Laboratory Manual

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAM-B.E

COMPUTER NETWORKS LAB (22ITC11)

(Common for CSE, CSE-AI&ML, CSE-IoT, AI&ML, IT, AI&DS)

SEMESTER-II

R-22 Regulation

Prepared by:**.....** ***Verified by:*****.....**

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Osmania University, Approved by AICTE,
Accredited by NAAC with A++ Grade and Programs Accredited by NBA)
Chaitanya Bharathi Post, Gandipet, Kokapet (Vill.), Hyderabad, Ranga Reddy - 500 075, Telangana
www.cbit.ac.in

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Osmania University, Approved by AICTE,
Accredited by NAAC with A++ Grade and Programs Accredited by NBA)
Chaitanya Bharathi Post, Gandipet, Kokapet (Vill.), Hyderabad, Ranga Reddy - 500 075, Telangana
www.cbit.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

Name of the Lab Course:
COMPUTER NETWORKS LAB (22ITC11)

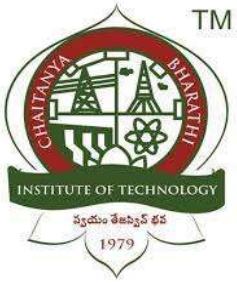
INDEX

LABORATORY / PRACTICAL

5	Installation set up of Network simulator software (NS2/NS3/NetSim/OPNET/QualNet/OMNet++/J-Sim and Cisco Packet Tracer).			
6	Simulation of Star topology.			
7	Simulation of Stop and Wait Protocol, Sliding Window Protocol			
8	Simulation of the Routing algorithms (LSR/DVR)			
9	Simulation of data transmission using TCP and evaluate the performance metric (Packet delivery ratio, Throughput, Jitter/Average end to end delay)			
10	Simulation of data transmission using UDP and evaluate the performance metric (Packet delivery ratio, Throughput, Jitter/Average end to end delay)			
11	Implement Socket Programming.			
12	Implement SMTP protocol.			

Note:

a) Minimum 12 experiments should be included in every course other than experiments beyond syllabus.
b) Add project/ case studies/ student defined experiments for the courses having experiments less than 12 in syllabus.
c) Include Minimum 2 experiments beyond prescribed experiments meeting industry problems to challenge student learning.



CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Osmania University, Approved by AICTE,
Accredited by NAAC with A++ Grade and Programs Accredited by NBA)
Chaitanya Bharathi Post, Gandipet, Kokapet (Vill.), Hyderabad, Ranga Reddy - 500 075, Telangana
www.cbit.ac.in

Vision of Institute

To be the Centre of Excellence in Technical Education and Research.

Mission of Institute

To address the Emerging needs through Quality Technical Education and Advanced Research.

Quality Policy

CBIT imparts value based Technical Education and Training to meet the requirements of students, Industry, Trade/ Profession, Research and Development Organizations for Self-sustained growth of Society.

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Osmania University, Approved by AICTE,
Accredited by NAAC with A++ Grade and Programs Accredited by NBA)

Chaitanya Bharathi Post, Gandipet, Kokapet (Vill.), Hyderabad, Ranga Reddy - 500 075, Telangana

www.cbit.ac.in

DEPARTMENT OF INFORMATIO TECHNOLOGY

Vision of the Department

Mission of the Department

Program Educational Objectives (PEOs)

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Osmania University, Approved by AICTE,
Accredited by NAAC with A++ Grade and Programs Accredited by NBA)
Chaitanya Bharathi Post, Gandipet, Kokapet (Vill.), Hyderabad, Ranga Reddy - 500 075, Telangana
www.cbit.ac.in

DEPARTMENT OF …….

Program Outcomes (POs)

PO1. ----------

PO2.----------

PO3.----------

PO4.----------

PO5.----------

PO6.----------

PO7.----------

PO8.----------

PO9.----------

PO10.----------

PO11.----------

PO12.----------

Program Specific Outcomes (PSOs)

PSO1.----------

PSO2.----------

PSO3.----------

COMPUTER NETWORKS LAB

22ITCXX

Instruction	3P Hours per week
Durationof SEE	3 Hours
CIE	50 Marks
SEE	50 Marks
Credits	1.0

CourseObjectives:

1. Toknow about the fundamentals of peer to peer networks.
2. Tofamiliarize with the installation and configuration of Physical systems and network connections.
3. Tolearn the implementation methodologies of Wire shark software packages.
4. Toexplore the concepts of simulations.
5. Toacquire knowledge onAlgorithms using Simulation software's.

Course Outcomes:

Uponsuccessfulcompletion ofthiscourse,students will be able to:

1. Describe the concepts of Peer to Peer Networks.
2. Implement the configuration of Physical hosts and sharing the network devices.
3. Analyze the network issues by using Wireshark Software.
4. Solve the Network Problems by using Simulators.
5. Illustrate various QoS metrics.

Mapping of CourseOutcomeswithProgramOutcomesandProgramSpecificOutcomes:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	2	2	3	-	-	-	-	-	-	1	3	3	3
CO2	3	3	2	2	3	-	-	-	-	-	-	1	3	3	2
CO3	2	2	2	3	3	3	1	-	-	-	-	-	3	3	3
CO4	2	2	2	3	3	3	1	-	-	-	-	-	3	3	2
CO5	2	2	2	3	3	3	1	-	-	-	-	-	3	3	3

LIST OF PROGRAMS

1. Configure Peer to Peer Network with at least three hosts.
2. Share Files/Folder, Devices and Printer in the Network and access the shared resources from the other node
3. Use Wireshark Packet sniffer software and captures TCP,UDP, IP, ARP, ICMP, Telnet, FTP packets
4. Write and analyze the output of various Network commands such as ping, ipconfig, arp, netstat, tracert, nslookup, hostname, systeminfo etc.,
5. Installation set up of Network simulator software (NS2/NS3/NetSim/OPNET/QualNet/OMNet++/J-Sim and Cisco Packet Tracer).
6. Simulation of Star topology.
7. Simulation of Stop and Wait Protocol,
8. Simulation of Sliding Window Protocol
9. Simulation of the Routing algorithms (Link State Routing/Distance Vector Routing)
10. Implement Socket Programming.
11. Implement SMTP protocol.

TextBooks:

1. AndrewS.Tanenbaum,Computer Networks,PearsonEducation,6th Edition,2021.
2. MichaelGregg, "BuildYourOwnSecurityLab",WileyPublishing, Inc.,2008.
3. Michael E.whitman, Herbert J.Mattord, Andrew Green ,"Hands on Information Security lab manual",CengageLearning,Fourthedition,December27, 2013.

SuggestedReading:

1. JamesF.Kurose,KeithW.Ross, "ComputerNetworking–ATop-DownApproachFeaturingtheInternet", 8th Edition,PearsonEducation,2022.

WebResources:

1. https://nmap.org
2. https://www.snort.org
3. https://www.wireshark.org
4. NS2 Projects Tutorials | How to install NS2 Software | Network Simulation Tools
5. Network Simulator 2 (NS2) : Steps For Installing NS2 (tutorialsweb.com)
6. The Network Simulator ns-2: Documentation (isi.edu)
7. Language (tcl.tk)

ASSESSMENT PROCEDURE AND AWARD OF CIE MARKS

Following is the subdivision for the internal marks (50) of the Lab:

(i) 20 marks for the lab internal tests

Two tests are to be conducted i.e one test after 1st cycle of experiments and second test after the second cycle. Average of two tests marks put together should be consider (20 maximum)

(ii) 30 marks for CIE

For the CIE 30 marks will be awarded based on the rubrics provided below.

This Rubrics are general guideline. Based on the lab type (programming or hardware) and complexity of the course Rubrics can be customized by the department in tune to program and course offered. Performance Indicators of the Rubrics also can be changed by the departments/Program based on the need.

RUBRICS

S. No	Parameter	Descriptors and Score				
		Outstanding	**Good**	**Fair**	**Poor**	**Very Poor**
1	Pre-Experiment Preparation work 5 M	Well prepared for the experimentation with a clear specifications, plan/design and additional information (5M)	prepared for the experimentation with clear specifications and plan/design (4M)	Adequately prepared for the experimentation without clear specifications and plan/design (3M)	Minimal preparation and without clear specifications and plan/design (2M)	Lacks preparation and without clear specifications and plan/design (1 M)
2	Experimentation (Problem Solving, Methodology of Conduction) 10 M	Student conducts experiment with all possible test cases in an optimized fashion.(10M)	Student solves the problem and conducts experiments with all possible test cases (8M)	Student solves the problem and conducts experiment with few possible test cases with complexity (6M)	Student solves the problem with few test cases with complexity (4M)	Student fails to solve the problem (2M)
3	Post Experiment Analysis [Viva, Inference] 5M	Demonstrates the simulation/ findings /Hardware results Infers and answer all the Questions posed by Instructor	Demonstrates results and inference; Able to answer Few Questions posed by Instructor (4M)	Demonstrates results and inference; Unable to answer the Questions posed by Instructor. (3M)	Demonstrates Partial results and inference; Unable to answer the Questions posed by Instructor (2M)	Failed to Demonstrates results and inference; Unable to answer the Questions posed by Instructor (1M)

S. No	Parameter	Descriptors and Score				
		Outstanding	**Good**	**Fair**	**Poor**	**Very Poor**
		(5M)				
4	Report Writing 5M	Report meets all requirements and it is prepared in original and creative way to engage readers (5M)	Report with well-organized content, visuals, graphics, citations and references (4M)	Report is complete and adequate with poor grammar. (3M)	Report is complete, poor grammar and inadequate and failed to organize thoughts (2M)	Report is incomplete, unclear, poor grammar and inadequate (1M)
5	Conduct (Ethics, Safety, Team Work)5M	Excellent team spirit, strictly follows ethics and safety precautions with good team work (5M)	Follows the safety precautions, practices ethics and poor team work (4M)	Follows safety precautions and ethical practices and failed to exhibit teamwork. (3M)	Followed minimum safety precautions and ethical practices and failed to exhibit team work. (2M)	Does not Follows safety precautions and ethical practices and failed to exhibit team work. (1M)
TOTAL SCORE						

ASSESSMENT PROCEDURE AND AWARD OF SEE MARKS

S. No.	Parameter	Descriptors and Score				
		Outstanding	**Good**	**Fair**	**Poor**	**Very**
1	Record 5 M	The content of all the experiments is well-organized, recorded the tables and neatly drawn graphs. Results and discussions are well presented. (5M)	The content of most of the experiments are well-organized, recorded the tables, neatly drawn graphs. Results and discussion are presented. (4M)	The content of the most of experiments are well-organized, recorded the tables, neatly drawn graphs. Results and discussion are not presented (3M)	The content of the few experiments is not well-organized, recorded the tables, neatly drawn graphs. Results and discussion are not presented (2M)	The conten all the exper not well-org recorded the neatly draw Results and discussion a presented (1
2	Write up about the experiment 10M	Presentation of the given experiment/program is very well organized with the required content, specifications, plan/procedure of the conduct and all the required additional information. (10M)	Presentation of the given experiment/program is organized with the required content, specifications, plan/procedure of the conduct. (8M)	Presentation of the given experiment/program is organized and is without clear specifications and plan/design. (6M)	Presentation of the given experiment/program is minimal with clear specifications and plan/design. (4M)	Presentation given experiment/ is minimum clear specifi and plan/de (2M)
3	Conduction of the experiment and observations 15M	Conducts experiment / Simulate the problem with proper connections / all possible test cases. All the possible observations are noted. (15M)	Conducts experiment / Simulate the problem with connections / with most of the test cases. Failed to note all observations(12M)	Conducts experiment / Simulate the problem with proper connections / less test cases. (9M)	Conducts experiment / Simulate the problem with improper connections / less test cases. (6M)	Failed to Co experiment Simulate the problem wit improper connections test cases. (3
4	Results & Analysis 10M	Demonstrates the experimental results with adequate analysis / simulation/ findings /obtained results /plotting the graphs. (10M)	Demonstrates the experimental results with required analysis / simulation/ findings /obtained results /plotting the graphs. (8M)	Demonstrates the experimental results, failed to maximum analysis / simulation/ findings. (6M)	Demonstrates the partial experimental results with least analysis / simulation/ findings. (4M)	Failed to Demonstrat results and analysis / simulation/ (2M)
5	Viva-Voce 10M	Answers most of the questions with good analytical explanation. (10M)	Answers most of the questions with good explanation. (8M)	Answers only few of the questions with good explanation. (6M)	Answers only few of the questions with nominal explanation. (4M)	Failed to ans questions. (2
TOTAL SCORE						

GENERAL INSTRUCTIONS FOR LABORATORY CLASSES

DO'S

1. Without Prior permission do not enter into the Laboratory.
2. While entering into the LAB students should wear their ID cards.
3. The Students should come with proper uniform.
4. Students should sign in the LOGIN REGISTER before entering into the laboratory.
5. Students should come with observation and record note book to the laboratory.
6. Students should maintain silence inside the laboratory.
7. After completing the laboratory exercise, make sure to shutdown the system properly

DONT'S

1. Students bringing the bags inside the laboratory..
2. Students wearing slippers/shoes insides the laboratory.
3. Students using the computers in an improper way.
4. Students scribbling on the desk and mishandling the chairs.
5. Students using mobile phones inside the laboratory.
6. Students making noise inside the laboratory.

EXPERIMENT / PRACTICAL -1

Configure Peer to Peer Network with at least three hosts.

AIM :

To Configure Peer-to-Peer Network with atleast three hosts

DESCRIPTION:

A peer-to-peer (P2P) network is a decentralized communications model in which each participant (referred to as a peer) has equal status and can initiate or complete transactions directly with other peers. Here are some key characteristics and features of P2P networks:

KEY CHARACTERISTICS:

1. **Decentralization**:
   - **No Central Authority**: There is no central server or authoritative control; each peer has equal power and responsibility.
   - **Autonomous Nodes**: Peers independently maintain the network and its functions.
2. **Direct Interaction**:
   - **Direct Data Exchange**: Peers can directly share resources, such as files or processing power, with one another.
   - **Self-Sustaining**: Peers collectively provide the necessary infrastructure to support network functions.
3. **Scalability**:
   - **Dynamic Expansion**: The network can grow or shrink as peers join or leave without significant impact on overall performance.
   - **Resilience**: The failure of one or several nodes does not necessarily disrupt the network.
4. **Distributed Resources**:
   - **Shared Resources**: Resources such as bandwidth, storage, and processing power are distributed across all peers.
   - **Load Balancing**: Tasks and data are distributed to balance the load among peers.

COMMON USES OF P2P NETWORKS:

1. **File Sharing**:
   - **Examples**: BitTorrent, Gnutella, Napster (early version).
   - **Function**: Allows users to share large files, like music, videos, and software, directly with each other.
2. **Communication Platforms**:
   - **Examples**: Skype (early versions), Discord (**Discord** is great for playing games and chilling with friends, or even building a worldwide community).
   - **Function**: Facilitates direct communication and data exchange without relying on centralized servers.
3. **Cryptocurrency**:
   - **Examples**: Bitcoin, Ethereum.
   - **Function**: Supports decentralized digital currencies and transactions through blockchain technology.
4. **Distributed Computing**:
   - **Examples**: SETI@home, Folding@home.

- **Function**: Utilizes the idle processing power of numerous computers to perform complex computations.

ADVANTAGES:

- **Redundancy and Reliability**: The network is less likely to experience downtime because there is no single point of failure.
- **Cost-Efficiency**: Reduces the need for central servers and infrastructure, lowering operational costs.
- **Enhanced Privacy**: Data is not stored on a central server, reducing the risk of mass data breaches.

DISADVANTAGES:

- **Security Risks**: Each peer is responsible for its own security, making the network vulnerable to malicious nodes.
- **Data Integrity**: Ensuring data consistency and integrity can be challenging due to the lack of centralized control.
- **Scalability Issues**: While P2P networks can scale dynamically, managing and coordinating a large number of peers can become complex.

Overall, P2P networks offer a flexible and resilient alternative to traditional client-server architectures, particularly suitable for scenarios requiring direct, decentralized interactions.

MinimumTheoreticalBackground

In Peer to Peer architecture every node is connected to other node directly for exchanging information instead of connected to central server.

Every computer node is referred as peer and they do the job of client as well as server both. Every peer provides services to other peers as well as uses services provided by other peers.

❖ **Configuring peer to peer network "Crossover cable is used"**

One end is used for transmitting and other end for receiving data. Peer-to-Peer networking is when all computers are on the same network. They are considered as peers and will have to be connected to a hub, switch or a router. There is no server, controller or one in charge. Computers in a workgroup shares resources such as the printer and files. Workgroup is automaticallyset up when you set up a network and they all share the same subnet. A workgroup is not protected by a password, no security is provided.

HARD WARE REQUIREMENTS:

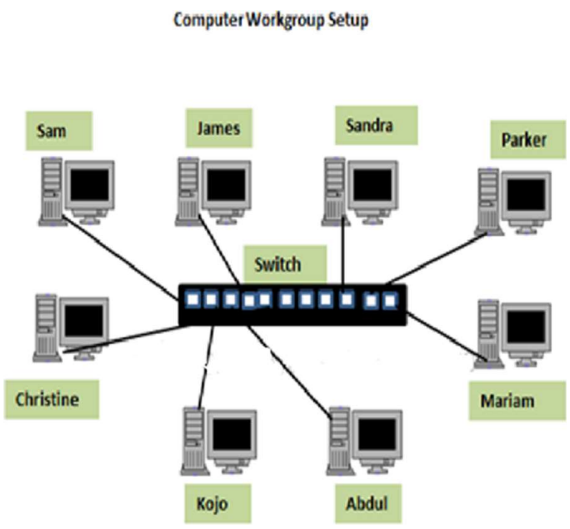
1. Computer with Configuration – CPU HDD capacity: 250 GB, RAM : 4 GB, i3processor
2. NetworkInterface Card - Manufacturer: Cisco
3. Switch(min.8 ports)
4. Crossover Cable

SOFTWARE REQUIREMENTS:

1. Windows Operating Systems

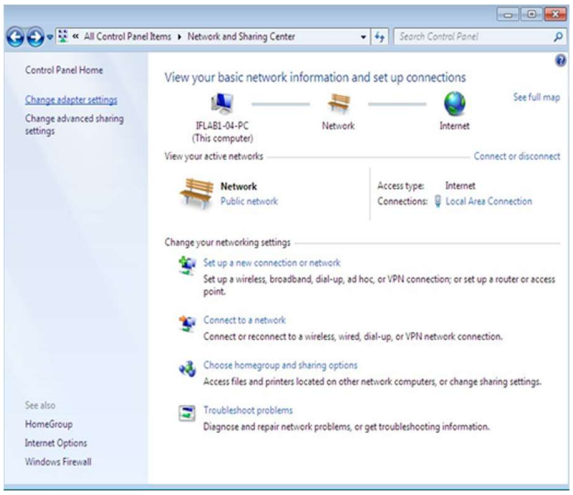
DIAGRAMS/EXPERIMENTALSET-UP/WORK SITUATION

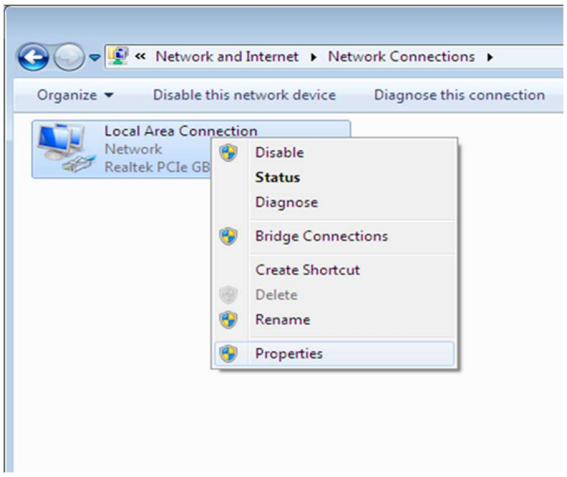
A typical exampleof aworkgroup is shown below:



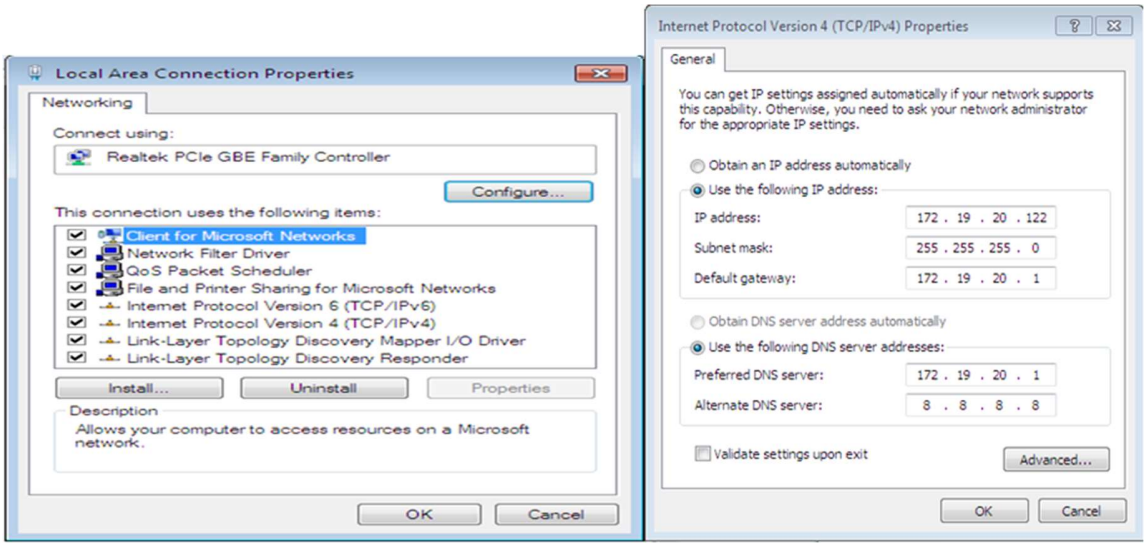
PROCEDURE

- **To set a static IP on your Windows computer**





1. Click **StartMenu>Control Panel>Network and Sharing Center**.Click **Change adapter settings**.
2. Right-click on **Local Area Connection** and clickon **Properties**.

