LEARN BEYOND

KPR Institute of Engineering and Technology

(Autonomous)

Avinashi Road, Arasur, Coimbatore - 641 407

B.E. - COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

LABORATORY RECORD

ACADEMIC YEAR: 2025-2026 [ODD SEM]

U21CSG05

COMPUTER NETWORKS

(R2021 Revised)

Department of CSE (Artificial Intelligence and Machine Learning)

KPR Institute of Engineering and Technology

KPRIET
Learn Beyond

LABORATORY RECORD

NAME	:	--
REGSITER NUMBER	:	--
ROLL NO	:	--
SUB. CODE/TITLE	:	--
YEAR/SEM	:	--

Certified that this is a Bonafide record of work done by ________________________ during the academic year________________.

Signature of the Course In-charge **Signature of the HoD**

Place:

Date:

He / She has submitted the record for the End Semester Practical Examination held on ________________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

VISION AND MISSION OF THE INSTITUTION

Vision

To become a premier institute of academic excellence by imparting technical, intellectual and professional skills to students for meeting the diverse need of the industry, society, the nation and the world at large

Mission

- Commitment to offer value-based education and enhancement of practical skills
- Continuous assessment of teaching and learning process through scholarly activities
- Enriching research and innovation activities in collaboration with industry and institute of repute
- Ensuring the academic process to uphold culture, ethics and social responsibilities

VISION AND MISSION OF THE DEPARTMENT

Vision

To establish as a technology hub of education, research and solution in artificial intelligence and machine learning.

- Provide an enriched educational experience in artificial intelligence and machine learning, that students are technically competent.
- Interact and collaborate with every industry segment and solving to mobilize the possibilities of artificial intelligence and machine learning.
- Create new computing technologies and solution for industry and society with high ethical and novel values.

Program Educational Objectives (PEOS)
The graduates of Bachelor of Engineering (Artificial Intelligence and Machine Learning) will be able to

- PEO1: Devise cutting edge solutions to the emerging technological problem.
- PEO2: Practice lifelong learning by upskilling in advanced research in artificial intelligence and machine learning technologies.
- PEO3: Function in their profession as socially responsible individuals adhering to the rich cultural and moral ethics.

Program Outcomes (POS)
Engineering Graduates will be able to:

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

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

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOS)

A graduate of Computer Science and Engineering (Artificial Intelligence and Machine Learning) student will be able to

- PSO1: Design and develop an intelligent automated system applying fundamental knowledge from mathematical, analytical programming and operational skills to solve the arising problems in the field of technology.

- PSO2: Efficiently apply machine learning techniques to fit various business situations.

RUBRICS FOR ASSESSMENT

Criteria		Excellent (4 Marks)	Good (3 Marks)	Adequate (2 Marks)	Inadequate (1 Mark)
A .Preparation & Observation	**Criterion #1** Ability to setup and conduct experiments	Able to develop contingency or alternative plans and anticipate problems during experiment.	Able to develop contingency or alternative plans.	Able to use theoretical framework, measurement techniques, testing apparatus or model.	Unable to identify theoretical framework, measurement techniques. Testing apparatus or model.
	Criterion #2 Ability to take measurements /readings and present data	Able to formulate, controls evaluate alternatives of the experiment. Able to evaluate data and relate to engineering phenomena decision-making.	Able to evaluate data and relate to engineering phenomena for decision-making.	Able to constraint apply and assumption into the experimental design. Able to conduct experiment correctly and collect data.	Unable to discuss experimental processes and protocols.
B .Results & Interpretation	**Criterion #3** Ability to analyze the data theoretically and logically to conclude experimental results	Able to combine /organize more than one set of data, interpret data and make meaningful conclusion.	Able to evaluate or compare data and make meaningful conclusion	Able to select and use and appropriate techniques apply methods to analyses the data.	Unable to select and describe the techniques or methods or of analyzing the data
	Criterion #4 Ability to interpret and discuss any discrepancies between theoretical and experimental results	Able to verify and/or validate several sets of data and relates to engineering phenomena for decision making.	Able to verify and/or validate data and relate to engineering phenomena for decision making.	Able to identify and verify how results relate/differ from for theory or previous results	Unable to identify how results relate/differ from theory or previous results.
C. Viva Voce	**Criterion #5** Demonstrate the ability to respond effectively to questions	Able to listen carefully and respond to questions appropriately; is able to explain and interpret results to the teacher	Able to listen carefully and respond to questions appropriately	Misunderstand the questions and does not respond appropriately to the teacher, or has some trouble in answering questions	Unable to listen carefully questions to and does not provide an appropriate answer, or is unable to answer questions

LIST OF EXPERIMENTS

EX.NO: 01	Network Commands
DATE:	

Aim

To gain practical experience in using and executing a variety of networking commands for diagnosing and resolving network issues on a Windows system.

1. Ping Command

The ping command is used to test network connectivity and diagnose network-related issues by sending Internet Control Message Protocol (ICMP) Echo Request packets to a target host or IP address and waiting for ICMP Echo Reply packets.

Syntax:
ping [options] destination
destination: The IP address or hostname (domain name) of the target host you want to ping.

Common Options:
-n <count>: Specify the number of ICMP Echo Request packets to send (default is 4).
-t: Ping continuously until manually stopped (press Ctrl + C to stop).
-l <size>: Set the size of the ICMP Echo Request packets in bytes.
-w <timeout>: Set the timeout in milliseconds to wait for each reply.
-4: Force the use of IPv4.

Examples:

1. Send four ICMP Echo Request packets to an IP address (e.g., 8.8.8.8):
ping -n 4 8.8.8.8

2. Continuously ping a host and display the results until manually stopped:
ping -t google.com

3. Set the size of the ICMP packets to 100 bytes:
ping -l 100 example.com

4. Set a longer timeout (e.g., 3000 milliseconds) for waiting for replies:
ping -w 3000 example.com

5. Force the use of IPv4 for the ping:
ping -4 example.com

2. <u>Traceroute Command:</u>

The tracert command is used to perform the equivalent of the traceroute command in Unix-like operating systems. tracert allows you to trace the route that packets take from your computer to a destination host or IP address by sending ICMP Echo Request packets with varying Time-to-Live (TTL) values and observing the ICMP Time Exceeded replies from routers along the path.

Syntax:
tracert [options] destination

destination: The IP address or hostname (domain name) of the target host you want to trace the route to.

Common Options:
-d: Perform a trace without resolving hostnames to IP addresses (numeric output).
-h <max_hops>: Set the maximum number of hops (TTL) to search for the target.
-w <timeout>: Set the timeout in milliseconds to wait for each reply.
-4: Force the use of IPv4.

Examples:

1. Trace the route to a domain (e.g., google.com):
   tracert google.com

2. Trace the route to an IP address (e.g., 8.8.8.8) using numeric output:
   tracert -d 8.8.8.8

3. Set the maximum number of hops (TTL) to 30:
   tracert -h 30 example.com

4. Set a longer timeout (e.g., 500 milliseconds) for waiting for replies:
   tracert -w 500 example.com

5. Force the use of IPv4 for the trace:
   tracert -4 example.com

3. <u>Netstat Command</u>

The netstat command in Windows is a network utility that allows you to display network statistics, active network connections, routing tables, and various network-related information on a Windows operating system. It's a versatile tool for troubleshooting network issues and monitoring network activities.

Syntax:

netstat [options]

Common Options:

-a: Display all active connections and listening ports.
-n: Display addresses and port numbers in numeric format (no hostname resolution).
-b: Display the executable involved in creating each connection or listening port.
-o: Display the owning process identifier (PID) for each connection.
-r: Display the routing table.
-s: Display per-protocol statistics (e.g., TCP, UDP).
-p <protocol>: Show connections for a specific protocol (e.g., -p tcp or -p udp).

