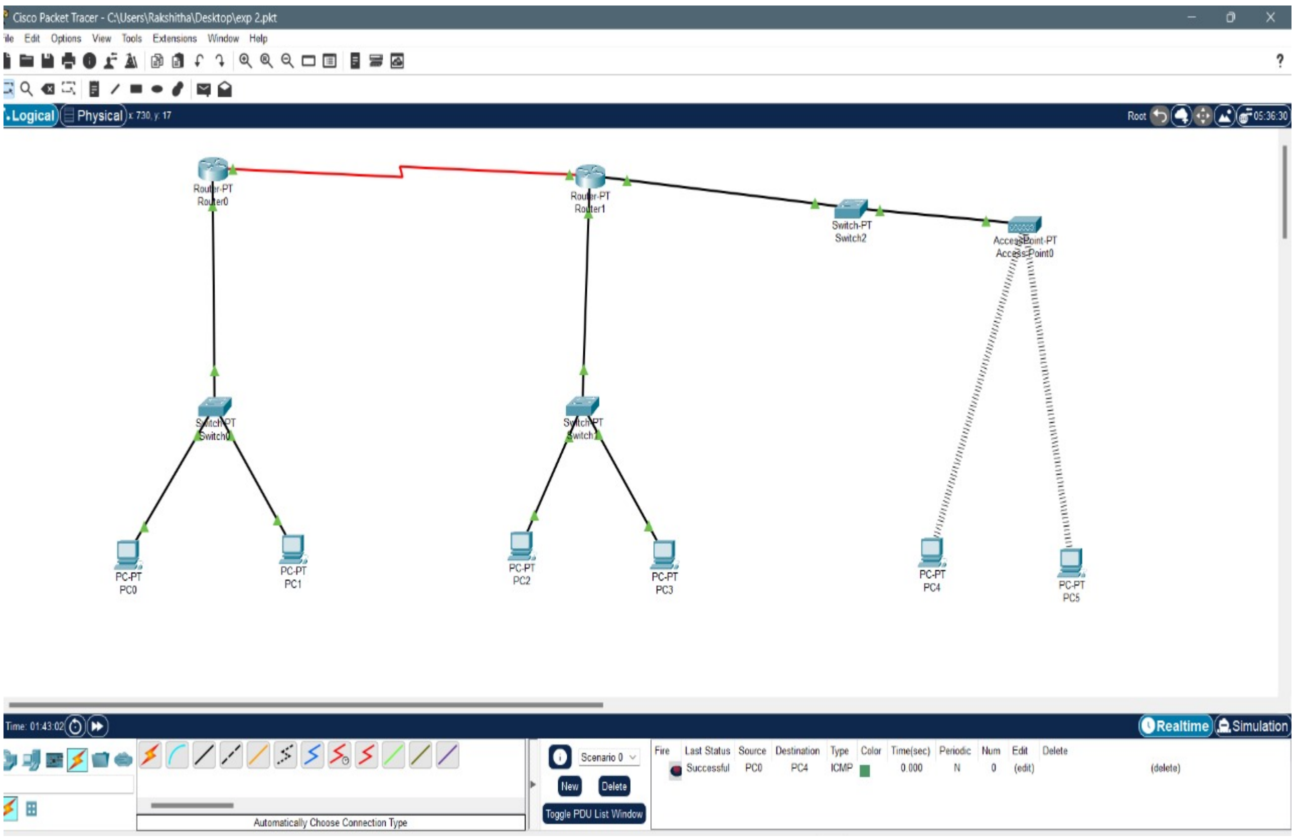
Fastethernet 0/1 :

IP:192.168.2.222, Subnet mask : 255.255.255.224

Se 2/0 :

IP:192.168.2.226, Subnet mask : 255.255.255.252

Simulation Output/Output:



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC4	ICMP	🟩	0.000	N	0	(edit)	(delete)

Result:

Thus, a Campus Area Network was established, employing subnetting and providing a reliable network for a new PG Block, along with 2 existing blocks, UG block and Admin Block. The design was implemented and communication simulated successfully, on Cisco Packet tracer platform.

Ex No: 3

Date:30/09/2024

DESIGN OF SUITABLE NETWORK TOPOLOGY FOR A MULTIFLOOR HYPERMARKET

Aim:

Build a network architecture for a Hypermarket with ‘m’ floors and ‘n’ systems. Choose and design the right Topology to build the architecture.

Theory:

Designing a network architecture for a hypermarket with multiple floors and systems involves considering several factors such as scalability, redundancy, security, and ease of management. Below is a step-by-step guide to building an effective network topology for such an environment:

1. Network Requirements

1. **Scalability**: The network should accommodate future expansion with minimal disruption.
2. **Redundancy**: To ensure high availability and reliability.
3. **Security**: Protect sensitive customer and business data.
4. **Performance**: Ensure fast and reliable access to network resources.
5. **Segmentation**: Differentiate between different departments and functions.
6. **Wireless Connectivity**: Provide Wi-Fi access to customers and staff.

2. Suggested Network Topology

Hierarchical Network Design: This topology divides the network into three main layers: Core, Distribution, and Access. It is scalable, manageable, and provides redundancy.

A. Core Layer

- **Function**: High-speed backbone connecting different distribution switches.
- **Components**:
  - High-performance switches/routers.
  - Redundant links and devices for fault tolerance.

B. Distribution Layer

- **Function**: Routing, filtering, and WAN access.
- **Components**:
    - Layer 3 switches for inter-VLAN routing.
    - Redundancy through dual links or stacking.

C. Access Layer

- **Function**: Directly connects to the end-user devices.
- **Components**:
    - Layer 2 switches.
    - Wireless Access Points (WAPs) for Wi-Fi.

3. Network Layout

Each Floor

- **Access Layer**:
    - Place Layer 2 switches on each floor.
    - Connect devices (PCs, PoS systems, IP cameras) to these switches.
    - Deploy Wireless Access Points for Wi-Fi coverage.

Inter-Floor Connectivity

- **Distribution Layer**:
    - Connect each floor’s access switches to distribution switches using fiber optic cables for high bandwidth and low latency.
    - Use multiple distribution switches for redundancy.

Data Center / Server Room

- **Core Layer**:
    - Connect distribution switches to core switches in the data center.
    - Host servers, storage, and network management systems.

4. Network Segmentation

- **LANs**:

  - Create LANs for different departments (e.g., marketing, inventory, management).

Procedure:

For the hierarchical star topology network design, selected for hypermarket with 2 floors and minimum of 2 LANs (Sales, Inventory) per floor, the following procedure needs to be followed.

- Open a new workspace to create a network in a packet tracer.
- From end devices choose PCs, laptops, mobile phone, tab, switches for each LAN and PT routers per floor and interfloor router.
- Connect these devices via cable.

For ground floor network:

- Configure the static IPs for the four end devices with 192.168.5.2, 192.168.5.3 etc for end devices connected to Ground floor inventory LAN, and add subnet mask as 255.255.255.0.
- Connect above end devices to Ground floor router through a switch. Click ground floor router, go to Config, and select FAST ETHERNET 1/0 . Set static IP as 192.168.5.1.
- Set 192.168.5.1 as default gateway for all end devices connected to this LAN.
- Configure the static IPs for the four end devices with 192.168.4.2, 192.168.4.3 etc for end devices connected to Ground floor sales LAN, and add subnet mask as 255.255.255.0.
- Connect above end devices to same Ground floor router through a switch. Click ground floor router, go to Config, and select FAST ETHERNET 0/0. Set static IP as 192.168.4.1.
- Set 192.168.4.1 as default gateway for all end devices connected to this LAN.
- To establish successful communication between switch and router, click router, go to Config, select FAST ETHERNET 0/0, click CLI, type NO SHUTDOWN and enter. Green arrows will appear between switch and router. Repeat the same for FAST ETHERNET 1/0.
- To connect wireless devices through access point, click access point, go to Config, select Port 1, give a name for SSID, Eg.ABC. Under authentication select WEP and enter a 10-digit WEP key. Eg. 0123456789.
- Click on the wireless device, go to Config, under global settings, give corresponding router static IP as default gateway.
- Under interface, select Wireless, enter the same SSID and WEP key entered for the access point.

For First Floor network:

- Configure the static IPs for the four end devices with 192.168.3.2, 192.168.3.3 etc for end devices connected to Ground floor inventory LAN, and add subnet mask as 255.255.255.0.
- Connect above end devices to Ground floor router through a switch. Click ground floor router, go to Config, and select FAST ETHERNET 1/0 . Set static IP as 192.168.3.1.
- Set 192.168.3.1 as default gateway for all end devices connected to this LAN.
- Configure the static IPs for the four end devices with 192.168.2.2, 192.168.2.3 etc for end devices connected to Ground floor sales LAN, and add subnet mask as 255.255.255.0.
- Connect above end devices to same Ground floor router through a switch. Click ground floor router, go to Config, and select FAST ETHERNET 0/0. Set static IP as 192.168.2.1.
- Set 192.168.2.1 as default gateway for all end devices connected to this LAN.
- To establish successful communication between switch and router, click router, go to Config, select FAST ETHERNET 0/0, click CLI, type NO SHUTDOWN and enter. Green arrows will appear between switch and router. Repeat the same for FAST ETHERNET 1/0.
- To connect wireless devices through access point, click access point, go to Config, select Port 1, give a different name for SSID, Eg.CBA. Under authentication select WEP and enter a 10-digit WEP key. Eg. 0123456789.
- Click on the wireless device, go to Config, under global settings, give corresponding router static IP as default gateway.
- Under interface, select Wireless, enter the same SSID and WEP key entered for the access point.

Inter floor router to router static routing:

Connect the interfloor router to ground floor and first floor routers, through Serial DTE cable. Select Serial 2/0 of Interfloor router, to connect to Serial 2/0 of 1st floor router. Select Serial 3/0 of Interfloor router, to connect to Serial 2/0 of Groundfloor router.

Click interfloor router, go to Config, select Serial 2/0, set IP address as 10.0.0.1 and subnet mask as 255.0.0.0. Select Serial3/0, set IP address as 11.0.0.1 and subnet mask as 255.0.0.0.

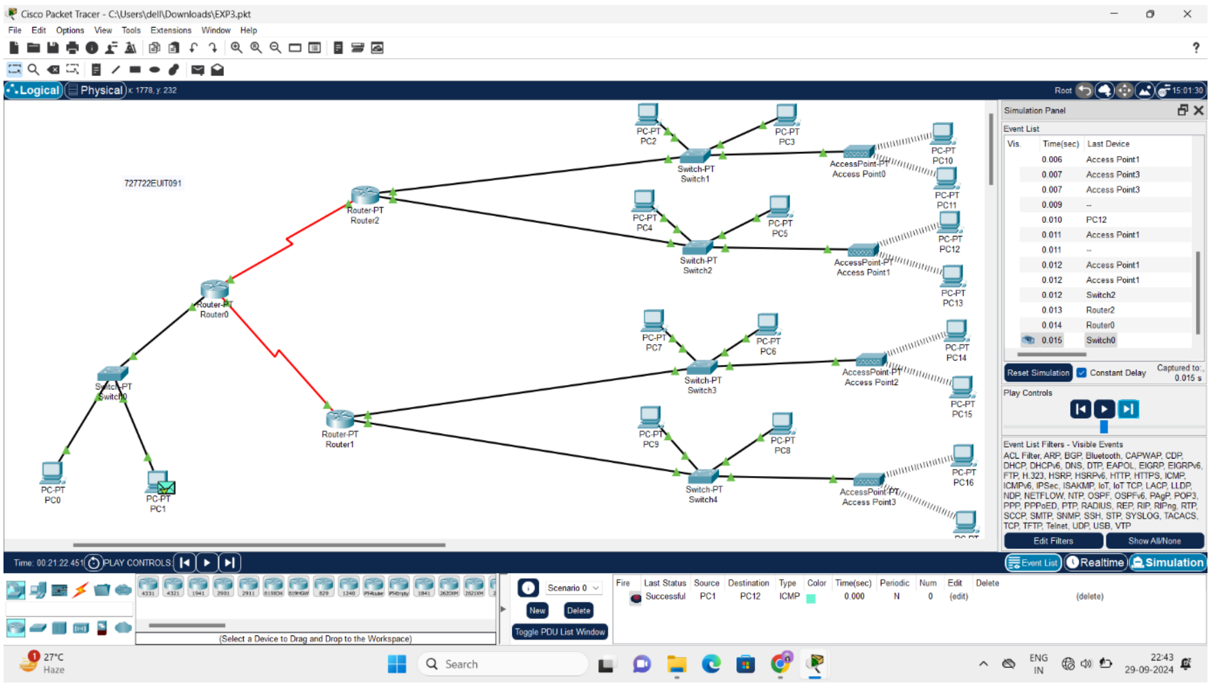
Click ground floor router, go to Config, select Serial 2/0, set IP address as 11.0.0.2 and subnet mask as 255.0.0.0. Click 1st floor router, go to Config, select Serial 2/0, set IP address as 10.0.0.2 and subnet mask as 255.0.0.0.

Click interfloor router, go to Config, under Routing, Click Static. Add the routes of packet transmission from interfloor router to destination subnet.
Repeat the same for ground floor and 1st floor routers.

To test the network:

- Use the ping command in the command prompt to check for successful transmission between the two networks.
- If the IP address gets pinged in other system successfully then the network is working successfully.
- Or alternatively, the simulation of simple PDU, from source device to destination device, can also be checked.

Simulation Output/Output:


Cisco Packet Tracer - C:\Users\dell\Downloads\EXP3.pkt

Event List

Vis.	Time(sec)	Last Device
	0.006	Access Point1
	0.007	Access Point3
	0.007	Access Point3
	0.009	--
	0.010	PC12
	0.011	Access Point1
	0.011	--
	0.012	Access Point1
	0.012	Access Point1
	0.012	Switch2
	0.013	Router2
	0.014	Router0
	0.015	Switch0

Captured to: 0.015 s

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC1	PC12	ICMP	🟩	0.000	N	0	(edit)	(delete)

Result:

Thus, the hierarchical network architecture which provides a robust, scalable, and secure solution for a hypermarket with multiple floors and systems was designed and communication simulated successfully, on Cisco Packet tracer platform.

Expt 4:

Consider a network with data frames transmitted from sender to receiver. Design a code for bit stuffing and un-stuffing mechanism considering the given input pattern.

Bit stuffing is a process used in data transmission protocols to prevent confusion with control information (like frame delimiters). When the sender detects a specific pattern in the data that could be mistaken for a control signal, it inserts a bit to break up the pattern. The receiver then removes this extra bit to retrieve the original data.

Let's assume we're working with a basic pattern like 01111110 (often used as a frame delimiter in protocols like HDLC), and we want to avoid having six consecutive 1 bits (111111) in our data.

Here's how you can implement bit stuffing and unstuffing in Java:

JAVA CODE:

```java
import java.util.Scanner;

public class BitStuffing {

    // Method to perform bit stuffing
    public static String bitStuffing(String data) {
        StringBuilder stuffedData = new StringBuilder();
        int consecutiveOnes = 0;

        for (int i = 0; i < data.length(); i++) {
            char bit = data.charAt(i);
            stuffedData.append(bit);

            if (bit == '1') {
                consecutiveOnes++;
                if (consecutiveOnes == 5) {
                    // Insert a '0' after five consecutive '1's
                    stuffedData.append('0');
                    consecutiveOnes = 0;
                }
            } else {
```

```
            consecutiveOnes = 0;
        }
    }

    return stuffedData.toString();
}

// Method to perform bit unstuffing
public static String bitUnstuffing(String stuffedData) {
    StringBuilder unstuffedData = new StringBuilder();
    int consecutiveOnes = 0;

    for (int i = 0; i < stuffedData.length(); i++) {
        char bit = stuffedData.charAt(i);
        unstuffedData.append(bit);

        if (bit == '1') {
            consecutiveOnes++;
            if (consecutiveOnes == 5) {
                // Skip the stuffed '0' after five consecutive '1's
                i++;  // Skip the next character (should be '0')
                consecutiveOnes = 0;
            }
        } else {
            consecutiveOnes = 0;
        }
    }

    return unstuffedData.toString();
}
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input binary data frame
        System.out.print("Enter the binary data frame: ");
        String data = scanner.nextLine();

        // Perform bit stuffing
        String stuffedData = bitStuffing(data);
        System.out.println("Stuffed Data: " + stuffedData);

        // Perform bit unstuffing
        String unstuffedData = bitUnstuffing(stuffedData);
        System.out.println("Unstuffed Data: " + unstuffedData);

        scanner.close();
    }
}
```

How the Code Works:

1. Bit Stuffing:
   - The code iterates through the input binary string (data).
   - It counts the consecutive 1s. If it detects five consecutive 1s, it stuffs (inserts) a 0 bit after them to break the pattern.
   - The resulting string (stuffedData) is the transmitted data that includes these extra 0 bits.
2. Bit Unstuffing:
   - When the receiver gets the stuffedData, it checks for sequences of five consecutive 1s.
   - If it finds such a sequence, it skips the next 0 (the stuffed bit) to retrieve the original data frame.