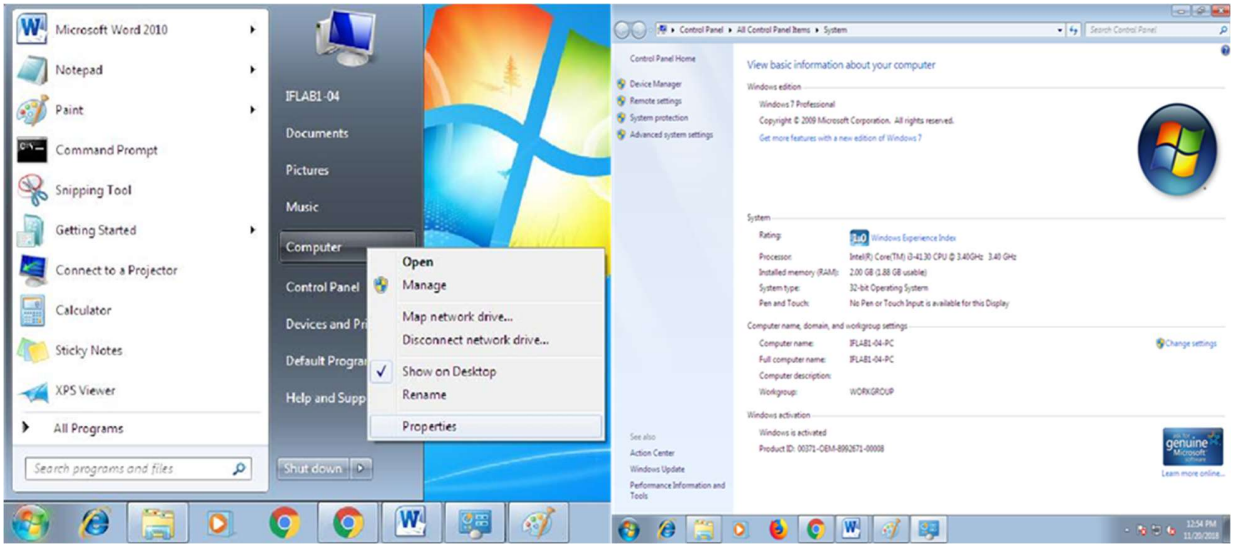


1. Select **Internet Protocol Version4(TCP/IPv4)**and click on **Properties**.
Select "**Use the following IP address"** and enter the IP address, Subnet Mask and DNS server.Click **OK** and close the **Local Area Connection** properties window.

❖ **How Workgroup works**

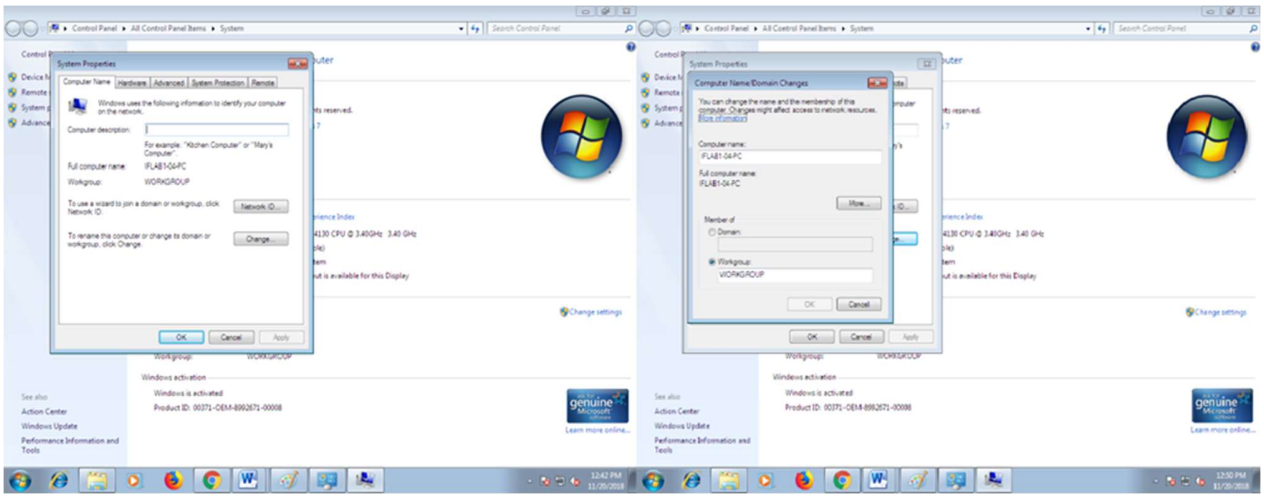
A computer joining a workgroup is assigned to the same workgroup name this process makes accessing the computers easier.

❖ **How to create a workgroup**



Clickon **Start** button Right-clickon **Computer** and then click **Properties**

2. Under **Computer name, domain, and workgroup settings**, click **Change settings**.



3. In the **SystemProperties** dialog box, click the **Computer Name** tab and then click **Change**.
4. In the Computer Name/Domain Changes dialog box, under Member of, click **Workgroup**
5. Then do one of the following: To join an **existing** workgroup, type the name of the workgroup thatyouwant to join, and thenclick **OK**. To create a **new** workgroup,type the name of the workgroup that you want to create,and then click **OK**

Note:Repeat the steps of setup of IP address and setup of Workgroup for third computer

❖ **Peer-to-peer applications**.

- **Skype**,an Internet telephony network,uses P2P technology.
- **Instant messaging** systems and **online chat** networks.
- **Bitcoin** and **PPCoin** are peer-to-peer-based digital currencies.
- **Dalesa** a peer-to-peer web cache for LANs(based onIPmulticasting).
- **OpenGarden**,connection sharing application that shares Internet access with other devices using Wi-Fi or Bluetooth.
- Streamingmedia.**P2PTV**and**PDT**

II. PRECAUTION

1. HandleComputerSystemandperipheralswithcare
2. Follow SafetyPractices

VIVA QUESTIONS

1. What is peer?
2. Whatispeertopeernetwork?
3. Howpeertopeerisdiffersfromclient-servernetwork?
4. Giveadvantagesof peertopeer network.
5. Givedisadvantagesofpeertopeer network.

RESULT:

The experiment has been conducted successfully.

EXPERIMENT / PRACTICAL -2

Share Files/Folder, Devices and Printer in the Network and access the shared resources from the other node

AIM:

Share Files/Folder, Devices and Printer in the Network and access the shared resources from the other node

DESCRIPTION:

Sharing files and folders between devices can be done in several ways, depending on your needs and the devices involved. Here are some common methods:

1. Using Cloud Services

- **Google Drive, Dropbox, OneDrive:** Upload files to these services and share links with others or sync files across devices.
- **iCloud:** For Apple devices, iCloud allows file sharing and syncing.

2. Network Sharing

- **Windows:** Use the "Share" feature or set up a HomeGroup to share files and folders over a local network.
- **Mac:** Use AirDrop for quick sharing, or enable file sharing in system preferences.
- **Linux:** Use Samba to share files with Windows or other Linux systems on the same network.

3. Direct Transfer

- **USB Drive/External Hard Drive:** Physically transfer files using an external storage device.
- **Bluetooth:** Pair devices via Bluetooth and send files wirelessly.
- **Email:** Attach files to an email and send them to yourself or others.

4. File Transfer Apps

- **AirDroid:** For Android devices, allows file transfer to and from PCs.
- **SHAREit, Xender:** Cross-platform apps that facilitate file sharing between smartphones and computers.

5. File Transfer Protocols

- **FTP/SFTP:** Set up an FTP server on one device and connect to it from another.
- **WebDAV:** Use WebDAV to share files over the internet or a local network.

6. Collaboration Tools

- **Slack, Microsoft Teams, Google Workspace:** These tools often have integrated file sharing features for team collaboration.

7. Command Line Tools

- **rsync, scp:** For advanced users, these command-line tools can be used for file transfer between devices.

STEPS FOR SOME COMMON METHODS:

Google Drive:

1. Upload the file to Google Drive.
2. Right-click the file and select "Share."
3. Enter the email address of the person you want to share with or get a shareable link.

Windows Network Sharing:

1. Right-click the folder you want to share and select "Properties."
2. Go to the "Sharing" tab and click "Share."
3. Choose the users to share with and set permissions.

AirDrop (Mac/iOS):

1. Open Finder (Mac) or Control Center (iOS) and enable AirDrop.
2. Select the file, click the "Share" button, and choose AirDrop.
3. Select the nearby device to send the file.

HARD WARE REQUIREMENTS:

1. Computer with Configuration – CPU HDD capacity: 250 GB RAM : 4 GB, i3 processor
2. Network Interface Card Manufacturer:Cisco
3. Switch (min. 8 ports)
4. Crossover Cable

SOFTWARE REQUIREMENTS:

Windows Operating Systems

PracticalSignificance

You can share Computer Resources

MinimumTheoreticalBackground

Are source,or system resource, is any physical or virtual component of limited availability within a computer system.Every device connected to a computer system is are source and every internal system component is also a resource.Major resource types are CPU time, Random access memory, Hard disk space, Network throughput, Electrical power, External Devices, Input/output operations. Virtual system resources include files, network connections and memory areas, where as a physical resource includes printer, scanner, fax machine etc.

Types of System Resources

1. Physical
2. Virtual

Types of Physical Resources

1. Printer
2. Scanner
3. FaxMachine

Typesof Virtual Resources

1. Memory
2. Files
3. CPU time

ResourceSharing

A shared resource or network share is a device or piece of information on a computer that can be remotely accessed from another computer typically via a local area network or an enterprise Intranet, transparently as if it were a resource in the local machine.

Examples are shared file access(also known as disk sharing and folder sharing),shared printer access (printer sharing), shared scanner access, etc.

Resource sharing means reduction in hardware costs. Shared files mean reduction in memory requirement, which indirectly means reduction in file storage expenses.

A network share can become a security liability when access to the shared files is gained(often by devious means)by those who should not have access to them.Many computer worms have spread through resource sharing.

Printer sharing is a feature which allows you to access and use a printer from other computers in network.If there are ten employees in an organization, each having their own computer, they will require ten printers if they want

to use there source at the same time.Printer sharing allows accessing the computers that can be interconnected using a network, and just one printer can efficiently provide the services to all ten users. Folder sharing is the public or private sharing of computer data or space in a network with various levels of access privilege.
A user sitting at one computer that is connected to network can easily see files present on another computers, provided he is authorized to do so.This saves him/her the hassle ofcarrying a storage device every time data needs to be transported from one system to another system

II. DIAGRAMS/EXPERIMENTAL SETUP/WORKSITUATION

Fig.Resource sharing

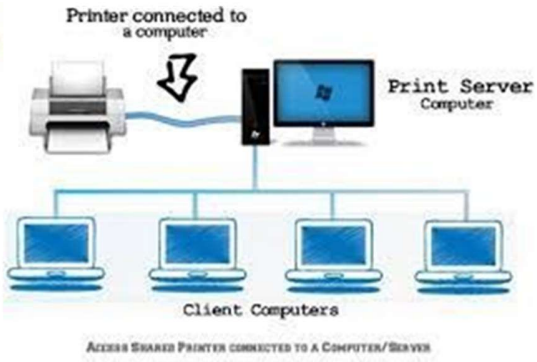
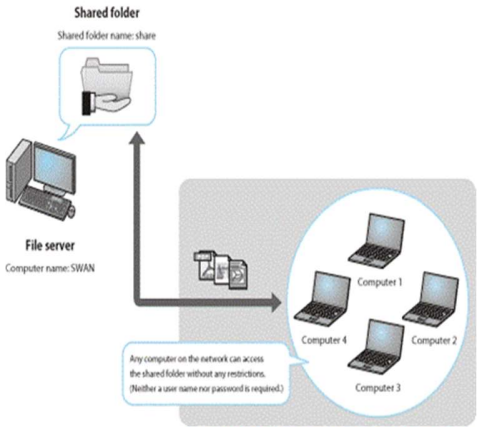


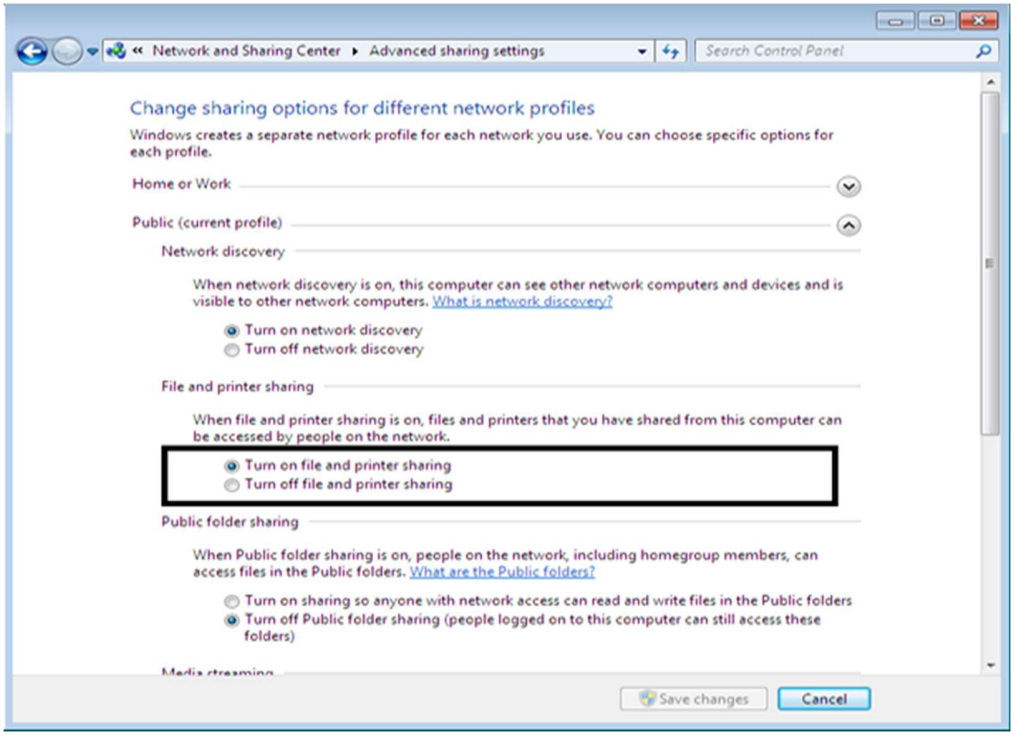
Fig.PrinterSharing



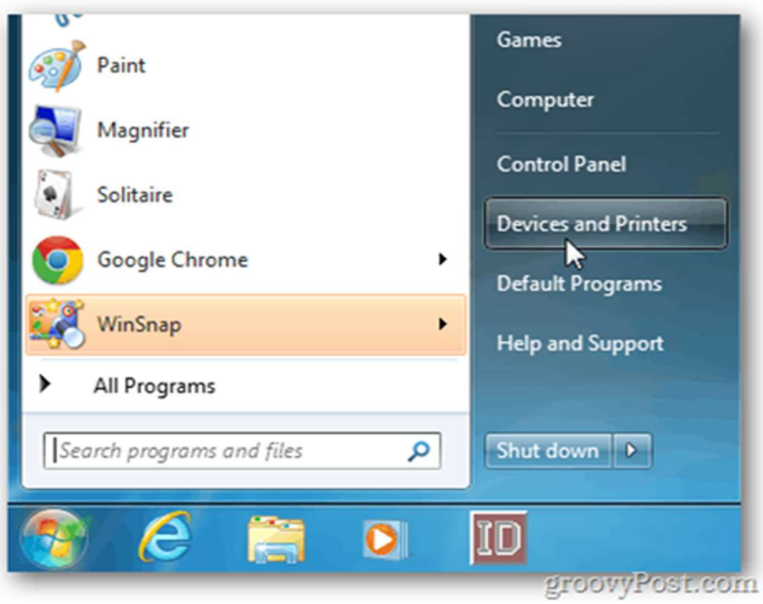
Folder Sharing

I. PROCEDURE

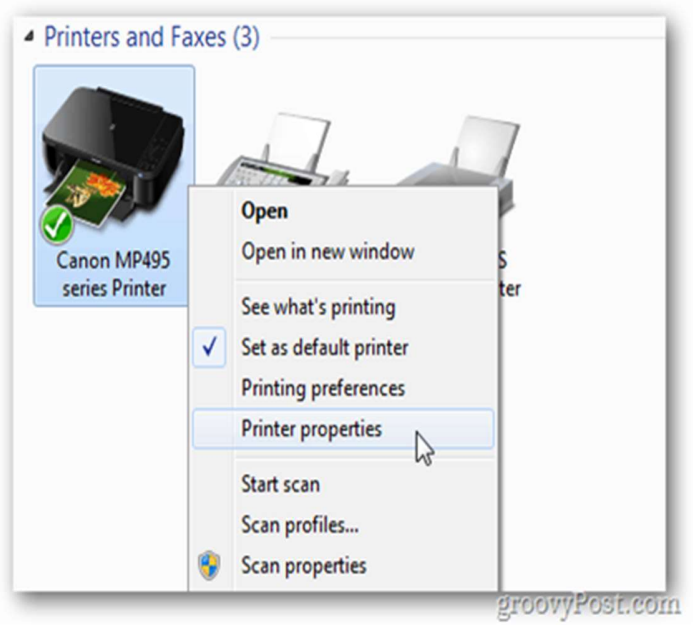
Share Printer and folder. Click on Start Button->Click on ControlPanel->Click on Network and Sharing Center->click on Change advanced sharing settings



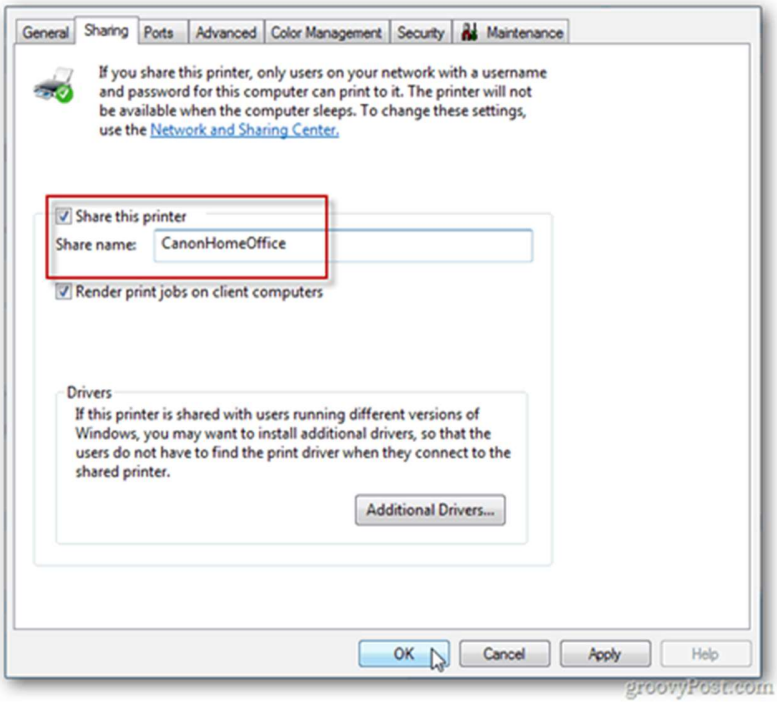
First start with the computer the printer is connected to.Make sure it‘s installed correctly with the latest drivers. click Start >> Devices and Printers.



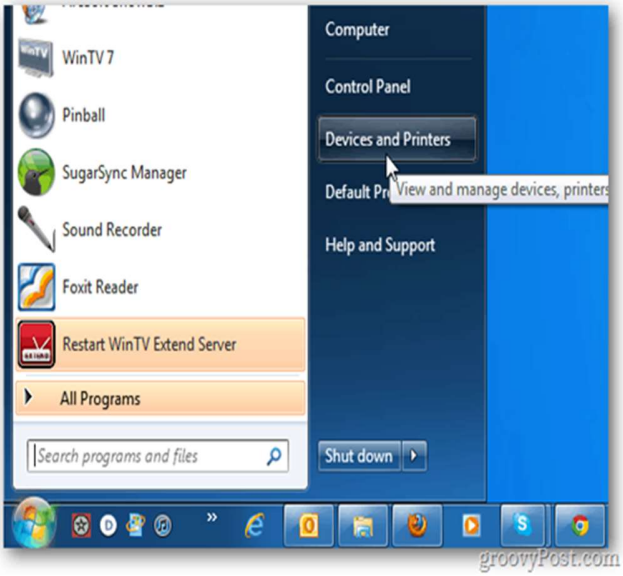
Next, right-click on the printer you want to share and select Printer Properties.