Examples:

1. Display all active connections and listening ports:
   netstat -a

2. Display all active connections and listening ports with numeric addresses and ports:
   netstat -an

3. Display listening ports and the associated processes:
   netstat -ab

4. Display routing table information:
   netstat -r

5. Display per-protocol statistics (e.g., TCP and UDP):
   netstat -s

6. Display active connections for a specific protocol, such as TCP:
   netstat -p tcp

4. Ipconfig

The ipconfig command in Windows is a command-line tool used to display and manage network configuration settings on a Windows computer. It provides information about the computer's IP configuration, including the IP address, subnet mask, default gateway, DNS servers, and more.

Basic Usage:

To display basic IP configuration information for all network interfaces, open Command Prompt and type:

ipconfig
This command will display information for all active network interfaces on your computer.

Common ipconfig Options:

1. /all: Displays detailed information for all network interfaces, including physical and virtual adapters.
   ipconfig /all

2. /release: Releases the DHCP lease for all network interfaces, relinquishing their IP addresses.
   ipconfig /release

3. /renew: Renews the DHCP lease for all network interfaces, obtaining new IP addresses if available.
   ipconfig /renew

4. /flushdns: Flushes the DNS resolver cache, which can be useful when troubleshooting DNS-related issues.
   ipconfig /flushdns

5. /displaydns: Displays the contents of the DNS resolver cache, showing the resolved domain names and their corresponding IP addresses.
   ipconfig /displaydns

5. Nslookup

The nslookup command is a network administration tool available in Windows for querying the Domain Name System (DNS) to obtain information about domain names, IP addresses, and other DNS-related records. It can help you troubleshoot and diagnose DNS-related issues or perform DNS lookups.

Basic options
nslookup domain_name
1. Replace domain_name with the domain or hostname you want to look up.

nslookup www.example.com
This will display the IP address(es) associated with the specified domain.
2. To perform a reverse DNS lookup (resolve an IP address to its domain name), type the following command:

nslookup IP_address
Replace IP_address with the actual IP address you want to look up. For example:

nslookup 8.8.8.8
This will display the domain name(s) associated with the specified IP address.

3. To check specific DNS record types for a domain, you can use the following format:

nslookup -type=record_type domain_name
Replace record_type with the specific DNS record type you want to query (e.g., A, MX, NS, TXT) and domain_name with the domain you want to query. For example:
nslookup -type=MX example.com
This will show the MX (Mail Exchange) records for the domain.

4. To change the DNS server used for the lookup (instead of using the default DNS server), you can specify the server using the following format:

nslookup domain_name dns_server
Replace dns_server with the IP address or hostname of the DNS server you want to use.
nslookup www.google.co.in 10.10.1.10

6. Arp Command:

arp command displays and manages the ARP (Address Resolution Protocol) cache, which maps IP addresses to MAC addresses on your local network.
arp –a

route: route command is to display and manage the local IP routing table.
route print

7. Tcpdump:

Tcpdump command is a famous network packet analysing tool that is used to display TCP\IP & other network packets being transmitted over the network attached to the system on which tcpdump has been installed. Tcpdump uses libpcap library to capture the network packets & is available on almost all Linux/Unix flavors.

tcpdump command can read the contents from a network interface or from a previously created packet file orcan also write the packets to a file to be used for later. One must use the tcpdump command as root or as a user with sudo privileges.

By default, tcpdump is available on almost all Linux distributions but if that’s not the case for you, install it on your system using the following method.

Step 1 – Download and install Windump
You will need to place your network card into promiscuous mode – for this, install WinPcap.
Step 2 – Download and install WinPcap

Step 3 – Open a Command Prompt with Administrator Rights
Start > Accessories > Command Prompt
Right Click > Run As Administrator
Change the directory to your download directory – normally in windows this is:
cd c:\Users\Smile\Downloads
Smile will be replaced with your username eg cd c:\Users\your username\Downloads
Step 4 – Run windump to locate your network adapter.
Windump will list your adapter with a number.
You may have several adapters listed. You select the interface to start running windump (as shown in step 5 using interface number 2).
Step 5 – Run windump to collect packets and write out to a file
windump -i 2 -q -w C:\perflogs\diagTraces -n -C 30 -W 10 -U -s 0
This will create a directory c:\perflogs\ and a file called diagTrace0.
The switches mean this:

- -i is the number of NIC selected in the previous step
- -q is quiet mode
- -w <name> is the prefix of the files to create
- -n the logging will not resolve host names, all data will be in IP address format
- -C the size in Millions of Bytes the logs files so grow to before moving to the next file
- -W the number of circular log files to retain in addition to the current log file, specify in <path> where the files are to be stored
- -U as each packet is saved, it will be written to the output file
- -s decreases the amount of packet buffering, set this to zero.

Step 6 – Use Wireshark to Open your file

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result

Thus, various networking commands were executed successfully.

EX.NO: 02	**Fabrication of Cables**
DATE:	

Aim:

To practice the colour code for different cables. Observe the Lan Tester and make the decision accordingly.

Theory:

A twisted pair consists of two insulated conductor twisted together in the shape of a spiral. It can be shielded or unshielded. The unshielded twisted pair cables are very cheap and easy to install. But they are very badly affected by the electromagnetic noise interference.

Twisting of wires will reduce the effect of noise or external interference. The induced emf into the two wires due to interference tends to cancel each other due to twisting. Number of twists per unit length will determine the quality of cable. More twists means better quality.

There are 3 types of UTP cables:-
1) Straight-through cable
2) Crossover cable
3) Roll-over cable

A. Straight-through cable
Straight-Through refers to cables that have the pin assignments on each end of the cable. In other words Pin 1 connector A goes to Pin 1 on connector B, Pin 2 to Pin 2 ect. Straight-Through wired cables are most commonly used to connect a host to client. Whentalk about cat5e patch cables, the Straight-Through wired cat5e patch cable is used to connect computers, printers and other network client devices to the router switch or hub (the host device in this instance).

B. Crossover cable
Crossover wired cables (commonly called crossover cables) are very much like Straight-Through cables with the exception that TX and RX lines are crossed (they are at oposite positions on either end of the cable. Using the 568-B standard as an example below you will see that Pin 1 on connector A goes to Pin 3 on connector B. Pin 2 on connector A goes to Pin 6 on connector B ect. Crossover cables are most commonly used to connect two hosts directly. Examples would be connecting a computer directly to another computer, connecting a switch directly to another switch, or connecting a router to a router.Note: While in the past when connecting two host devices directly a crossover cable was required. Now days most devices have auto sensing technology that detects the cable and device and crosses pairs when needed.

C. Roll-over cable
Rollover wired cables most commonly called rollover cables, have opposite Pin assignments on each end of the cable or in other words it is "rolled over". Pin 1 of connector A would be

connected to Pin 8 of connector B. Pin 2 of connector A would be connected to Pin 7 of connector B and so on. Rollover cables, sometimes referred to as Yost cables are most commonly used to connect to a devices console port to make programming changes to the device. Unlike crossover and straight-wired cables, rollover cables are not intended to carry data but instead create an interface with the device.

Procedure:

1) The aim is to Fabricate a UTP Cable.
2) To perform the experiment follow the below steps
3) A choice list would be give that which type of cable is to be fabricated
4) Select the choice out of the three choices given
5) Once a selection is done then the user have to make the cable ready
6) In-order to do so select the color codes on both the sides i.e Switch port and PC port.
7) After assigning the color codes click on the Start button to observe that the cable made is correct or not.
8) Based on the observations made select if cable made is correct or not.