Example Usage:

- Input:

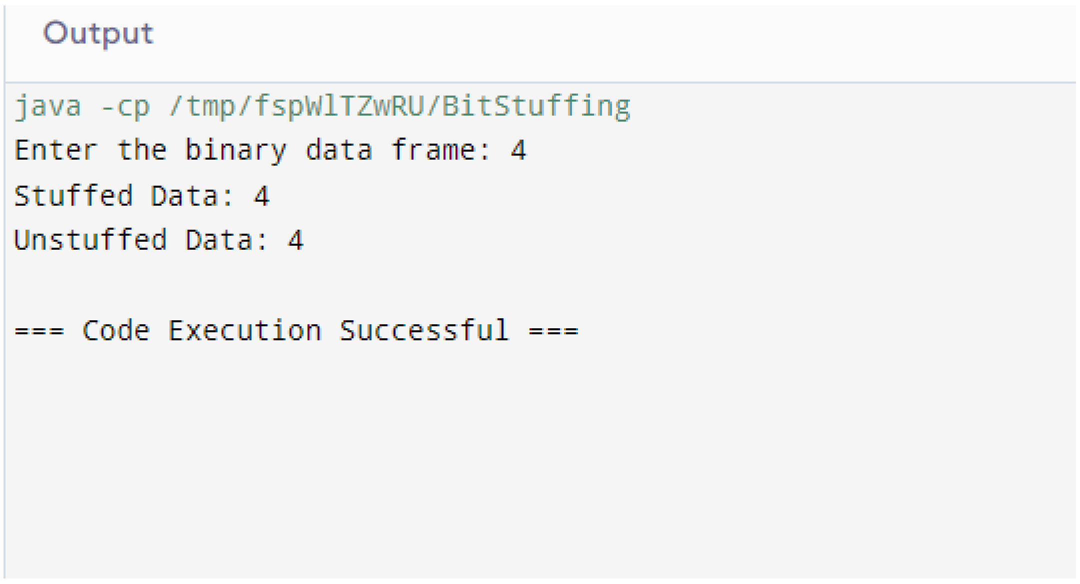
   - Original Data: 011111101111110
- Output:
   - Stuffed Data: 0111110101111100
   - Unstuffed Data: 011111101111110

The stuffed data contains the extra 0s inserted after sequences of five consecutive 1s. The unstuffing process removes these 0s, returning the data to its original form.

Application:

This method is particularly useful in communication protocols to ensure that patterns in the data don't interfere with control sequences, such as frame delimiters.

Output:



```
java -cp /tmp/fspWlTZwRU/BitStuffing
Enter the binary data frame: 4
Stuffed Data: 4
Unstuffed Data: 4

=== Code Execution Successful ===
```

Ex No: 5

Date:30/09/2024

ERROR DETECTION IN DATA FRAME TRANSMISSION BETWEEN SENDER AND RECEIVER USING CHECKSUM.

Aim:

To develop a java code, for data frame transmission, to calculate the checksum value and to check whether the data frames are received in the receiver without error, in a network, assuming that the sender sends 'm' frames each of 'n' bits.

Theory:

Checksum is the error detection method used by upper layer protocols and is considered to be more reliable than LRC, VRC and CRC. This method makes the use of Checksum Generator on Sender side and Checksum Checker on Receiver side.

At the Sender side, the data is divided into equal subunits of n bit length by the checksum generator. This bit is generally of 16-bit length. These subunits are then added together using one's complement method. This sum is of n bits. The resultant bit is then complemented. This complemented sum which is called checksum is appended to the end of original data unit and is then transmitted to receiver.

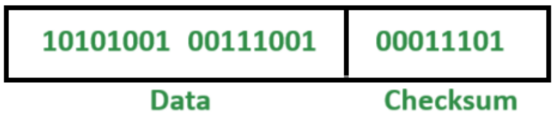
The Receiver after receiving data + checksum passes it to checksum checker. Checksum checker divides this data unit into various subunits of equal length and adds all these subunits. These subunits also contain checksum as one of the subunits. The resultant bit is then complemented. If the complemented result is zero, it means the data is error-free. If the result is non-zero it means the data contains an error and Receiver rejects it.

Example – If the data unit to be transmitted is 10101001 00111001, the following procedure is used at Sender site and Receiver site.

Sender Site:

10101001	subunit 1
00111001	subunit 2
11100010	sum (using 1s complement)
00011101	checksum (complement of sum)

Data transmitted to Receiver is:



Receiver Site :

```
10101001       subunit 1
00111001       subunit 2
00011101       checksum
11111111          sum
00000000    sum's complement
```

Result is zero, it means no error.

Algorithm:

1. Start

- Initialize the program by taking input from the user for the number of frames (m) and the number of bits per frame (n).

2. Input Data Frames

- Ask the user to input m frames, each consisting of n bits.
- Store these frames in an array frames[].

3. Checksum Calculation at Sender

- Initialize the checksum with the value of the first frame.
- For each subsequent frame:
    - Add the binary values of the current checksum and the frame using binary addition.
    - Handle overflow (if the result exceeds n bits), add the carry back to the least significant bit.
- After all frames are added, compute the one's complement of the final sum to get the checksum.

4. Transmit Frames and Checksum

- Display the calculated checksum to the user.
- Ask the user to input the received frames, including the checksum (i.e., m+1 frames in total).

5. Checksum Verification at Receiver

- Initialize the sum with the first received frame.

- For each subsequent received frame (including the checksum):
    - Add the binary values of the current sum and the frame.
    - Handle overflow if needed.
- Check if the final sum is all ones:
    - If yes, the data was received without error.
    - If no, there was an error during transmission.

6. End

Program:

```
import java.util.Scanner;
public class ChecksumCalculator {
    // Method to add two binary strings
    public static String addBinary(String a, String b) {
        StringBuilder result = new StringBuilder();
        int carry = 0;

        // Ensure both binary strings have the same length by padding with zeros using
StringBuilder
        StringBuilder aBuilder = new StringBuilder(a);
        StringBuilder bBuilder = new StringBuilder(b);

        // Padding 'a' with leading zeros if needed
        while (aBuilder.length() < bBuilder.length()) {
            aBuilder.insert(0, '0');
        }

        // Padding 'b' with leading zeros if needed
        while (bBuilder.length() < aBuilder.length()) {
            bBuilder.insert(0, '0');
        }

        // Perform binary addition from right to left
        for (int i = aBuilder.length() - 1; i >= 0; i--) {
            int bitA = aBuilder.charAt(i) - '0';
            int bitB = bBuilder.charAt(i) - '0';
            int sum = bitA + bitB + carry;
            result.append(sum % 2); // Append the sum mod 2 (0 or 1)
            carry = sum / 2;        // Calculate carry (0 or 1)
        }
        // If there's still a carry, append it
        if (carry > 0) {
            result.append(carry);
        }

        // Reverse the result to get the correct order
        return result.reverse().toString();
    }

    // Method to calculate the one's complement of a binary string
    public static String onesComplement(String binary) {
        StringBuilder complement = new StringBuilder();
        for (char bit : binary.toCharArray()) {
            complement.append(bit == '0' ? '1' : '0');
        }
        return complement.toString();
    }

    // Method to check if all bits are ones (used for validation at receiver side)
    public static boolean allOnes(String binary) {
        for (char bit : binary.toCharArray()) {
            if (bit != '1') {
                return false;
            }
        }
        return true;
    }

    // Main method to calculate checksum and verify frames
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input number of frames and bits per frame
        System.out.println("Enter the number of frames: ");
        int m = scanner.nextInt();
        System.out.println("Enter the number of bits per frame: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        // Input the frames
        String[] frames = new String[m];
        System.out.println("Enter the frames (each frame should be " + n + " bits): ");
        for (int i = 0; i < m; i++) {
            frames[i] = scanner.nextLine();
        }

        // Sender side: calculate the checksum
        String checksum = frames[0];
        for (int i = 1; i < m; i++) {
            checksum = addBinary(checksum, frames[i]);
            // Handle carry overflow (wrap around addition)
            if (checksum.length() > n) {
                checksum = addBinary(checksum.substring(1), "1");
            }
        }
        checksum = onesComplement(checksum);
        System.out.println("Calculated checksum at sender: " + checksum);

        // Receiver side: verify the frames including checksum
        System.out.println("Enter received frames (including the checksum): ");
        String[] receivedFrames = new String[m + 1];
        for (int i = 0; i <= m; i++) {
            receivedFrames[i] = scanner.nextLine();
        }

        // Add the received frames
        String receivedSum = receivedFrames[0];
        for (int i = 1; i <= m; i++) {
            receivedSum = addBinary(receivedSum, receivedFrames[i]);
            // Handle carry overflow (wrap around addition)
            if (receivedSum.length() > n) {
                receivedSum = addBinary(receivedSum.substring(1), "1");
            }
        }

        // Check if the result is all ones (no error)
        if (allOnes(receivedSum)) {
            System.out.println("No error in received data.");
        } else {
            System.out.println("Error detected in received data.");
        }
        scanner.close();
    }
}
```

Sample Input & Output:

10011001

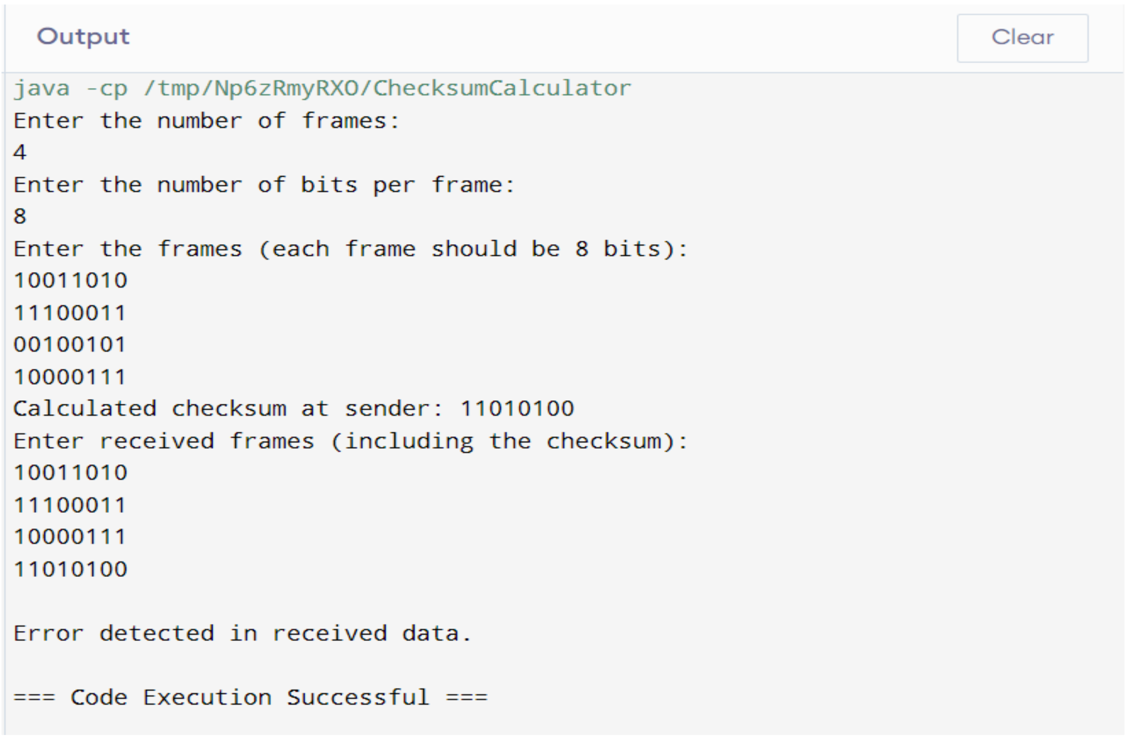
11100010

00100100

10000100

Checksum:11011010

Screenshot of output:



```
java -cp /tmp/Np6zRmyRXO/ChecksumCalculator
Enter the number of frames:
4
Enter the number of bits per frame:
8
Enter the frames (each frame should be 8 bits):
10011010
11100011
00100101
10000111
Calculated checksum at sender: 11010100
Enter received frames (including the checksum):
10011010
11100011
10000111
11010100

Error detected in received data.

=== Code Execution Successful ===
```

Result:

Thus the Java code to calculate the checksum value at sender side, and to check whether the data frames are received in the receiver end without error, for a dataframe transmission in a network, where sender sends ‘m’ frames each of ‘n’ bits, was developed and executed successfully in IntelliJ Idea Java IDE.

Ex No: 6

Date:

IMPLEMENTATION OF SLIDING WINDOW PROTOCOL

Aim:

Assume a network scenario where N is the sender's window size = 'm' & damaged frame is 'x';

(a) At this situation, the sender sends frame 1 to 'm' before receiving the knowledge of frame 1. All the frames are numbered to deal with the most and duplicate frames. If the sender does not receive the receiver's acknowledgement, then all the frames available in the current window ie., 1 to 'm' will be retransmitted.
(b) The sender will resend only the damaged frame ie., frame 'x'

(a) Go- Back-N Protocol

Theory:

Go-Back-N ARQ uses the concept of protocol pipelining, i.e. the sender can send multiple frames before receiving the acknowledgement for the first frame. There is a finite number of frames, and the frames are numbered sequentially. The number of frames that can be sent depends on the sender's window size.

Go-Back-N ARQ is a specific instance of the automatic repeat request (ARQ) protocol, in which the sending process continues to send a number of frames specified by a *window size* even without receiving an acknowledgement (ACK) packet from the receiver. It is a special case of the general sliding window protocol with the transmit window size of N and receive window size of even 1. It can transmit N frames to the peer before requiring an ACK.

Algorithm:

SENDER:

Assuming Total no. of frames to be transmitted to be **'nf',** with a window size of **'sws'**

Initialize Server:

- Create a ServerSocket to listen for connections on a specific port (port 10 in this case).

- Accept a connection from the client (Socket s).

Input Setup:

- Set up BufferedReader objects to read inputs from the console (user) and from the client.
- Set up an array sbuff[] to store the frames to be sent.

Window Size and Frame Information:

- Ask the user to input the sliding window size (sws).
- Ask the user to input the total number of frames to be sent (nf).
- Send the total number of frames (nf) to the receiver.

Frame Data Input:

- Ask the user to input the actual messages to send for each frame and store them in sbuff[].

Sliding Window Transmission:

- Initialize base (the starting frame of the window) to 0 and nextFrame to 0 (the next frame to be sent).
- While not all frames are acknowledged (base < nf):
    - **Send Frames:**
        - Send all frames within the sliding window where nextFrame - base < sws.
        - For each frame, print the frame number and message, and send it to the receiver.
    - **Wait for Acknowledgment:**
        - After sending the frames, wait for an acknowledgment from the receiver.
        - Implement a timeout of 10 seconds to detect if the acknowledgment is received within the time frame.
    - **Acknowledgment Handling:**

    - If an acknowledgment is received for a frame (ano), move the window base to ano + 1.
    - If all frames are acknowledged, terminate the process.
    - If no acknowledgment is received within the timeout, retransmit all frames starting from base.

Frame Resending (Timeout Handling):

- If the acknowledgment is not received within the timeout period, retransmit all the frames in the window starting from base.

Termination:

- Once all frames are acknowledged, print a message indicating that all frames have been successfully acknowledged.
- Close the socket connection.

<u>RECEIVER:</u>

Assuming the **damaged frame number to be 2.**

Establish Connection:

- Create a Socket to connect to the sender on the same machine (InetAddress.getLocalHost()) using port 10.

Input and Output Setup:

- Set up BufferedReader to receive data (frames) from the sender.
- Set up a PrintStream to send acknowledgment back to the sender.

Initialization:

- Initialize expectedFrame to 0, which represents the frame number the receiver is expecting.
- Create a buffer rbuf[] to store the received frames.
- Initialize a flag damagedFrameReceived to simulate a damaged frame (in this case, assume that frame 2 is damaged).

Receive Total Number of Frames:

- Receive the total number of frames (totalFrames) from the sender.
- Print the total number of frames expected.

Receiving and Handling Frames:

- While expectedFrame is less than totalFrames, keep receiving frames from the sender.
    - **Receive Frame:**
        - Read the incoming message from the sender and split it into two parts: the frame number (frameNumber) and the frame content (frameContent).
    - **Simulate Frame Damage:**
        - If the frame number is the predefined damaged frame (damagedFrame) and it hasn't been previously received:
            - Simulate the damage by not sending an acknowledgment for the frame.
            - Skip the rest of the loop to avoid further processing of the damaged frame.
    - **Frame Handling:**
        - If the received frame is the expected frame (frameNumber == expectedFrame):
            - Store the frame content in the buffer rbuf[].
            - Print a message indicating the correct receipt of the frame.
            - Send an acknowledgment (expectedFrame) to the sender for the successfully received frame.
            - Increment the expectedFrame to move to the next frame.

Termination:

- Once all frames have been successfully received and acknowledged (expectedFrame >= totalFrames), print a message indicating that all frames have been received.

- Close the socket connection.