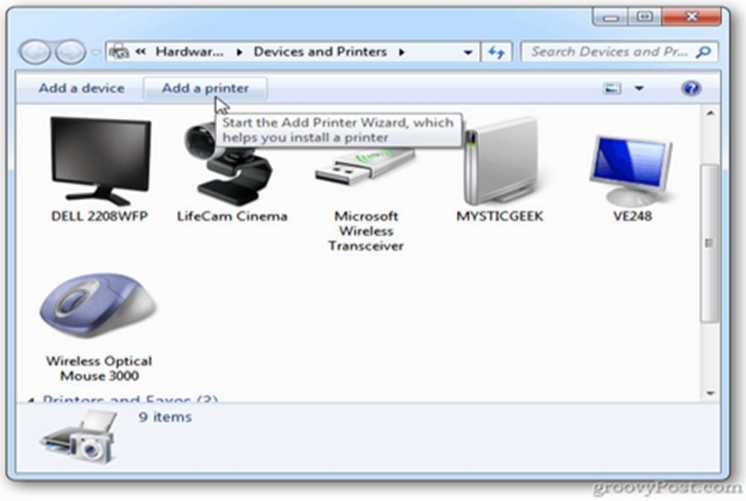
Click the Sharing tab.Make sure Share this Printer is checked and give it an easy to remember share name. Click OK.



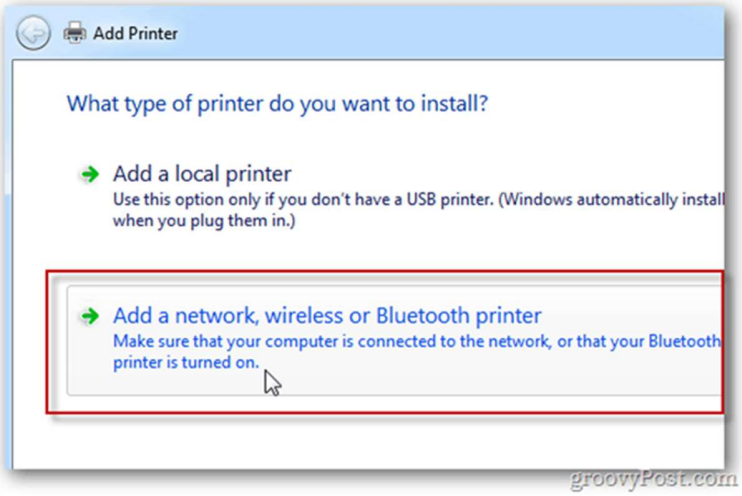
The computer the printer is attached to will need to be powered on to find and print to it. Now go to the other computer you want to print from.Click *Start>>Devices and Printers*



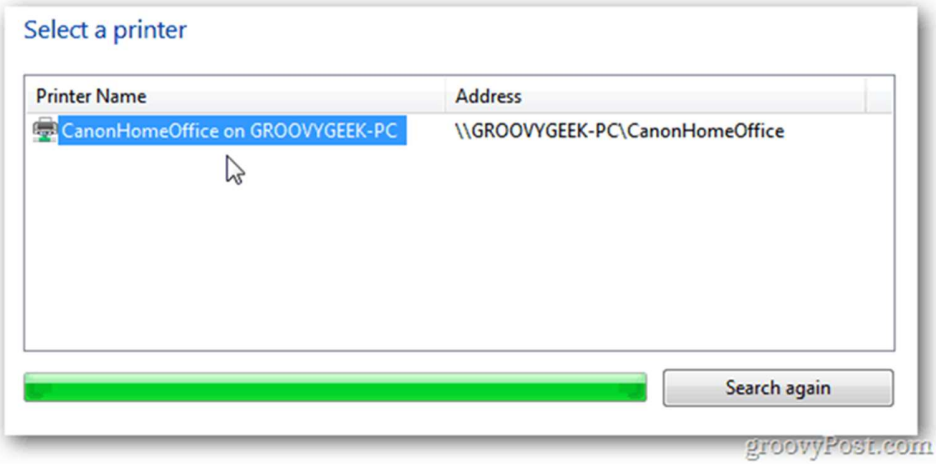
Click Add a Printer.



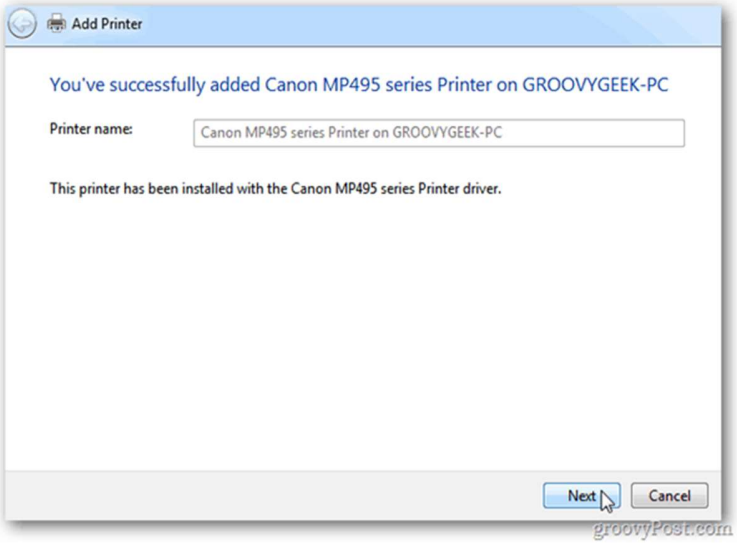
Next, click Add aNetwork,Wireless or Bluetooth Printer.



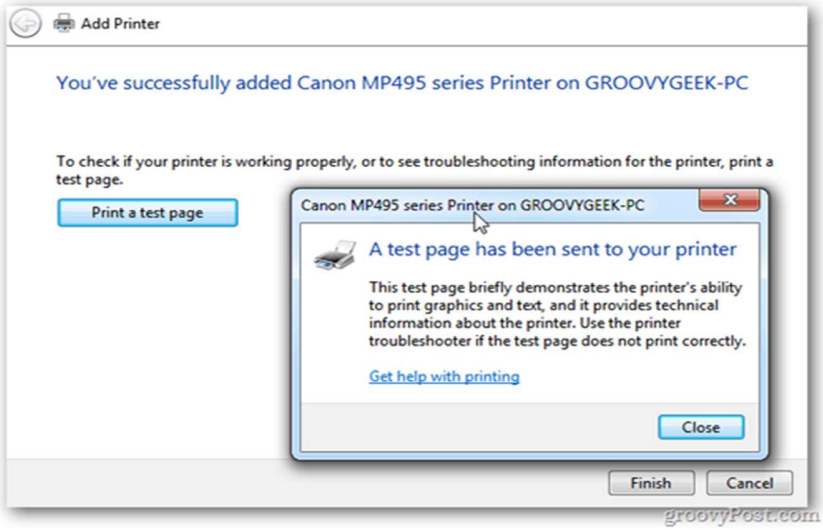
The system will search you rnet work for the shared printer.When it finds the printer, highlight it and click Next.



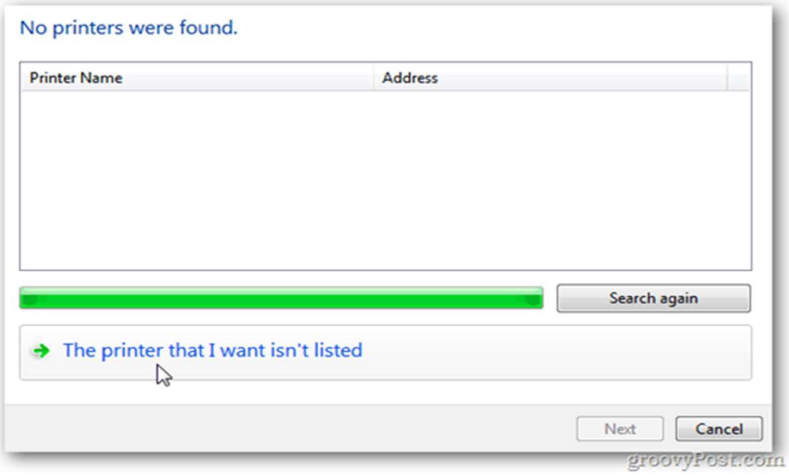
Success.Click Next.



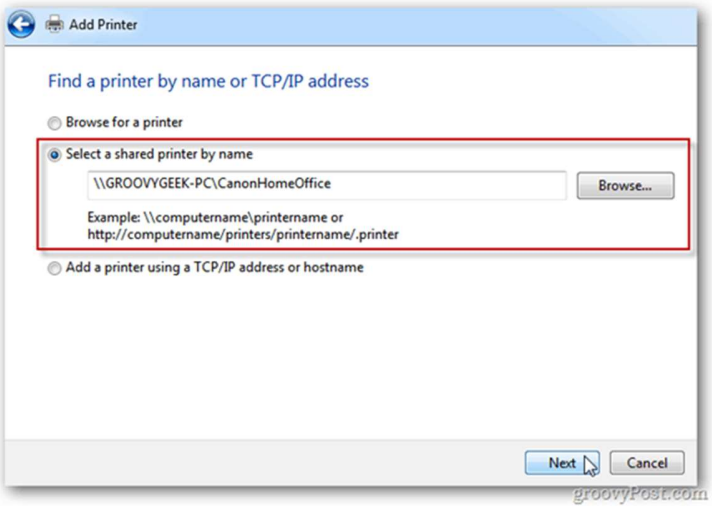
Back in Devices and Printers,you‘ll find the printer listed.Send a test page to the printer to verify it‘s working.



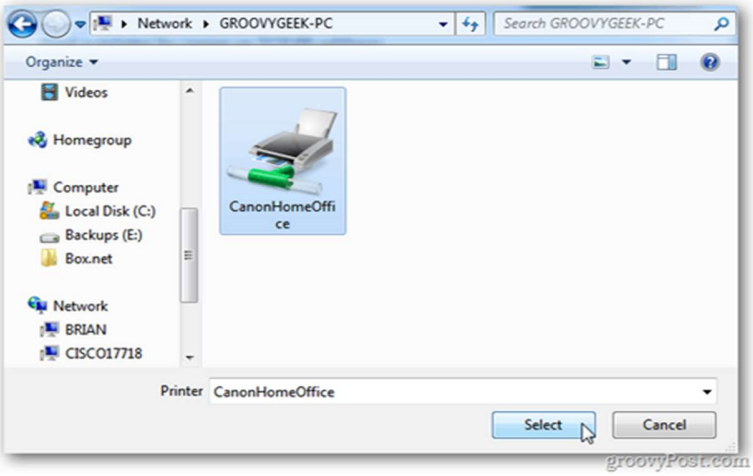
If Windows doesn‘t automatically find the printer, click ThePrinter That I Want Isn‘t Listed.



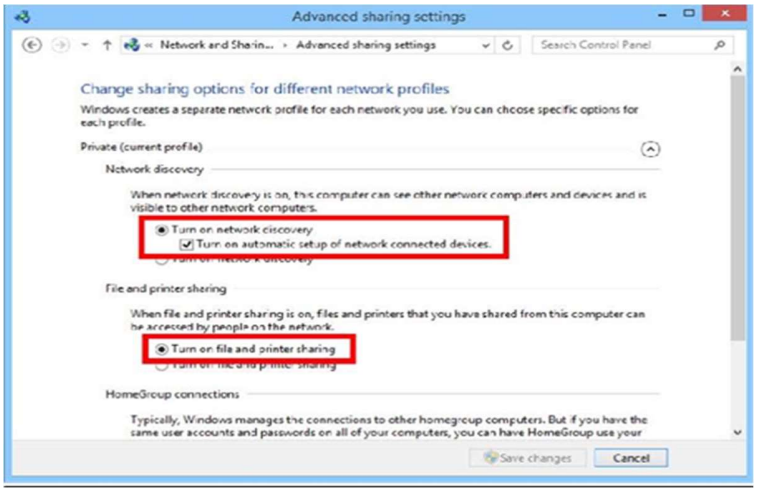
Check Select a Shared Printer byname and type the path in directly.



Or click Browse to find the printer and select it.



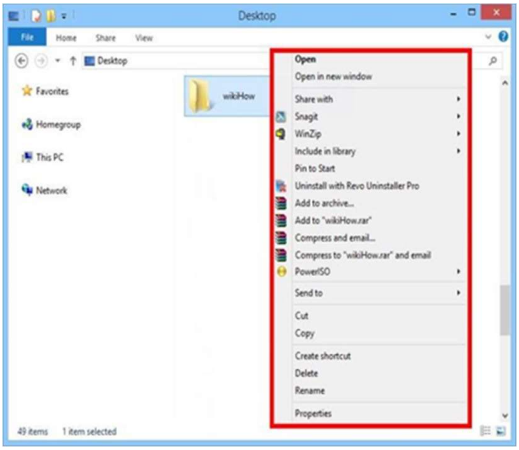
SharingSpecificFolders



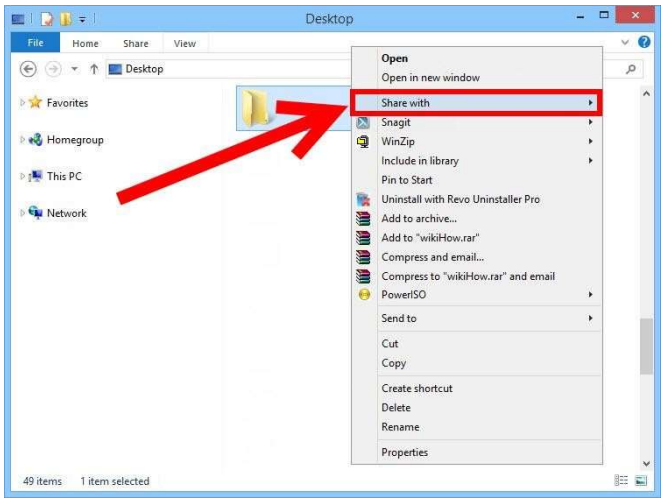
Ensure that File and Printer Sharing is enabled. In order to share specific folders, you will need to have this feature enabled. The method for enabling it varies slightly depending on which version of Windows you are using. It is highly recommended that you do not enable folder sharing when on a public network such as a school or coffee shop.

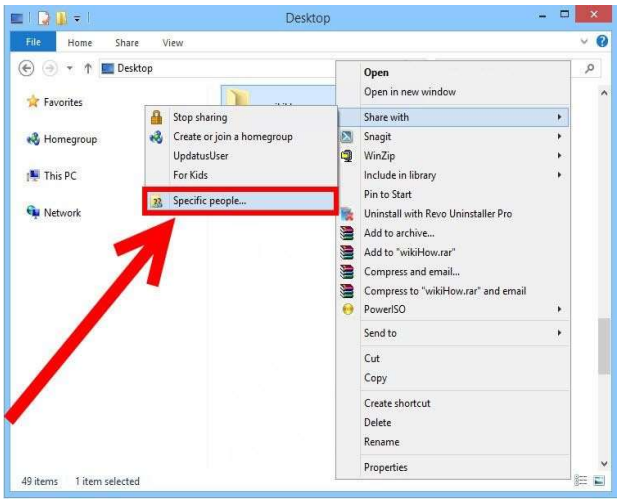
Windows 7 - Click the Start button, type "control panel", and press ↵ Enter. Double-click the "Network and Sharing Center"icon. Click the"Change advanced sharing settings"link. Expand the profile that you want to enable sharing on (Home/Work or Public). Turn on both "Network discovery" and "File and printer sharing". Click the "Save changes" button and enter your administrator password if necessary.

Find the folder you wish to share. Once File and Printer Sharing has been enabled, you can share any folder on your hard drive with other people on your network. Navigate to the folder that you want to share using Explorer. Right-click on it.



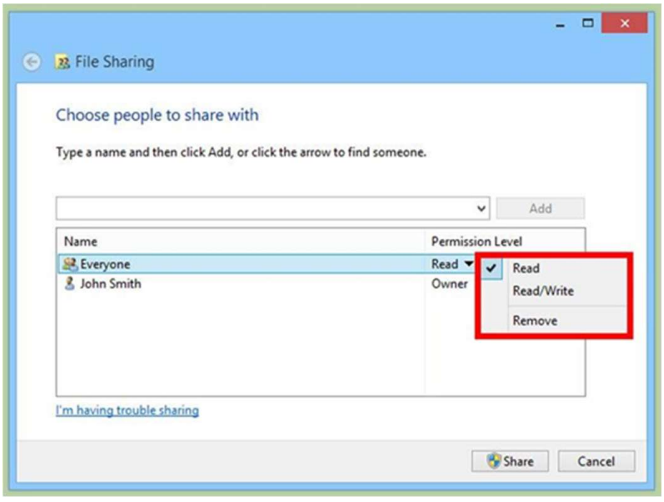
3. **Selectthe"Sharewith"option.** This will open the Sharing sub menu.You can choose to share it with everyone in your Homegroup or select specific people to share it with.When choosing a Homegroup option,you can allow other Homegroup members to both read and write to the folder, or limit them to just read from it.





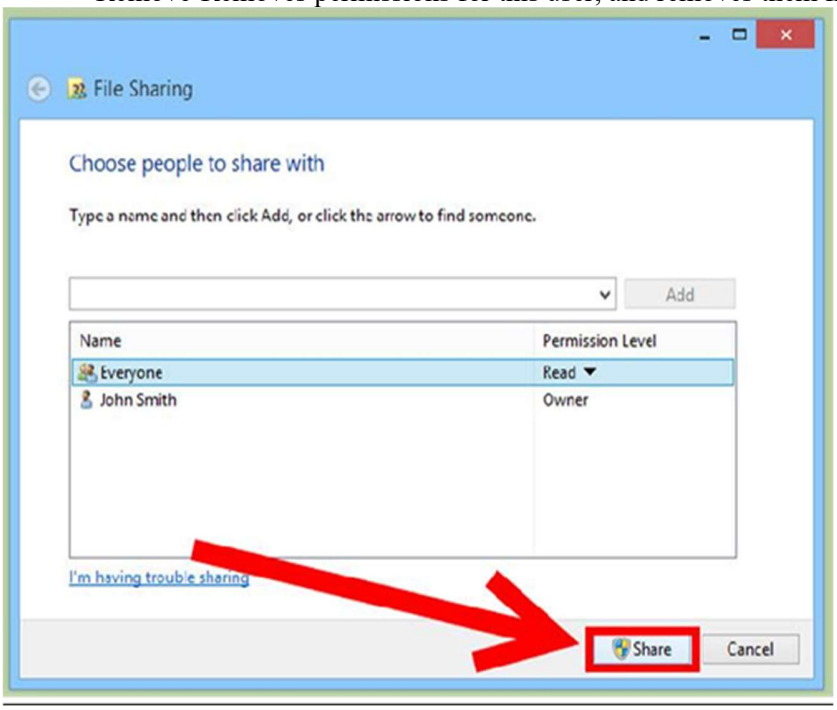
Click the"Specificpeople"option to select which users you want to sharewith. This will open a new window with a list of all the users that currently have access to the folder. You can add users to this list and give them specific permissions for the folder.

- To share the folder with everyone, click the dropdown menu at the top and select "Everyone". Click the Add button.
- To share with specific users,click the dropdown menu and select the mortype in the name and click Add.



Set permissions for users on the list. Find a user on the list that you want to change the permissions for. Look in the Permissions Level column, and click the arrow next to the existing permission. Select the new one from the list.

- Read-Usercansee,copy,and open files from the folder,but cannot change files or add new ones.
- Read/Write-Besides Readabilities,users can change files and addnew files to the shared folder. Files can be deleted by users with Read/Write permissions.
- Remove-Removes permissions for this user, and removes them from the list.



VIVAQUESTIONS

1. Define system resource. List resources that can be shared in network?
2. Give the examples of physical and virtual resource.Define resource sharing and state its needs.
3. Give advantages and disadvantages of printer sharing and folder sharing.
4. How security is measure issue in resource sharing?
5. Which are different privileges associated w ith folder?

RESULT:

The experiment has been conducted successfully.

EXPERIMENT / PRACTICAL -3

Use Wireshark Packet sniffer software and capture TCP, UDP, IP, ARP, ICMP, Telnet, FTP packets with sample output

AIM:

Use Wireshark Packet sniffer software and captures TCP,UDP, IP, ARP, ICMP, Telnet, FTP packets with sample output

DESCRIPTION:

Wireshark is a powerful network protocol analyzer that can capture and display data packets traveling through a network. It supports various protocols and can be used to troubleshoot network issues, analyze network performance, and ensure security. Here's an overview of how Wireshark handles different types of packets:

1. **TCP (Transmission Control Protocol)**: Wireshark can capture TCP packets, which are used for reliable, ordered, and error-checked delivery of data between applications. TCP is commonly used for protocols such as HTTP, HTTPS, FTP, and Telnet.
2. **UDP (User Datagram Protocol)**: Wireshark captures UDP packets, which provide a connectionless communication model with a minimal protocol mechanism. UDP is used for time-sensitive transmissions such as DNS queries and streaming media.
3. **IP (Internet Protocol)**: Wireshark captures IP packets, which are used to deliver data across network boundaries. Both IPv4 and IPv6 packets can be analyzed.
4. **ARP (Address Resolution Protocol)**: Wireshark can capture ARP packets, which are used for mapping network addresses (IP) to physical addresses (MAC).
5. **ICMP (Internet Control Message Protocol)**: Wireshark captures ICMP packets, which are used for diagnostic and error-reporting purposes in network operations (e.g., ping).
6. **Telnet**: Wireshark can capture Telnet packets, which provide bidirectional interactive text-oriented communication using a virtual terminal connection over TCP.
7. **FTP (File Transfer Protocol)**: Wireshark captures FTP packets, which are used for the transfer of files between a client and server on a network.

With Wireshark, users can inspect individual packets, follow protocol streams, and apply various filters to narrow down the data for more precise analysis. This makes it an invaluable tool for network administrators, cybersecurity professionals, and developers.

PROCEDURE:

Wireshark is a powerful tool used for network packet analysis. It allows you to capture and interactively browse the traffic running on a computer network. Below are the steps to capture various types of packets (TCP, UDP, IP, ARP, ICMP, Telnet, FTP) using Wireshark, along with sample outputs.

