05.05.24 CN Lab.

Exp 4b - To write a program that performs DNS lookup.

Algorithm -

1. Start :
   - Begin the DNS lookup process.
2. Initialize variables :
   - Initialize the variable 'link' to "a".
3. Loop until 'end' is entered :
   - Enter a while loop that continues until the user inputs "end".
   - Prompt the user to enter a website name
   - Read the input & assign it to the variable 'link'.
   - If the input is "end", exit the loop.
4. Perform DNS lookup :
   - Within the loop :
   - Try to perform a DNS lookup for the entered website name ('link')
   - If successful, retrieve the IP address associated with the website using 'socket.gethostbyname(link)'
   - If an error occurs during the lookup ('socket.gaierror'), print an error message indicating that the website name could not be resolved.
5. Close :
   - After the loop ends (either by entering "end" or encountering an error), print "closed".
6. End :
   - End of the DNS lookup process.

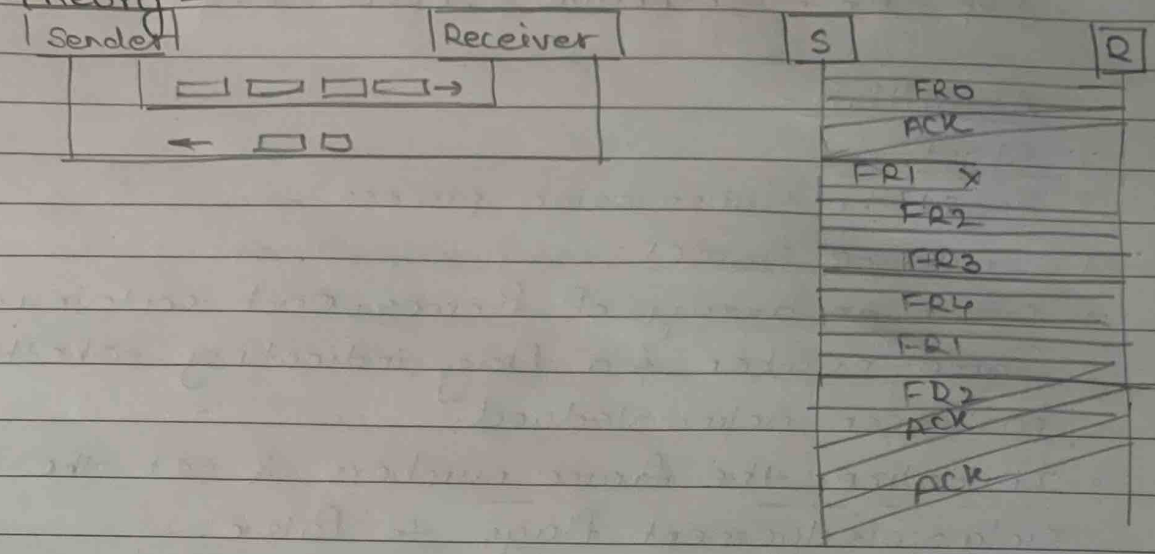
Exp 5a - To understand basics of error-recovery using window based control & simulate Go-Back-N protocol.

Algorithm -

1. Start:
   - Begin the transmission process.
2. Input Window size:
   - Prompt the user to enter the window size.
   - Read & store the window size.
3. Transmission loop:
   - Enter an infinite loop.
   - Initialize 'sent' to 1.
   - Repeat until 'sent' reaches window size
   - Print "Frame [sent] has been transmitted."
   - Increment 'sent' by 1.
4. Receive Acknowledgement:
   - Prompt user to enter the last acknowledgement received.
   - Read & store the Acknowledgement in 'ack'.
5. Check acknowledgement:
   - If the received acknowledgement equals the window size:
   - Exit the loop (transmission is complete)
   - Else
   - set 'sent' to Acknowledgement + 1 (to resume transmission to next frame)
6. End:
   - End of transmission process.

Rainbow

Theory -



- Go back N ARQ can send several frames before receiving an acknowledgement.
- We keep a copy of the frames until the acknowledgement arrives.
- The frames are numbered sequentially, Frame 0 is sent & after that FR1 to FR5 are sent.
- If FR1 is lost, all the frames from FR1 to FR5 are resent because the previous frames were discarded.

Exp - 5b - Selective - Repeat ARQ.