Program:

SENDER:

```
import java.net.*;
import java.io.*;

public class SLIDSENDER {
    public static void main(String[] a) throws Exception {
        ServerSocket ser = new ServerSocket(10);
        Socket s = ser.accept();

        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        BufferedReader in1 = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        String sbuff[] = new String[100];
        PrintStream p;

        int sws, nf, ano, i, base = 0, nextFrame = 0;
        boolean ackReceived = false;

        // Get the window size input from the user
        System.out.print("Enter the window size: ");
        sws = Integer.parseInt(in.readLine());

        // Get the total number of frames
        System.out.print("Enter the total number of frames to send: ");
        nf = Integer.parseInt(in.readLine());

        // Inform the receiver about the total number of frames
        p = new PrintStream(s.getOutputStream());
        p.println(nf);

        // Reading all frames from the user and storing them in the buffer
        System.out.println("Enter " + nf + " Messages to send:");
        for (i = 0; i < nf; i++) {
            sbuff[i] = in.readLine();
        }

        do {
            // Send frames within the window
            while (nextFrame < nf && (nextFrame - base) < sws) {
                p = new PrintStream(s.getOutputStream());
                System.out.println("Sending Frame " + nextFrame + ": " +
sbuff[nextFrame]);
                p.println(nextFrame + ":" + sbuff[nextFrame]);  // Send frame number and
message
                nextFrame++;
            }

            // Waiting for acknowledgment
            System.out.println("Waiting for acknowledgment...");

            long startTime = System.currentTimeMillis();
            int timeout = 5000;  // 5 seconds timeout

            while (!ackReceived && (System.currentTimeMillis() - startTime) < timeout) {
                // Check if acknowledgment is received
                if (in1.ready()) {
                    ano = Integer.parseInt(in1.readLine());
                    System.out.println("Acknowledgment received for Frame " + ano);

                    // Slide the window base to the next unacknowledged frame
                    base = ano + 1;
                    ackReceived = true;

                    // If all frames are acknowledged, exit the loop
                    if (base >= nf) {
                        System.out.println("All frames acknowledged.");
                        break;
                    }
                }
            }

            // If acknowledgment is not received, timeout occurred
            if (!ackReceived) {
                System.out.println("Timeout! No acknowledgment received for Frame " +
base);
                System.out.println("Retransmitting all frames in the window starting from
Frame " + base);
```

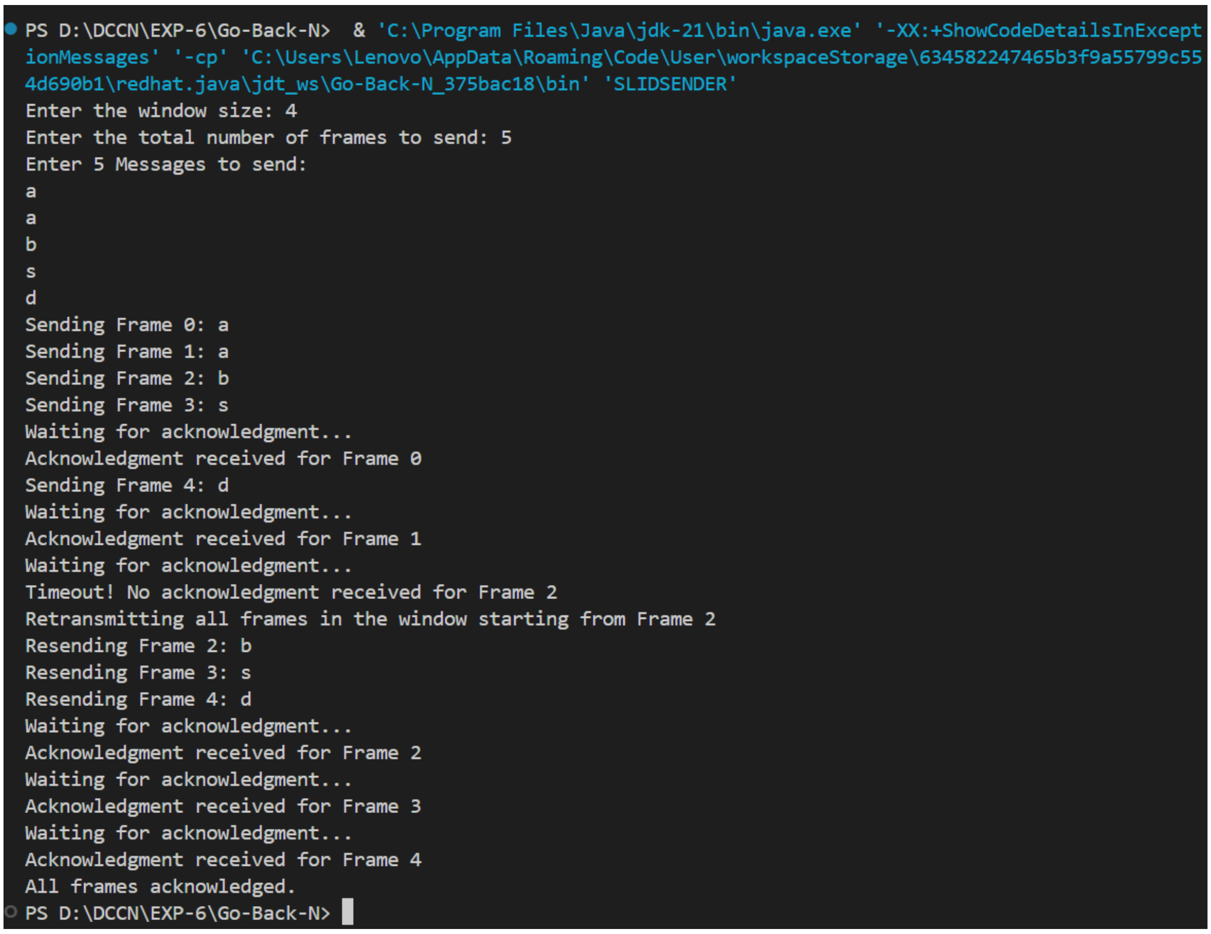
```
            // Resend all frames in the window starting from base
            for (i = base; i < nextFrame; i++) {
                p = new PrintStream(s.getOutputStream());
                System.out.println("Resending Frame " + i + ": " + sbuff[i]);
                p.println(i + ":" + sbuff[i]);  // Send frame number and message
            }
        } else {
            ackReceived = false;
        }

    } while (base < nf);  // Continue until all frames are acknowledged

    s.close();
  }
}
```

RECEIVER:

```
import java.net.*;
import java.io.*;

class SLIDRECEIVER {
    public static void main(String[] a) throws Exception {
        Socket s = new Socket(InetAddress.getLocalHost(), 10);
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        PrintStream p = new PrintStream(s.getOutputStream());

        int expectedFrame = 0, totalFrames;
        String rbuf[] = new String[100];
        int damagedFrame = 2;  // Assume frame 2 is damaged on first attempt
        boolean damagedFrameReceived = false;  // Flag to track if the damaged frame
has been received

        System.out.println("Receiver is ready...");

        // Read the total number of frames to be received (sent by the sender)
        totalFrames = Integer.parseInt(in.readLine());
        System.out.println("Total number of frames to receive: " + totalFrames);

        while (expectedFrame < totalFrames) {
            // Receive the frame from the sender
            String frameMessage = in.readLine();
            // Split the incoming message to get the frame number and content
            String[] parts = frameMessage.split(":");
            int frameNumber = Integer.parseInt(parts[0]);
            String frameContent = parts[1];

            // Simulate frame damage for the first attempt
            if (frameNumber == damagedFrame && !damagedFrameReceived) {
                System.out.println("Frame " + frameNumber + " is damaged. No
acknowledgment sent.");
                damagedFrameReceived = true;  // Mark that the damaged frame has
been received
                continue;  // Don't send an acknowledgment for the damaged frame
            }

            // If the frame is expected
            if (frameNumber == expectedFrame) {
                System.out.println("Received Frame " + expectedFrame + ": " +
frameContent);
                rbuf[expectedFrame] = frameContent;

                // Send acknowledgment for the correctly received frame
                System.out.println("Sending acknowledgment for Frame " +
expectedFrame);
                p.println(expectedFrame);
                expectedFrame++;
            }
        }

        System.out.println("All frames received and acknowledged.");
        s.close();  // Close the socket when done
    }
}
```

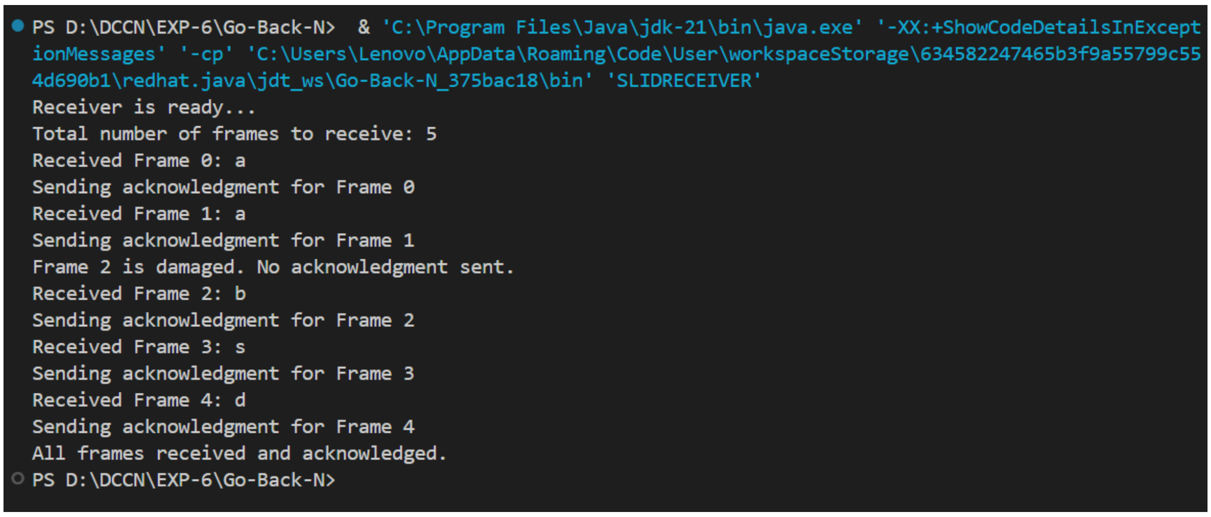
Screenshot of output:

Sender:



```
PS D:\DCCN\EXP-6\Go-Back-N>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\634582247465b3f9a55799c554d690b1\redhat.java\jdt_ws\Go-Back-N_375bac18\bin' 'SLIDSENDER'
Enter the window size: 4
Enter the total number of frames to send: 5
Enter 5 Messages to send:
a
a
b
s
d
Sending Frame 0: a
Sending Frame 1: a
Sending Frame 2: b
Sending Frame 3: s
Waiting for acknowledgment...
Acknowledgment received for Frame 0
Sending Frame 4: d
Waiting for acknowledgment...
Acknowledgment received for Frame 1
Waiting for acknowledgment...
Timeout! No acknowledgment received for Frame 2
Retransmitting all frames in the window starting from Frame 2
Resending Frame 2: b
Resending Frame 3: s
Resending Frame 4: d
Waiting for acknowledgment...
Acknowledgment received for Frame 2
Waiting for acknowledgment...
Acknowledgment received for Frame 3
Waiting for acknowledgment...
Acknowledgment received for Frame 4
All frames acknowledged.
PS D:\DCCN\EXP-6\Go-Back-N>
```

Receiver:



```
PS D:\DCCN\EXP-6\Go-Back-N>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\634582247465b3f9a55799c554d690b1\redhat.java\jdt_ws\Go-Back-N_375bac18\bin' 'SLIDRECEIVER'
Receiver is ready...
Total number of frames to receive: 5
Received Frame 0: a
Sending acknowledgment for Frame 0
Received Frame 1: a
Sending acknowledgment for Frame 1
Frame 2 is damaged. No acknowledgment sent.
Received Frame 2: b
Sending acknowledgment for Frame 2
Received Frame 3: s
Sending acknowledgment for Frame 3
Received Frame 4: d
Sending acknowledgment for Frame 4
All frames received and acknowledged.
PS D:\DCCN\EXP-6\Go-Back-N>
```

Result:

Thus Go-Back-N sliding window protocol has been simulated successfully in Java IDE.

(b) Selective Repeat window protocol

Theory

• In sliding window method, multiple frames are sent by sender at a time before needing an acknowledgment.

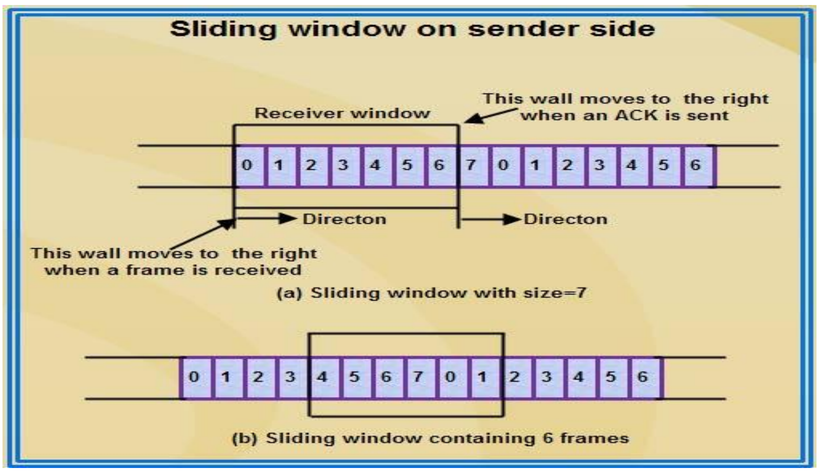
• Multiple frames sent by source are acknowledged by receiver using a single ACK frame.

Sender:

At the beginning of a transmission, the sender's window contains n-l frames.

• As the frames are sent by source, the left boundary of the window moves inward, shrinking the size of window. This means if window size is w, if four frames are sent by source after the last acknowledgment, then the number of frames left in window is w-4.

• When the receiver sends an ACK, the source's window expand i.e. (right boundary moves outward) to allow in a number of new frames equal to the number of frames acknowledged by that ACK.


Sliding window on sender side
Receiver window
This wall moves to the right when an ACK is sent
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6
Directon
Directon
This wall moves to the right when a frame is received
(a) Sliding window with size=7
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6
(b) Sliding window containing 6 frames

Receiver:

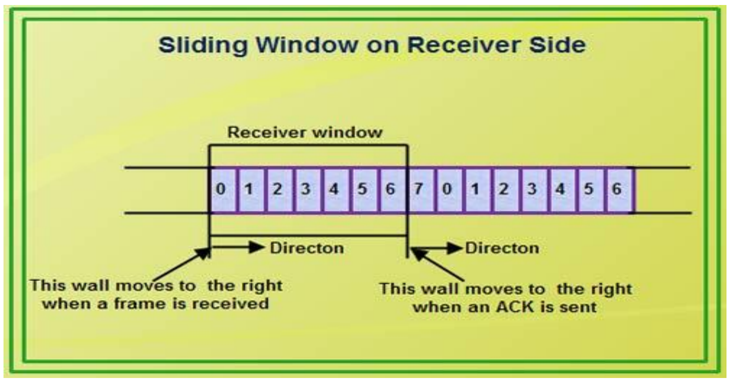
At the beginning of transmission, the receiver's window contains n-1 spaces for frame but not the frames.

• As the new frames come in, the size of window shrinks.

• Therefore the receiver window represents not the number of frames received but the number of frames that may still be received without an acknowledgment ACK must be sent.

• Given a window of size w, if three frames are received without an ACK being returned, the number of spaces in a window is w-3.

• As soon as acknowledgment is sent, window expands to include the number of frames equal to the number of frames acknowledged.


Sliding Window on Receiver Side
Receiver window
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6
Directon
Directon
This wall moves to the right when a frame is received
This wall moves to the right when an ACK is sent

Algorithm:

<u>SENDER:</u>

Initialization:

- Create a ServerSocket and accept a connection.
- Initialize input/output streams for communication with the receiver.
- Create an array sbuff[] to store messages (frames).
- Create a boolean array ack[] to keep track of whether each frame has been acknowledged.
- Define variables for:
    - sws: the sliding window size
    - nf: the total number of frames
    - base: the lowest numbered frame in the window that has yet to be acknowledged
    - nextFrame: the next frame to be sent

Get User Input:

- Prompt the user to enter the window size (sws).
- Prompt the user to enter the total number of frames (nf).
- Send the total number of frames to the receiver.

Read Frames:

- For i from 0 to nf-1, do the following:
    - Read the message from the user and store it in sbuff[i].
    - Set ack[i] to false (indicating that the frame is not yet acknowledged).

S**ending Loop**:

- Repeat until all frames are acknowledged (base < nf):
    1. **Send Frames**:
        - While nextFrame < nf and the difference (nextFrame - base) < sws:
            - If ack[nextFrame] is false:
                - Print a message indicating the frame is being sent.
                - Send the frame (nextFrame + ":" + sbuff[nextFrame]) to the receiver.
            - Increment nextFrame.
    2. **Wait for Acknowledgment**:
        - Print a message indicating the sender is waiting for acknowledgment.
        - Record the current time (startTime).
        - Set timeout to 5000 milliseconds (5 seconds).
        - While the current time minus startTime is less than timeout:
            - If acknowledgment is ready to be read from the input stream:
                - Read the acknowledgment number (ano) from the receiver.
                - Print a message indicating which frame has been acknowledged.
                - Mark ack[ano] as true (the frame has been acknowledged).
                - **Slide the Window**:

                - While ack[base] is true (the base frame has been acknowledged):
                    - Increment base (slide the window).
            - If base is greater than or equal to nf, print a message indicating that all frames have been acknowledged and exit the loop.

3. **Retransmit Unacknowledged Frames**:
    - For i from base to nextFrame - 1:
        - If ack[i] is false:
            - Print a message indicating the frame is being retransmitted.
            - Resend the unacknowledged frame (i + ":" + sbuff[i]) to the receiver.

Close Connection:

- After all frames are acknowledged, close the socket.

<u>RECEIVER:</u>

Initialization:

- Create a Socket to connect to the sender at a specified port.
- Initialize input/output streams for communication with the sender.
- Create an array rbuf[] to store received messages (frames).
- Create a boolean array received[] to track which frames have been received.
- Define variables for:
    - expectedFrame: the next frame number expected to be received.
    - totalFrames: the total number of frames to be received.
    - damagedFrame: simulate a damaged frame (e.g., frame number 2).
    - damagedFrameReceived: a flag to indicate whether the damaged frame has been received.

Get Ready:

- Print a message indicating that the receiver is ready to receive frames.

Read Total Frames:

- Read the total number of frames to be received from the sender.
- Print the total number of frames.

Receiving Loop:

- Repeat until all frames are received (expectedFrame < totalFrames):
    1. **Receive Frame**:
        - Read the incoming frame message from the sender.
        - Split the incoming message to extract the frame number and its content.
    2. **Check for Damaged Frame**:
        - If the received frame number equals damagedFrame and damagedFrameReceived is false:
            - Print a message indicating that the frame is damaged.
            - Set damagedFrameReceived to true.
            - **Skip Acknowledgment**: Continue to the next iteration of the loop without sending an acknowledgment.
    3. **Process Received Frame**:
        - If the frame number is greater than or equal to expectedFrame and it has not been received yet (received[frameNumber] is false):
            - Print a message indicating that the frame has been received.
            - Store the frame content in rbuf[frameNumber].
            - Mark the frame as received (received[frameNumber] = true).
            - **Send Acknowledgment**:
                - Print a message indicating that an acknowledgment is being sent for the received frame.

            - Send the frame number back to the sender as acknowledgment.
        - **Slide the Window**:
            - While received[expectedFrame] is true (the expected frame has been received):
                - Increment expectedFrame (slide the window to the next expected frame).

Completion:

- Once all frames are received, print a message indicating that all frames have been acknowledged.

Close Connection:

- Close the socket when done.