Steps to Capture Packets Using Wireshark

1. **Install Wireshark**:
   - Download and install Wireshark from the official website.
2. **Start Wireshark**:
   - Open Wireshark on your machine.
   - Select the network interface you want to capture traffic on (e.g., Ethernet, Wi-Fi).
3. **Begin Capture**:
   - Click on the interface to start capturing packets.
   - You can stop the capture by clicking on the red square button.

4. **Apply Display Filters**:
   - Use Wireshark's display filters to focus on specific types of traffic.

CAPTURING SPECIFIC PACKET TYPES

1. TCP Packets

- **Filter:** `tcp`
- **Sample Output:**

```
plaintext
Copy code
No.     Time        Source                Destination           Protocol Length Info
1       0.000000    192.168.1.2           192.168.1.1           TCP      66     55914 →
80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=12345678 TSecr=0 WS=128
```

2. UDP Packets

- **Filter:** `udp`
- **Sample Output**:

```
plaintext
Copy code
No.     Time        Source                Destination           Protocol Length Info
2       0.001234    192.168.1.2           192.168.1.3           UDP      58     50624 →
53 Len=16
```

3. IP Packets

- **Filter:** `ip`
- **Sample Output:**

```
plaintext
Copy code
No.     Time        Source                Destination           Protocol Length Info
3       0.002345    192.168.1.2           192.168.1.4           IP       98
192.168.1.2 → 192.168.1.4 [SYN] Seq=0 Win=64240 Len=0
```

4. ARP Packets

- **Filter:** `arp`
- **Sample Output:**

```
plaintext
Copy code
No.     Time        Source                Destination           Protocol Length Info
4       0.003456    192.168.1.1           192.168.1.2           ARP      42     Who has
192.168.1.2? Tell 192.168.1.1
```

5. ICMP Packets

- **Filter: i**`cmp`
- **Sample Output:**

```
plaintext
Copy code
No.     Time        Source                Destination           Protocol Length Info
5       0.004567    192.168.1.2           192.168.1.1           ICMP     74     Echo
(ping) request  id=0x1234, seq=1/256, ttl=64
```

6. Telnet Packets

- **Filter:** `telnet`
- **Sample Output:**

```
plaintext
Copy code
No.     Time        Source                Destination           Protocol Length Info
6       0.005678    192.168.1.2           192.168.1.5           TELNET   74     Telnet
Data
```

7. FTP Packets

- **Filter:** `ftp`
- **Sample Output:**

```
plaintext
Copy code
No.     Time        Source                Destination           Protocol Length Info
7       0.006789    192.168.1.2           192.168.1.6           FTP      78     Request:
USER anonymous
```

Additional Tips

- **Filtering**: You can combine filters using logical operators, e.g., `tcp || udp` to show both TCP and UDP packets.

- **Analysis**: Use the "Statistics" menu for in-depth analysis and graphical representations of the traffic.

RESULT:
The experiment has been conducted successfully.

EXPERIMENT / PRACTICAL -4

Implement Write and analyze the output of various Network commands such as ping, ipconfig, arp, netstat, tracert, nslookup, hostname, systeminfo etc., with sample outputs

AIM:
Implement Write and analyze the output of various Network commands such as ping, ipconfig, arp, netstat, tracert, nslookup, hostname, systeminfo etc., with sample outputs

DESCRIPTION:
Network commands are instructions used to configure, manage, and troubleshoot network devices and connections. These commands can be issued through command-line interfaces (CLI) on network devices such as routers, switches, firewalls, and computers. Here's a detailed description of some commonly used network commands:

Basic Network Commands

1. **ping**:
   - **Purpose**: Tests connectivity between two devices on a network.
   - **Usage**: `ping [hostname/IP address]`
   - **Example**: `ping 192.168.1.1`
2. **traceroute/tracert**:
   - **Purpose**: Traces the path packets take from one device to another.
   - **Usage**:
     - Unix/Linux: `traceroute [hostname/IP address]`
     - Windows: `tracert [hostname/IP address]`
   - **Example**: `traceroute google.com` or `tracert google.com`
3. **ipconfig/ifconfig**:
   - **Purpose**: Displays network configuration details.
   - **Usage**:
     - Windows: `ipconfig`
     - Unix/Linux: `ifconfig`
   - **Example**: `ipconfig /all` or `ifconfig eth0`
4. **netstat**:

- **Purpose**: Displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- **Usage**: `netstat [options]`
- **Example**: `netstat -a`

5. **nslookup**:
   - **Purpose**: Queries the Domain Name System (DNS) to obtain domain name or IP address mapping.
   - **Usage**: `nslookup [hostname/IP address]`
   - **Example**: `nslookup example.com`
6. **hostname**:
   - **Purpose**: Displays or sets the system's hostname.
   - **Usage**: `hostname [new-hostname]`
   - **Example**: `hostname`

Network Configuration Commands

1. **route**:
   - **Purpose**: Displays or modifies the IP routing table.
   - **Usage**: `route [add/delete] [destination] [gateway]`
   - **Example**: `route add 192.168.1.0 mask 255.255.255.0 192.168.1.1`
2. **arp**:
   - **Purpose**: Displays and modifies the ARP (Address Resolution Protocol) cache.
   - **Usage**: `arp [options] [hostname/IP address]`
   - **Example**: `arp -a`
3. **iptables**:
   - **Purpose**: Configures the IP packet filter rules of the Linux kernel firewall.
   - **Usage**: `iptables [options]`
   - **Example**: `iptables -L`
4. **ifup/ifdown**:
   - **Purpose**: Activates or deactivates network interfaces on Unix/Linux systems.
   - **Usage**: `ifup [interface]` or `ifdown [interface]`
   - **Example**: `ifup eth0` or `ifdown eth0`

Advanced Network Commands

1. **tcpdump**:
   - **Purpose**: Captures and displays packets on a network.
   - **Usage**: `tcpdump [options]`
   - **Example**: `tcpdump -i eth0`
2. **nmap**:
   - **Purpose**: Scans networks to discover hosts and services.
   - **Usage**: `nmap [options] [hostname/IP address]`
   - **Example**: `nmap -sP 192.168.1.0/24`
3. **wireshark**:
   - **Purpose**: Network protocol analyzer for capturing and analyzing traffic.
   - **Usage**: Typically used with a graphical interface, but can be run with command-line options.
   - **Example**: `wireshark` (GUI) or `tshark -i eth0` (CLI)

Windows-Specific Commands

1. **netsh**:
   - **Purpose**: Configures and displays the status of network components.

- **Usage**: `netsh [context] [command]`
- **Example**: `netsh interface ip show config`

2. **Get-NetIPAddress**:
   - **Purpose**: Displays IP address configuration on Windows PowerShell.
   - **Usage**: `Get-NetIPAddress`
   - **Example**: `Get-NetIPAddress`

Unix/Linux-Specific Commands

1. **dig**:
   - **Purpose**: Queries DNS name servers for information.
   - **Usage**: `dig [hostname/IP address]`
   - **Example**: `dig example.com`

2. **ip**:
   - **Purpose**: Manages IP addresses, network devices, and routing tables.
   - **Usage**: `ip [command]`
   - **Example**: `ipaddr show`

These commands are fundamental tools for network administrators and IT professionals, enabling them to manage and troubleshoot network infrastructure efficiently.

PROCEDURE:

1. `ping`

Description: The `ping` command is used to check the network connectivity between the host computer and a target (usually another computer or a server).

Command:

```
bash
Copy code
ping google.com
```

Sample Output:

```
plaintext
Copy code
Pinging google.com [142.250.64.78] with 32 bytes of data:
Reply from 142.250.64.78: bytes=32 time=14ms TTL=54
Reply from 142.250.64.78: bytes=32 time=13ms TTL=54
Reply from 142.250.64.78: bytes=32 time=15ms TTL=54
Reply from 142.250.64.78: bytes=32 time=12ms TTL=54

Ping statistics for 142.250.64.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 12ms, Maximum = 15ms, Average = 13ms
```

Analysis:

- The target (`google.com`) resolves to the IP address `142.250.64.78`.

- All packets sent were received (0% packet loss).
- The round-trip time (latency) varies between 12ms and 15ms, indicating a stable and responsive connection.

2. `ipconfig`

Description: The `ipconfig` command displays the network configuration details of the local machine.

Command:

```
bash
Copy code
ipconfig
```

Sample Output:

```
plaintext
Copy code
Windows IP Configuration

Ethernet adapter Ethernet:
   Connection-specific DNS Suffix  . : example.local
   Link-local IPv6 Address . . . . . : fe80::1c9e:5fbc:b001:6833%12
   IPv4 Address. . . . . . . . . . . : 192.168.1.10
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

Analysis:

- The computer has both an IPv4 address (`192.168.1.10`) and a link-local IPv6 address.
- The subnet mask is `255.255.255.0`, indicating a standard class C network.
- The default gateway is `192.168.1.1`, typically the router's address.

3. `arp`

Description: The `arp` command is used to view and manipulate the ARP (Address Resolution Protocol) cache, which maps IP addresses to MAC addresses.

Command:

```
bash
Copy code
arp -a
```

Sample Output:

```
plaintext
Copy code
Interface: 192.168.1.10 --- 0x12
  Internet Address      Physical Address      Type
  192.168.1.1           00-14-22-01-23-45     dynamic
  192.168.1.15          00-25-96-FF-EE-77     dynamic
```

Analysis:

- The ARP cache shows that the IP address `192.168.1.1` maps to the MAC address `00-14-22-01-23-45`.
- Another device on the network with IP `192.168.1.15` maps to the MAC address `00-25-96-FF-EE-77`.

4. `netstat`

Description: The `netstat` command displays network statistics, including current connections, routing tables, and interface statistics.

Command:

```
bash
Copy code
netstat
```

Sample Output:

```
plaintext
Copy code
Active Connections

Proto  Local Address          Foreign Address        State
  TCP    192.168.1.10:55478     ec2-52-87-185-95:443   ESTABLISHED
  TCP    192.168.1.10:55479     216.58.206.14:443      TIME_WAIT
  TCP    192.168.1.10:55480     172.217.10.46:443      ESTABLISHED
```

Analysis:

- The computer has active TCP connections to various remote addresses.
- The state `ESTABLISHED` indicates ongoing communication, while `TIME_WAIT` shows the connection is closing.

5. `tracert`

Description: The `tracert` command traces the path packets take to reach a network host.

Command:

```
bash
Copy code
tracert google.com
```

Sample Output:

```
plaintext
Copy code
Tracing route to google.com [142.250.64.78] over a maximum of 30 hops:
```

```
  1     1 ms<1 ms<1 ms  192.168.1.1
  2    12 ms    11 ms    12 ms  10.0.0.1
  3    13 ms    12 ms    11 ms  172.16.0.1
  4    14 ms    13 ms    12 ms  172.217.10.46
  5    15 ms    14 ms    13 ms  google.com [142.250.64.78]
```

Analysis:

- The command shows the route and time taken to reach each hop.
- The final destination (`google.com`) is reached in 5 hops.

6. `nslookup`

Description: The `nslookup` command queries DNS servers to obtain domain name or IP address mapping.

Command:

```
bash
Copy code
nslookup google.com
```

Sample Output:

```
plaintext
Copy code
Server:  resolver1.opendns.com
Address:  208.67.222.222

Non-authoritative answer:
Name:    google.com
Addresses:  142.250.64.78
```

Analysis:

- The DNS server used for the query is `208.67.222.222`.
- The resolved IP address for `google.com` is `142.250.64.78`.

7. `hostname`

Description: The `hostname` command displays the name of the current host (computer).

Command:

```
bash
Copy code
hostname
```

Sample Output:

```
plaintext
Copy code
MyComputer
```

Analysis:

- The name of the host machine is `MyComputer`.

8. `systeminfo`

Description: The `systeminfo` command provides detailed information about the computer's system configuration.

Command:

```
bash
Copy code
systeminfo
```

SAMPLE OUTPUT:

```
plaintext
Copy code
Host Name:                 MyComputer
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.19041 N/A Build 19041
OS Manufacturer:           Microsoft Corporation
System Manufacturer:       Dell Inc.
System Model:              XPS 15 9570
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 158 Stepping 10
GenuineIntel ~2200 Mhz
BIOS Version:              Dell Inc. 1.12.0, 1/22/2021
```

ANALYSIS:

- The computer's hostname is `MyComputer`.
- It is running Windows 10 Pro, version 19041.
- The system is a Dell XPS 15 9570 with an Intel processor.

These commands provide a comprehensive view of network and system information, useful for diagnostics and troubleshooting.

RESULT:

The experiment has been conducted successfully.

EXPERIMENT / PRACTICAL -5

Installation set up of Network simulator software (NS2 / NS3 / NetSim / OPNET /QualNet/OMNet++/J-Sim and Cisco Packet Tracer). Installation Procedure for Ns2 on Windows10/11

AIM:

Installation set up of Network simulator software (NS2/NS3/NetSim/OPNET/QualNet/OMNet++/J-Sim and Cisco Packet Tracer). Installation Procedure for Ns2onWindows10/11

DESCRIPTION:

NS2, or Network Simulator 2, is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks. Here are some key features and aspects of NS2:

1. **Discrete Event Simulation**: NS2 uses discrete event simulation techniques to model the operation of networks. Events are processed in a sequential manner, which helps in studying the behavior of the network under different conditions.
2. **Support for Various Protocols**: It supports a wide range of network protocols, including TCP, UDP, and routing algorithms. This allows researchers to simulate complex network behaviors and evaluate the performance of different protocols.

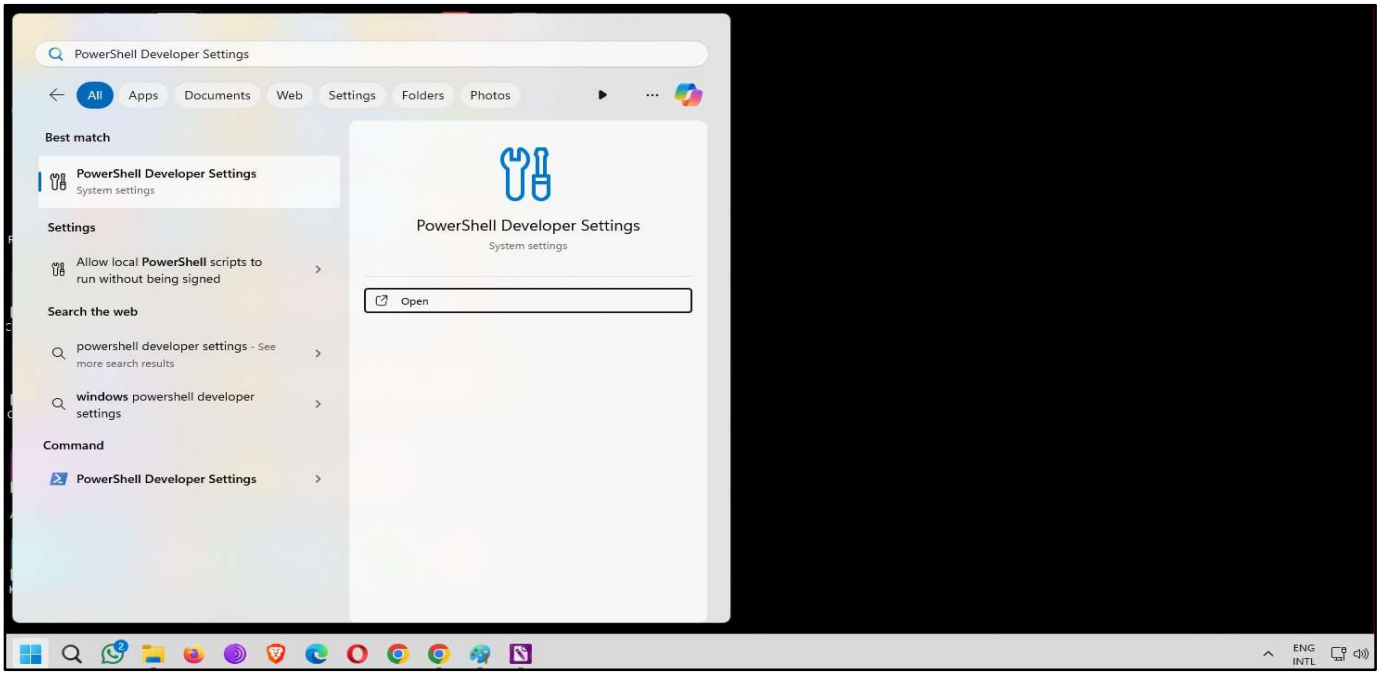
3. **Wired and Wireless Network Simulations**: NS2 can simulate both wired and wireless network scenarios. It includes models for various types of networks, such as LANs, WANs, ad-hoc networks, and sensor networks.
4. **Extensibility**: The simulator is highly extensible, allowing users to create and integrate new modules and protocols. This flexibility makes NS2 a valuable tool for researchers who need to test novel networking ideas.
5. **Visualization Tools**: NS2 comes with tools like NAM (Network Animator) for visualizing the simulation. These tools help users understand the behavior of the network and debug their simulations.
6. **Scripting Language**: NS2 simulations are typically defined using the OTcl scripting language, which is an object-oriented extension of Tcl. This allows for easy and flexible setup of simulation scenarios.
7. **Wide Usage and Community Support**: NS2 is widely used in the academic and research community. A large number of research papers and studies have been based on simulations conducted with NS2, and it has a strong user base that contributes to its ongoing development and support.
8. **Open Source**: NS2 is an open-source project, which means its source code is available for users to modify and extend according to their needs.

Overall, NS2 is a powerful and versatile tool for network simulation, making it a popular choice for researchers and students in the field of computer networking.

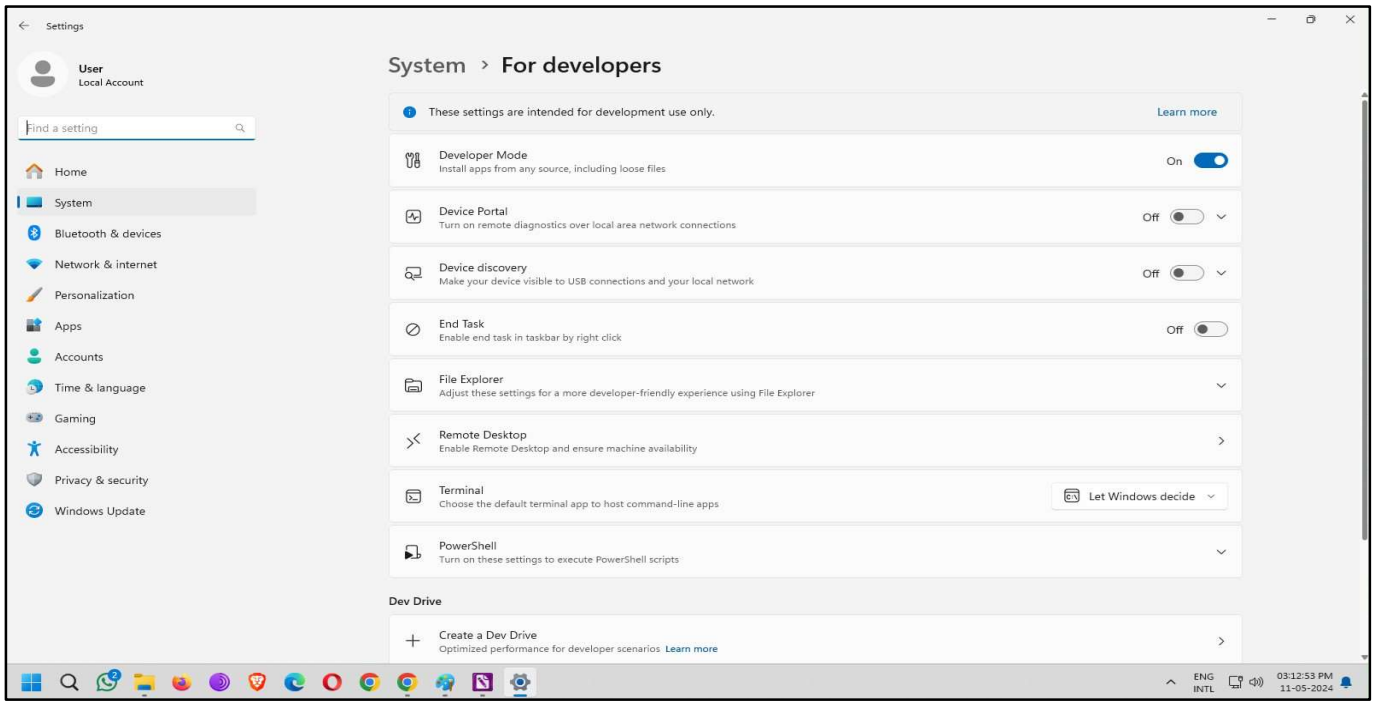
PROCEDURE

STEP 1:

Turnon Developer Mode Open search(Windows+S)and type“Power Shell Developer Settings”and click on Open as shown below



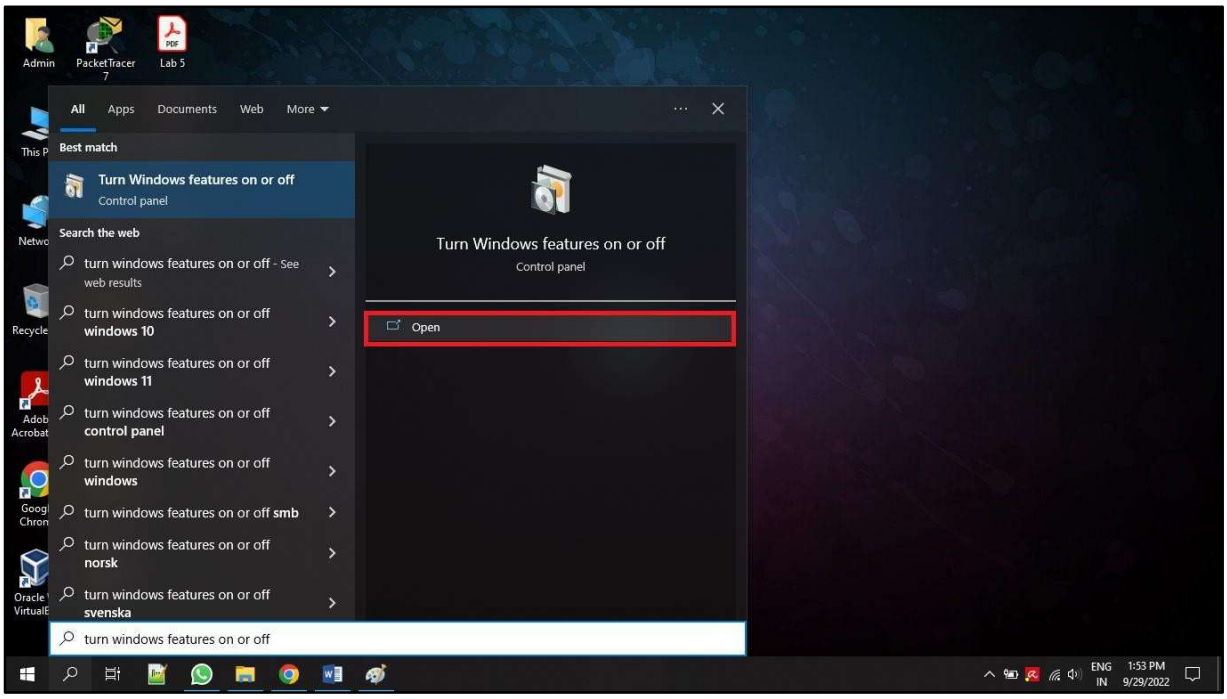
Turnon the option“Install apps from any source, including loose files”as shown below, click YES and close the window.


Settings
User
Local Account
Find a setting
Home
System
Bluetooth & devices
Network & internet
Personalization
Apps
Accounts
Time & language
Gaming
Accessibility
Privacy & security
Windows Update
System › For developers
These settings are intended for development use only.
Learn more
Developer Mode
Install apps from any source, including loose files
On
Device Portal
Turn on remote diagnostics over local area network connections
Off
Device discovery
Make your device visible to USB connections and your local network
Off
End Task
Enable end task in taskbar by right click
Off
File Explorer
Adjust these settings for a more developer-friendly experience using File Explorer
Remote Desktop
Enable Remote Desktop and ensure machine availability
Terminal
Choose the default terminal app to host command-line apps
Let Windows decide
PowerShell
Turn on these settings to execute PowerShell scripts
Dev Drive
Create a Dev Drive
Optimized performance for developer scenarios Learn more
ENG
INTL
03:12:53 PM
11-05-2024

STEP2

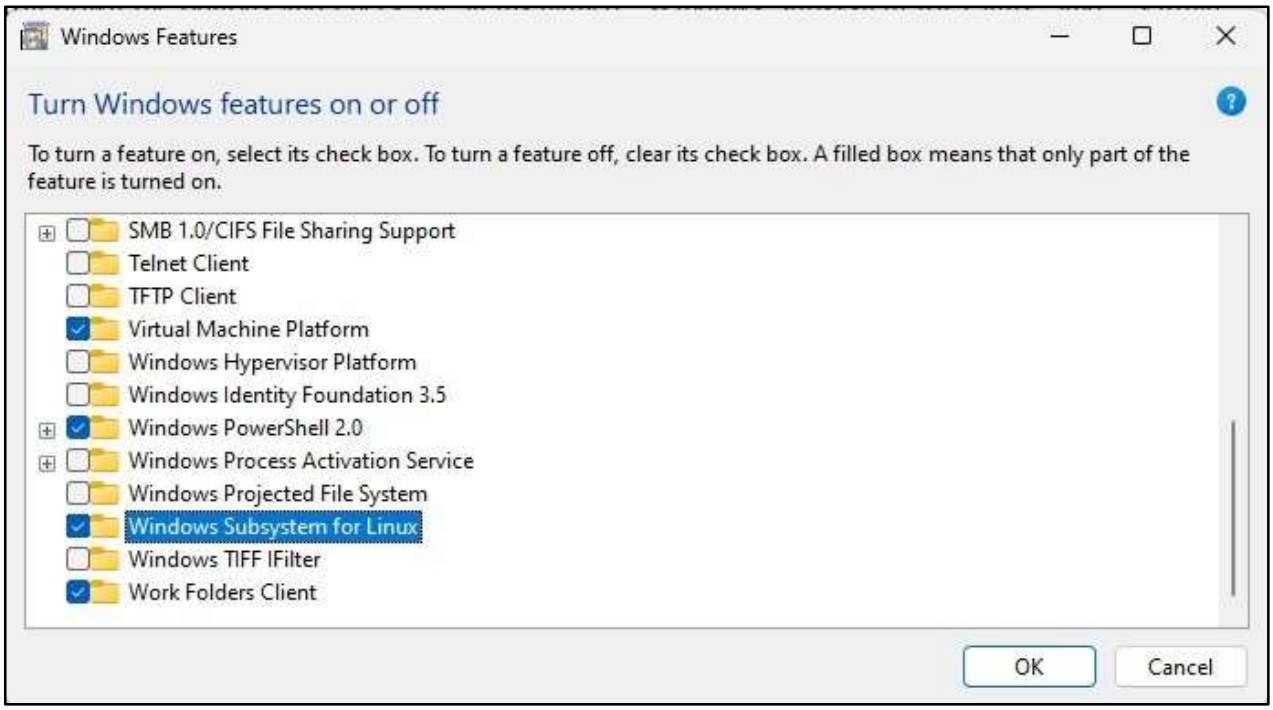
TurnonWSL(WindowsSubsystemforLinux)andVirtualMachinePlatform

Open search and type“turn windows features on or off”and clickon Open as shown below:



Scroll down the options and check the items named “Windows Subsystem for Linux” and “Virtual Machine Platform” and click on OK.

Restart the System.



STEP3

Open search and type“Windows Power Shell”,and run as administrator

Type the following commands inWindows Power Shell

wsl--install

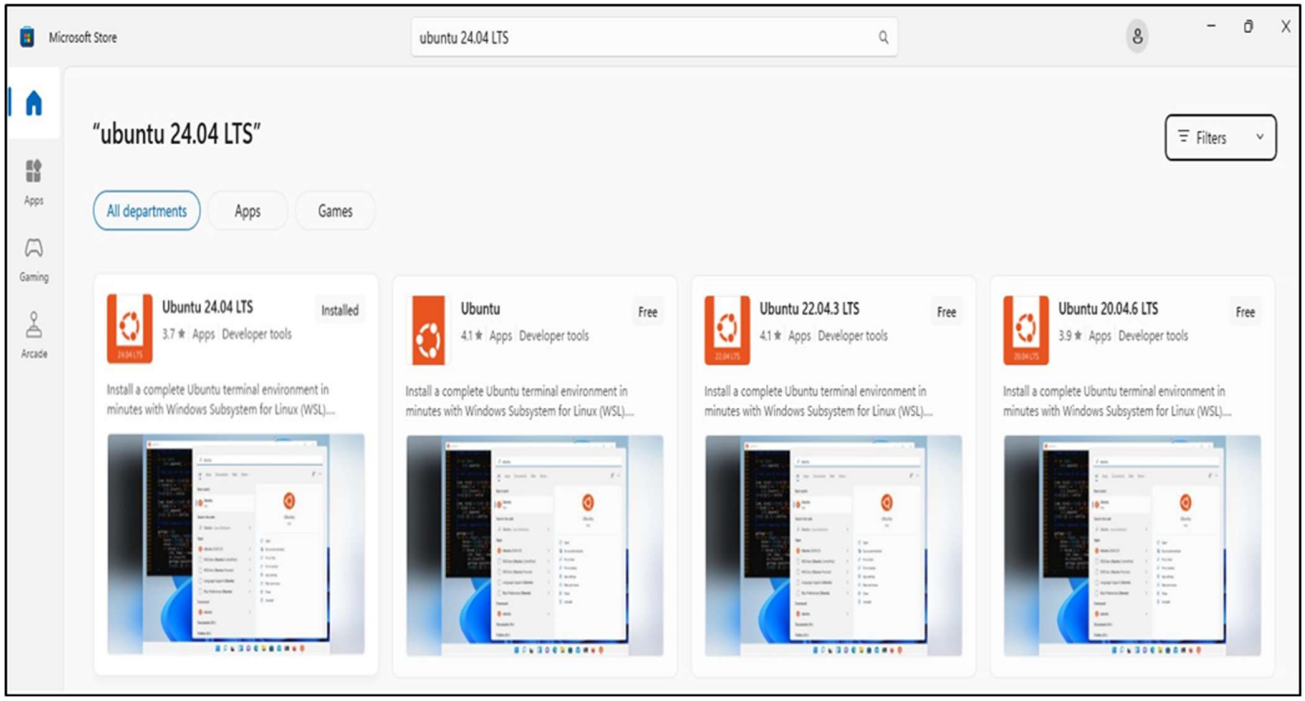
wsl--update

STEP4

Install Ubuntu from Microsoft Store.

Click on search and type“Microsoft Store”and clickon Open.

In the search bar at the top,type“Ubuntu24.04LTS”and clickon“Get”button as shown below:



It will take sometime to download and install the Ubuntu app as it is nearly400 MB.

STEP5

Open search and type“Ubuntu”and clickon”Runas administrator”.

As it is the first time you are opening Ubuntu, it will ask for User name and Password.

Give a username and password of your choice and hit enter key.

Now you should see a prompt as shown below:

Type the following commands one by one and hit enter after each command:

1. sudoapt update
2. sudoaptupgrade
3. sudoapt-getinstall ns2
4. sudoaptinstalltcl

Download nam_1.14_amd64.debornam_1.15-10-ubuntu14_amd64.debfilesfromtheinternet.

Please download nam_1.14_amd64.deb from the below website

https://drive.google.com/file/d/0B7S255p3kFXNNmtLeXhsaG5hXzQ/edit?resourcekey=0-NJmLwKSqlXCAjXSHr6ILiA

Pleasedownload nam_1.15-10-ubuntu14_amd64.debfromthebelowwebsite

https://drive.google.com/file/d/0B7S255p3kFXNdmxzSmRzaVRWb28/view?usp=sharing

Open terminal and navigate to the folder where nam_1.14_amd 64.debornam_1.15-10-ubuntu14_amd64.deb file is downloaded and run the following command.

$sudodpkg--installnam_1.14_amd64.deb

$sudodpkg--installnam_1.15-10-ubuntu14_amd64.deb (Recommendedtoinstallnam1.15)

STEP 6

Changing the location to working directory.
For this example, for working directory name is“NS2”and it is located on my“Desktop”.

The complete path to “NS2” directory on my PC is “C:\Users\User\Desktop\NS2Programs”.

To change the location to the above directory,type the following command and hit Enter key.

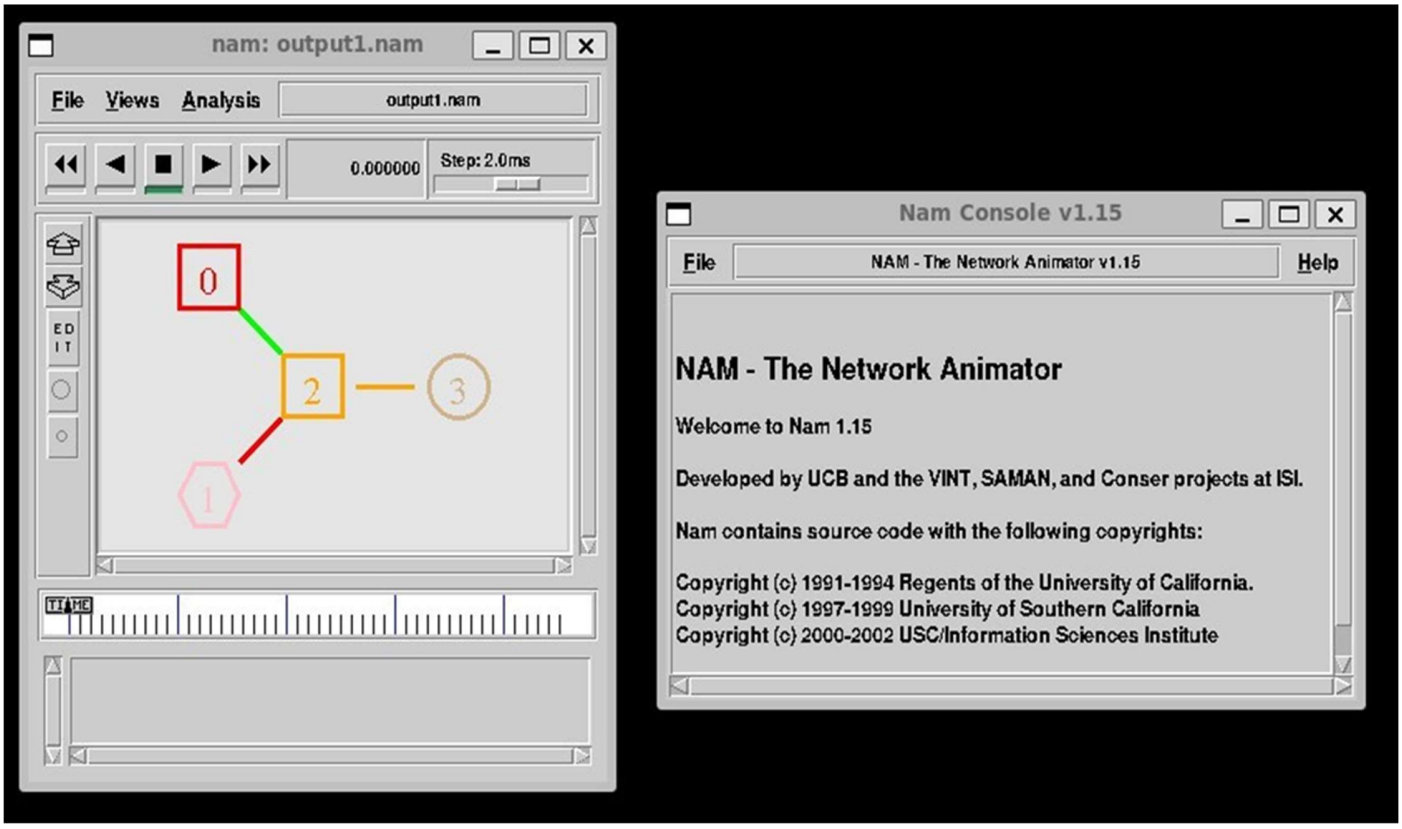
cd/mnt/c/Users/User/Desktop/’NS2Programs’

STEP 7

Running the program using“ns” command.

ns Program0-label.tcl

Now, you should be able to see the network animator (nam) window with the topology as shown in the below figure.



RESULT:

The experiment has been conducted successfully.

EXPERIMENT / PRACTICAL -6

Write a TCL script to simulate Star topology

AIM:

Write a TCL script to simulate Star topology

DESCRIPTION:

Star networks are one of the most common computer network topologies. In its simplest form, a star networkconsists of one central switch, hub or computer, which acts as a conduit to transmit messages. This consists of a central node, to which all other nodes are connected; this central node provides a common connection point for all nodes through a hub.In star topology, every node(computer workstation or anyother peripheral)is connected to a central node called a hub or switch. The switch is the server and the peripherals are the clients. Thus, the hub and leaf nodes, and the transmission lines between them, form a graph with the topology of a star. If the central node is passive, the originating node must be able to tolerate the reception of an echo of its own transmission, delayed by the two-way transmission time (i.e. to and from the central node) plus any delay generated in the central node. An active star network has an active central node that usually has the means to prevent echo-related problems.

The star topology reduces the damage caused by line failure by connecting all of the systems to a central node.When applied to a bus-based network, this central hub rebroadcasts all transmissions received from any peripheral node to all peripheral nodes on the network, sometimes including the originating node. All peripheral nodes may thus communicate with all others by transmitting to, and receiving from, the central node only. The failure of a transmission line linking any peripheral node to the central node will result in the isolation of that peripheral node from all others,but the rest of the systems will be unaffected.

HARD WARE REQUIREMENTS:

1. Computer with Configuration – CPU HDD capacity: 250 GB RAM: 4 GB, i3 processor

SOFTWARE REQUIREMENTS:

Windows 11 Operating Systems,Ubuntu 20.04 Linux, NS 2.35 Simulator

ALGORITHM:

1. Create a simulator object
2. Define different colors for different dataflows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on tracefile.
4. Create six nodes that forms a network numbered from 0 to 5
5. Create duplex links between the nodes to form a Star Topology
6. Setup TCP Connection between 1 and n3
7. Apply CBR Traffic over TCP
8. Schedule events and run the program.

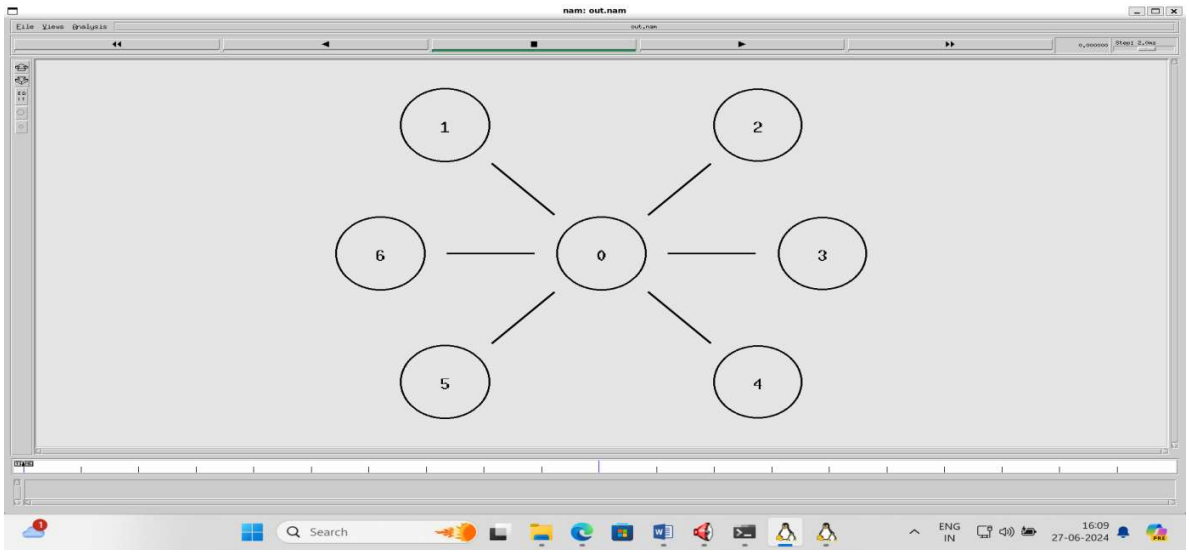
SOURCE CODE:

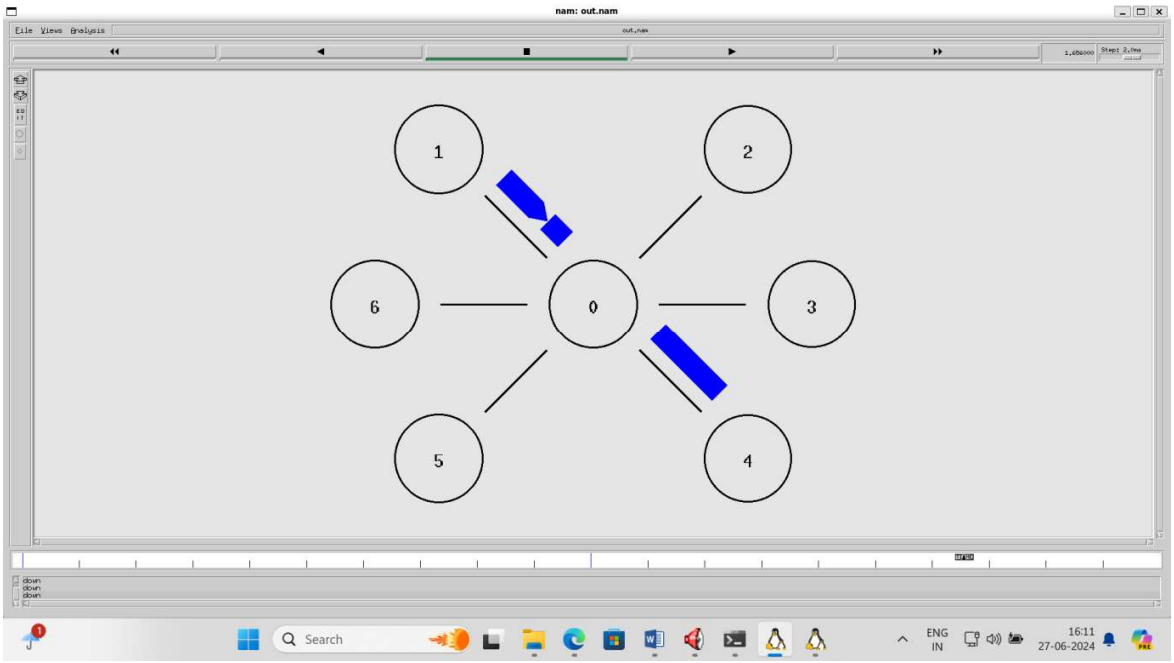
```
#This program will create a Star Topolgy using for loop in tcl in order to use less statements

set ns  [new Simulator]