Algorithm -

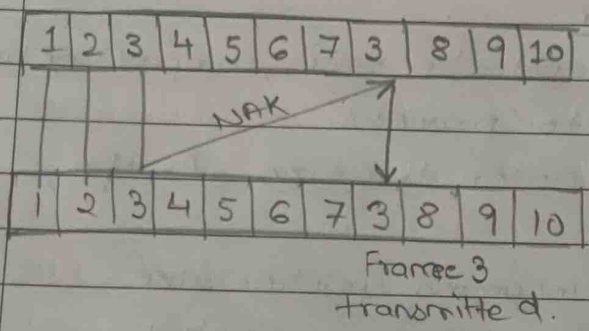
1. Start:
   - Begin the transmission process.
2. Initialize Frames:
   - Create an array of frames, each containing a frame number & a flag indicating whether it has been acknowledged.
   - Initialize the frame numbers & set the acknowledgement flags to false.
3. Set Initial Pointers:
   - Initialize variables 'next_to_send' to 0 (indicating the next frame to send) & 'last_acked' to -1 (indicating the last acknowledged frame).
4. Transmission Loop:
   - Enter an infinite loop:
   - Calculate the no. of frames to send based on the difference bet^n 'WINDOW_SIZE', 'next_to_stand', & 'last_acked'.
   - Send the frames within the window from 'next_to_send' to 'end'.
   - Update 'next_to_send' to index of next frame to send.
5. Receive Acks:
   - Prompt user to enter acknowledgement received.
   - If acknowledgement is within window size:
   - Mark corresponding frame as acknowledged.
   - Update 'last_acked' to the index of the last acknowledged frame
   - If the acknowledgement is invalid, display an error message

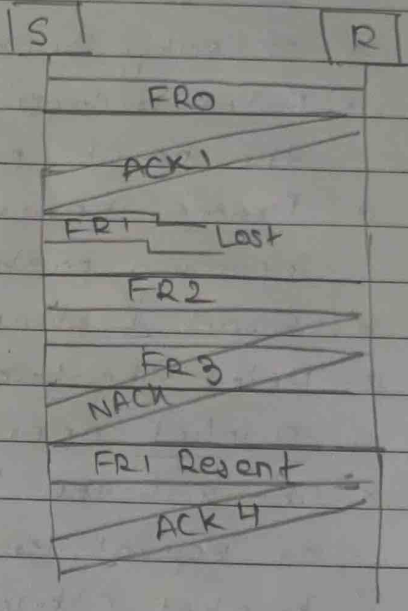
Rainbow

6. Check completion:
- Check if all frames within the window have been acknowledged.
- If yes, exit the loop.

7. End:
- print a message indicating that all the frames have been acknowledged successfully.
- End of transmission process.

Theory -





- Disadvantage of Go-back-N ARQ was it transmits frames continuously & the NAK signal takes some time to reach to the sender till that time the sender has already sent some frames.
- In selective Repeat ARQ, only the specific damaged or lost frame is retransmitted.

Rainbow

Exp-6a - Leaky Bucket Algorithm [Traffic Shaping Algorithm]

Algorithm -

1. Start:
   - Begin the leaky bucket algorithm.
2. Input parameters
   - Prompt the user to enter the bucket size, outgoing rate, & no. of inputs.
   - Read & store these values.
3. While Leaky Bucket loop:
   - Enter a loop that iterates 'n' times, representing the no. of packets to process.
   - Prompt the user to enter the size of the incoming packet.
   - Read & store the incoming packet size.
   - Print the size of the incoming packet.
4. Bucket operation:
   - Check if the incoming packet size is less than or equal to the space available in the bucket ('bucket_size - store')
   - If yes:
   - Add the incoming packet size to the current contents of the bucket ('store').
   - Print the updated bucket buffer size.
   - If no (i.e bucket is full):
   - calculate no. of packets to drop ('incoming - (bucket_size - store)').
   - Print the no. of packets dropped & the updated bucket buffer size (which remains at maximum capacity.)
   - Set the bucket contents to the maximum capacity ('bucket_size') since the bucket is full.