Program:

<u>SENDER:</u>

```
import java.net.*;
import java.io.*;

public class SLIDSELSENDER {
    public static void main(String[] a) throws Exception {
        ServerSocket ser = new ServerSocket(10);
        Socket s = ser.accept();

        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        BufferedReader in1 = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        String sbuff[] = new String[100];
        boolean ack[] = new boolean[100]; // To keep track of acknowledgments for
each frame
        PrintStream p;

        int sws, nf, ano, i, base = 0, nextFrame = 0;
        // Get the window size input from the user
        System.out.print("Enter the window size: ");
        sws = Integer.parseInt(in.readLine());

        // Get the total number of frames
        System.out.print("Enter the total number of frames to send: ");
        nf = Integer.parseInt(in.readLine());

        // Inform the receiver about the total number of frames
        p = new PrintStream(s.getOutputStream());
        p.println(nf);

        // Reading all frames from the user and storing them in the buffer
        System.out.println("Enter " + nf + " Messages to send:");
        for (i = 0; i < nf; i++) {
            sbuff[i] = in.readLine();
            ack[i] = false; // Initially no frame is acknowledged
        }

        do {
            // Send frames within the window
            while (nextFrame < nf && (nextFrame - base) < sws) {
                if (!ack[nextFrame]) { // Send only if the frame is not acknowledged
                    p = new PrintStream(s.getOutputStream());
                    System.out.println("Sending Frame " + nextFrame + ": " +
sbuff[nextFrame]);
                    p.println(nextFrame + ":" + sbuff[nextFrame]);  // Send frame number
and message
                }
                nextFrame++;
            }

            // Waiting for acknowledgment
```

```
System.out.println("Waiting for acknowledgment...");

long startTime = System.currentTimeMillis();
int timeout = 5000;  // 5 seconds timeout

while ((System.currentTimeMillis() - startTime) < timeout) {
    // Check if acknowledgment is received
    if (in1.ready()) {
        ano = Integer.parseInt(in1.readLine());
        System.out.println("Acknowledgment received for Frame " + ano);

        // Mark the frame as acknowledged
        ack[ano] = true;

        // Slide the window base if base frame is acknowledged
        while (ack[base]) {
            base++;
        }

        // If all frames are acknowledged, exit the loop
        if (base >= nf) {
            System.out.println("All frames acknowledged.");
            break;
        }
    }
}

// Check for unacknowledged frames and retransmit them
for (i = base; i < nextFrame; i++) {
    if (!ack[i]) {
        System.out.println("Retransmitting Frame " + i + ": " + sbuff[i]);
        p.println(i + ":" + sbuff[i]);  // Resend unacknowledged frame
    }
}
```

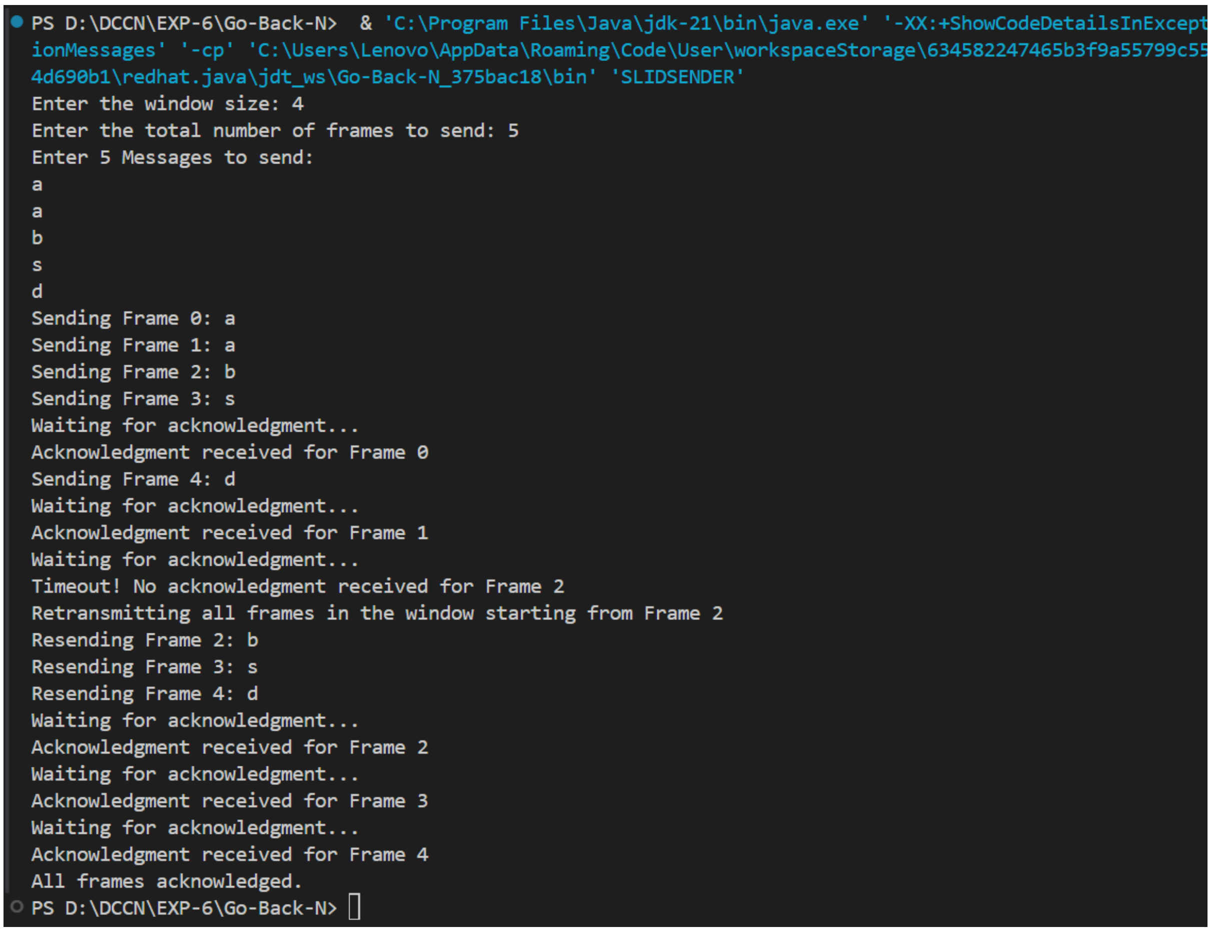
```
        } while (base < nf);  // Continue until all frames are acknowledged


        s.close();
    }
}
```

RECEIVER:

```
import java.net.*;
import java.io.*;

class SLIDSELRECEIVER {
    public static void main(String[] a) throws Exception {
        Socket s = new Socket(InetAddress.getLocalHost(), 10);
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        PrintStream p = new PrintStream(s.getOutputStream());

        int expectedFrame = 0, totalFrames;
        String rbuf[] = new String[100];
        boolean received[] = new boolean[100]; // To track which frames have been
received
        int damagedFrame = 2;  // Assume frame 2 is damaged on first attempt
        boolean damagedFrameReceived = false;  // Flag to track if the damaged frame
has been received

        System.out.println("Receiver is ready...");

        // Read the total number of frames to be received (sent by the sender)
        totalFrames = Integer.parseInt(in.readLine());
        System.out.println("Total number of frames to receive: " + totalFrames);

        while (expectedFrame < totalFrames) {
            // Receive the frame from the sender
            String frameMessage = in.readLine();

            // Split the incoming message to get the frame number and content
            String[] parts = frameMessage.split(":");
            int frameNumber = Integer.parseInt(parts[0]);
            String frameContent = parts[1];
            // Simulate frame damage for the first attempt
            if (frameNumber == damagedFrame && !damagedFrameReceived) {
                System.out.println("Frame " + frameNumber + " is damaged. No
acknowledgment sent.");
                damagedFrameReceived = true;  // Mark that the damaged frame has
been received
                continue;  // Don't send an acknowledgment for the damaged frame
            }

            // If the frame is in range and not already received
            if (frameNumber >= expectedFrame && !received[frameNumber]) {
                System.out.println("Received Frame " + frameNumber + ": " +
frameContent);
                rbuf[frameNumber] = frameContent; // Store the frame content
                received[frameNumber] = true; // Mark the frame as received

                // Send acknowledgment for the received frame
                System.out.println("Sending acknowledgment for Frame " +
frameNumber);
                p.println(frameNumber);

                // Slide the expected frame pointer to the next missing frame
                while (received[expectedFrame]) {
                    expectedFrame++;
                }
            }
        }

        System.out.println("All frames received and acknowledged.");
        s.close();  // Close the socket when done
    }
}
```

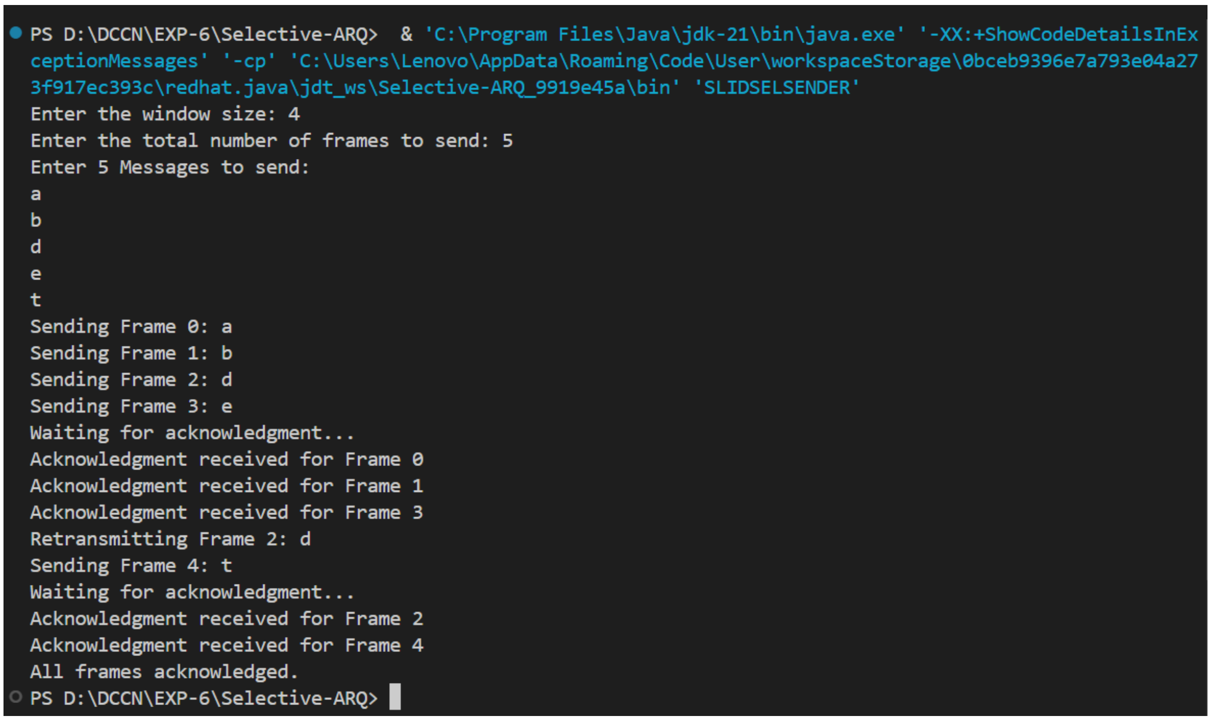
Screenshot of output:

Sender:



```
PS D:\DCCN\EXP-6\Go-Back-N>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExcept
ionMessages' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\634582247465b3f9a55799c55
4d690b1\redhat.java\jdt_ws\Go-Back-N_375bac18\bin' 'SLIDSENDER'
Enter the window size: 4
Enter the total number of frames to send: 5
Enter 5 Messages to send:
a
a
b
s
d
Sending Frame 0: a
Sending Frame 1: a
Sending Frame 2: b
Sending Frame 3: s
Waiting for acknowledgment...
Acknowledgment received for Frame 0
Sending Frame 4: d
Waiting for acknowledgment...
Acknowledgment received for Frame 1
Waiting for acknowledgment...
Timeout! No acknowledgment received for Frame 2
Retransmitting all frames in the window starting from Frame 2
Resending Frame 2: b
Resending Frame 3: s
Resending Frame 4: d
Waiting for acknowledgment...
Acknowledgment received for Frame 2
Waiting for acknowledgment...
Acknowledgment received for Frame 3
Waiting for acknowledgment...
Acknowledgment received for Frame 4
All frames acknowledged.
PS D:\DCCN\EXP-6\Go-Back-N> 
```

Receiver:



```
PS D:\DCCN\EXP-6\Selective-ARQ>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInEx
ceptionMessages' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\0bceb9396e7a793e04a27
3f917ec393c\redhat.java\jdt_ws\Selective-ARQ_9919e45a\bin' 'SLIDSELSENDER'
Enter the window size: 4
Enter the total number of frames to send: 5
Enter 5 Messages to send:
a
b
d
e
t
Sending Frame 0: a
Sending Frame 1: b
Sending Frame 2: d
Sending Frame 3: e
Waiting for acknowledgment...
Acknowledgment received for Frame 0
Acknowledgment received for Frame 1
Acknowledgment received for Frame 3
Retransmitting Frame 2: d
Sending Frame 4: t
Waiting for acknowledgment...
Acknowledgment received for Frame 2
Acknowledgment received for Frame 4
All frames acknowledged.
PS D:\DCCN\EXP-6\Selective-ARQ> 
```

Result:

Thus the Selective Repeat sliding window protocol is simulated successfully in Java IDE.

Ex No: 7

Date:

TCP/IP SOCKET PROGRAMMING IN CLIENT SERVER MODEL FOR FILE TRANSFER APPLICATION

Aim:

To write a Java program to implement the client server model for file transfer application using TCP/IP protocol

Theory:

The **File Transfer Protocol (FTP)** is a standard network protocol used for transferring files between a client and a server over a TCP/IP network.

Key Concepts:

1. **Client-Server Architecture**:
   - **Client**: Requests files from the server.
   - **Server**: Listens for client requests and sends the requested files.
2. **Process Flow**:
   - **Connection Establishment**: The client establishes a connection to the server using a specific port (usually port 21 for FTP).
   - **Command-Response Protocol**: The client sends commands (such as requests for file transfer), and the server responds with either the requested file or a status message (e.g., file not found, transfer successful).
   - **File Transfer**: Data is transferred from server to client (or vice versa) in binary or text mode. In FTP, this usually happens on a separate data connection.
   - **Connection Termination**: After the transfer, both the client and server close the connection.
3. **Two Modes of FTP**:
   - **Active Mode**: The server actively opens a connection to the client for the data transfer.
   - **Passive Mode**: The client initiates both control and data connections to the server.
4. **Reliability**: FTP ensures that files are transferred reliably using TCP, which handles packet delivery, error correction, and reordering.

Algorithm:

Server

Step 1: **Start the server** on port 8081.
Step 2: **Accept client connection**.
Step 3: **Read the requested file name** from the client.
Step 4: **Check if the file exists**:

- If the file exists:
  - Open the file and **send its contents** to the client using a buffer.
  - **Flush and close** the output stream.
  - **Notify** the server-side that the file was transferred successfully.
- If the file does not exist:
  - **Send an error message** ("ERROR: File not found") to the client.

Step 5: **Close** the client socket and server socket.

Client

Step 1: **Connect to the server** on port 8081.
Step 2: **Prompt the user** for the file name.
Step 3: **Send the file name** to the server.
Step 4: **Read the server response**:

- If the response starts with "ERROR":
  - **Display the error message** to the user.
- If no error:
  - **Receive the file data** from the server.
  - **Write the received data** into a new file on the client system.
  - **Notify** the client-side that the file was received successfully.

Step 5: **Close** the socket and file streams.

Run the server code first, then client code.
Make sure the requested filename is present in the directory where the server code runs.