$ns color 1 blue
```

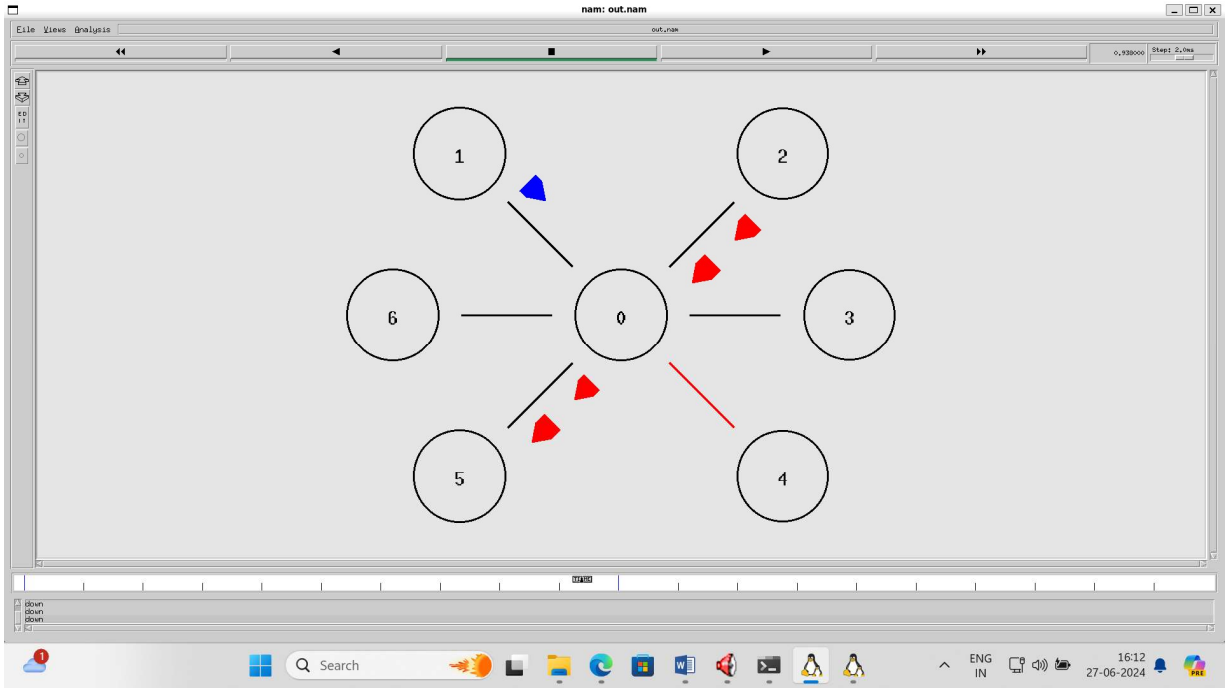
```
$ns color 2 red

$ns rtproto DV

setnf [open out.nam w]
$ns namtrace-all $nf

proc finish {} {
global ns nf
	$ns flush-trace
close $nf
execnamout.nam
exit 0
	}

#creating Nodes
for {set i 0} {$i<7} {incri} {
set n($i) [$ns node]
}

#Creating Links
for {set i 1} {$i<7} {incri} {
$ns duplex-link $n(0) $n($i) 512Kb 10ms SFQ
}

#Orienting The nodes
$ns duplex-link-op $n(0) $n(1) orient left-up
$ns duplex-link-op $n(0) $n(2) orient right-up
$ns duplex-link-op $n(0) $n(3) orient right
$ns duplex-link-op $n(0) $n(4) orient right-down
$ns duplex-link-op $n(0) $n(5) orient left-down
$ns duplex-link-op $n(0) $n(6) orient left

#TCP_Config
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n(1) $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink0

$ns connect $tcp0 $sink0

#UDP_Config
set udp0 [new Agent/UDP]
$udp0 set class_ 2
$ns attach-agent $n(2) $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0

$ns connect $udp0 $null0

#CBR Config
```

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set rate_ 256Kb
$cbr0 attach-agent $udp0

#FTP Config
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0


#Scheduling Events
$ns rtmodel-at 0.5 down $n(0) $n(5)
$ns rtmodel-at 0.9 up $n(0) $n(5)

$ns rtmodel-at 0.7 down $n(0) $n(4)
$ns rtmodel-at 1.2 up $n(0) $n(4)

$ns at 0.1 "$ftp0 start"
$ns at 1.5 "$ftp0 stop"

$ns at 0.2 "$cbr0 start"
$ns at 1.3 "$cbr0 stop"

$ns at 2.0 "finish"
$ns run
```

OUTPUT:




nam: out.nam
File Views Analysis
1
2
6
0
3
5
4
Search
ENG
IN
16:11
27-06-2024


nam: out.nam
File Views Analysis
1
2
6
0
3
5
4
Search
ENG
IN
16:12
27-06-2024

VIVA QUESTIONS:

1. What are the Different topologies available in networks?
2. Which topology requires multipoint connection?
3. Datacommunication system with in a campus is called as?
4. What is meant by WAN?
5. Explain the working of Ring topology?

RESULT:

The experiment has been conducted successfully.

EXPERIMENT / PRACTICAL -7

Simulation of Stop and Wait Protocol, Simulation of Sliding Window Protocol

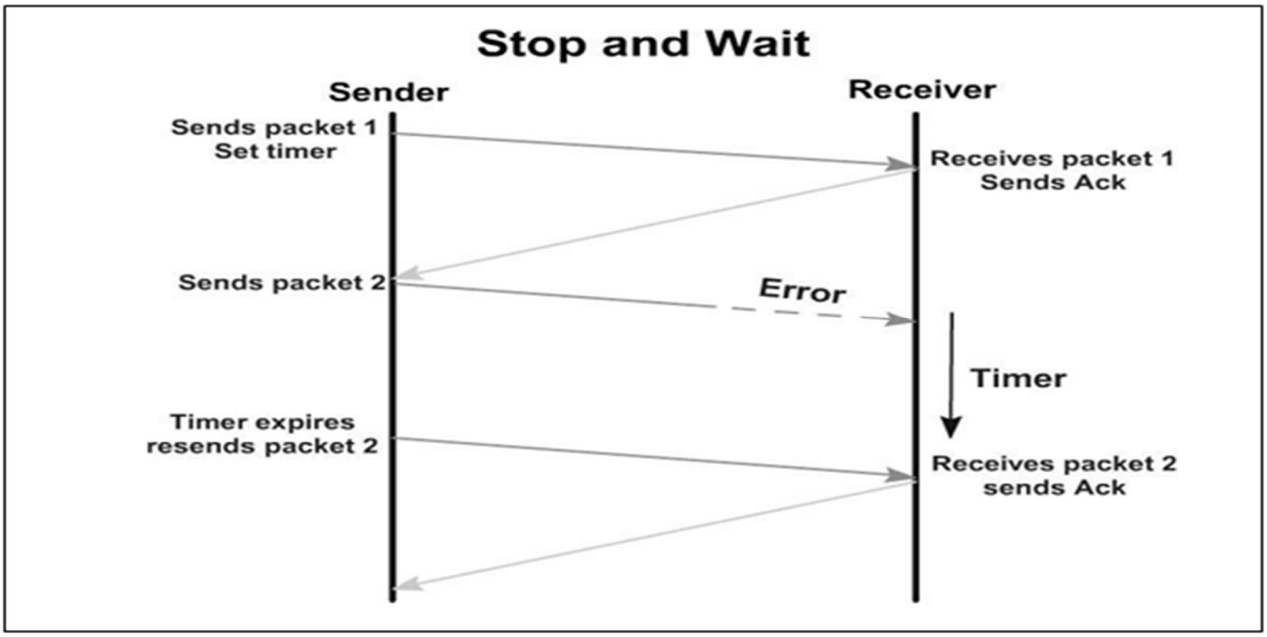
a) AIM :

Simulation of Stop and Wait Protocol

DESCRIPTION:

Stop and Wait is a reliable transmission flow control protocol. This protocol works only in Connection Oriented(Point to Point) Transmission. The Source node has window size of ONE. After transmission of a frame thetransmitting (Source) node waits for an Acknowledgement from the destination node. If the transmitted frame reaches the destination without error, the destination transmits a positive acknowledgement. If the transmittedframe reaches the Destination with error, the receiver destination does not transmit an acknowledgement. If thetransmitter receives a positive acknowledgement, it transmits the next frame if any. Else if its acknowledgementreceivetimerexpires, it retransmits the same frame.

1. Start with the window size of 1from the transmitting (Source)node
2. After transmission of a frame the transmitting(Source) node waits for a reply(Acknowledgement) from the receiving (Destination) node.
3. If the transmitted frame reaches the receiver(Destination) without error, the receiver(Destination) transmits a Positive Acknowledgement.
4. If the transmitted frame reaches the receiver(Destination)with error,the receiver(Destination) do not transmit acknowledgement.
5. If the transmitter receives a positive acknowledgement, it transmits the next frame if any. Else if the transmission timer expires, it retransmits the same frame again.
6. If the transmitted acknowledgment reaches the Transmitter(Destination) without error, the Transmitter (Destination) transmits the next frame if any.
7. If the transmitted frame reaches the Transmitter(Destination) with error, the Transmitter(Destination) transmits the same frame.
8. This concept of the Transmitting (Source) node waiting after transmission for a reply from the receiver is known as STOP and WAIT.



HARD WARE REQUIREMENTS:

1. Computer with Configuration – CPU HDD capacity: 250 GB RAM: 4 GB, i3 processor

SOFTWARE REQUIREMENTS:

Windows 11 Operating Systems, Ubuntu 20.04 Linux, NS 2.35 Simulator

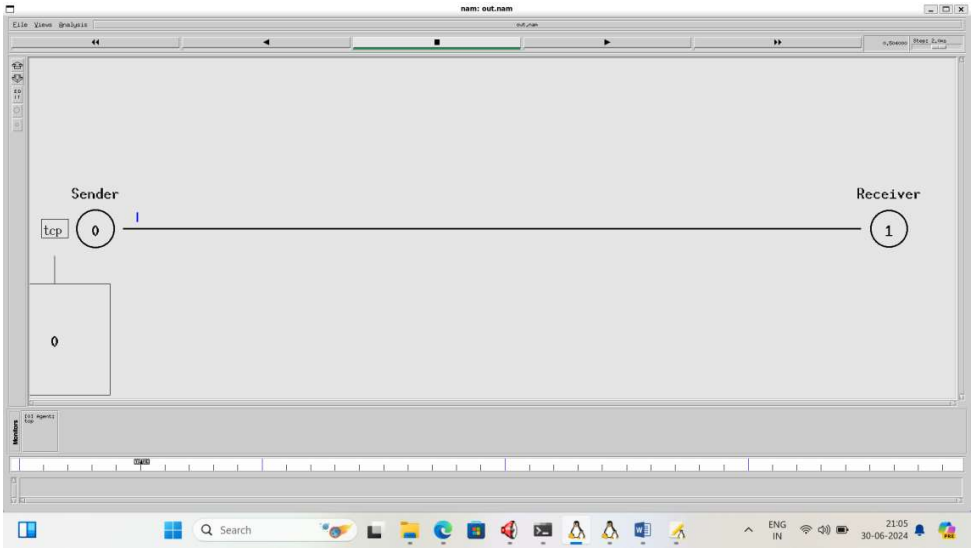
ALGORITHM:

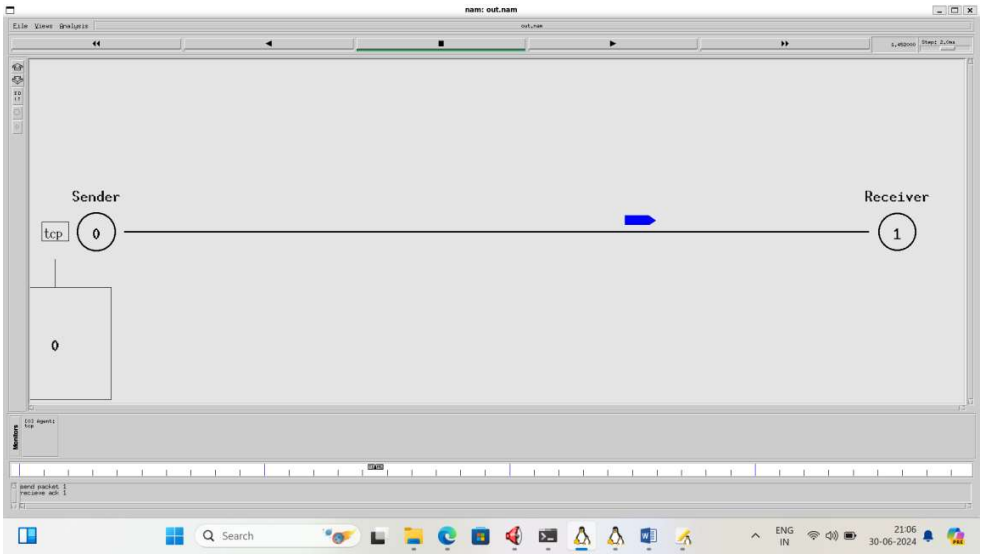
1. Create a simulator object
2. Define different colors for different dataflows
3. Open a nam trace file and define finish procedure then close the trace file,and execute nam on trace file.
4. Create two nodes that forms a network numbered 0 and 1
5. Create duplex links between the nodes n0 and n1
6. Setup TCP Connection between n0 and n1
7. Apply CBR Traffic over TCP
8. Schedule events and run the program.

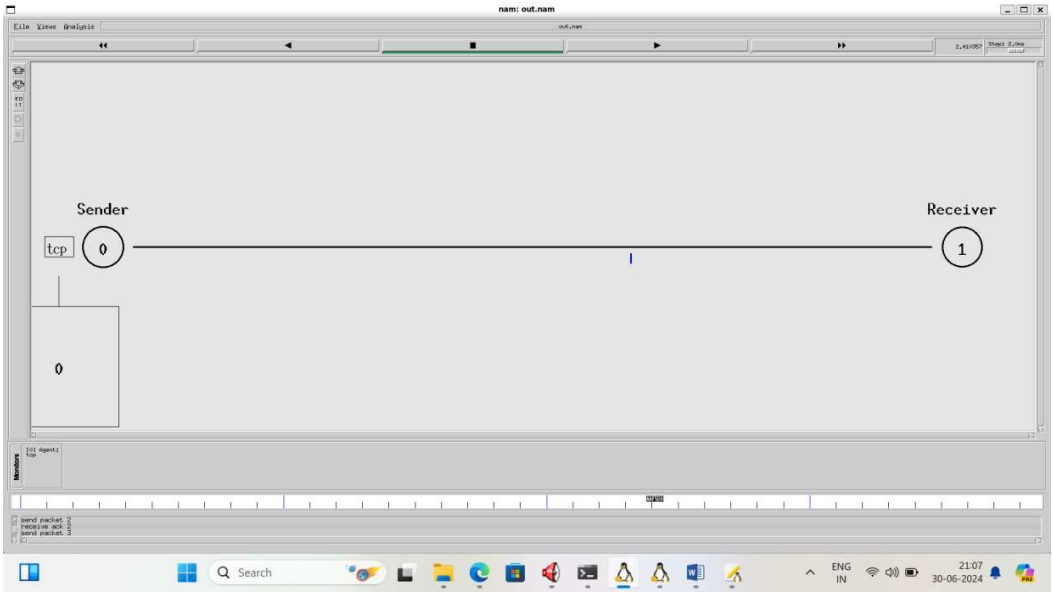
SOURCE CODE:

```
set ns [new Simulator]
$ns color 1 Blue
# set nam output file
setnf [open out.nam w]
$ns namtrace-all $nf

# destructor
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        execnamout.nam&
        exit 0
}

# create two new nodes and create labels for them
set n0 [$ns node]
set n1 [$ns node]
$ns at 0.0 "$n0 label \" Sender \" "
$ns at 0.0 "$n1 label \"Receiver\" "

# set up a new duplex link
$ns duplex-link $n0 $n1 1Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right

# create a new TCP agent
settcp [new Agent/TCP]
# attach the agent to first node
```

```
$ns attach-agent $n0 $tcp
$tcp set fid_ 1
$tcp set window_ 1
$tcp set maxcwnd_ 1
$ns add-agent-trace $tcptcp
$ns monitor-agent-trace $tcp
settcpsink [new Agent/TCPSink]
$ns attach-agent $n1 $tcpsink

$ns connect $tcp $tcpsink
set ftp [new Application/FTP]
$ftp attach-agent $tcp

$ns at 0.5 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $tcpsink "
$ns at 1.0 "$ns trace-annotate \"send packet 1\""
$ns at 1.4 "$ns trace-annotate \"recieveack 1\""
$ns at 2.0 "$ns trace-annotate \"send packet 2\""
$ns at 2.5 "$ns trace-annotate \"receive ack 2\""
$ns at 3.2 "$ns trace-annotate \"send packet 3\""
$ns at 3.5 "$ns trace-annotate \"receive ack 3\""
$ns at 3.8 "$ns trace-annotate \"send packet 4\""
$ns at 4.0 "finish"
$ns run
```







RESULT:

The experiment has been conducted successfully.

b) AIM :

Simulation of Sliding Window Protocol

DESCRIPTION:

A sliding window protocol is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the Data Link Layer (OSI model) as well as in the Transmission Control Protocol(TCP).

Conceptually, each portion of the transmission (packets in most data link layers, but bytes in TCP) is assigned a unique consecutive sequence number, and the receiver uses the numbers to place received packets in the correctorder, discarding duplicate packets and identifying missing ones. The problem with this is that there is no limit on the size of the sequence number that can be required.

By placing limits on the number of packets that can be transmitted or received at any given time, a sliding window protocol allows an unlimited number of packets to be communicated using fixed-size sequence numbers. The term "window" on the transmitter side represents the logical boundary of the total number of packets yet to be acknowledged by the receiver. The receiver informs the transmitter in each acknowledgment packet the current maximum receiver buffer size (window boundary). The TCP header uses a 16-bit field to report the receive window size to the sender. Therefore, the largest window that can be used is 216=64 kilo bytes. In slow-start mode, the transmitter starts with low packet count and increases the number of packets in each transmission after receiving acknowledgment packets from receiver. For every ack packet received, the window slides by one packet (logically) to transmit one new packet. When the window threshold is reached, thetransmitter sends one packet for one ack packet received. If the window limit is 10 packets then in slow start mode the transmitter may start transmitting one packet followed by two packets(before transmitting two packets,one packet ack has to be received), followed by three packets and so on until 10 packets. But after reaching 10 packets, further transmissions are restricted to one packet transmitted for one ack packet received. In a simulation this appears as if the window is moving by one packet distance for every ack packet received.
On the receiver side also, the window moves one packet for every packet received. The sliding window method ensures that traffic congestion on the network is avoided. The application layer will still be offering data for transmission to TCP without worrying about the network traffic congestion issues as the TCP on sender and receiver side implement sliding windows of packet buffer.The window size may vary dynamically depending on network traffic.

For the highest possible throughput, it is important that the transmitter is not forced to stop sending by the sliding window protocol earlier than one round-trip delay time (RTT). The limit on the amount of data that it can send before stopping to wait for an acknowledgment should be larger than the bandwidth-delay product of the communications link. If it is not, the protocol will limit the effective bandwidth of the link.

HARD WARE REQUIREMENTS:

1. Computer with Configuration – CPU HDD capacity: 250 GB RAM: 4 GB, i3 processor

SOFTWARE REQUIREMENTS:

Windows 11 Operating Systems, Ubuntu 20.04 Linux, NS 2.35 Simulator

ALGORITHM:

Sliding Window Protocol Algorithm

1. **Initialization:**
   - Define the window size `W`.
   - Initialize sender window: `S_window = 0`.
   - Initialize receiver window: `R_window = 0`.
   - Set a timer for each packet.
2. **Sending Data:**
   - The sender can send up to `W` packets without receiving an acknowledgment (ACK).
   - For each packet `i` (where `i` ranges from `S_window` to `S_window + W - 1`):
     - Send packet `i`.
     - Start a timer for packet `i`.
3. **Receiving Data:**
   - When the receiver receives a packet `i`:
     - If packet `i` is within the `R_window`:
       - Send an acknowledgment (ACK) for packet `i`.
       - If packet `i` is the expected packet, move the `R_window` forward.
     - If packet `i` is outside the `R_window`, ignore the packet.
4. **Acknowledgment Handling:**
   - When the sender receives an ACK for packet `i`:
     - Mark packet `i` as acknowledged.
     - Slide the sender window `S_window` forward to the next unacknowledged packet.
5. **Timeout Handling:**
   - If the timer for packet `i` expires before an ACK is received:
     - Retransmit packet `i`.
     - Restart the timer for packet `i`.