Sample Cable Fabrication:

http://vlabs.iitb.ac.in/vlabs-dev/labs_local/computer-networks/labs/exp1/exp1.html

2 b. Peer to Peer Topology

Aim:

To construct Peer to Peer Topology.

Theory:

The word physical network topology is used to explain the manner in which a network is physically connected. Devices or nodes in a network get connected to each other via communication links and all these links are related to each other in one way or the other. The geometric representation of such a relationship of links and nodes is known as the topology of that network.

These topologies can be classifies into two types:-

1. Peer to peer
2. Primary - Secondary

Peer to peer is the relationship where the devices share the link equally. The examples are ring and mesh topologies.

In Primary - Secondary relationship, one device controls and the other devices have to transmit through it. For example star and tree topology.

Features of Peer to peer:-

In peer to peer architecture every node is connected to other node directly.

Every computer node is referred as peer.
Every peer provides services to other peers as well as uses services of them.
There is no central server present.

Advantages of Peer to peer:-

1) It is easy to install and so is the configuration of computers on this network,
2) All the resources and contents are shared by all the peers
3) P2P is more reliable as central dependency is eliminated. Failure of one peer doesn’t affect the functioning of other peers.
4) There is no need for full-time System Administrator. Every user is the administrator of his machine. User can control their shared resources.

Disadvantages of Peer to peer:-
1) In this network, the whole system is decentralised thus it is difficult to administer. That is one person cannot determine the whole accessibility setting of whole network.
2) Data recovery or backup is very difficult. Each computer should have its own back-up system.

Procedure:

1) The aim is to create the topology.
2) To perform the experiment follow the below steps
3) A blank square area would be given which defines the working area
4) A series of components would be given
5) In order to build a topology first select on the component and then immediately click on the working area to place it
6) To draw a line between two components first select the line click on the port of first component and then immediately click on the port of second component
7) Once the topology is build then click on the Submit button to test whether the give topology is built correctly or not.

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result:
Thus, the colour codes for different cables were studied.

EX.NO: 03	**Configuration of LAN using Cisco 2960 Switch**
DATE:	

Aim:

Configure Cisco 2960 switch with the two PCs and observe that the data transfer between two computers is reliable.

Requirements:

- Windows PC / Laptop
- CISCO Packet Tracer Software (Student Version)

Procedure:

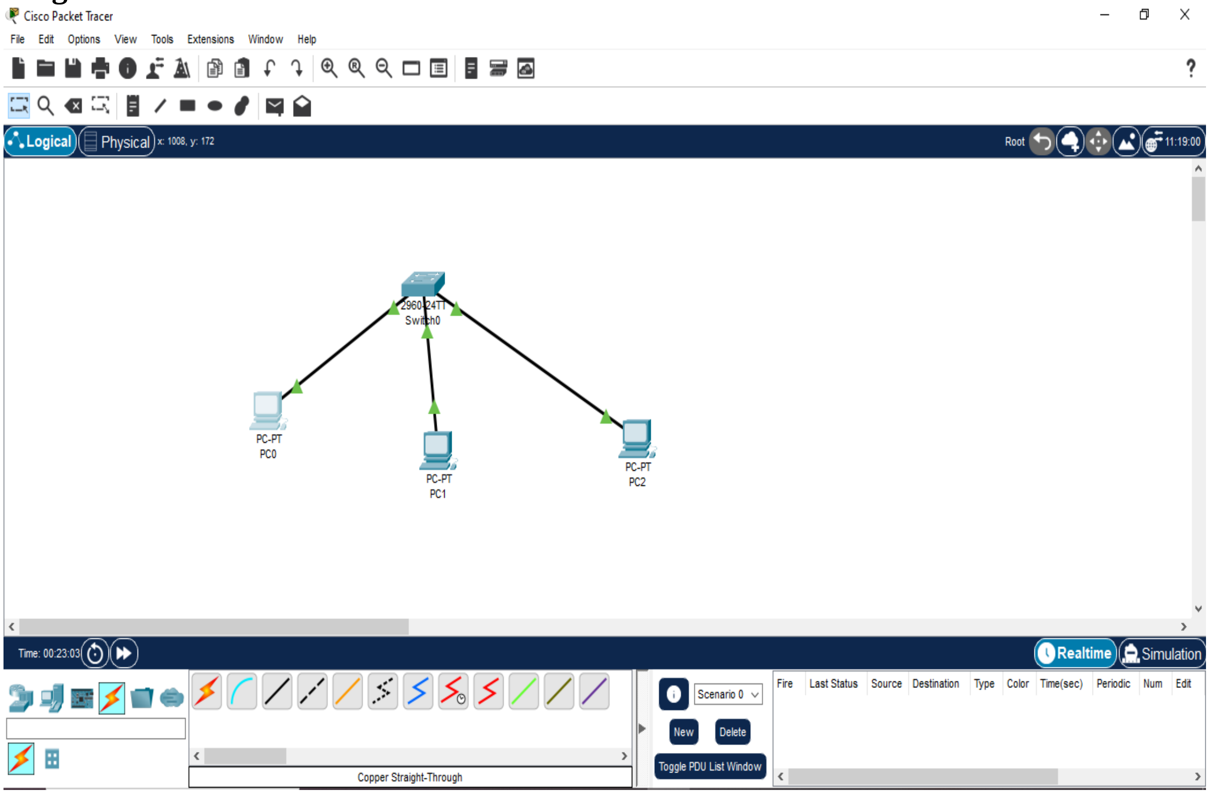
- Open the CISCO Packet tracer software
- Drag and drop 2 pcs using End Device Icons on the left corner
- Cisco 2960 switch from switch icon list in the left bottom corner
- Make the connections using Straight through Ethernet cables
- Give IP address of the PC1 and PC2 as 192.168.1.2 and 192.168.1.3 respectively, ping between PCs and observe the transfer of data packets in real and simulation mode.

Theory:

A local area network (LAN) is a collection of devices connected together in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school.

A LAN comprises cables, access points, switches, routers, and other components that enable devices to connect to internal servers, web servers, and other LANs via wide area networks.
The advantages of a LAN are the same as those for any group of devices networked together. The devices can use a single Internet connection, share files with one another, print to shared printers, and be accessed and even controlled by one another.

Diagram:



OUTPUT (PINGING FROM PC0-PC1):

```
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time=13ms TTL=128
Ping statistics for 192.168.1.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 13ms, Average = 3ms
```

LAN - MAC ADDRESS TABLE

```
Switch>en
Switch#show mac-address-table
Mac Address Table
-------------------------------------------

Vlan Mac Address Type Ports
```

```
---- ----------- -------- -----

1 0040.0b86.916b DYNAMIC Fa0/1
1 0090.213e.7d06 DYNAMIC Fa0/2
```

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result:

Hence, Cisco 2960 switch is configured with the two PCs and observed that the data transfer between two computers was reliable.

EX.NO: 04	Configuration of WAN using Cisco 1841 Router
DATE:	

Two computers, PC1(192.168.1.2) and PC2 (192.168.2.3), from different networks must be configured to communicate with each other. In this experiment, you will use the router as an intermediate device. Configure Cisco 1841 ISR router with the two PCs mentioned above. Experiment and observe that the data transfers between two computers are reliable.

Aim:

Configure Cisco 1841 router with the two PCs and observe that the data transfer between two computers are reliable.

Requirements

- Windows PC / Laptop
- CISCO Packet Tracer Software (Student Version)

Procedure

- Open the CISCO Packet tracer software

LAN 1

- Drag and drop 2 pcs using End Device Icons on the left corner
- Cisco 2960 switch from switch icon list in the left bottom corner
- Make the connections using Straight through Ethernet cables
- Give IP address of the PC1 and PC2 as 192.168.1.2 and 192.168.1.3 respectively, ping between PCs and observe the transfer of data packets in real and simulation mode.