Program:

Server:

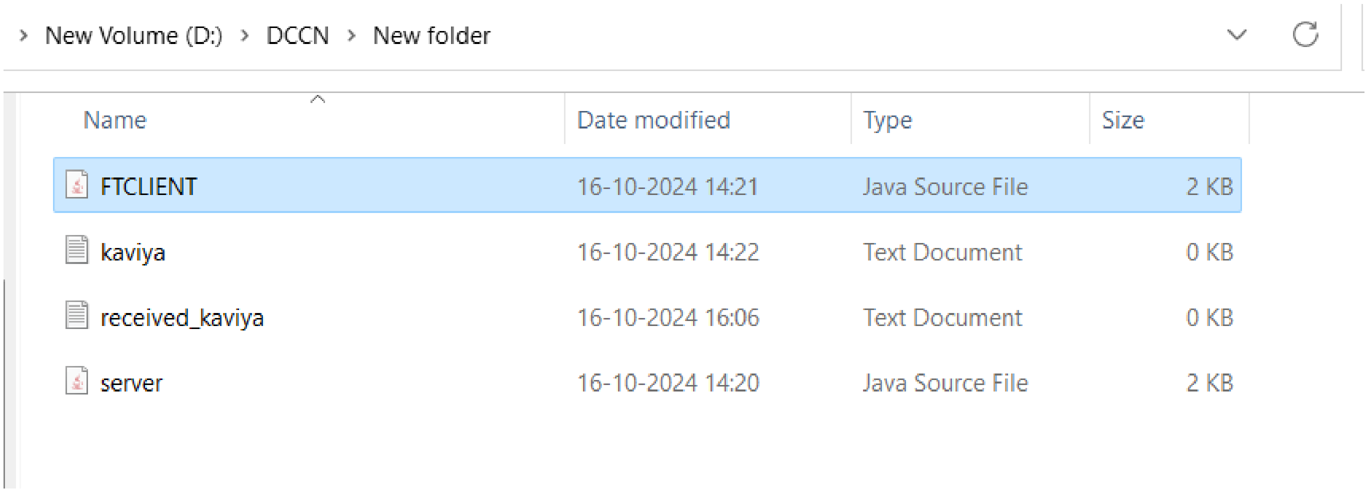
```
import java.io.*;
import java.net.*;

public class FTSERVER {
   public static void main(String args[]) throws IOException {
      ServerSocket ss = new ServerSocket(8081);
      System.out.println("Server started, waiting for connection...");

      Socket cs = ss.accept();  // Wait for client connection
      System.out.println("Client connected.");

      BufferedReader st = new BufferedReader(new
InputStreamReader(cs.getInputStream()));
      String requestedFile = st.readLine();
      System.out.println("The requested file is: " + requestedFile);
File file = new File(requestedFile);
        PrintWriter put = new PrintWriter(cs.getOutputStream(), true);  // Send messages to
the client

        if (file.exists()) {
            // Send file content using a BufferedOutputStream
            BufferedInputStream fileReader = new BufferedInputStream(new
FileInputStream(file));
            BufferedOutputStream outputStream = new
BufferedOutputStream(cs.getOutputStream());

            byte[] buffer = new byte[4096];  // Buffer for reading file content
            int bytesRead;
            while ((bytesRead = fileReader.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }

            fileReader.close();
            outputStream.flush();  // Ensure all data is sent
            outputStream.close();
            System.out.println("File transferred successfully.");
        } else {
            // Send an error message to the client
            put.println("ERROR: File not found");
            System.out.println("File does not exist.");
        }

        put.close();
        cs.close();  // Close client socket
        ss.close();  // Close server socket
    }
}
```

Client:

```
import java.io.*;
import java.net.*;

public class FTCLIENT {
    public static void main(String args[]) throws IOException {
        Socket s = new Socket("localhost", 8081);

        PrintWriter put = new PrintWriter(s.getOutputStream(), true);
        BufferedReader serverResponse = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        BufferedReader userInputReader = new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("Enter the file name to transfer from the server:");
        String fileName = userInputReader.readLine();  // Get file name from user
        put.println(fileName);  // Send file name to server

        // Check the first line of the server response
        String response = serverResponse.readLine();
        if (response != null && response.startsWith("ERROR")) {
            System.out.println("Server error: " + response);
        } else {
            // Prepare to receive the file content if no error message is received
            File receivedFile = new File("received_" + fileName);
            FileOutputStream fileOut = new FileOutputStream(receivedFile);
            BufferedInputStream fileReader = new BufferedInputStream(s.getInputStream());

            byte[] buffer = new byte[4096];
            int bytesRead;
            while ((bytesRead = fileReader.read(buffer)) != -1) {
                fileOut.write(buffer, 0, bytesRead);
            }

            fileOut.close();  // Close file output
            fileReader.close();  // Close input stream

            System.out.println("File received successfully.");
        }

        s.close();  // Close socket
    }
}
```

SCREENSHOTS OF OUTPUT:

SERVER CODE DIRECTORY:

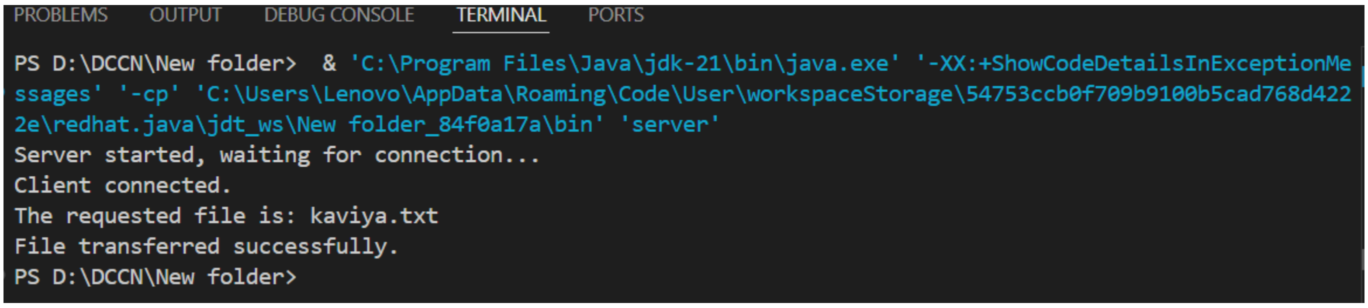


› New Volume (D:) › DCCN › New folder

Name	Date modified	Type	Size
FTCLIENT	16-10-2024 14:21	Java Source File	2 KB
kaviya	16-10-2024 14:22	Text Document	0 KB
received_kaviya	16-10-2024 16:06	Text Document	0 KB
server	16-10-2024 14:20	Java Source File	2 KB

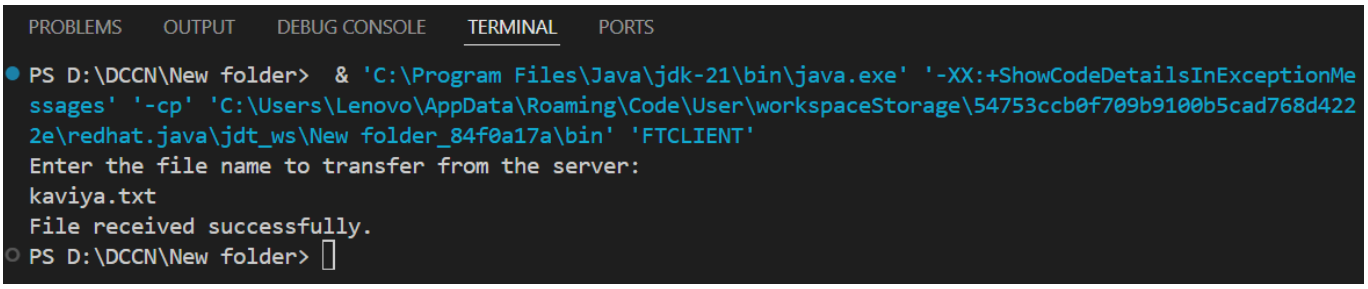
<u>When requested file is available:</u>

SERVER:



```
PS D:\DCCN\New folder>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMe
ssages' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\54753ccb0f709b9100b5cad768d422
2e\redhat.java\jdt_ws\New folder_84f0a17a\bin' 'server'
Server started, waiting for connection...
Client connected.
The requested file is: kaviya.txt
File transferred successfully.
PS D:\DCCN\New folder>
```

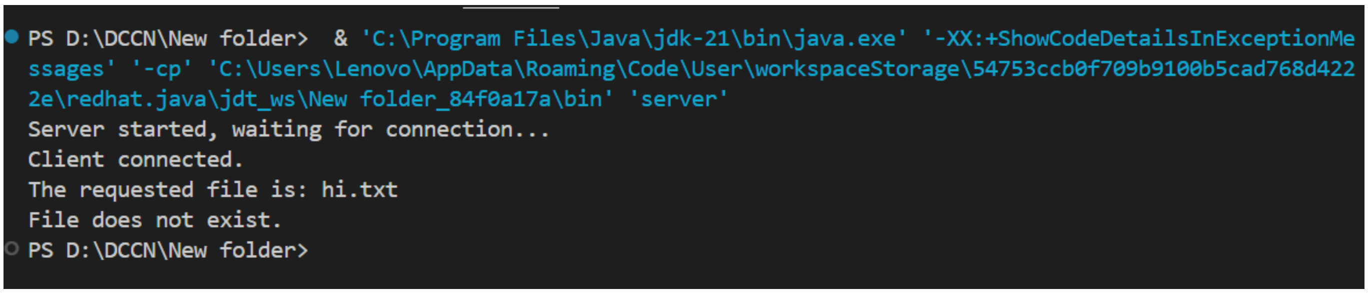
CLIENT:



```
PS D:\DCCN\New folder>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMe
ssages' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\54753ccb0f709b9100b5cad768d422
2e\redhat.java\jdt_ws\New folder_84f0a17a\bin' 'FTCLIENT'
Enter the file name to transfer from the server:
kaviya.txt
File received successfully.
PS D:\DCCN\New folder>
```

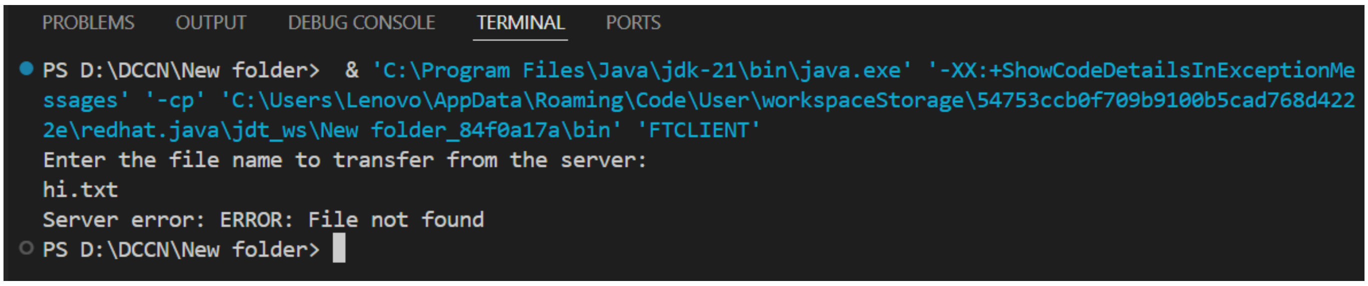
<u>When requested file is not available:</u>

SERVER:



```
PS D:\DCCN\New folder>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMe
ssages' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\54753ccb0f709b9100b5cad768d422
2e\redhat.java\jdt_ws\New folder_84f0a17a\bin' 'server'
Server started, waiting for connection...
Client connected.
The requested file is: hi.txt
File does not exist.
PS D:\DCCN\New folder>
```

CLIENT:



```
PS D:\DCCN\New folder>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMe
ssages' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\54753ccb0f709b9100b5cad768d422
2e\redhat.java\jdt_ws\New folder_84f0a17a\bin' 'FTCLIENT'
Enter the file name to transfer from the server:
hi.txt
Server error: ERROR: File not found
PS D:\DCCN\New folder>
```

Result:

Thus the echo client server model for file transfer application using TCP/IP protocol, was successfully implemented in Java coding and output obtained.

Ex No: 8

Date:

DEVELOPMENT OF SUITABLE PROTOCOL TO SEND, RECEIVE AND SYNCHRONIZE MAIL.

Aim:

To write a Java program to implement the client server model to send, receive and synchronize mail.

Theory:

SMTP and POP3 are both protocols used to transmit email data between devices, but they have different purposes and functions:

- SMTP (Simple Mail Transfer Protocol)

Used to send emails from the sender's device to the recipient's email server. SMTP is also used by servers to forward messages to their final destination. SMTP is a push protocol.

- POP3 (Post Office Protocol 3)

Used to retrieve emails from the recipient's email server to the recipient's device. POP3 is also known as a pop protocol. POP3 is an inbound protocol.

Here are some other differences between SMTP and POP3:

- Encryption

POP3 has standard and encrypted ports, with 110 being the standard and 995 being the SSL-encrypted alternative. SMTP has ports 25 and 587, with 25 being the standard for outgoing emails and 587 being registered as a secure SMTP port.

- Benefits of POP3

POP3 allows for offline access to emails, is easy to set up, and is bandwidth-friendly.

- Benefits of SMTP

SMTP ensures simplicity, speed, and reliability with notifications for delivery failures.

Algorithm:

TO SEND MAIL: SMTP PROTOCOL

Initialize Constants:

- Set the SMTP server (SMTP_SERVER) to "smtp.gmail.com".
- Set the SMTP port (SMTP_PORT) to 465.
- Set the email USER and PASSWORD for authentication.

Create SSL Socket:

- Get an instance of SSLSocketFactory.
- Use the SSLSocketFactory to create an SSLSocket connected to the SMTP server at the specified port.

Create Input and Output Streams:

- Create a BufferedReader for reading responses from the server.
- Create a BufferedWriter for sending commands to the server.

Read Server Response:

- Read and print the server's initial response after connecting.

Send SMTP Commands:

- Send the HELO command to the SMTP server and read the response.
- Send the AUTH LOGIN command to initiate authentication and read the response.

Authenticate User:

- Encode the USER (email address) in Base64 and send it to the server.
- Read and print the server's response.
- Encode the PASSWORD in Base64 and send it to the server.
- Read and print the server's response.

Specify Sender and Recipient:

- Send the MAIL FROM command with the user's email address and read the response.
- Send the RCPT TO command with the recipient's email address and read the response.

Send Email Data:

- Send the DATA command to indicate the start of the email content and read the response.
- Send the email headers (Subject, From, To) followed by the email body content.
- End the email content with a single period . on a new line, then read the server's response.

Terminate the Session:

- Send the QUIT command to terminate the session and read the server's response.

Close Resources:

- Close the BufferedWriter, BufferedReader, and SSLSocket to release resources.

Handle Exceptions:

- Use a try-catch block to handle exceptions that may occur during the execution of the code, printing the stack trace if an exception is caught.

<u>TO RECEIVE MAIL: POP3 PROTOCOL</u>

Initialize Constants:

- Set the POP3 server (POP3_SERVER) to "pop.gmail.com".
- Set the POP3 port (POP3_PORT) to 995.
- Set the email USER (email address) and PASSWORD for authentication.

Create SSL Socket:

- Get an instance of SSLSocketFactory.
- Use the SSLSocketFactory to create an SSLSocket connected to the POP3 server at the specified port.

Create Input and Output Streams:

- Create a BufferedReader for reading responses from the server.
- Create a BufferedWriter for sending commands to the server.

Read Server Greeting:

- Read and print the server's greeting message after connecting.

Authenticate User:

- Send the USER command with the email address and read the server's response.
- Send the PASS command with the password and read the server's response.
- Check if the authentication was successful by verifying if the response starts with -ERR.
- If authentication fails, print an error message and exit the program.

Check Number of Messages:

- Send the STAT command to get the total number of messages in the mailbox and read the response.
- Extract the number of total messages from the response.

Determine Messages to Fetch:

- Calculate the number of messages to fetch (last 5 messages) using Math.min(5, totalMessages).

Fetch Latest Emails:

- Loop from the total number of messages down to the number of messages to fetch:
  - Send the RETR command for each message to retrieve its content.
  - Read and print the server's response (which contains the email content).
  - Read the email content until a line with just . is encountered, indicating the end of the email.
  - Store the email content in a StringBuilder.

Parse Email Headers:

- Split the email content into lines and extract relevant headers:
  - Initialize variables for subject, from, to, and date.
  - Loop through the headers, extracting the required information based on header prefixes.

Display Extracted Fields:

- Print the email index along with the extracted subject, from, to, and date fields.

Terminate the Session:

- Send the QUIT command to end the POP3 session and read the server's response.

Close Resources:

- Close the BufferedWriter, BufferedReader, and SSLSocket to release resources.

Handle Exceptions:

- Use a try-catch block to handle exceptions that may occur during the execution of the code, printing the stack trace if an exception is caught.

NOTE: Use only gmail email id. Generate 16 key App Password in your google account, which must be entered in the program, as password.

Program:

<u>TO SEND MAIL: SMTP PROTOCOL</u>

```
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import java.io.*;
import java.util.Base64;

public class SENDMAIL {
   private static final String SMTP_SERVER = "smtp.gmail.com";
   private static final int SMTP_PORT = 465;
   private static final String USER = "yourmail@gmail.com";
   private static final String PASSWORD = "your 16key app password"; // Use App
Password if using 2FA

   public static void main(String[] args) {
      try {
         // Create an SSL socket to connect to the SMTP server
         SSLSocketFactory ssf = (SSLSocketFactory) SSLSocketFactory.getDefault();
         SSLSocket socket = (SSLSocket) ssf.createSocket(SMTP_SERVER,
SMTP_PORT);
         BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
         BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));

         System.out.println("Response: " + reader.readLine());

         writer.write("HELO " + SMTP_SERVER + "\r\n");
         writer.flush();
         System.out.println("Response: " + reader.readLine());

         writer.write("AUTH LOGIN\r\n");
         writer.flush();
         System.out.println("Response: " + reader.readLine());

         // Send username (Base64 encoded)
         writer.write(Base64.getEncoder().encodeToString(USER.getBytes()) + "\r\n");
         writer.flush();
         System.out.println("Response: " + reader.readLine());

         // Send password (Base64 encoded)
         writer.write(Base64.getEncoder().encodeToString(PASSWORD.getBytes()) +
"\r\n");
         writer.flush();
         System.out.println("Response: " + reader.readLine());

         writer.write("MAIL FROM:<" + USER + ">\r\n");
         writer.flush();
         System.out.println("Response: " + reader.readLine());

         writer.write("RCPT TO:<ngianna28@gmail.com>\r\n");
         writer.flush();
         System.out.println("Response: " + reader.readLine());

         writer.write("DATA\r\n");
         writer.flush();
```

```
                System.out.println("Response: " + reader.readLine());

                writer.write("Subject: Test Email\r\n");
                writer.write("From: " + USER + "\r\n");
                writer.write("To: recipientmail@gmail.com\r\n");
                writer.write("\r\n");
                writer.write("HI THIS IS A TEST MAIL\r\n");
                writer.write(".\r\n");
                writer.flush();
                System.out.println("Response: " + reader.readLine());

                writer.write("QUIT\r\n");
                writer.flush();
                System.out.println("Response: " + reader.readLine());

                writer.close();
                reader.close();
                socket.close();

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

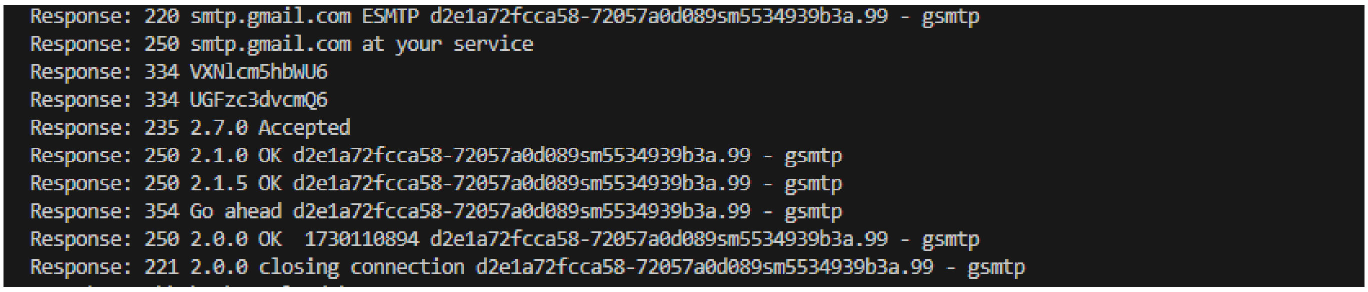
TO RECEIVE MAIL: POP3 PROTOCOL