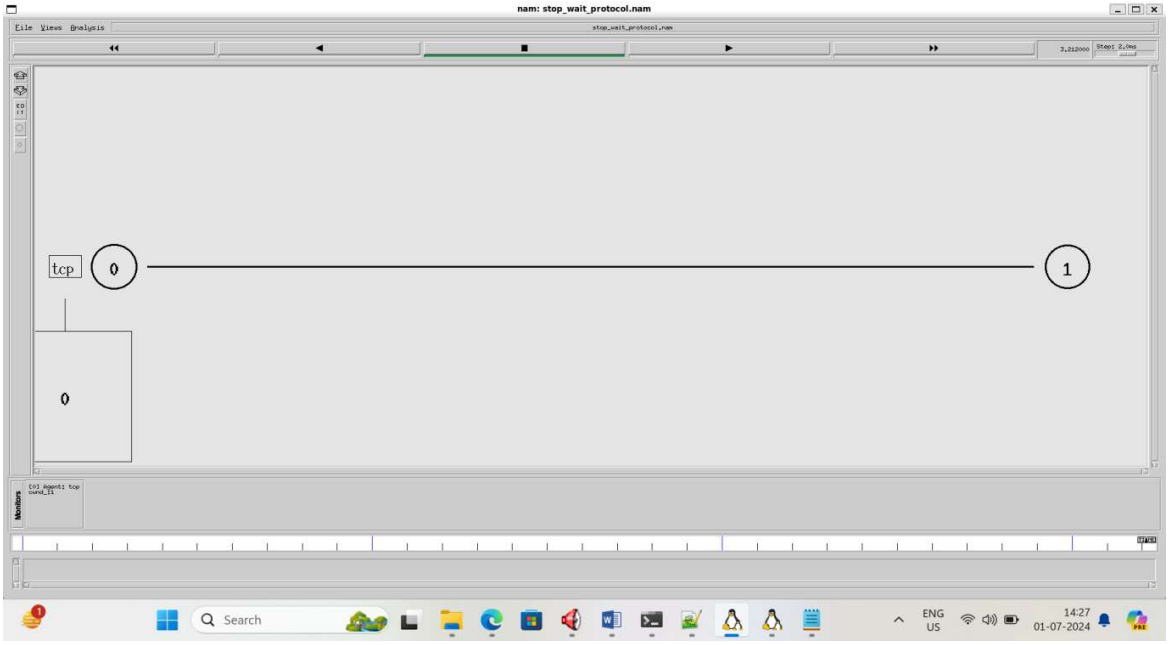
SOURCE CODE:

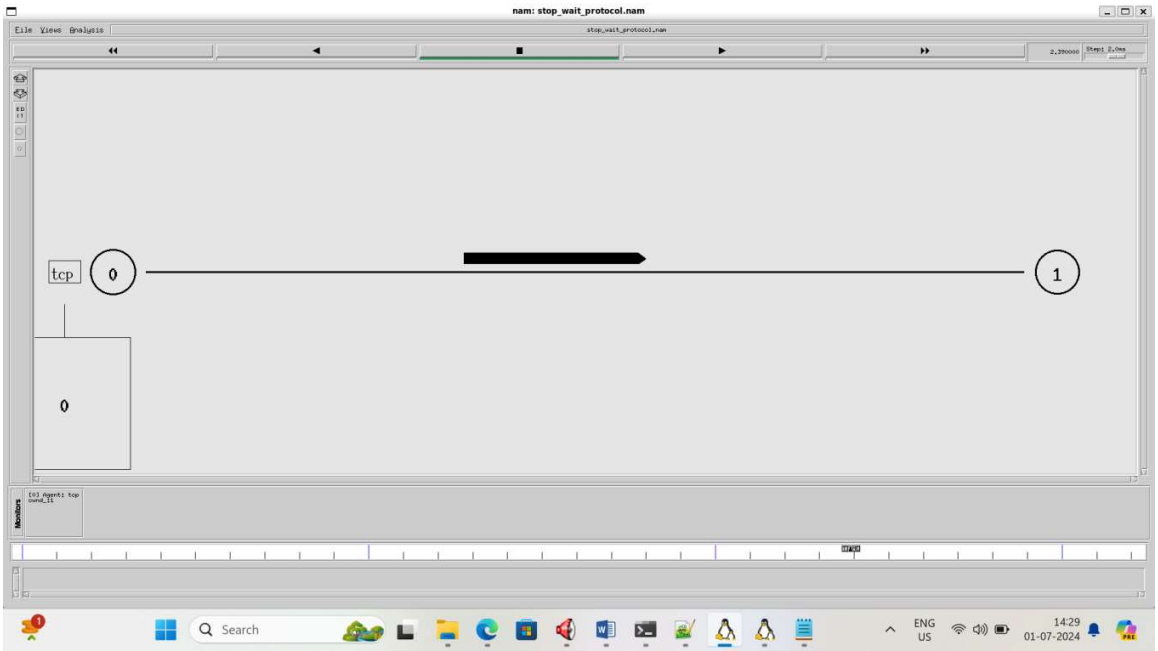
```
set ns [new Simulator]

#puts "Starting program..."

# Creating Output files
setnamf [open stop_wait_protocol.nam w]
$ns namtrace-all $namf

settrf [open stop_wait_protocol.tr w]
$ns trace-all $trf

# Finishing procedures
proc finish {} {
    global ns namftrf
    $ns flush-trace
    #puts "Saving namf and trf..."
```

```
    close $namf
    close $trf
    #puts "Opening Network animator..."
    execnamstop_wait_protocol.nam&
    #puts "Closing program..."
    exit 0
}

# Creating nodes
set n0 [$ns node]
set n1 [$ns node]

# Setting links and setting queue size
$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns queue-limit $n0 $n1 10

# Defining agents and setting src and dest
settcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n1 $sink
$ns connect $tcp $sink

# Creating FTP
set ftp [new Application/FTP]
$ftp attach-agent $tcp

# Stop and wait protocol
$tcp set window_ 1
$tcp set maxcwnd_ 1

# Use below 2 # for sliding window
#$tcp set windowInit_ 4
#$tcp set maxcwnd_ 4

# I dont know what these are.
$tcp set nam_tracevar_ true
$ns add-agent-trace $tcptcp
$ns monitor-agent-trace $tcp
$tcptracevarcwnd_

# Finishing and closing procedures
$ns at 0.1 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp"
```

$ns at 3.0 "$ns detach-agent $n1 $sink"
$ns at 3.5 "$ftp stop"
$ns at 5.0 "finish"
$ns run

OUTPUT:





VIVA QUESTIONS:

1. What is ARQ?
2. What is stop and wait protocol?
3. What is stop and wait ARQ?
4. What is usage of sequence number in reliable transmission?
5. Whatis sliding window?

RESULT:

The experiment has been conducted successfully.

EXPERIMENT / PRACTICAL -8

Simulation of the Routing algorithms (LSR/DVR)

AIM:

Simulation of Distance Vector/Link State Routing.

DESCRIPTION:

Distance Vector routing protocol

Routing is the process of selecting best paths in a network.In the past, the termrouting was also used to mean forwarding network traffic among networks. However this latter function is much better described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology. In packet switching networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such asrouters, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain are cord of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multi path routing techniques enable the use of multiple alternative paths. Incase of overlapping/equal routes, the following elements are considered in order to decide which routes get installed into the routing table(sorted by priority):

1. Prefix-Length: where longer subnet masks are preferred (independent of whether it is within a routing protocol or over different routing protocol)
2. Metric: where a lower metric/cost is preferred (only valid within one and the Same routing protocol)
3. Administrative distance: where a lower distance is preferred(only valid between different routing protocols)Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Structured addresses allow a single routing table entry to represent the route to a group of devices. In large networks, structured addressing(routing, in the narrow sense)out performs unstructured addressing(bridging). Routing has become the dominant form of addressing on the Internet. Bridging is still widely used within localized environments.

ALGORITHM:

There are several variants of flooding algorithm. Most work roughly as follows:

1. Each node acts as both a transmitter and a receiver.
2. Each node tries to forward every message to every one of its neighbours except the source node. This result sin every message eventually being delivered to all reachable parts of the network. Algorithms may need to be more complex than this, since, in some case,precautions have to be taken to avoid wasted duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system. A variant of flooding called selective flooding partially addresses these issues by only sending packets to routers in the same direction.In selective flooding the routers don't send every incoming packet on everyline but only on those lines which are going approximately in the right direction.

PROGRAM:

```
setns[newSimulator]

setnf[openout.namw]
$nsnamtrace-all$nf

settr[openout.trw]
$nstrace-all$tr

procfinish{}{
    globalnfnstr
    $nsflush-trace
    close$tr
    execnamout.nam&
    exit0
    }

set    n0    [$ns
    node]set  n1
    [$ns
    node]set  n2
    [$ns
    node]setn3[
    $nsnode]

$nsduplex-link $n0$n110Mb10msDropTail
$nsduplex-link $n1$n310Mb10msDropTail
$nsduplex-link $n2$n110Mb10msDropTail

$nsduplex-link-op$n0$n1orientright-down
```

$nsduplex-link-op $n1$n3orientright
$nsduplex-link-op$n2$n1 orientright-up

settcp[newAgent/TCP]
$nsattach-agent$n0$tcp

setftp[newApplication/FTP]
$ftpattach-agent$tcp

setsink[newAgent/TCPSink]
$nsattach-agent$n3$sink

setudp [newAgent/UDP]
$nsattach-agent$n2$udp

setcbr[newApplication/Traffic/CBR]
$cbrattach-agent$udp

setnull[newAgent/Null]
$nsattach-agent$n3$null

$nsconnect$tcp$sink
$nsconnect$udp$null

$nsrtmodel-at1.0down $n1$n3
$nsrtmodel-at2.0up$n1$n3

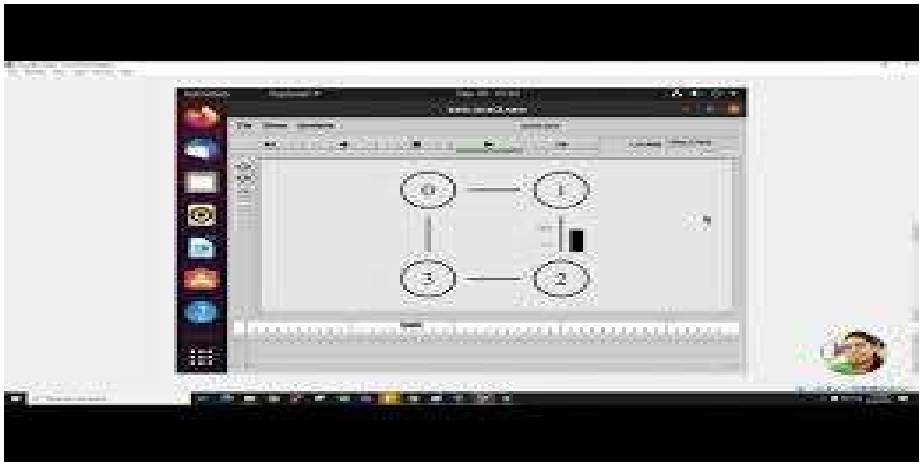
$nsrtprotoDV

$nsat 0.0 "$ftpstart"
$nsat 0.0 "$cbrstart"

$nsat 5.0"finish"

$nsrun

RESULT:



EXPERIMENT / PRACTICAL -9

Simulation of data transmission using TCP and evaluate the performance metric (Packet delivery ratio, Throughput, Jitter/Average end to end delay)

AIM:

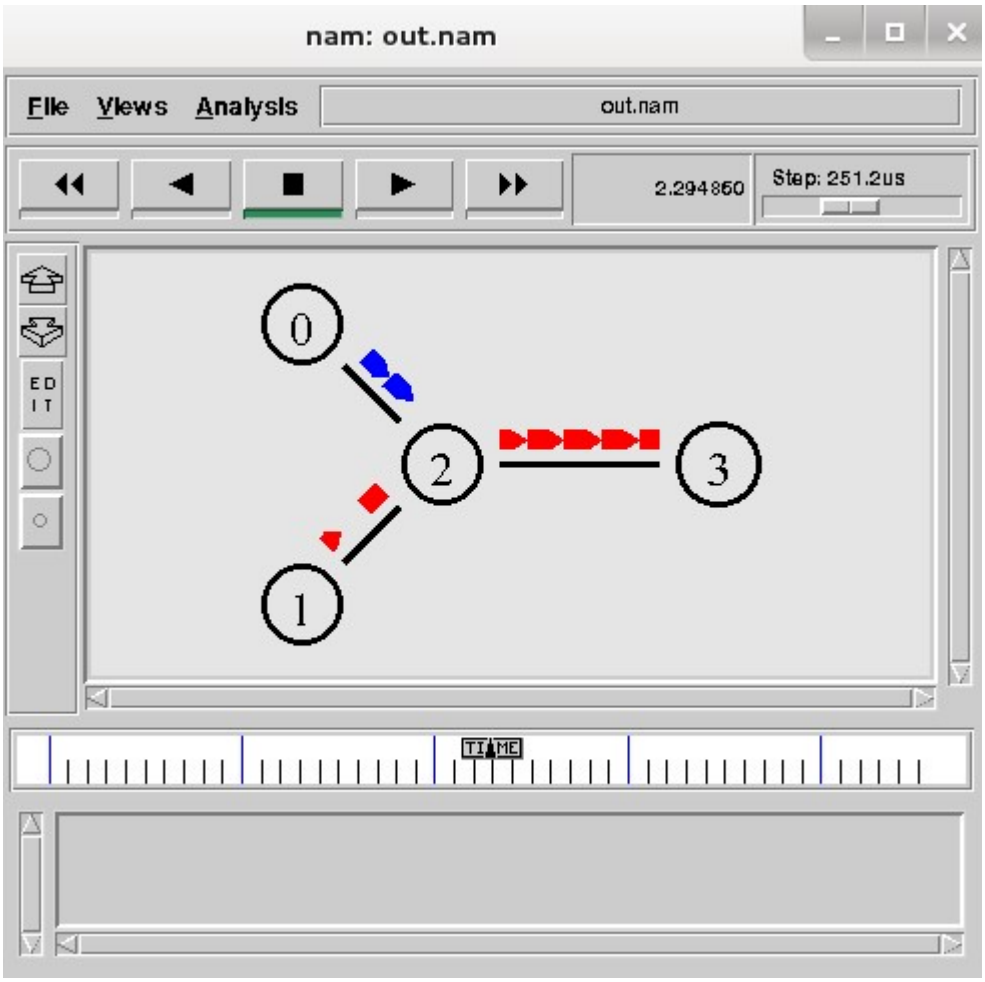
Simulation of data transmission using TCP and evaluate the performance metric (Packet delivery ratio, Throughput, Jitter/Average end to end delay)

CODE:

```
# Create a simulator object
set ns [new Simulator]

# Define different colors
# for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

# Open the NAM trace file
setnf [open out.nam w]
$ns namtrace-all $nf

# Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace

# Close the NAM trace file
close $nf

# Execute NAM on the trace file
execnamout.nam&
exit 0
}

# Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

# Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

# Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

```
# Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5


# Setup a TCP connection
settcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


/* # Setup a UDP connection
setudp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null] */

$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

# Setup a CBR over UDP connection
setcbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false


# Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

# Detachtcp and sink agents
# (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

# Call the finish procedure after
# 5 seconds of simulation time
$ns at 5.0 "finish"

# Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

# Run the simulation
$ns run
```

OUTPUT:


nam: out.nam
File
Views
Analysis
out.nam
2.294860
Step: 251.2us
0
1
2
3
TIME

EXPERIMENT / PRACTICAL -10

Simulation of data transmission using UDP and evaluate the performance metric (Packet delivery ratio, Throughput, Jitter/Average end to end delay)

AIM:

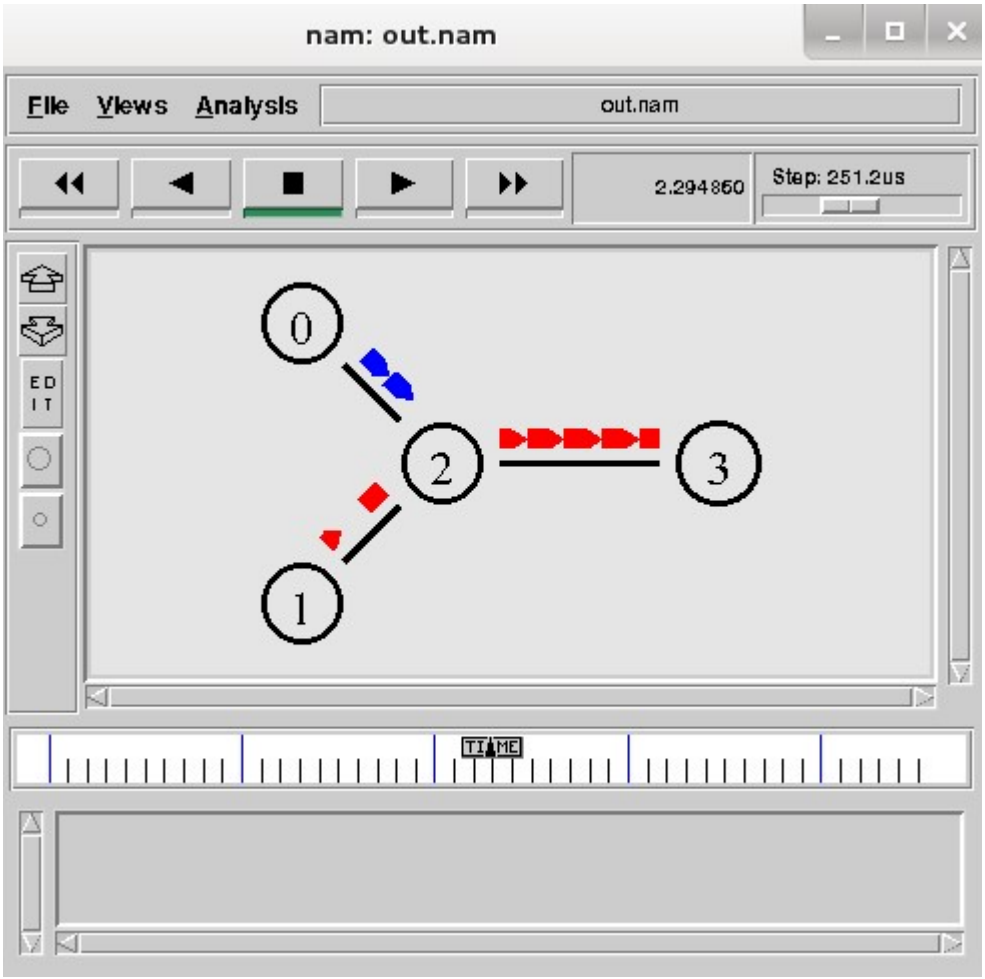
Simulation of data transmission using UDP and evaluate the performance metric (Packet delivery ratio, Throughput, Jitter/Average end to end delay)

CODE:

```
# Create a simulator object
set ns [new Simulator]

# Define different colors
# for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

# Open the NAM trace file
setnf [open out.nam w]
$ns namtrace-all $nf

# Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace

# Close the NAM trace file
close $nf

# Execute NAM on the trace file
execnamout.nam&
exit 0
}

# Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

# Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

# Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
```

```
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

# Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


# Setup a UDP connection
setudp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]

$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

# Setup a CBR over UDP connection
setcbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false


# Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

# Detachtcp and sink agents
# (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

# Call the finish procedure after
# 5 seconds of simulation time
$ns at 5.0 "finish"

# Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

# Run the simulation
$ns run
```

OUTPUT:


nam: out.nam
File
Views
Analysis
out.nam
2.294860
Step: 251.2us
0
1
2
3
TIME

EXPERIMENT / PRACTICAL -11

Implement Socket Programming.

AIM:
Implement Socket Programming.

DESCRIPTION:
TCP Client-Server Implementation in C

4th September 2021  Nikhil Tomar

In this tutorial, you will learn to implement a TCP client-server program in the C programming language. Here, the client and server would exchange messages and communicate with each other in an interactive way. In addition, you will also learn about the client-server architecture and other related concepts.

Table of Content

What is a Socket?

A socket is a structure that allows communication between different processes on the same computer or different computers. A socket allows us to communicate by sending and receiving data over the network. In simple terms, it is a way for a computer to talk to other computers on the network.

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

What is TCP?

TCP refers to the Transmission Control Protocol. It is one of the main protocol used for communication over the internet. Some of the features of this protocol are:

- It is a connection-oriented communication protocol.
- It used three-way handshake to establish reliable connections.
- TCP gurantees the delivery of the data packets.

Client-server Architecture

A client-server architecture is a model in computer networking, where the server provides some service to the client. In this architecture, the client computer sends a request to the server computer through a network (internet). The server accepts this request and sends the required data to the client.In simple terms, we can say

- Server is a remote computer which provides some services.
- Client is the one that request the server for these services.

Implementation

Here, we will start the implementation of the client-server implementation in C. First, we start by implementing the server-side and then we begin with the client-side code.

SERVER-SIDE CODE

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <unistd.h>
5   #include <arpa/inet.h>
6
7   int main(){
8
9     char *ip = "127.0.0.1";
10    int port = 5566;
11
12    intserver_sock, client_sock;
13    structsockaddr_inserver_addr, client_addr;
14    socklen_taddr_size;
15    char buffer[1024];
16    int n;
17
18    server_sock = socket(AF_INET, SOCK_STREAM, 0);
19    if (server_sock< 0){
20      perror("[-]Socket error");
21      exit(1);
22    }
23    printf("[+]TCP server socket created.\n");
24
25    memset(&server_addr, '\0', sizeof(server_addr));
26    server_addr.sin_family = AF_INET;
27    server_addr.sin_port = port;
28    server_addr.sin_addr.s_addr = inet_addr(ip);
29
30    n = bind(server_sock, (structsockaddr*)&server_addr, sizeof(server_addr));
31    if (n < 0){
32      perror("[-]Bind error");
33      exit(1);
34    }
35    printf("[+]Bind to the port number: %d\n", port);
36
37    listen(server_sock, 5);
38    printf("Listening...\n");
39
40    while(1){
41      addr_size = sizeof(client_addr);
42      client_sock = accept(server_sock, (structsockaddr*)&client_addr, &addr_size);
43      printf("[+]Client connected.\n");
44
45      bzero(buffer, 1024);
46      recv(client_sock, buffer, sizeof(buffer), 0);
47      printf("Client: %s\n", buffer);
48
```

```
49          bzero(buffer, 1024);
50          strcpy(buffer, "HI, THIS IS SERVER. HAVE A NICE DAY!!!");
51          printf("Server: %s\n", buffer);
52          send(client_sock, buffer, strlen(buffer), 0);
53
54          close(client_sock);
55          printf("[+]Client disconnected.\n\n");
56
57        }
58        return 0;
59      }
```

The server-side process can be broken down into the following steps:

1. socket() – create TCP socket.
2. bind() – bind the TCP socket to the server address (ip and port).
3. listen() – waiting for the clients.
4. accept() – connection is established between the client and server.
5. recv() and send () – communicate with each other.
6. close() – close the connection from the client.