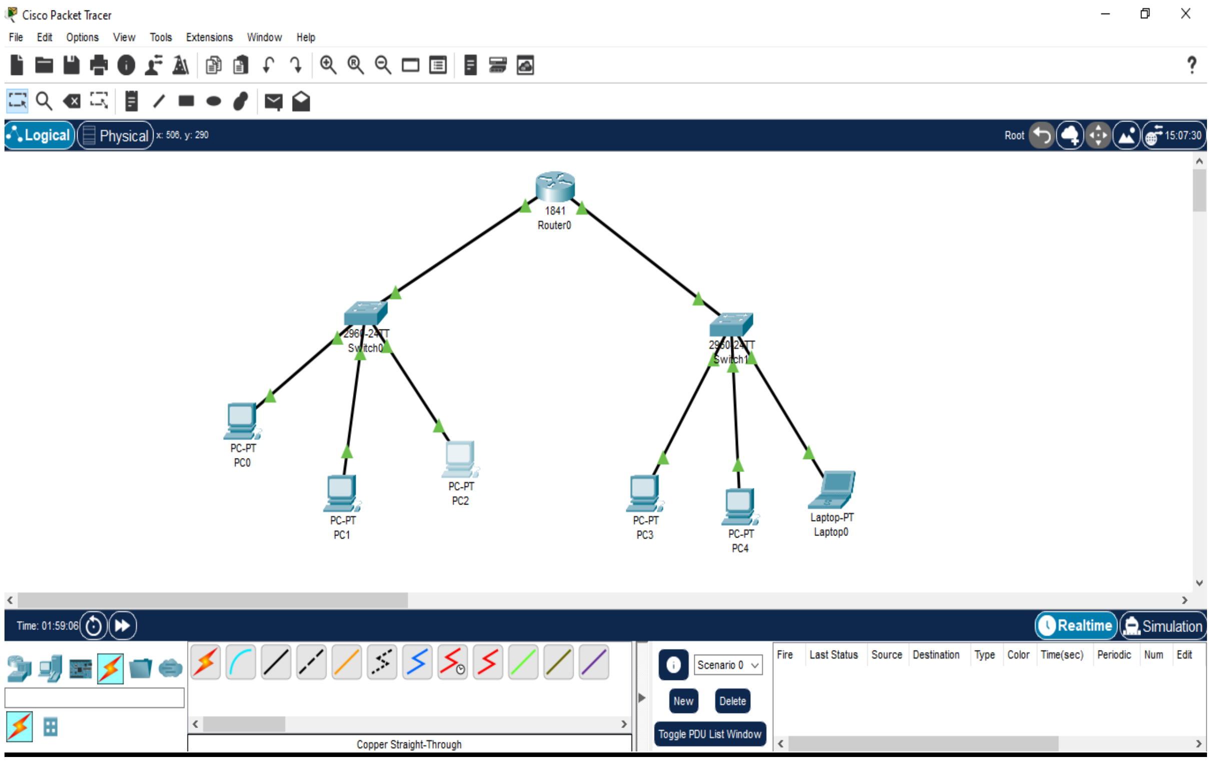
LAN 2

- Drag and drop 2 pcs using End Device Icons on the left corner
- Cisco 2960 switch from switch icon list in the left bottom corner
- Make the connections using Straight through Ethernet cables
- Give IP address of the PC1 and PC2 as 192.168.2.2 and 192.168.2.3 respectively, ping between PCs and observe the transfer of data packets in real and simulation mode.

Internet

- Drag Cisco 1841 router from router icon list in the left bottom corner
- Make the connections using straight through Ethernet cables.
- Give the interface of LAN1 switch to router as 192.168.1.1 and LAN2 switch to router as 192.168.2.1.

- Then add static gateway to the PC's connected in LAN1 as 192.168.1.1 and PC's connected in LAN2 as 192.168.2.1.
- Ping between two LAN's and observe the transfer of data packets in real and simulation mode.



Theory

A local area network (LAN) is a collection of devices connected together in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school.

A LAN comprises cables, access points, switches, routers, and other components that enable devices to connect to internal servers, web servers, and other LANs via wide area networks.

The advantages of a LAN are the same as those for any group of devices networked together. The devices can use a single Internet connection, share files with one another, print to shared printers, and be accessed and even controlled by one another.

OUTPUT (PINGING FROM PC0-PC1)

```
        Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:
```

```
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time=13ms TTL=128

Ping statistics for 192.168.1.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 13ms, Average = 3ms
```

LAN - MAC ADDRESS TABLE

```
Switch>en
Switch#show mac-address-table
Mac Address Table
-------------------------------------------

Vlan Mac Address Type Ports
---- ----------- -------- -----

1 0040.0b86.916b DYNAMIC Fa0/1
1 0090.213e.7d06 DYNAMIC Fa0/2
```

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result

Hence, Cisco 1891 router is configured with the two LANs and observed that the data transfer between two computers was reliable.

EX.NO: 05	Configuration of NAT
DATE:	

The communication between LAN and WAN is to be configured through Network Address Translation (NAT) as a border router. Create a NAT topology with three routers, RT1, RT2, and RT3. Configure static NAT on Router 2(RT2) while Router RT1 is configured in LAN and RT3 is configured in WAN. Use the following IP address to configure the router

Router 1 (RT1) IP address: 192.168.1.2 (local)
Router 3 (RT3) IP address: 110.120.1.2 (local)
Router 2 (RT2) IP address: 110.120.1.2 (global)

After configuring the IP address, check the packet situation by opening debug with the "debug ip icmp" command. Observe and under the displayed.
Now configure the NAT using packet tracer (use the manual to configure). Experiment with different configuration scenarios and check the packet situation between LAN and WAN.

Aim:
To Configure Network Address Translation (NAT) using 3 Cisco Routers.

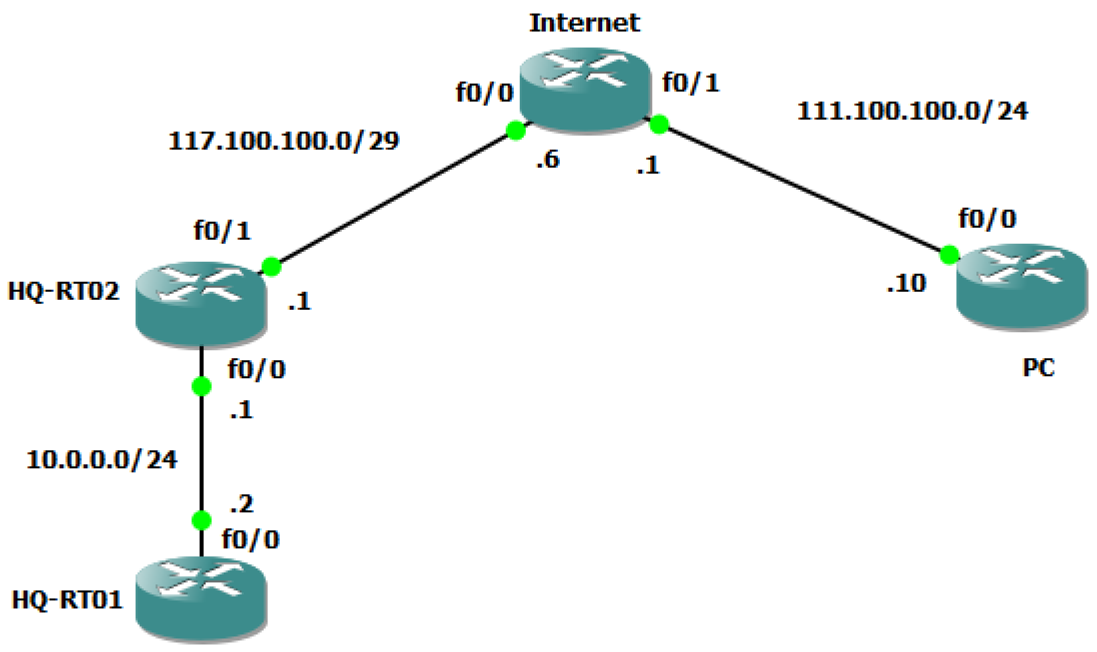
Theory:
The function of Network Address Translation (NAT) is to translation a private IP address to into a public IP address that connected to the internet before packets are forwarded to another network. NAT can advertise a single public IP address for the entire local private network to the internet and providing a security by hiding the entire internal network behind that address.