```
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import java.io.*;

public class RECEIVEMAIL {
    private static final String POP3_SERVER = "pop.gmail.com"; // POP3 server for Gmail
    private static final int POP3_PORT = 995; // POP3 port with SSL
    private static final String USER = "yourmail@gmail.com";
    private static final String PASSWORD = "your 16key app password"; // Use App Password if
using 2FA


    public static void main(String[] args) {
        try {
            // Create an SSL socket to connect to the POP3 server
            SSLSocketFactory ssf = (SSLSocketFactory) SSLSocketFactory.getDefault();
            SSLSocket socket = (SSLSocket) ssf.createSocket(POP3_SERVER, POP3_PORT);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
```

```
            // Read server greeting
            System.out.println("Response: " + reader.readLine());

            // Send USER command
            writer.write("USER " + USER + "\r\n");
            writer.flush();
            System.out.println("Response: " + reader.readLine());

            // Send PASS command
            writer.write("PASS " + PASSWORD + "\r\n");
            writer.flush();
            String response = reader.readLine();
            System.out.println("Response: " + response);

            // Check if authentication was successful
            if (response.startsWith("-ERR")) {
                System.out.println("Authentication failed. Please check your username and
password.");
                return; // Exit the program
            }

            // Send STAT command to get the number of messages
            writer.write("STAT\r\n");
            writer.flush();
            response = reader.readLine();
            System.out.println("Response: " + response);

            // Extract the number of messages
            int totalMessages = Integer.parseInt(response.split(" ")[1]);
            System.out.println("Total messages: " + totalMessages);

            // Determine how many messages to fetch (last 5)
            int messagesToFetch = Math.min(5, totalMessages);

            // Fetch the latest emails
            for (int i = totalMessages; i > totalMessages - messagesToFetch; i--) {
                // Send RETR command to fetch the email
                writer.write("RETR " + i + "\r\n");
                writer.flush();
                response = reader.readLine();
                System.out.println("Response: " + response); // This shows the email content

                // Read the email content
                StringBuilder emailContent = new StringBuilder();
                String line;
                while (!(line = reader.readLine()).equals(".")) {
                    emailContent.append(line).append("\n");
                }

                // Parse the email content for headers
                String[] headers = emailContent.toString().split("\n");
                String subject = "", from = "", to = "", date = "";

                // Extract headers
                for (String header : headers) {
                    if (header.startsWith("Subject:")) {
                        subject = header.substring(9).trim();
                    } else if (header.startsWith("From:")) {
                        from = header.substring(6).trim();
                    } else if (header.startsWith("To:")) {
                        to = header.substring(4).trim();
                    } else if (header.startsWith("Date:")) {
                        date = header.substring(6).trim();
                    }
                }

                // Display extracted fields
                System.out.println("Email " + i + ":");
                System.out.println("Subject: " + subject);
                System.out.println("From: " + from);
                System.out.println("To: " + to);
                System.out.println("Date: " + date);
                System.out.println("\n--- End of Email " + i + " ---\n");
            }

            // Send QUIT command
            writer.write("QUIT\r\n");
            writer.flush();
            System.out.println("Response: " + reader.readLine());

            writer.close();
            reader.close();
            socket.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

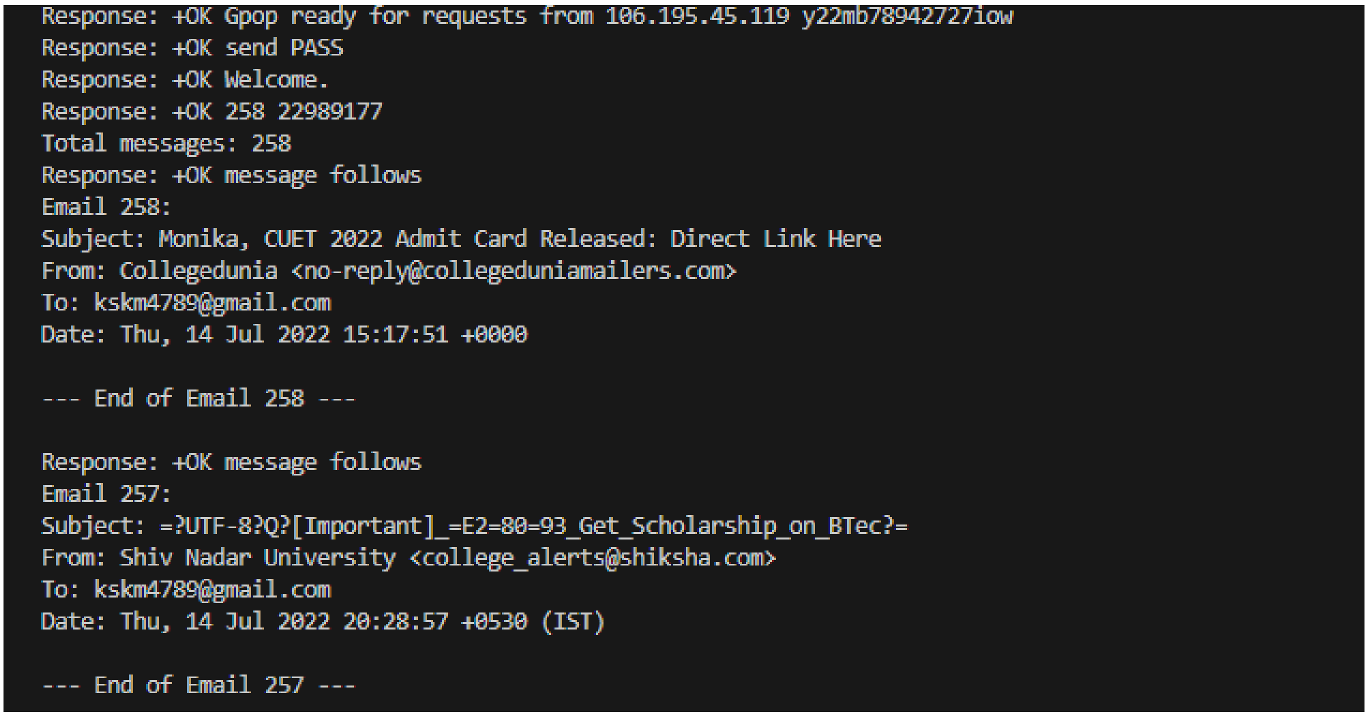
SCREENSHOTS OF OUTPUT:

SENDING MAIL



```
Response: 220 smtp.gmail.com ESMTP d2e1a72fcca58-72057a0d089sm5534939b3a.99 - gsmtp
Response: 250 smtp.gmail.com at your service
Response: 334 VXNlcm5hbWU6
Response: 334 UGFzc3dvcmQ6
Response: 235 2.7.0 Accepted
Response: 250 2.1.0 OK d2e1a72fcca58-72057a0d089sm5534939b3a.99 - gsmtp
Response: 250 2.1.5 OK d2e1a72fcca58-72057a0d089sm5534939b3a.99 - gsmtp
Response: 354 Go ahead d2e1a72fcca58-72057a0d089sm5534939b3a.99 - gsmtp
Response: 250 2.0.0 OK  1730110894 d2e1a72fcca58-72057a0d089sm5534939b3a.99 - gsmtp
Response: 221 2.0.0 closing connection d2e1a72fcca58-72057a0d089sm5534939b3a.99 - gsmtp
```

RECEIVING MAIL



```
Response: +OK Gpop ready for requests from 106.195.45.119 y22mb78942727iow
Response: +OK send PASS
Response: +OK Welcome.
Response: +OK 258 22989177
Total messages: 258
Response: +OK message follows
Email 258:
Subject: Monika, CUET 2022 Admit Card Released: Direct Link Here
From: Collegedunia <no-reply@collegeduniamailers.com>
To: kskm4789@gmail.com
Date: Thu, 14 Jul 2022 15:17:51 +0000

--- End of Email 258 ---

Response: +OK message follows
Email 257:
Subject: =?UTF-8?Q?[Important]_=E2=80=93_Get_Scholarship_on_BTec?=
From: Shiv Nadar University <college_alerts@shiksha.com>
To: kskm4789@gmail.com
Date: Thu, 14 Jul 2022 20:28:57 +0530 (IST)

--- End of Email 257 ---
```

Result:

Thus the client server model for mail transfer application using TCP/IP protocol, was successfully implemented in Java coding and output obtained.

Ex No: 9

Date:

WRITE A PROGRAM TO FIND THE SHORTEST PATH BETWEEN VERTICES USING OPEN SHORTEST PATH FIRST ROUTING ALGORITHM.

Aim:

To simulate the process of finding the shortest path between vertices in a network using the Open Shortest Path First (OSPF) algorithm.

Theory:

After the completion of this experiment, students will be able to:

- Understand the OSPF algorithm and its role in routing within networks.
- Simulate the process of finding the shortest path using Dijkstra's algorithm, the core of OSPF.
- Analyze how the shortest path is determined in a network of interconnected nodes.
- Interpret the relationship between link costs and routing decisions in maintaining optimal network performance.
- Understand the importance of link-state routing protocols in managing dynamic network topologies.

The OSPF algorithm is a link-state routing protocol used in Internet Protocol (IP) networks. It uses Dijkstra's algorithm to compute the shortest path tree for each route. The OSPF algorithm works by finding the shortest path between a given source node and all other nodes in the network based on link costs.

Algorithm:

1. **Initialization:**
    - Start with the source node and set the distance to itself as 0.
    - Set the distance to all other nodes as infinity.
    - Mark all nodes as unvisited.
2. **Selection:**
    - Choose the unvisited node with the smallest tentative distance and mark it as the current node.
3. **Distance Calculation:**
    - For the current node, consider all its unvisited neighbors.

- Calculate the tentative distance from the source node to each neighbor.
- If the calculated distance is less than the known distance, update the shortest distance to that neighbor.

4. **Mark as Visited:**
    - Once all neighbors of the current node have been considered, mark the current node as visited.
5. **Repeat:**
    - Repeat steps 2 to 4 until all nodes have been visited or the smallest tentative distance among the unvisited nodes is infinity.
6. **Termination:**
    - The algorithm terminates when all nodes have been visited, and the shortest path from the source node to all other nodes is determined.

Example:

Consider a network of 6 nodes with the following edges and costs:

- (0, 1) with cost 4
- (0, 2) with cost 3
- (1, 2) with cost 1
- (1, 3) with cost 2
- (2, 3) with cost 4
- (3, 4) with cost 2
- (4, 5) with cost 6

Steps:

- Initialize distances from the source node (e.g., Node 0) to all other nodes.
- For each node, find the shortest path to its neighbors and update the distances.
- Repeat until all nodes have been visited.

Example Calculation:

For the source node 0:

- Distance to 1: 4
- Distance to 2: 3
- Distance to 3: 6
- Distance to 4: 8
- Distance to 5: 14

These distances represent the shortest path from node 0 to all other nodes.

Program:

```
import java.util.*;

class OSPF {
    private int vertices;
    private LinkedList<Edge>[] adjList;

    // Edge class to represent a weighted edge in the graph
    static class Edge {
        int target, weight;

        Edge(int target, int weight) {
            this.target = target;
            this.weight = weight;
        }
    }

    // Constructor to initialize the graph with a given number of vertices
    OSPF(int vertices) { // Changed from Graph to OSPF
        this.vertices = vertices;
        adjList = new LinkedList[vertices];
        for (int i = 0; i < vertices; i++) {
            adjList[i] = new LinkedList<>();
        }
    }

    // Method to add an edge to the graph
    void addEdge(int source, int target, int weight) {
        adjList[source].add(new Edge(target, weight));
        adjList[target].add(new Edge(source, weight)); // for undirected graph
    }

    // Method to find the shortest path using Dijkstra's algorithm
    void dijkstra(int source, int target) {
        PriorityQueue<Edge> priorityQueue = new PriorityQueue<>(vertices,
Comparator.comparingInt(e -> e.weight));
        int[] distances = new int[vertices];
        Arrays.fill(distances, Integer.MAX_VALUE);
        distances[source] = 0;
        priorityQueue.add(new Edge(source, 0));
        while (!priorityQueue.isEmpty()) {
            int u = priorityQueue.poll().target;

            for (Edge edge : adjList[u]) {
                int v = edge.target;
                int weight = edge.weight;

                if (distances[u] + weight < distances[v]) {
                    distances[v] = distances[u] + weight;
                    priorityQueue.add(new Edge(v, distances[v]));
                }
            }
        }

        // Print the shortest path to the specified target node
        System.out.println("Shortest path from vertex " + source + " to vertex " + target
+ " is " + distances[target]);
    }

    // Main method to demonstrate the program
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        OSPF graph = new OSPF(6); // Changed from Graph to OSPF
        graph.addEdge(0, 1, 4);
        graph.addEdge(0, 2, 3);
        graph.addEdge(1, 2, 1);
        graph.addEdge(1, 3, 2);
        graph.addEdge(2, 3, 4);
        graph.addEdge(3, 4, 2);
        graph.addEdge(4, 5, 6);

        System.out.print("Enter the source vertex: ");
        int source = scanner.nextInt();

        System.out.print("Enter the target vertex: ");
        int target = scanner.nextInt();

        graph.dijkstra(source, target);
    }
}
```

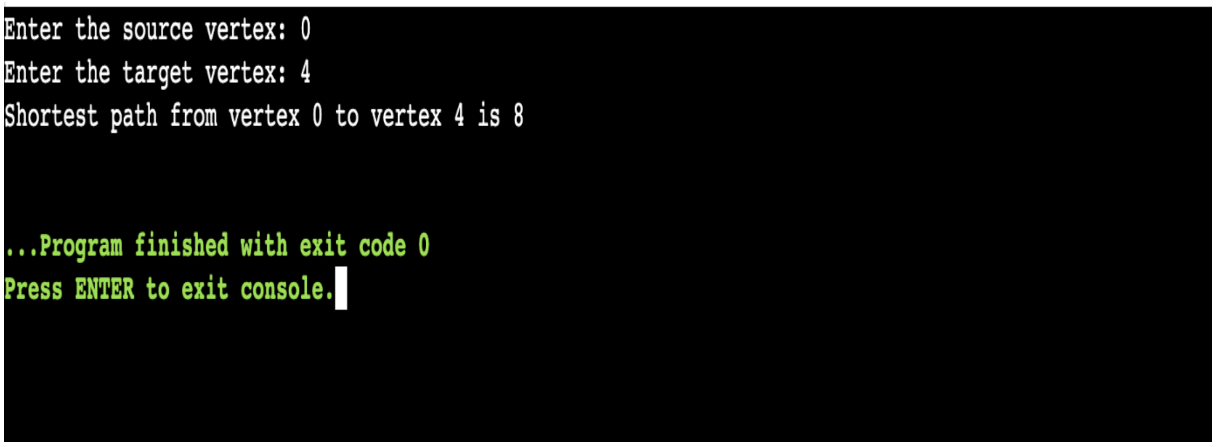
Sample Output:

Enter the source vertex: 0

Enter the target vertex: 4

Shortest path from vertex 0 to vertex 4 is 8

Screenshot of output:



```
Enter the source vertex: 0
Enter the target vertex: 4
Shortest path from vertex 0 to vertex 4 is 8


...Program finished with exit code 0
Press ENTER to exit console.
```

Result:

The OSPF algorithm effectively calculates the shortest path between nodes, ensuring optimal routing decisions in IP networks. This simulation demonstrates how the algorithm dynamically adjusts to find the most efficient routes, which is crucial in maintaining network performance.

Ex No: 10

Date:

WRITE A PROGRAM FOR CONGESTION CONTROL IN A NETWORK USING LEAKY BUCKET ALGORITHM.

Aim:

To simulate network congestion control using the Leaky Bucket algorithm.

Theory:

After the completion of this experiment, student will be able to

- Understand the Leaky Bucket algorithm and its role in network congestion control.
- Simulate the process of packet regulation using the Leaky Bucket mechanism.
- Analyze how packet loss occurs when the bucket capacity is exceeded.
- Interpret the relationship between input packet size, bucket size, and output packet size in maintaining steady network transmission.
- Understand the importance of traffic shaping techniques in managing network bandwidth and preventing congestion.

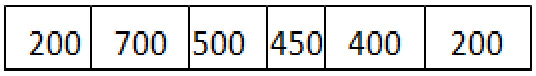
A simple leaky bucket algorithm can be implemented using FIFO queue. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

Algorithm:

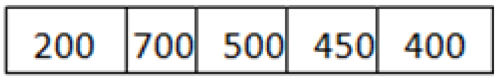
```
Step-1: Initialize a counter to n at the tick of the clock.

Step-2: Repeat until n is smaller than the packet size of the packet at the head
of the queue.

        • Pop a packet out of the head of the queue, say P.
        • Send the packet P, into the network
        • Decrement the counter by the size of packet P.
Step-3: Reset the counter and go to step 1.
```

Example: Let n=1000

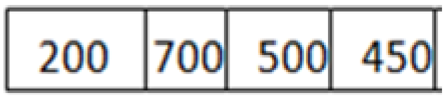
Packet=



Since n > size of the packet at the head of the Queue, i.e. n > 200
Therefore, n = 1000-200 = 800
Packet size of 200 is sent into the network.



Now, again n > size of the packet at the head of the Queue, i.e. n > 400
Therefore, n = 800-400 = 400
Packet size of 400 is sent into the network.