Include the required header files.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <arpa/inet.h>
```

Define the IP (Internet Protocol) address and port number, that would be used to create a socket. Here, we are using the localhost address for both the server and the client.

```
1 char *ip = "127.0.0.1";
2 int port = 5566;
```

Here, we define the required variables used later in this program.

```
1 intserver_sock, client_sock;
2 structsockaddr_inserver_addr, client_addr;
3 socklen_taddr_size;
4 char buffer[1024];
5 int n;
```

Create a TCP (Transmission Control Protocol) socket that returns a socket descriptor. This socket descriptor would be used to communicate with the client.

```
1 server_sock = socket(AF_INET, SOCK_STREAM, 0);
2 if (server_sock< 0){
3   perror("[-]Socket error");
4   exit(1);
5 }
6 printf("[+]TCP server socket created.\n");
```

Here, we initialize the server address by providing the required IP and port number. The server keeps all the address information for both the server and the client in the sockaddr_in struct.

```
1 memset(&server_addr, '\0', sizeof(server_addr));
2 server_addr.sin_family = AF_INET;
3 server_addr.sin_port = port;
4 server_addr.sin_addr.s_addr = inet_addr(ip);
```

Binding the socket descriptor with the server address information.

```
1 n = bind(server_sock, (structsockaddr*)&server_addr, sizeof(server_addr));
2 if (n < 0){
3 perror("[-]Bind error");
4 exit(1);
```

```
5  }
6  printf("[+]Bind to the port number: %d\n", port);
```

Now, we listen for incoming connections from the clients.

```
1  listen(server_sock, 5);
2  printf("Listening...\n");
```

The server handles only one client at a time. So, only one client would communicate and the rest would have to wait for the communication to get completed.

```
1  while(1){
2    addr_size = sizeof(client_addr);
3    client_sock = accept(server_sock, (structsockaddr*)&client_addr, &addr_size);
4    printf("[+]Client connected.\n");
5
6    bzero(buffer, 1024);
7    recv(client_sock, buffer, sizeof(buffer), 0);
8    printf("Client: %s\n", buffer);
9
10   bzero(buffer, 1024);
11   strcpy(buffer, "HI, THIS IS SERVER. HAVE A NICE DAY!!!");
12   printf("Server: %s\n", buffer);
13   send(client_sock, buffer, strlen(buffer), 0);
14
15   close(client_sock);
16   printf("[+]Client disconnected.\n\n");
17   }
```

To communicate with all the clients, we start a while loop and the following things happen:

- Accept the client connection.
- We receive a message from the client and print it on the console.
- Next, the server send a reply message to the client.
- Close connection from the client.

The following sequences go on with other clients.

CLIENT-SIDE CODE

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <unistd.h>
5   #include <arpa/inet.h>
6
7   int main(){
8
9     char *ip = "127.0.0.1";
10    int port = 5566;
11
12    int sock;
13    structsockaddr_inaddr;
14    socklen_taddr_size;
15    char buffer[1024];
16    int n;
17
18    sock = socket(AF_INET, SOCK_STREAM, 0);
19    if (sock < 0){
20      perror("[-]Socket error");
21      exit(1);
22    }
23    printf("[+]TCP server socket created.\n");
```

```
24
25    memset(&addr, '\0', sizeof(addr));
26    addr.sin_family = AF_INET;
27    addr.sin_port = port;
28    addr.sin_addr.s_addr = inet_addr(ip);
29
30    connect(sock, (structsockaddr*)&addr, sizeof(addr));
31    printf("Connected to the server.\n");
32
33    bzero(buffer, 1024);
34    strcpy(buffer, "HELLO, THIS IS CLIENT.");
35    printf("Client: %s\n", buffer);
36    send(sock, buffer, strlen(buffer), 0);
37
38    bzero(buffer, 1024);
39    recv(sock, buffer, sizeof(buffer), 0);
40    printf("Server: %s\n", buffer);
41
42    close(sock);
43    printf("Disconnected from the server.\n");
44
45    return 0;
46  }
```

The client-side process can be broken down into the following steps:

1. socket() – create TCP socket.
2. connect() – connect to the server.
3. recv() and send () – communicate with each other.
4. close() – close the connection.

We begin by including all the required header files, defining the IP (Internet Protocol) address and the port number. We also define the required variable used later in this program.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <unistd.h>
5   #include <arpa/inet.h>
```

NOTE: Make sure that the IP address and the port number for the client are the same as the server.

```
1   char *ip = "127.0.0.1";
2   int port = 5566;
3
4   int sock;
5   structsockaddr_inaddr;
6   socklen_taddr_size;
7   char buffer[1024];
8   int n;
```

We start by creating a TCP (Transmission Control Protocol) socket. The socket would be used to connect to the server and start communication.

```
1   sock = socket(AF_INET, SOCK_STREAM, 0);
2   if (sock < 0){
3     perror("[-]Socket error");
4     exit(1);
5   }
6   printf("[+]TCP server socket created.\n");
```

We provide the required IP address and the port number to the required data structures.

```
1   memset(&addr, '\0', sizeof(addr));
```

```
2   addr.sin_family = AF_INET;
3   addr.sin_port = port;
4   addr.sin_addr.s_addr = inet_addr(ip);
```

We send a connection request to the server and wait for the server to accept the request.

```
1   connect(sock, (structsockaddr*)&addr, sizeof(addr));
2   printf("Connected to the server.\n");
```

We send a message to the server and wait for the reply.

```
1   bzero(buffer, 1024);
2   strcpy(buffer, "HELLO, THIS IS CLIENT.");
3   printf("Client: %s\n", buffer);
4   send(sock, buffer, strlen(buffer), 0);
```

We receive the reply from the server and print it on the console.

```
1   bzero(buffer, 1024);
2   recv(sock, buffer, sizeof(buffer), 0);
3   printf("Server: %s\n", buffer);
```

At last, we close the connection from the server.

```
1   close(sock);
2   printf("Disconnected from the server.\n");
```

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

int main(){

  char *ip = "127.0.0.1";
  int port = 5566;

  int server_sock, client_sock;
  struct sockaddr_in server_addr, client_addr;
  socklen_t addr_size;
  char buffer[1024];
  int n;

  server_sock = socket(AF_INET, SOCK_STREAM, 0);
  if (server_sock < 0){
    perror("[-]Socket error");
    exit(1);
  }
  printf("[+]TCP server socket created.\n");

  memset(&server_addr, '\0', sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = port;
  server_addr.sin_addr.s_addr = inet_addr(ip);

  n = bind(server_sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
  if (n < 0){
    perror("[-]Bind error");
    exit(1);
```

```
  }
  printf("[+]Bind to the port number: %d\n", port);

  listen(server_sock, 5);
  printf("Listening...\n");

  while(1){
    addr_size = sizeof(client_addr);
    client_sock = accept(server_sock, (struct sockaddr*)&client_addr, &addr_size);
    printf("[+]Client connected.\n");

    bzero(buffer, 1024);
    recv(client_sock, buffer, sizeof(buffer), 0);
    printf("Client: %s\n", buffer);

    bzero(buffer, 1024);
    strcpy(buffer, "HI, THIS IS SERVER. HAVE A NICE DAY!!!");
    printf("Server: %s\n", buffer);
    send(client_sock, buffer, strlen(buffer), 0);

    close(client_sock);
    printf("[+]Client disconnected.\n\n");

  }
  return 0;
}
```

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

int main(){

  char *ip = "127.0.0.1";
  int port = 5566;

  int sock;
  struct sockaddr_in addr;
  socklen_t addr_size;
  char buffer[1024];
  int n;

  sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock < 0){
    perror("[-]Socket error");
    exit(1);
  }
  printf("[+]TCP server socket created.\n");

  memset(&addr, '\0', sizeof(addr));
  addr.sin_family = AF_INET;
  addr.sin_port = port;
```

```
    addr.sin_addr.s_addr = inet_addr(ip);

    connect(sock, (struct sockaddr*)&addr, sizeof(addr));
    printf("Connected to the server.\n");

    bzero(buffer, 1024);
    strcpy(buffer, "HELLO, THIS IS CLIENT.");
    printf("Client: %s\n", buffer);
    send(sock, buffer, strlen(buffer), 0);

    bzero(buffer, 1024);
    recv(sock, buffer, sizeof(buffer), 0);
    printf("Server: %s\n", buffer);

    close(sock);
    printf("Disconnected from the server.\n");

    return 0;
}
```

EXPERIMENT / PRACTICAL -12

Implement SMTP protocol.

AIM:
Implement SMTP protocol

DESCRIPTION:
Simple Mail Transfer Protocol (SMTP)

Pre-Requisite: Application Layer

Email is emerging as one of the most valuable services on the internet today. Most internet systems use SMTP as a method to transfer mail from one user to another. SMTP is a push protocol and is used to send the mail whereas POP (post office protocol) or IMAP (internet message access protocol) is used to retrieve those emails at the receiver's side.

SMTP Fundamentals

SMTP is an application layer protocol. The client who wants to send the mail opens a TCP connection to the SMTP server and then sends the mail across the connection. The SMTP server is an always-on listening mode. As soon as it listens for a TCP connection from any client, the SMTP process initiates a connection through port 25. After successfully establishing a TCP connection the client process sends the mail instantly.

SMTP Protocol

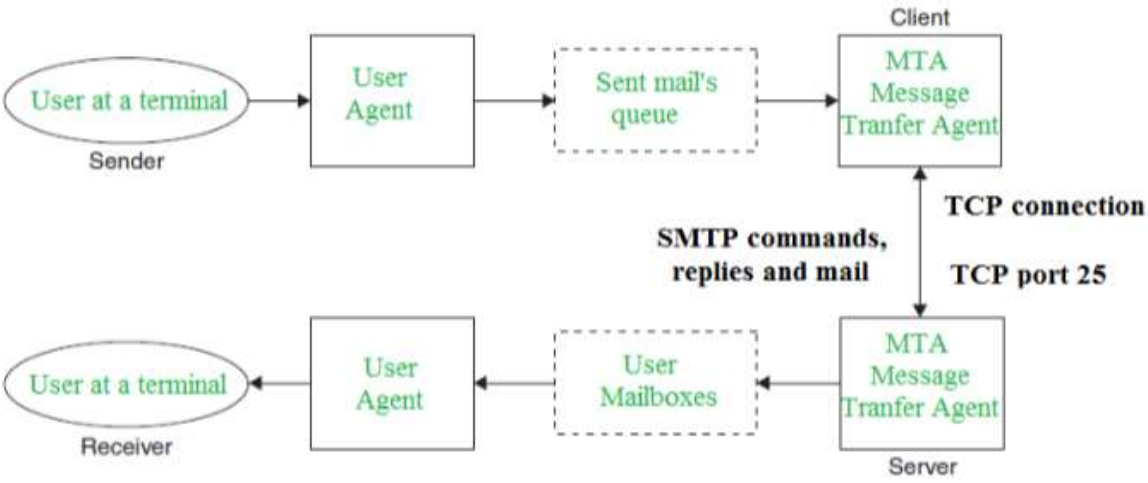
The SMTP model is of two types:

1. End-to-end method
2. Store-and-forward method

The end-to-end model is used to communicate between different organizations whereas the store and forward method is used within an organization. An SMTP client who wants to send the mail will contact the destination's host SMTP directly, in order to send the mail to the destination. The SMTP server will keep the mail to itself until it is successfully copied to the receiver's SMTP.

The client SMTP is the one that initiates the session so let us call it the client-SMTP and the server SMTP is the one that responds to the session request so let us call it receiver-SMTP. The client-SMTP will start the session and the receiver SMTP will respond to the request.

Model of SMTP System
In the SMTP model user deals with the user agent (UA), for example, Microsoft Outlook, Netscape, Mozilla, etc. In order to exchange the mail using TCP, MTA is used. The user sending the mail doesn't have to deal with MTA as it is the responsibility of the system admin to set up a local MTA. The MTA maintains a small queue of mail so that it can schedule repeat delivery of mail in case the receiver is not available. The MTA delivers the mail to the mailboxes and the information can later be downloaded by the user agents.



SMTP Model

Components of SMTP

1. Mail User Agent (MUA)
2. Mail Submission Agent (MSA)
3. Mail Transfer Agent (MTA)
4. Mail Delivery Agent (MDA)

1. Mail User Agent (MUA): It is a computer application that helps you in sending and retrieving mail. It is responsible for creating email messages for transfer to the mail transfer agent(MTA).

2. Mail Submission Agent (MSA): It is a computer program that basically receives mail from a Mail User Agent(MUA) and interacts with the Mail Transfer Agent(MTA) for the transfer of the mail.

3. Mail Transfer Agent(MTA): It is basically software that has the work to transfer mail from one system to another with the help of SMTP.

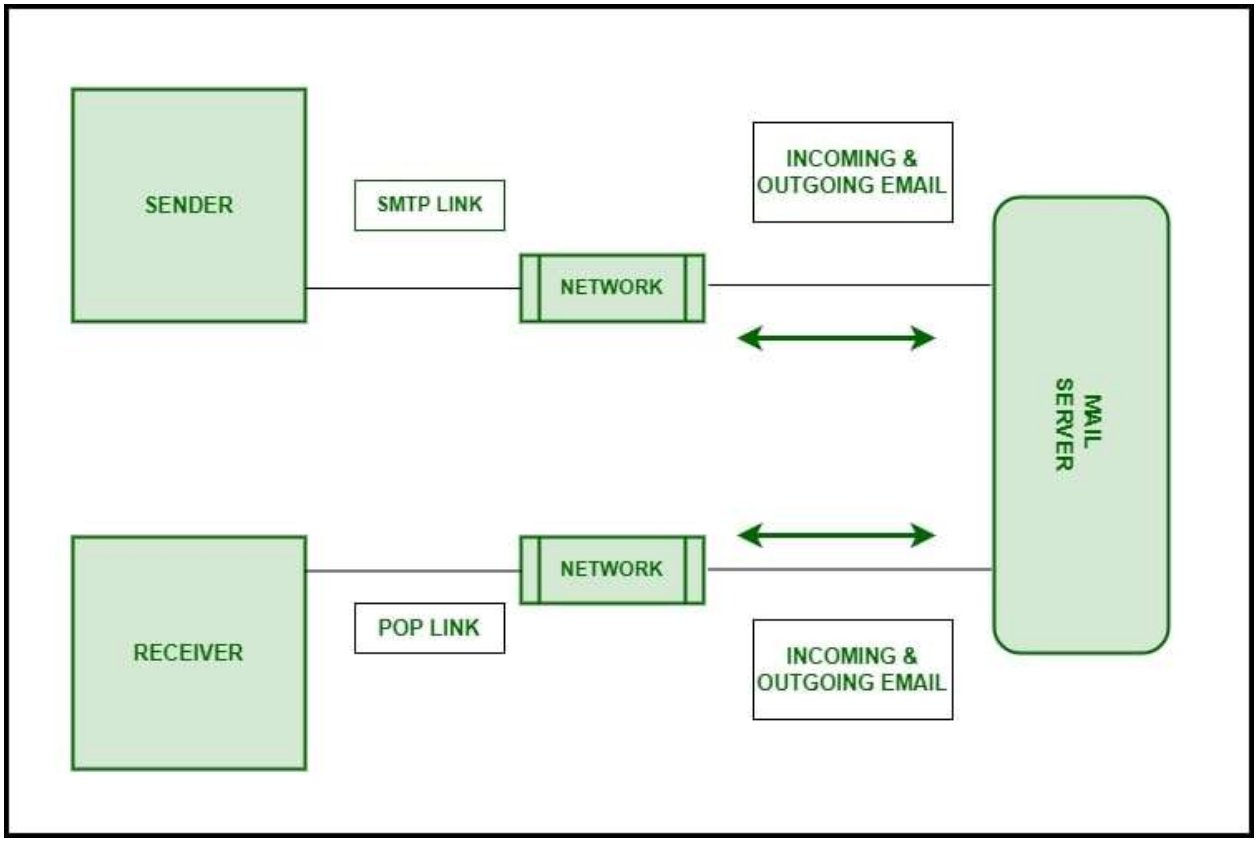
4. Mail Delivery Agent(MDA): A mail Delivery agent or Local Delivery Agent is basically a system that helps in the delivery of mail to the local system.

Working of SMTP

1. Communication between the sender and the receiver : The sender's user agent prepares the message and sends it to the MTA. The MTA's responsibility is to transfer the mail across the network to the receiver's MTA. To send mail, a system must have a client MTA, and to receive mail, a system must have a server MTA.

2. Sending Emails: Mail is sent by a series of request and response messages between the client and the server. The message which is sent across consists of a header and a body. A null line is used to terminate the mail header and everything after the null line is considered the body of the message, which is a sequence of ASCII characters. The message body contains the actual information read by the receipt.

3. Receiving Emails: The user agent on the server-side checks the mailboxes at a particular time of intervals. If any information is received, it informs the user about the mail. When the user tries to read the mail it displays a list of emails with a short description of each mail in the mailbox. By selecting any of the mail users can view its contents on the terminal.



Working of SMTP

Some SMTP Commands

- HELO – Identifies the client to the server, fully qualified domain name, only sent once per session
- MAIL – Initiate a message transfer, the fully qualified domain of the originator
- RCPT – Follows MAIL, identifies an addressee, typically the fully qualified name of the addressee, and for multiple addressees use one RCPT for each addressee
- DATA – send data line by line

For more, you can refer to SMTP Commands.

DIFFERENCE BETWEEN SMTP AND EXTENDED SMTP

Extended STMP is an extended version of SMTP. Extended SMTP is a set of protocols for sending and receiving electronic messages on the internet. First, Email is sent from sender to sender-server through ESTMP and from sender-server to receiver-server on the internet through ESTMP. ESMTP follows the same protocols as SMTP. It adds more functionality, security, and authentication than SMTP.
Let's see some basic differences between them.

SMTP	Extended SMTP
Users were not verified in SMTP as a result of massive-scale scam emails being sent.	In Extended SMTP, authentication of the sender is done.
We cannot attach a Multimedia file in SMTP directly without the help of MMIE.	We can directly attach Multimedia FIle in ESMTP.

SMTP	Extended SMTP
We cannot reduce the size of the email in Extended SMTP.	We can reduce the size of the email in Extended SMTP.
SMTP clients open transmission with the command HELO.	The main identification feature for ESMTP clients is to open a transmission with the command EHLO (Extended HELLO).

ADVANTAGES OF SMTP

- If necessary, the users can have a dedicated server.
- It allows for bulk mailing.
- Low cost and wide coverage area.
- Offer choices for email tracking.
- Reliable and prompt email delivery.

DISADVANTAGES OF SMTP

- SMTP's common port can be blocked by several firewalls.
- SMTP security is a bigger problem.
- Its simplicity restricts how useful it can be.
- Just 7-bit ASCII characters can be used.
- If a message is longer than a certain length, SMTP servers may reject the entire message.
- Delivering your message will typically involve additional back-and-forth processing between servers, which will delay sending and raise the likelihood that it won't be sent.

// Implementation of SMTP

==========

```
int main()
{
int i,sockfd;
struct sockaddr_in server;
char buf[MAX];
char tmp[1024];

sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
{
perror("Error:");
exit(0);
}
server.sin_family=AF_INET;
server.sin_port=htons(25);
server.sin_addr.s_addr=inet_addr("192.168.10.55");

i=connect(sockfd,(struct sockaddr *)&server,sizeof(server));

if(i<0)
{
perror("Error:");
close(sockfd);
```

```
exit(0);
}

memset(buf,0,MAX);
read(sockfd,buf,MAX);
printf("\n%s",buf);
sleep(1);
memset(buf,0,MAX);
strcpy(buf,"HELO localhost.localdomain\r\n");
write(sockfd,buf,strlen(buf));
memset(buf,0,MAX);
read(sockfd,buf,MAX);
printf("\n%s\nSender mail:",buf);
scanf("%s",tmp);
strcat(tmp,"\r\n");
sleep(1);
memset(buf,0,MAX);
strcpy(buf,"MAIL FROM:");
strcat(buf,tmp);
write(sockfd,buf,strlen(buf));
memset(buf,0,MAX);
read(sockfd,buf,MAX);
printf("\n%s\nReceipient mail:",buf);
scanf("%s",tmp);
strcat(tmp,"\r\n");
sleep(1);
memset(buf,0,MAX);
strcpy(buf,"RCPT TO:");
strcat(buf,tmp);
write(sockfd,buf,strlen(buf));
memset(buf,0,MAX);
read(sockfd,buf,MAX);
printf("\n%s",buf);
memset(buf,0,MAX);
strcpy(buf,"DATA\r\n");
write(sockfd,buf,strlen(buf));
memset(buf,0,MAX);
read(sockfd,buf,MAX);
printf("\n%s",buf);
memset(buf,0,MAX);
while(1)
{
memset(tmp,0,1024);
gets(tmp);
if(!strcmp(tmp,"."))
break;
strcat(buf,tmp);
}
strcat(buf,"\r\n.\r\n");
write(sockfd,buf,strlen(buf));
memset(buf,0,MAX);
read(sockfd,buf,MAX);
printf("\n%s",buf);
sleep(1);
close(sockfd);
return 0;
}
```

GAP ANALYSIS BY THE COURSE FACULTY / COORDINATOR

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

Faculty name & Signature with date

HoD/BoS Chairman

remarks:……

……

……

……

HoD/BoS chairman Signature with date