In this article, will configure different configuration of static NAT and Dynamic NAT on Cisco router using GNS3.

Prerequisites:
In this document, it is supposed that:
a. Check GNS3 VM installed up and running on your computer. Installing GNS3 VM on VMware Workstation
b. Configure SSH remote management on Cisco router.

Lab Scenario Set up:
Set up a Lab to configure NAT as show in the following diagram. Configure NAT on “HQ-RT02”. There is one router in the LAN with the host name as “HQ-RT01” and this host is acting as an inside LAN server. Will configure SSH and Telnet server on this host. There is one router act as the internet and another router act as computer in the public network.

The following is the basic configuration of each devices.



On HQ-RT01

First all, configure host name and IP as the following.

```
# hostname HQ-RT01
# int f0/0
   ip add 10.0.0.2 255.255.255.0
   no sh
# ip route 0.0.0.0 0.0.0.0 10.0.0.1
```

Then, let’s configure SSH server for remote management as the following.

```
# username netadmin privilege 15 secret 1111
# enable secret 2222
# service password-encryption
# ip domain-name techspacekh.com
# crypto key generate rsa
# line vty 0 4
  login local
  transport input all
# ip ssh version 2
# aaa new-model
```

On HQ-RT02

On HQ-RT02 let’s configure the basic configuration as the following.

```
# hostname HQ-RT02
# int f0/0
   no sh
   ip add 10.0.0.1 255.255.255.0
# int f0/1
   no sh
   ip add 117.100.100.1 255.255.255.248
# ip route 0.0.0.0 0.0.0.0 117.100.100.6
```

On router acts as the Internet

Configure IP address on each interface on the Internet router.

```
# int f0/0
  no sh
  ip add 117.100.100.6 255.255.255.248
# int f0/1
  no sh
  ip add 111.100.100.1 255.255.255.0
```

On router acts as a computer in public network

On PC router, let’s configure the basic configuration as the following.

```
# int f0/0
  no sh
  ip add 111.100.100.10 255.255.255.0
# ip route 0.0.0.0 0.0.0.0 111.100.100.1
```

Dynamic NAT

Configure Dynamic NAT To Interface IP

Before configure dynamic NAT, HQ-RT01 is not able to ping to IP address of the internet router.

```
HQ-RT01#ping 117.100.100.6
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 117.100.100.6, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Now let start to configure dynamic NAT on HQ-RT02. First,need to create an ACL to contain the IP address to be NATed. In below ACL, allow all IP in the LAN can access to the internet.

```
# ip access-list standard ACL-DNAT
   permit 10.0.0.0 0.0.0.255
```

Then, need to configure dynamic NAT using the created ACL above.

```
# int f0/0
   ip nat inside
# int f0/1
   ip nat outside
# ip nat inside source list ACL-DNAT interface f0/1
```

Now, HQ-RT01 should be able to ping to IP address of the internet router.

```
HQ-RT01#ping 117.100.100.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 117.100.100.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/30/32 ms
```

Ifshow the NAT translation,should see something as the following.

```
#sh ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
icmp 117.100.100.1:9   10.0.0.2:9         117.100.100.6:9    117.100.100.6:9
```

4.2 Configure Dynamic NAT With IP Pool

It is also possible to translate our LAN IP subnet to pool of public IP address. In the following command will create a NAT pool with IP address rang from 117.100.100.4 to 117.100.100.5.

```
# ip nat pool DNAT-POOL01 117.100.100.4 117.100.100.5 netmask 255.255.255.248
```

Then use the pool created above to do the NAT configuration.

```
# no ip nat inside source list ACL-DNAT interface FastEthernet0/1 overload
# ip nat inside source list ACL-DNAT pool DNAT-POOL01
```

5. Static NAT

5.1 Configure Static NAT To Interface IP

To publish the access of the internal server to the internet, configure static NAT on our router "HQ-RT02".

In the following command will configure a static NAT for remote SSH access from the internet using the IP address that is assigned to interface Fa0/1 of HQ-RT01.

```
# ip nat inside source static tcp 10.0.0.2 22 interface f0/1 22
```

Now, iftest telnet port 22 to the public IP configure on interface Fa0/1 of HQ-RT02 router from router PC, should get the following successful result.

```
# telnet 117.100.100.1 22
Trying 117.100.100.1, 22 ... Open
SSH-2.0-Cisco-1.25
```

To login to HQ-RT01 from the internet, can use the following commands.

```
# ssh -l netadmin 117.100.100.1

Password:

HQ-RT01>
```

5.2 Configure Static NAT To IP In The Same Interface Subnet

Usually, there are many local services to publish to be accessible from the internet. In this case, can configure NAT the local IP to any available public IP within the same subnet of IP that assigned to the router "HQ-RT02" interface Fa0/1 which it is connect directly to the internet.

Now , can configure NAT as the following to NAT public IP 117.100.100.2 port 2323 to the private IP 10.0.0.2 port 23.

```
# ip nat inside source static tcp 10.0.0.2 23 117.100.100.2 2323
```

We can test telnet from PC router as the following.

```
# telnet 117.100.100.2 2323
Trying 117.100.100.2, 2323 ... Open


User Access Verification

Username: netadmin
Password:

HQ-RT01>
```

In some case, might want to NAT one private IP to one public IP for any services and can do as the following.

```
# no ip nat inside source static tcp 10.0.0.2 22 interface FastEthernet0/1 22
# no ip nat inside source static tcp 10.0.0.2 23 117.100.100.2 2323 extendable
# clear ip nat translation *

# ip nat inside source static 10.0.0.2 117.100.100.3
```

Now, ping IP 117.100.100.3 from PC router, should get the following result.

```
#ping 117.100.100.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 117.100.100.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 104/113/144 ms
```

If test SSH from PC router, should get the following result.

```
#ssh -l netadmin 117.100.100.3
Password:

HQ-RT01>
```

If test telnet from PC router, should get the following result.