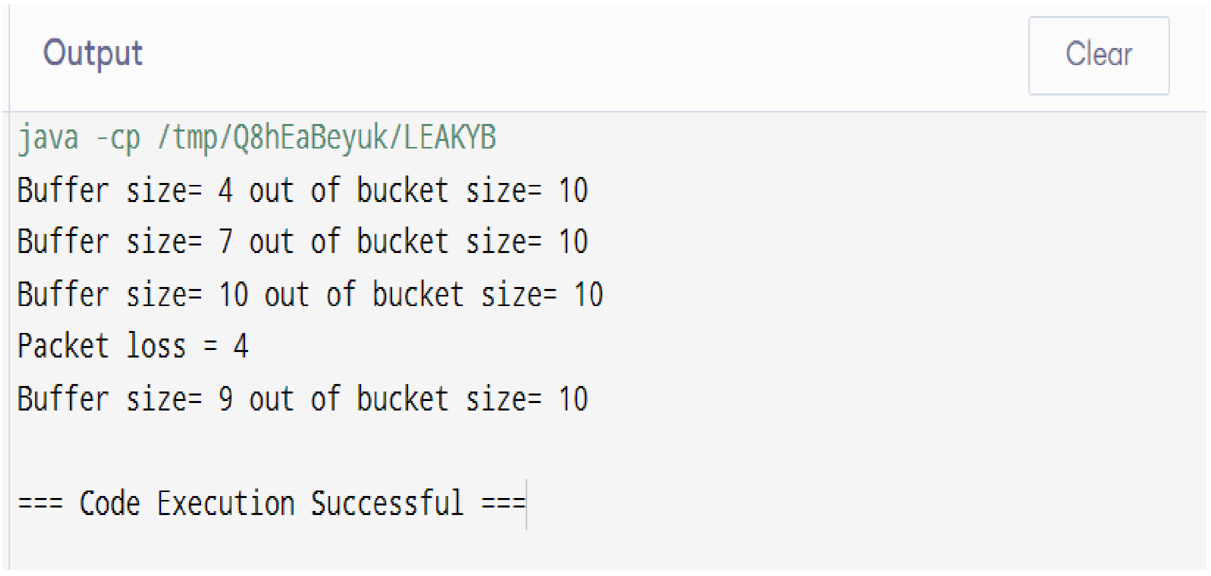
Since, n < size of the packet at the head of the Queue, i.e. n < 450
Therefore, the procedure is stopped.

Initialise n = 1000 on another tick of the clock.
This procedure is repeated until all the packets are sent into the network.

Program:

```
import java.io.*;

import java.util.*;

class LEAKYB {

    public static void main(String[] args) {

        int no_of_queries, storage, output_pkt_size;

        int input_pkt_size, bucket_size, size_left;


        // initial packets in the bucket
        storage = 0;


        // total no. of times bucket content is checked

        no_of_queries = 4;


        // total no. of packets that can be accommodated in the bucket

        bucket_size = 10;


        // no. of packets that enters the bucket at a time

        input_pkt_size = 4;


        // no. of packets that exits the bucket at a time

        output_pkt_size = 1;


        for (int i = 0; i < no_of_queries; i++) {

            size_left = bucket_size - storage;  // space left


            if (input_pkt_size <= size_left) {

                storage += input_pkt_size;

            } else {

                System.out.println("Packet loss = " + input_pkt_size);

            }


            System.out.println("Buffer size= " + storage + " out of bucket size= " +
bucket_size);

            storage -= output_pkt_size;
            // Ensure storage doesn't go below 0
            if (storage < 0) {
                storage = 0;
            }
        }
    }
}
```

Screenshot of output:



```
java -cp /tmp/Q8hEaBeyuk/LEAKYB
Buffer size= 4 out of bucket size= 10
Buffer size= 7 out of bucket size= 10
Buffer size= 10 out of bucket size= 10
Packet loss = 4
Buffer size= 9 out of bucket size= 10

=== Code Execution Successful ===
```

Result:

Thus the program for network congestion control using the Leaky Bucket algorithm has been simulated successfully, by implementing java code.

Ex No: 11

Date:26/10/24

STUDY OF NETWORK SIMULATORS AND EMULATORS

Aim:

To Study of Network simulator platform as simulator and emulator.

Theory:

Ns overview

- Ns programming: A Quick start
- Case study I: A simple Wireless network
- Case study II: Create a new agent in Ns
- Ns Status
- Periodical release (ns-2.26, Feb 2003)
- Platform support
- FreeBSD, Linux, Solaris, Windows and Mac

Ns Functionalities

Routing, Transportation, Traffic sources,queuing disciplines, QoS Wireless Ad hoc routing, mobile IP, sensor-MAC Tracing, visualization and various utilitiesNS(Network Simulators) Most of the commercial simulators are GUI driven, while some network simulators are CLI driven. The network model / configuration describe the state of the network (nodes, routers, switchesand links) and the events (data transmissions, packet error etc.). The important outputs of simulations are the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node. Simulation of networks is a very complex task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variants" and "importance sampling" have been developed to speed simulation.

Examples of network simulators

There are many both free/open-source and proprietary network simulators. Examples of notable network simulation software are, ordered after how often they are mentioned in research papers:

- ns (open source)
- OPNET (proprietary software)
- NetSim (proprietary software)

Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance,simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly usefulin allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc. Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies likeEthernet, token rings etc., can all be simulated with a typical simulator and the user cantest, analyse various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface, but may permit more advanced forms of customization.

Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit error and spurious packets caused due to noise. Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss duringthe steady state is another important property of a congestion control scheme. The larger the value of packet loss, the more difficult it is for transport layer protocols to maintain high bandwidths, the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself.

Throughput

This is the main performance measure characteristic, and most widely used.In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bits per second (bit/s orbps), and sometimes in data packets per second or data packets per time slot This measure how soon the receiver is able to get a certain amount of data send by the sender. It is determined as the ratio of the total data received to the end to end delay. Throughput is an important factor which directly impacts the network performance

Delay

Delay is the time elapsed while a packet travels from one point e.g., source premise or network ingress to destination premise or network degrees. The larger the value of delay, the more difficult it is for transport layer protocols to maintain high bandwidths. We will calculate end to end delay

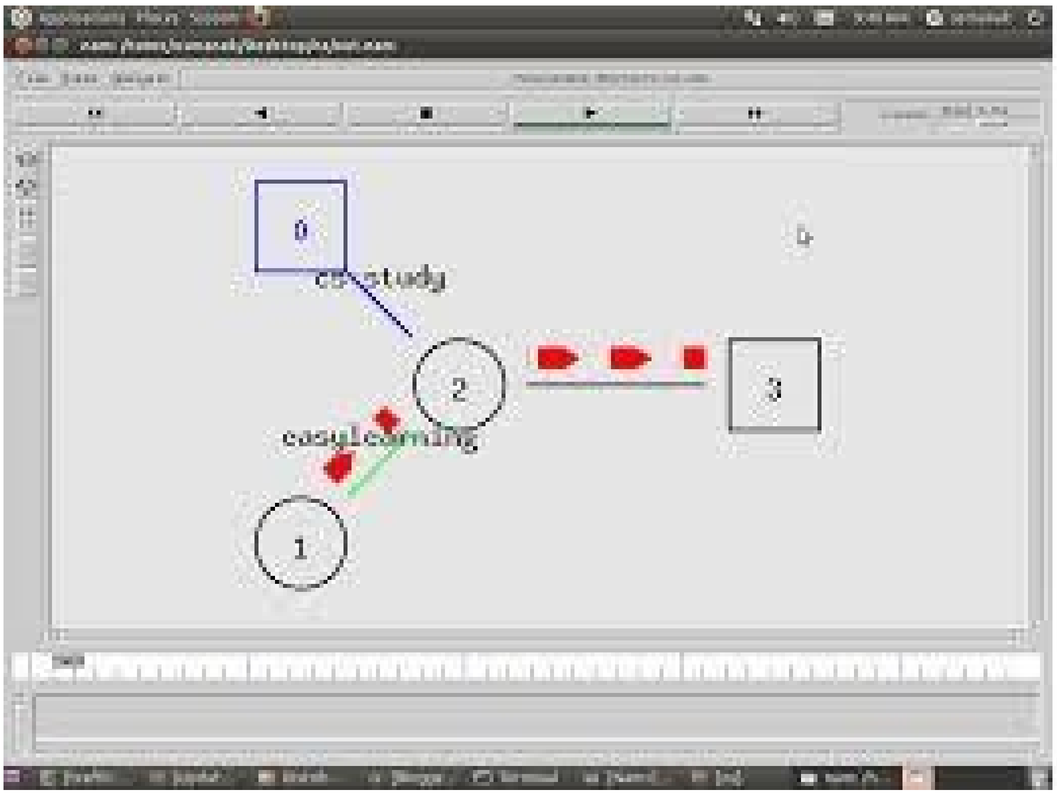
Queue Length

A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served. Thus queue length is very important characteristic to determine that how well the active queue management of the congestion control algorithm has been working.

1. Creating UDP Data traffic for wired network

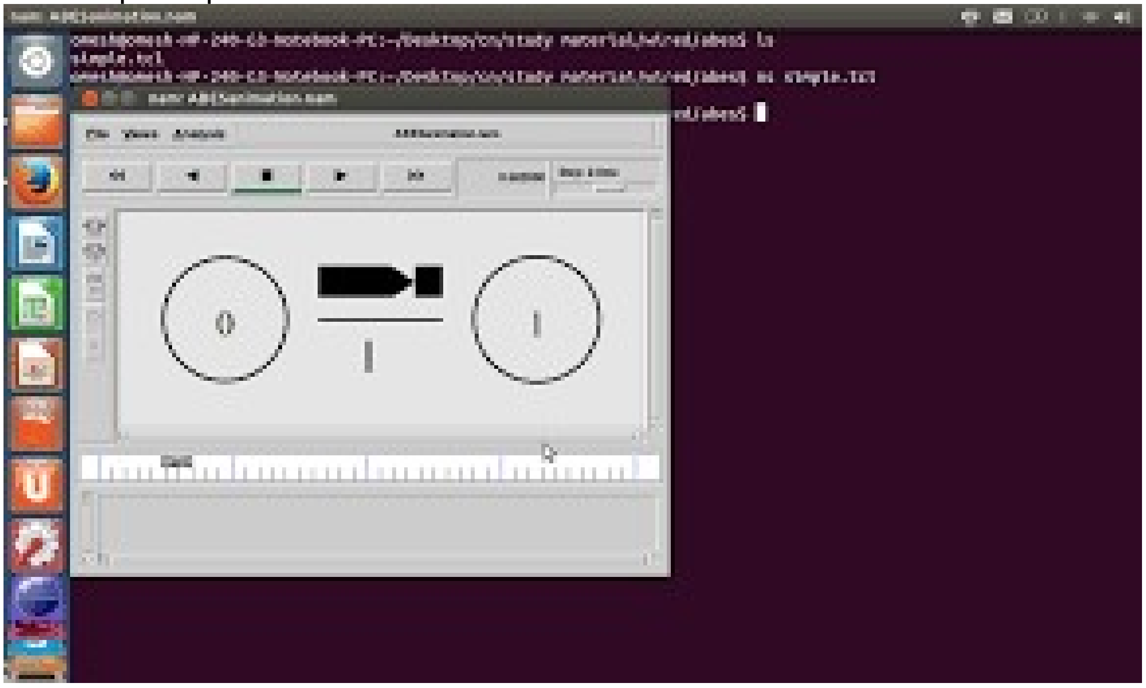
```
set ns [new Simulator]
#create file for analysis mode
set tr [open outl3.tr w]
$ns trace-all $tr
#create file for Animation Mode
set namtr [open outl3.nam w]
$ns namtrace-all $namtr
#Create Node
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Create Link
$ns duplex-link $n0 $n1 10Mb 5ms DropTail
$ns duplex-link $n2 $n0 10Mb 5ms DropTail
$ns duplex-link $n3 $n0 10mb 5ms DropTail
#Create Orientation
```

```
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n0 $n2 orient left-up
$ns duplex-link-op $n0 $n3 orient left-down
#create UDP Source
set udp0 [new Agent/UDP]
$ns attach-agent $n3 $udp0
#create UDP Destination
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
#connecting UDP Source & Destination
$ns connect $udp0 $null0
#create application traffic
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
#Application start time
$ns at 1.0 "$cbr0 start"
#Application Stop time
$ns at 5.0 "$cbr0 stop"

$ns at 10.0 "$ns halt"
$ns run
```



2. Creating tcp data traffic for wired network

```
#create TCP Source
set tcp0 [new Agent/TCP]
$ns attach-agent $n2 $tcp0
#create TCP Destination
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0
#create application traffic
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
#Application start & stop time
$ns at 2.0 "$ftp0 start"
$ns at 5.0 "$ftp0 stop"
```



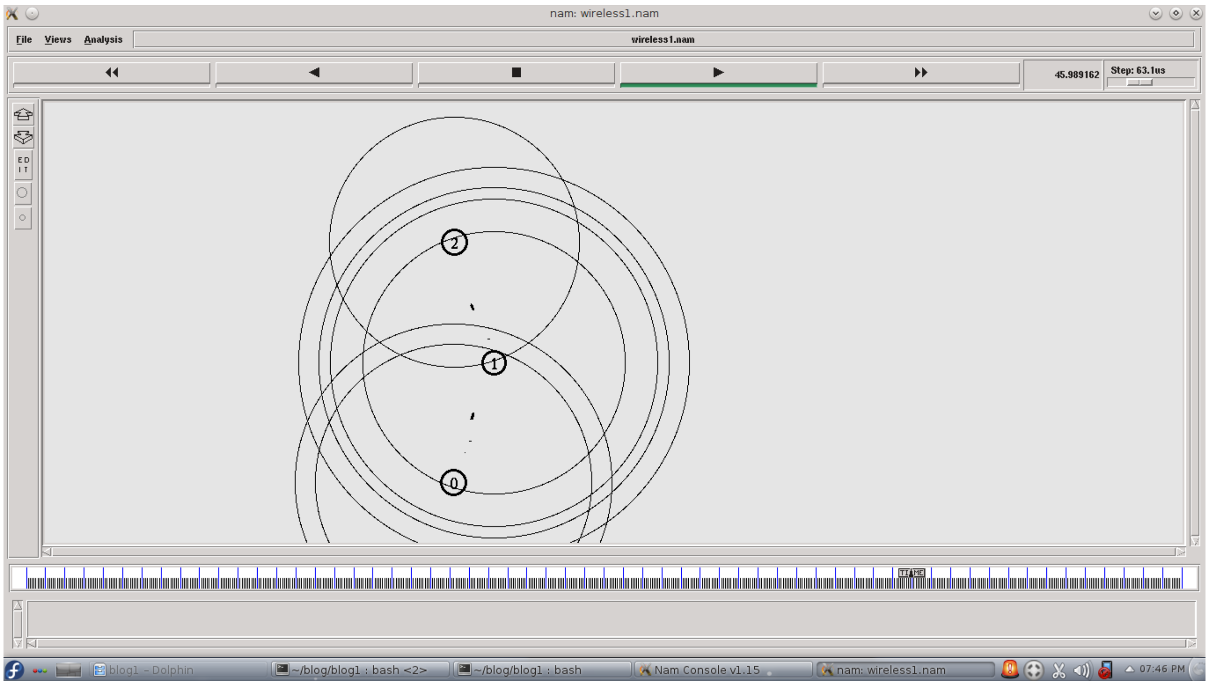
3. Wireless Network

```
#parameter initialization
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/Shadowing
set val(netif)  Phy/WirelessPhy
set val(ant)    Antenna/OmniAntenna
set val(mac)  Mac/802_11
set val(ifq)     Queue/DropTail/PriQueue
set val(ll)       LL
set val(x)       300
set val(y)       300
set val(ifqlen) 100
set val(nn)     10
set val(stop) 300.0
set val(rp)      AODV
```

```
#Sheduler Creation
set ns [new Simulator]
set tracefd [open wireless.tr w]
$ns trace-all $tracefd
set namtrace [open wireless.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#Topography
set prop [new $val(prop)]
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

$ns node-config -adhocRouting $val(rp) \
        -llType $val(ll) \
        -macType $val(mac) \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif) \
        -channelType $val(chan) \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace ON \
        -movementTrace ON

#Creating node objects..

for {set i 0} { $i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
}

for {set i 0} { $i < $val(nn) } { incr i } {
$node_($i) color darkgreen
$ns at 0.0 "$node_($i) color darkgreen"
}


##provide Initial location of wireless nodes
#Nodes:15, pause time:25.00, Max speed:3.00, Max X:100.0, MaxY:100.0
$node_(0) set X_ 271.057487973002
$node_(0) set Y_ 106.288323320442
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 96.023940800917
$node_(1) set Y_ 223.736201025809
$node_(1) set Z_ 0.000000000000
…..
$node_(9) set X_ 75.928619711928
```

```
$node_(9) set Y_ 203.805071579166
$node_(9) set Z_ 0.000000000000

#Assigning Mobility to Wireless Nodes

#Assigning Traffic -TCP
#Wireless Nodes:15, max connection:5, Send rate;0.0, seed:0
$ns at 50.000000000000 "$node_(0) setdest 234.760870186207 250.399813866979
1.818754605553"
$ns at 50.000000000000 "$node_(1) setdest 14.519853731648 84.887787293796
0.343160886173"
…..
$ns at 295.147357982603 "$node_(7) setdest 122.314673415723 126.467685356103
0.000000000000"


#0 connecting to 1 at time 133.84
set tcp_(0) [$ns create-connection TCP $node_(0) TCPSink $node_(1) 0]
$tcp_(0) set window_ 32
$tcp_(0) set packetSize_ 512
set ftp_(0) [$tcp_(0) attach-source FTP]
$ns at 133.8466 "$ftp_(0) start"

#0 connecting to 2 at time 33.84
set tcp_(1) [$ns create-connection TCP $node_(0) TCPSink $node_(2) 0]
$tcp_(1) set window_ 32
$tcp_(1) set packetSize_ 512
set ftp_(1) [$tcp_(1) attach-source FTP]
$ns at 33.8466 "$ftp_(1) start"

……

#Define node initial position in nam..
for {set i 0} { $i < $val(nn) } { incr i } {
#2 defines the node size for nam..
$ns initial_node_pos $node_($i) 2
}

#Tellin nodes when the simulation ends.
for {set i 0} {$i < $val(nn) } { incr i } {
$ns at $val(stop) "$node_($i) reset";
}

#Ending nam and the simulation..
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 299.01 "puts\"end simulation\"";#$ns halt
```

```
#stop procedure.

proc stop {} {
global ns tracefd namtrace
$ns flush-trace
close $tracefd
close $namtrace
exec nam 50nodes.nam &
exit 0
}

$ns run
```



Result:

Thus the Network Simulator 3 was studied for specific implementations as simulator and emulator.