```
#telnet 117.100.100.3
Trying 117.100.100.3 ... Open

User Access Verification
Username: netadmin
Password:

HQ-RT01>
```

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result:

Thus, the NAT was configured using three Cisco routers.

EX.NO: 06	Configuration of Routing Information protocol
DATE:	

The routing information protocol (RIP) is used in this experiment to understand the hop count as a routing metric to find the most suitable route between the source and destination network. Configure RIP across the network and set up end devices to communicate on the network by enabling and verifying RIP commands. Create a routing table consisting of the following parameters: device name, IP address, subnet mask, and default gateway. Assign RIP route to a particular router and verify the network by pinging the IP address of any PC.

Aim:
To understand the concept and operation of Routing Information Protocol (RIP)

Requirements

- Windows pc – 2 Nos
- CISCO Packet Tracer Software (Student Version)
- 8 port switch – 2 No
- Router – 2 Nos
- Cat-5 LAN cable

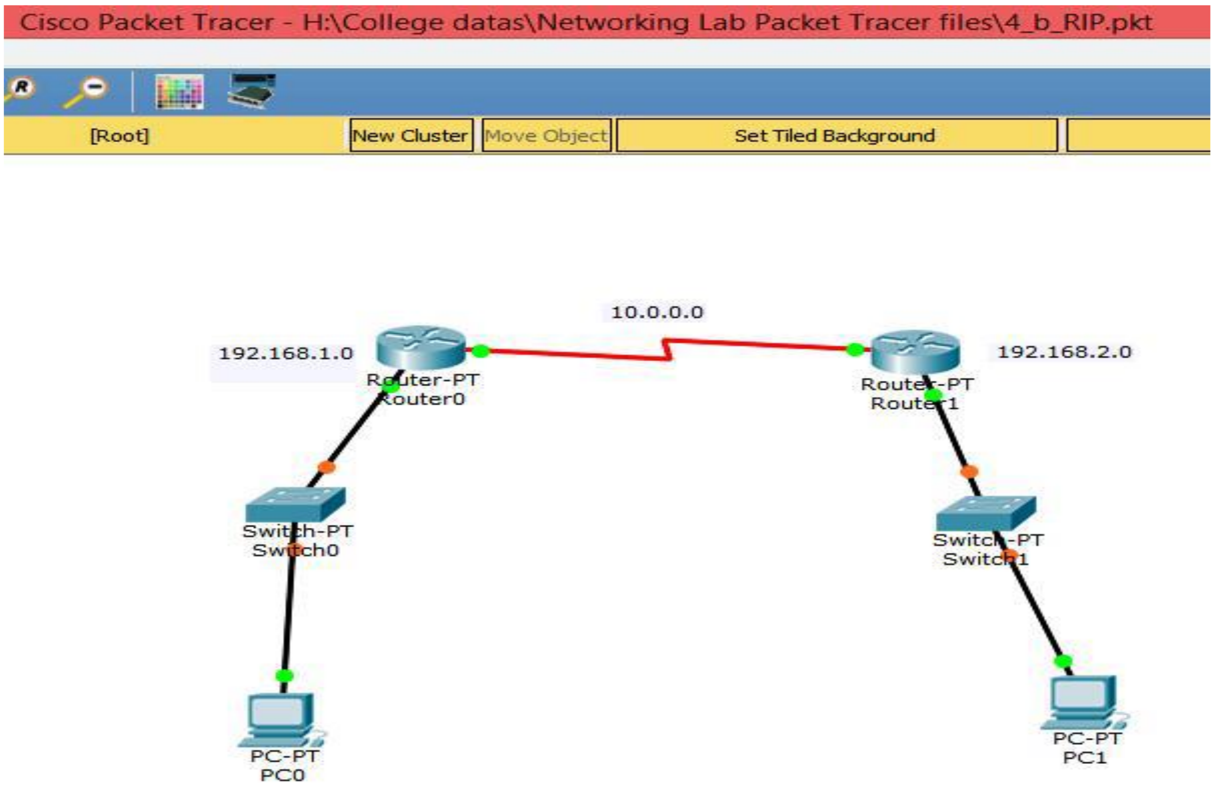
Procedure

- Open the CISCO Packet tracer software
- Drag and drop 5 pcs using End Device Icons on the left corner
- Select 8 port switch from switch icon list in the left bottom corner
- Select Routers and Give the IP address for serial ports of router and apply clock rate as per the table.
- Make the connections using Straight through Ethernet cables
- Ping between PCs and observe the transfer of data packets in real and simulation mode.

Theory

RIP (Routing Information Protocol) is one of the oldest distance vector routing protocols. It is usually used on small networks because it is very simple to configure and maintain, but lacks some advanced features of routing protocols like OSPF or EIGRP. Two versions of the protocol exists: version 1 and version 2. Both versions use hop count as a metric and have the administrative distance of 120. RIP version 2 is capable of advertising subnet masks and uses multicast to send routing updates, while version 1 doesn't advertise subnet masks and uses broadcast for updates. Version 2 is backwards compatible with version 1. RIPv2 sends the entire routing table every 30 seconds, which can consume a lot of bandwidth. RIPv2 uses multicast address of 224.0.0.9 to send routing updates, supports authentication and triggered updates (updates that are sent when a change in the network occurs).

Network Topology Diagram for RIP



Input Details for RIP

PC0

IP Address :

192.168.1.2

Gate way :

192.168.1.1

PC1

IP Address:

192.168.2.2

Gate way :

192.168.2.1

Router 0

Fast Ethernet 0/0

IP Address:

192.168.1.1

Serial 2/0 :

10.0.0.1 at 6400

clock rate
Router 1
Fast Ethernet 0/0
IP Address :
192.168.2.1
Serial 2/0 :
10.0.0.2 no clock rate

OUTPUT:
RIP (PINGING FROM PC0 TO PC1):

```
C:\>ping 192.168.2.2
Pinging 192.168.2.2 with 32 bytes of data:
Reply from 192.168.2.2: bytes=32 time=11ms TTL=126
Reply from 192.168.2.2: bytes=32 time=12ms TTL=126
Reply from 192.168.2.2: bytes=32 time=13ms TTL=126
Reply from 192.168.2.2: bytes=32 time=11ms TTL=126
Ping statistics for 192.168.2.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 11ms, Maximum = 13ms, Average = 11ms
```

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result:
Thus, understand the concept and operation of RIP and pinged from PC in are networks to PC to another network.

EX.NO: 07	**Client Server Model using UDP**
DATE:	

The client-server communication is studied in this experiment using socket programming to understand the UDP protocol. The server program and client program is executed separately. Initially, the UDP socket is created at the server and client sides. The binding is carried with the server address. Ensure that the client initiates the communication. Check the response of the client from the server side. Process the datagram packet and send a reply to the client. Observer that the data transfer between Client and Server occurred.

Aim:

To write a socket programming to establish a connection between client and server using UDP protocol.

Algorithm

Server-Side Algorithm (Java):

1. Initialize the server-side:
   - Create a DatagramSocket to listen for incoming datagrams on a specific port.
2. Set up a DatagramPacket for receiving data:
   - Create a byte array to hold the received data.
   - Create a DatagramPacket with the byte array to receive data.
3. Receive data from the client:
   - Use the DatagramSocket's receive method to receive data into the DatagramPacket.
   - Extract the data from the DatagramPacket's byte array.
4. Process the received data:
   - Process the received data as needed (e.g., modify or analyze it).
5. Prepare a response message:
   - Create a byte array to hold the response message.
   - Pack the response message into the byte array.
6. Send a response to the client:
   - Create a DatagramPacket with the response message and the client's address and port.
   - Use the DatagramSocket's send method to send the DatagramPacket to the client.
7. Close resources (optional):
   - The UDP protocol does not require explicit closing of resources like TCP. You can choose to close the DatagramSocket if necessary.

Client-Side Algorithm (Java):

1. Initialize the client-side:
    - Create a DatagramSocket to send and receive datagrams.
2. Prepare a message to be sent:
    - Create a byte array to hold the message to be sent.
    - Pack the message into the byte array.
3. Send the message to the server:
    - Create a DatagramPacket with the message and the server's address and port.
    - Use the DatagramSocket's send method to send the DatagramPacket to the server.
4. Set up a DatagramPacket for receiving data:
    - Create a byte array to hold the received data.
    - Create a DatagramPacket with the byte array to receive data.
5. Receive a response from the server:
    - Use the DatagramSocket's receive method to receive data into the DatagramPacket.
    - Extract the data from the DatagramPacket's byte array.
6. Process the received data:
    - Process the received data as needed (e.g., modify or analyze it).
7. Close resources (optional):
    - The UDP protocol does not require explicit closing of resources like TCP. You can choose to close the DatagramSocket if necessary.

Program

UDP Server

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

public class UDPServer {
    public static void main(String[] args) {
        DatagramSocket serverSocket = null;
        try {
            // Create a UDP socket
            serverSocket = new DatagramSocket(9876);

            byte[] receiveData = new byte[1024];

            System.out.println("Server is waiting for data...");
            while (true) {
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                serverSocket.receive(receivePacket);

                String clientMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                InetAddress clientAddress = receivePacket.getAddress();
                int clientPort = receivePacket.getPort();

                System.out.println("Received from client at " + clientAddress + ":" + clientPort + ":
" + clientMessage);
                // Process the received data (you can add your logic here)
                // Send a reply back to the client
                String serverReply = "Hello from the server!";
                byte[] sendData = serverReply.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
clientAddress, clientPort);
                serverSocket.send(sendPacket);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (serverSocket != null && !serverSocket.isClosed()) {
                serverSocket.close();
            }
        }
    }
}
```

UDP Client

```
import java.io.*;
import java.net.*;
public class UDPClient {
    public static void main(String[] args)
{
        DatagramSocket clientSocket = null;
     try {
            // Create a UDP socket
            clientSocket = new DatagramSocket();
            InetAddress serverAddress = InetAddress.getByName("127.0.0.1");
            int serverPort = 9876;
            BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
            while (true) {
                System.out.print("Enter a message to send to the server (or 'exit' to quit): ");
                String message = userInput.readLine();
                if (message.equals("exit")) {
                    break;
                }
                byte[] sendData = message.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverAddress, serverPort);
                clientSocket.send(sendPacket);
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                clientSocket.receive(receivePacket);
                String serverReply = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                System.out.println("Received from server: " + serverReply);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (clientSocket != null && !clientSocket.isClosed()) {
                clientSocket.close();
            }
        }
    }
}
```

Output:

Server:

```
C:\Program Files\Java\jdk1.8.0_162\bin>javac UDPServer.java
C:\Program Files\Java\jdk1.8.0_162\bin>java UDPServer
Server is waiting for data...
Received from client at /127.0.0.1:55140: Hello
Received from client at /127.0.0.1:55140: Welcome to Networks Lab
```

Client:

```
C:\Program Files\Java\jdk1.8.0_162\bin>javac UDPClient.java
C:\Program Files\Java\jdk1.8.0_162\bin>java UDPClient
```

```
Enter a message to send to the server (or 'exit' to quit): Hello
Received from server: Hello from the server!
Enter a message to send to the server (or 'exit' to quit): Welcome to Networks Lab
Received from server: Hello from the server!
Enter a message to send to the server (or 'exit' to quit): exit
C:\Program Files\Java\jdk1.8.0_162\bin>
```

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result

Thus a socket programming to establish a connection between client and server using UDP protocol was executed successfully.

EX.NO: 08	**Client Server Model using TCP**
DATE:	

The client-server communication is studied in this experiment using socket programming to understand the TCP protocol. The server program and client program are executed separately. Initially, the TCP socket is created on the server and client sides. The binding is carried with the server address. Ensure that the client initiates the communication. Check the response of the client from the server side. Process the packet and send a reply to the client. Observer and ensure that the data transfers between Client and Server are reliable.

Aim:

To write a socket programming to establish a connection between client and server using TCP protocol.

Algorithm

Server-Side Algorithm (Java):

1. Initialize the server-side:
   - Create a ServerSocket to listen for incoming connections on a specific port.
   - Wait for a client to connect.
2. Accept the client connection:
   - When a client connection is accepted, obtain a Socket object to communicate with the client.
   - Display a message indicating that the client has connected.
3. Set up input and output streams:
   - Create BufferedReader and PrintWriter objects to read from and write to the client's socket.
4. Receive data from the client:
   - Read data from the client using the BufferedReader.
   - Process the received data as needed (e.g., modify or analyze it).
5. Send a response to the client:
   - Prepare a response message.
   - Send the response message to the client using the PrintWriter.
6. Close resources:
   - Close the input and output streams.
   - Close the client's socket.
   - Close the ServerSocket.

Client-Side Algorithm (Java):

1. Initialize the client-side:
    - Create a Socket object to connect to the server's IP address and port.
2. Set up input and output streams:
    - Create PrintWriter and BufferedReader objects to write to and read from the server's socket.
3. Send data to the server:
    - Prepare a message to be sent to the server.
    - Send the message to the server using the PrintWriter.
4. Receive a response from the server:
    - Read the server's response using the BufferedReader.
    - Process the received response as needed.
5. Close resources:
    - Close the input and output streams.
    - Close the client's socket.

Program

TCP Server

```
import java.io.*;
import java.net.*;
public class TCPServer {
    public static void main(String[] args) {
        int port = 8080; // Port number to listen on

        try {
            // Create a server socket
            ServerSocket serverSocket = new ServerSocket(port);
            System.out.println("Server is listening on port " + port);

            // Wait for a client to connect
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected from " +
clientSocket.getInetAddress().getHostAddress());

            // Set up input and output streams for communication
            BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

            // Read data from the client
            String clientMessage = in.readLine();
            System.out.println("Received from client: " + clientMessage);
            // Process the received data (You can implement your own logic here)
            // For example, you can modify the message or perform some calculations
            String responseMessage = "Hello, Client!"; // Change this message as needed

            // Send a reply back to the client
            out.println(responseMessage);
            System.out.println("Sent to client: " + responseMessage);

            // Close the sockets and streams
            in.close();
            out.close();
            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TCP Client

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

public class TCPClient {
    public static void main(String[] args) {
        String serverAddress = "127.0.0.1"; // Server's IP address
        int serverPort = 8080; // Server's port number

        try {
            // Create a socket and connect to the server
            Socket socket = new Socket(serverAddress, serverPort);
            System.out.println("Connected to the server.");

            // Set up input and output streams for communication
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

            // Send a message to the server
            String message = "Hello, Server!"; // Message to send to the server
            out.println(message);
            System.out.println("Sent to server: " + message);

            // Read the server's response
            String serverResponse = in.readLine();
            System.out.println("Received from server: " + serverResponse);

            // Close the socket and streams
            out.close();
            in.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Server:

```
C:\Program Files\Java\jdk1.8.0_162\bin>javac TCPServer.java

C:\Program Files\Java\jdk1.8.0_162\bin>java TCPServer
Server is listening on port 8080
Client connected from 127.0.0.1
Received from client: Hello, Server!
Sent to client: Hello, Client!

C:\Program Files\Java\jdk1.8.0_162\bin>
```

Client:

```
C:\Program Files\Java\jdk1.8.0_162\bin>javac TCPClient.java

C:\Program Files\Java\jdk1.8.0_162\bin>java TCPClient
Connected to the server.
Sent to server: Hello, Server!
Received from server: Hello, Client!