Ex No: 12

Date:

SIMULATION OF LAN TO CREATE A SIMPLE NETWORK

Aim:

To simulate a LAN to Create a simple network and show the transfer of packets from one node to another node, using Cisco Packet Tracer software.

Theory:

Creating a Local Area Network (LAN) can involve both wired and wireless components.

1. Required Equipment

- **Router**: Central device to manage both wired and wireless connections.
- **Ethernet Switch** (optional): Expands the number of Ethernet ports if needed.Hubs can also be used.
- **Ethernet Cables**: For wired connections (Cat5e, Cat6, or higher for better speeds).
- **Wireless Access Point (WAP)**: If the router doesn't have built-in Wi-Fi, or if you need to extend wireless coverage.
- **Network Devices**: PCs, laptops, smartphones, etc., to connect to the network.

2. Setting Up the Wired LAN

- **Position Your Router**: Place your router in a central location if possible. This will optimize the wireless coverage and minimize the length of cables for wired devices.
- **Connect to the Internet**: Connect the router to your modem (if separate) using an Ethernet cable. This provides internet access to the network.
- **Wired Device Connections**: Use Ethernet cables to connect your devices (e.g., computers, printers) to the router. If your router has limited Ethernet ports, connect an Ethernet switch to the router, then connect additional devices to the switch.
- **Configuration**: Access the router's web interface by entering the router's IP address in a browser (commonly 192.168.1.1 or 192.168.0.1). Follow the instructions to set up basic network settings like IP addressing (usually DHCP).

3. Setting Up the Wireless LAN

- **Configure the Router's Wi-Fi**: If your router has built-in wireless capabilities, configure the wireless network by setting up an SSID (network name) and a secure password (WPA2 or WPA3 encryption).

- **Connect Wireless Devices**: On each wireless device (e.g., laptops, smartphones), search for the SSID and connect using the password you set.
- **Extending Wireless Coverage (Optional)**: If needed, install additional Wireless Access Points (WAPs) connected to the router via Ethernet cables to extend wireless coverage. Ensure WAPs are configured with the same SSID and security settings to allow seamless roaming between them.

4. Testing the Network : This can be done wither by pinging from source host to destination host, or by placing the data packet on the source and destination, and check if simulation is successful.

Procedure:

WIRED LAN

1. First, we will download Cisco Packet Tracer from netacad.com (latest version).
2. After downloading we will open it and now in this window, we see there are multiplesmall windows where we can select component and create our own particular computer network.
3. Select the components that are listed on the left bottom corner.
4. Select the 2950T switch from the components and place it on the white screen.
5. Place the PC's and laptops from the components and place it on the white screen.
6. Now select the wire from the connections and select copper straight through wire andconnect fastethernet from PC to the switch.

CONFIGURING THE NETWORK

- Now assign ip address to each of the PC and laptops and set the subnet mask to 255.255.2550.
- Under fastethernet tab when you double click on the PC you will able to see fastethernet and under that set IPv4 Address to the 192.168.1.101, 192.168.1.102,192.168.1.103, 192.168.1.104, 192.168.1.105 and for laptops 192.168.1.106 and 192.168.1.107.

TESTING THE NETWORK

- Choose the device you want to test and double click on that and under desktop youwill see the command prompt option
- Click on that and type the command ping "host ip"(the ip of any other device in thenetwork).
- The data packets are successfully sent from the source to destination.

WIRELESS LAN

- Select the components that are listed on the left bottom corner.

- Select the Home router from the components under wireless devices and place it onthe white screen.

- Place the PC's and laptops from the components and place it on the white screen. For router select wireless router WRT300N.

CONFIGURING THE NETWORK

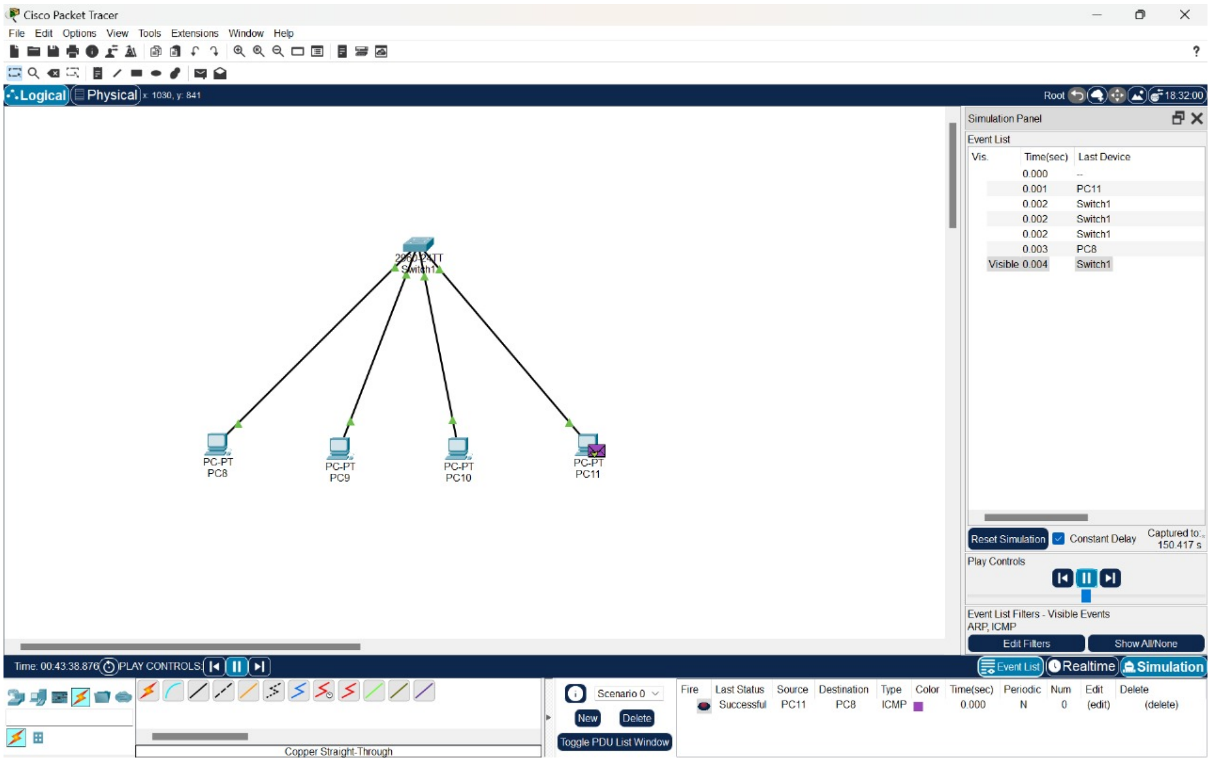
- Configure the static IPs for the end devices with for eg. 11.0.0.2 , 11.0.0.3 and so on and add corresponding subnet mask as 255.0.0.0. etc
- Now in each of the end devices , make sure to power off and remove the LAN port and add "WMP300N" and then power on the devices.
- In the config section of the wireless router , in Wireless 2.4G section , provide the SSID for the wireless router and add the same SSID in the end devices of the network

TESTING THE NETWORK

- Use the ping command in the command prompt to check for successful transmission between the two networks.
- If the IP address gets pinged in other system successfully then the network is working successfully.

- Transmission can also be checked by simulating the particular of transfer of data packet from source to destination.

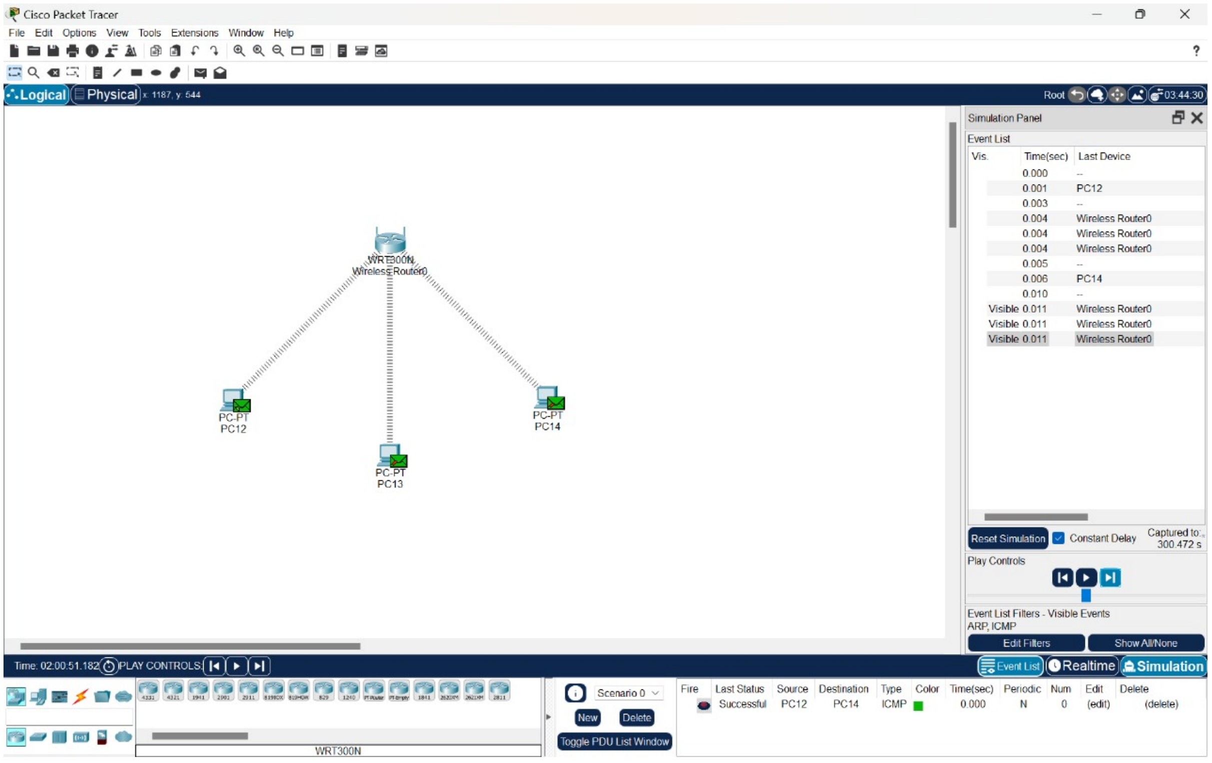
Simulation Output:



Event List

Vis.	Time(sec)	Last Device
	0.000	--
	0.001	PC11
	0.002	Switch1
	0.002	Switch1
	0.002	Switch1
	0.003	PC8
Visible	0.004	Switch1

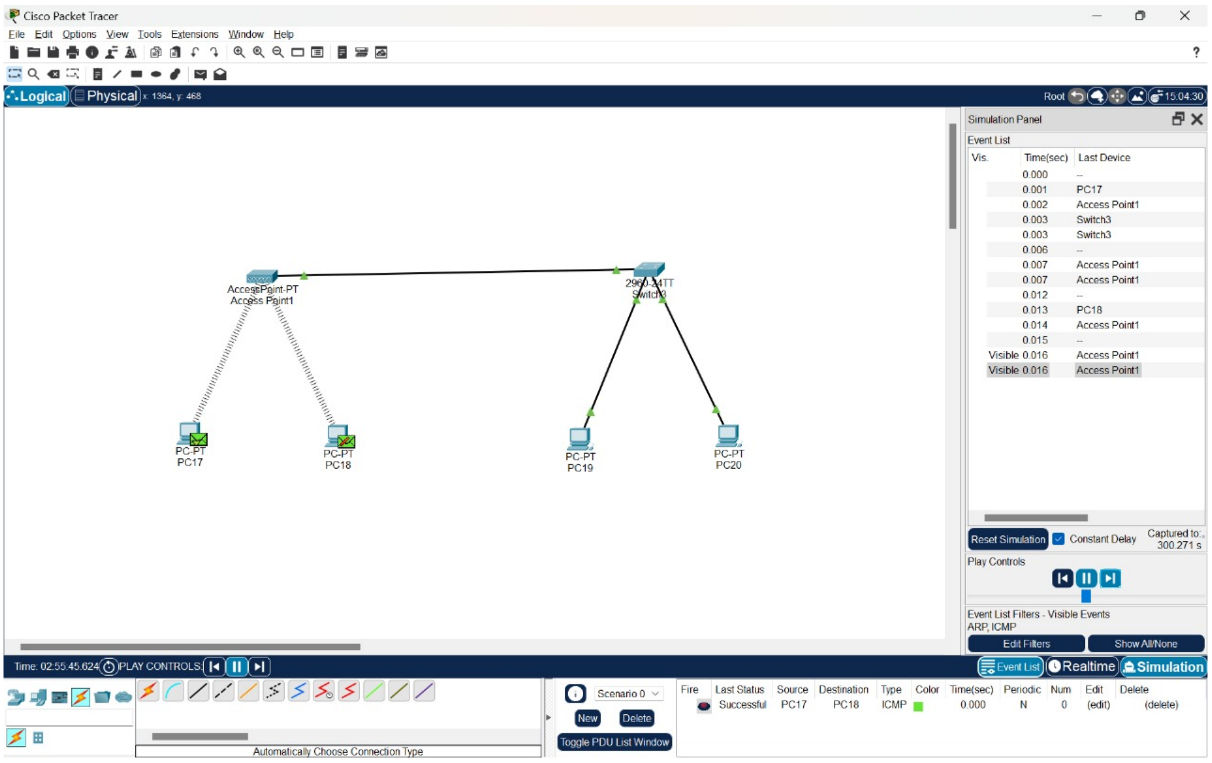
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC11	PC8	ICMP	🟪	0.000	N	0	(edit)	(delete)



Event List

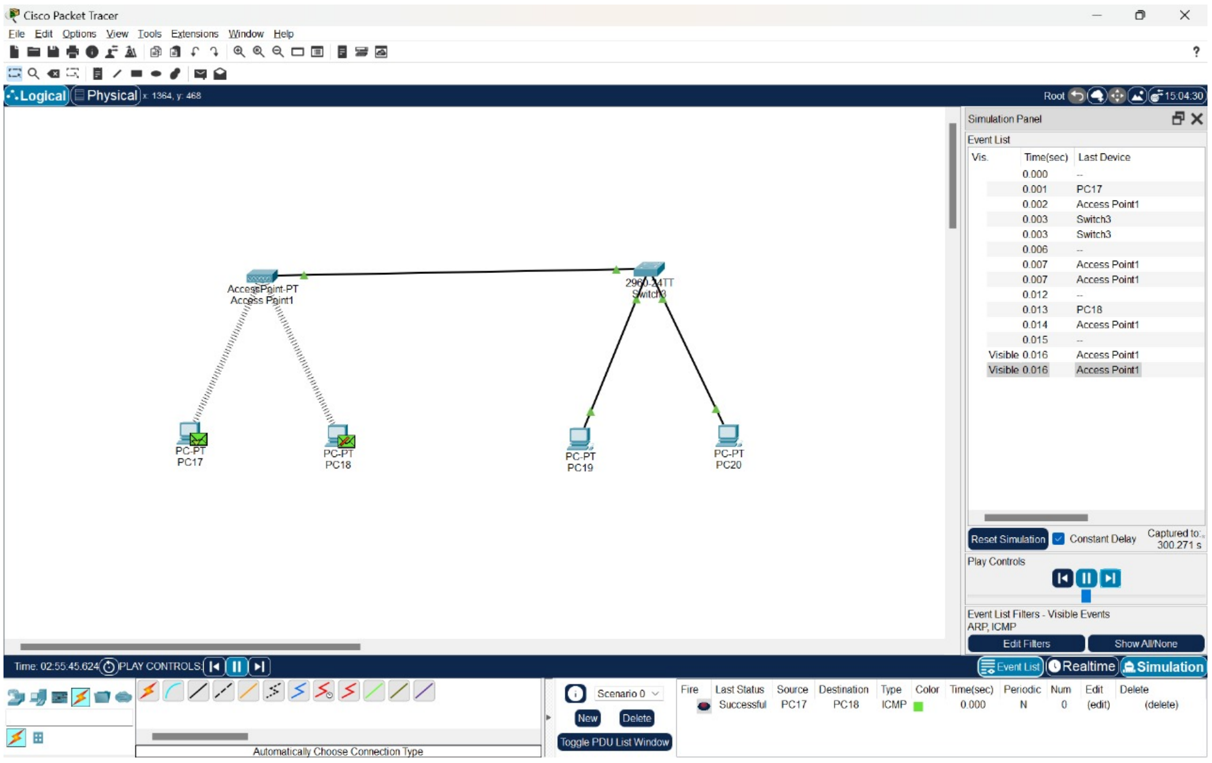
Vis.	Time(sec)	Last Device
	0.000	--
	0.001	PC12
	0.003	--
	0.004	Wireless Router0
	0.004	Wireless Router0
	0.004	Wireless Router0
	0.005	--
	0.006	PC14
	0.010	--
Visible	0.011	Wireless Router0
Visible	0.011	Wireless Router0
Visible	0.011	Wireless Router0

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC12	PC14	ICMP	🟩	0.000	N	0	(edit)	(delete)



Vis.	Time(sec)	Last Device
	0.000	--
	0.001	PC17
	0.002	Access Point1
	0.003	Switch3
	0.003	Switch3
	0.006	--
	0.007	Access Point1
	0.007	Access Point1
	0.012	--
	0.013	PC18
	0.014	Access Point1
	0.015	--
Visible	0.016	Access Point1
Visible	0.016	Access Point1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC17	PC18	ICMP	🟩	0.000	N	0	(edit)	(delete)



Vis.	Time(sec)	Last Device
	0.000	--
	0.001	PC17
	0.002	Access Point1
	0.003	Switch3
	0.003	Switch3
	0.006	--
	0.007	Access Point1
	0.007	Access Point1
	0.012	--
	0.013	PC18
	0.014	Access Point1
	0.015	--
Visible	0.016	Access Point1
Visible	0.016	Access Point1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC17	PC18	ICMP	🟩	0.000	N	0	(edit)	(delete)

Result:

Thus, simple LAN networks, with both wired and wireless communication, were implemented and communication between host devices were established and tested successfully**.**