C:\Program Files\Java\jdk1.8.0_162\bin>
```

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result

Thus a socket programming to establish a connection between client and server using TCP protocol was executed successfully.

EX.NO: 09	**File Transfer Protocol**
DATE:	

The active and passive File Transfer Protocol is studied in this experiment to understand the basic communication architecture between client and server. The server and client program with the following IP address should be used

Server: 127.0.0.1

Client: 192.168.x.x

Active FTP: Write the client and server program and ensure that client initiates a session via a command channel request and the server creates a data connection back to the client and begins transferring data. Use Wireshark to snip the data packets. Experiment and observer the above by enabling and disabling the local firewall.

Passive FTP: Write the client and server program and ensure that server uses the command channel to send the client information to open the data channel and ensure that the transfer has begun. Use Wireshark to snip the data packets. Experiment and observer the above by enabling and disabling the local firewall.

Create an FTP Server

Let's start by creating the FTP server using Apache FtpServer:

Download the Apache FtpServer library from the Apache FtpServer website (https://mina.apache.org/ftpserver-project/).

Create a directory for your FTP server project.

Inside this directory, create a configuration file for the server. For example, ftpd-typical.xml. Customize it according to your needs.

Create a user properties file (e.g., ftp-users.properties) to define users and their permissions. Customize this file accordingly.

Create the FTP Server Code

Create a Java class to set up and start the FTP server

```
import org.apache.ftpserver.FtpServer;
import org.apache.ftpserver.FtpServerFactory;
import org.apache.ftpserver.listener.ListenerFactory;
import org.apache.ftpserver.usermanager.PropertiesUserManagerFactory;
import java.io.File;

public class FTPServerExample {
    public static void main(String[] args) {
        FtpServerFactory serverFactory = new FtpServerFactory();
        ListenerFactory factory = new ListenerFactory();
        factory.setPort(21);

        serverFactory.addListener("default", factory.createListener());

        PropertiesUserManagerFactory userManagerFactory = new
PropertiesUserManagerFactory();
        userManagerFactory.setFile(new File("ftp-users.properties"));

        serverFactory.setUserManager(userManagerFactory.createUserManager());

        FtpServer server = serverFactory.createServer();

        try {
            server.start();
            System.out.println("FTP server started.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Create an FTP Client

For the FTP client, use Apache Commons Net. Ensure that you've added the library to your classpath.

Create the FTP Client Code

Create a Java class to set up and use the FTP client

```
import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import java.io.FileOutputStream;
import java.io.IOException;
```

```java
public class FTPClientExample {
    public static void main(String[] args) {
        String server = "ftp.example.com";
        int port = 21;
        String username = "your-username";
        String password = "your-password";
        String remoteFile = "/path/to/remote/file.txt";
        String localFile = "path/to/local/file.txt";

        FTPClient ftpClient = new FTPClient();

        try {
            ftpClient.connect(server, port);
            ftpClient.login(username, password);
            ftpClient.enterLocalPassiveMode();
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);

            FileOutputStream localFileStream = new FileOutputStream(localFile);
            boolean downloaded = ftpClient.retrieveFile(remoteFile, localFileStream);

            if (downloaded) {
                System.out.println("File downloaded successfully.");
            } else {
                System.out.println("File download failed.");
            }

            localFileStream.close();
            ftpClient.logout();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (ftpClient.isConnected()) {
                    ftpClient.disconnect();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Run the FTP Server and Client

Compile and run the FTP server program first.
Compile and run the FTP client program to connect to the server and transfer files.

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result:
Thus the Socket program to perform file transfer was executed successfully.

EX.NO: 10	Dijkstra’s Algorithm
DATE:	

Aim:

To perform Dijkstra's algorithm to find the shortest path in a given graph using Java.

Procedure:

Step 1: Represent the Graph

First, represent the graph. You can use an adjacency matrix to represent the graph, where each cell graph[i][j] represents the weight (or distance) between node i and node j. If there is no edge between the nodes, set the value to infinity or a very large number.

Step 2: Initialize Data Structures

Initialize data structures to keep track of distances and visited nodes:

- Create an array distance[] to store the shortest distance from the source node to all other nodes. Initialize it with infinity except for the source node, which should be set to 0.
- Create a Boolean array visited[] to keep track of visited nodes. Initialize all values to false.

Step 3: Implement Dijkstra's Algorithm

- Start a loop to iterate through all nodes:
- Find the node with the minimum distance from the source node. This node is your current node (let's call it u).
- Mark u as visited by setting visited[u] to true.
- Update the distance to all neighboring nodes of u if a shorter path is found. The new distance will be the sum of the distance to u and the weight of the edge to the neighboring node.
- Continue this loop until all nodes have been visited.

Step 4: Print the Shortest Paths

After running Dijkstra's algorithm, you will have the shortest distance from the source node to all other nodes in the distance[] array. You can print these distances to display the shortest paths.

Program:

```
import java.util.Arrays;
```

```
public class DijkstraAlgorithm {
    public static void dijkstra(int[][] graph, int source) {
        int V = graph.length;
        int[] distance = new int[V];
        boolean[] visited = new boolean[V];

        Arrays.fill(distance, Integer.MAX_VALUE);
        distance[source] = 0;

        for (int count = 0; count < V - 1; count++) {
            int u = minDistance(distance, visited);

            visited[u] = true;

            for (int v = 0; v < V; v++) {
                if (!visited[v] && graph[u][v] != 0 && distance[u] != Integer.MAX_VALUE
                        && distance[u] + graph[u][v] < distance[v]) {
                    distance[v] = distance[u] + graph[u][v];
                }
            }
        }

        printSolution(distance);
    }

    private static int minDistance(int[] distance, boolean[] visited) {
        int V = distance.length;
        int min = Integer.MAX_VALUE, minIndex = -1;

        for (int v = 0; v < V; v++) {
            if (!visited[v] && distance[v] <= min) {
                min = distance[v];
                minIndex = v;
            }
        }
        return minIndex;
    }

    private static void printSolution(int[] distance) {
        System.out.println("Vertex \t Distance from Source");
        for (int i = 0; i < distance.length; i++) {
            System.out.println(i + " \t " + distance[i]);
        }
    }

    public static void main(String[] args) {
        int[][] graph = {
            {0, 4, 0, 0, 0, 0, 0, 8, 0},
            {4, 0, 8, 0, 0, 0, 0, 11, 0},
            {0, 8, 0, 7, 0, 4, 0, 0, 2},
            {0, 0, 7, 0, 9, 14, 0, 0, 0},
            {0, 0, 0, 9, 0, 10, 0, 0, 0},
            {0, 0, 4, 14, 10, 0, 2, 0, 0},
            {0, 0, 0, 0, 0, 2, 0, 1, 6},
            {8, 11, 0, 0, 0, 0, 1, 0, 7},
            {0, 0, 2, 0, 0, 0, 6, 7, 0}
        };

        dijkstra(graph, 0);
    }
}
```

Output:

```
C:\Program Files\Java\jdk1.8.0_162\bin>javac DijktraAlgorithm.java

C:\Program Files\Java\jdk1.8.0_162\bin>java DijktraAlgorithm
Vertex   Distance from Source
0        0
1        4
2        12
3        19
4        21
5        11
6        9
7        8
8        14

C:\Program Files\Java\jdk1.8.0_162\bin>
```

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result:
Thus the java program to perform Dijkstra's algorithm to find the shortest path for the given graph was executed successfully.

EX.NO: 11	**Checksum Calculation**
DATE:	

Aim:

To write a Java Program to compute checksum for the given string.

Program:

```
import java.util.Scanner;

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

        // Ask the user for input
        System.out.print("Enter a string: ");
        String userInput = scanner.nextLine();

        // Calculate the checksum
        int checksum = calculateChecksum(userInput.getBytes());

        // Display the checksum
        System.out.println("Checksum: " + checksum);

        scanner.close();
    }

    public static int calculateChecksum(byte[] data) {
        int sum = 0;

        for (int i = 0; i < data.length; i++) {
            sum += data[i];
        }

        return sum;
    }
}
```

Output:

```
C:\Program Files\Java\jdk1.8.0_162\bin>javac UIC.java

C:\Program Files\Java\jdk1.8.0_162\bin>java UIC
```

```
Enter a string: Hello
Checksum: 500

C:\Program Files\Java\jdk1.8.0_162\bin>java UIC
Enter a string: Hi
Checksum: 177
```

Department of CSE		
Preparation (Algorithm)	**4**	
Observation (Program)	**4**	
Results (Output)	**4**	
Interpretation (Validation)	**4**	
Viva - Voce	**4**	
Total	**20**	

Result:

Thus, the java program to compute checksum for the given string was executed successfully.