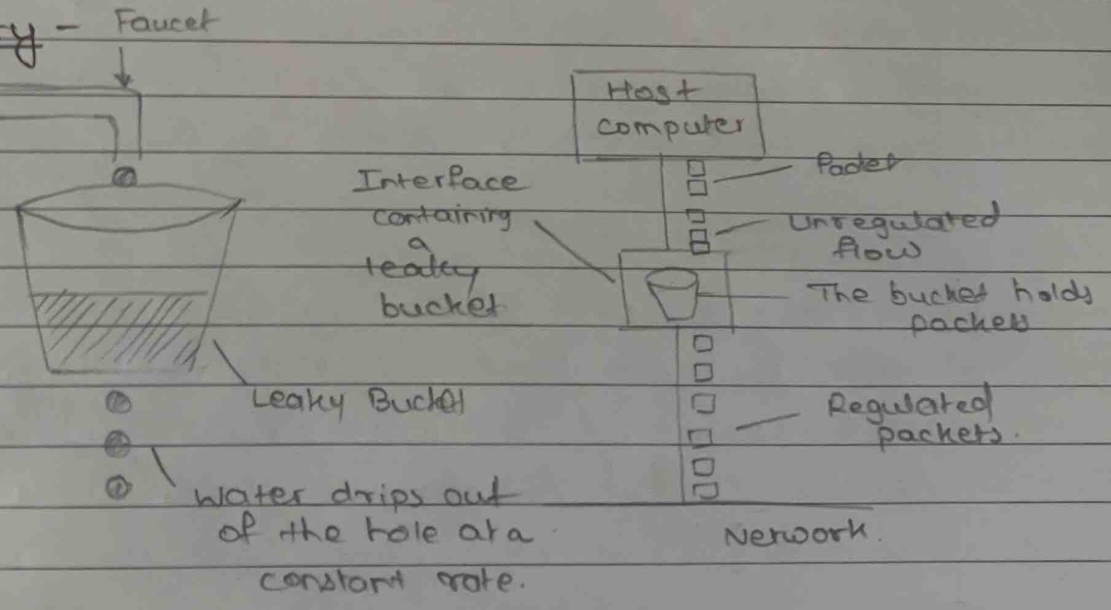
5. Outgoing Packets:
- Subtract the outgoing rate from the current contents of the bucket ('store')
- Ensure that the bucket contents don't go negative.
- Print the no. of packets remaining in the bucket after outgoing packets are removed.

6. Decrement Loop Counter:
- Decrement loop counter 'n'.

7. End loop:
- Repeat steps 3 to 6 uptill all packets have been processed.

8. End.
- End of leaky bucket algorithm.

Theory -



1. The way Leaky Bucket method works is by controlling the speed at which data packets are sent over a network.
2. It functions using the idea of a symbolic bucket with a finite capacity to hold data.

Rainbow

3. Packets are added to this container when they arrive.
4. But if bucket turns into when filled, extra packets are stored or thrown away.
5. It consistently leakes packets from the bucket, independent of the rate at which they are added.
6. By ensuring a constant & predictable packet flow, this helps to avoid unexpected spikes in network traffic that may cause congestion.

Exp 7a - Implementation of Dynamic routing protocols - Distance vector routing

Algorithm -

1. Start:
   - Begin the process of computing the shortest paths using the Distance vector Routing algorithm.
2. Input number of Nodes:
   - Prompt the user to enter the number of nodes in the network.
   - Read & store the value in variable 'n'.
3. Input cost matrix:
   - Prompt the user to enter the cost matrix representing the distance between nodes.
   - Read & store the values in the 2D array 'dmat'.
   - Initialize the diagonal elements of the cost matrix to 0.
4. Initialize Distance vectors:
   - For each node:
   - Initialize the distance vector ('rt[i].dist') with the corresponding row of the cost matrix.
   - ~~Initialize the diagonal elements of the cost matrix to 0.~~
4. ~~Initialize distance vectors~~:
   - ~~For each node~~:
   - ~~Initialize the distance vector~~
   - 5. Initialize the next-hop vector ('rt[i].from') with the node itself.
5. Compute shortest paths:
   - Enter a do-while loop that continues until no further updates are made to the distance vectors.
   - Reset the update counter 'count' to 0.

Rainbow

- For each node 'i':
- For each pair of nodes '(i,j)':
- For each intermediate node 'k':
- check if the dist. from 'i' to 'j' thr. 'k' is shorter than the current shortest distance from 'i' to 'j'.
- If yes. update the distance vector & next-hop vector for node 'i'.
- Increment the update counter 'count'.

6. Output shortest paths:
- For each node 'i':
- Print the state value for router 'i'.
- For each ~~router~~ destination node 'j':
- Print the distance from node 'i' to node 'j' via the next-hop node & the distance.

7. End
- End of the process of computing shortest paths using the distance vector routing algorithm.

Theory:

In ~~this~~ distance vector routing algorithm, routers construct accurate routing tables maintaining a detailed database of the network's current topology & state.

Exp-7b - Link State Algorithm - Dijkstra's algorithm

Algorithm -

1. Start.
   - Begin Dijkstra's algorithm.
2. Input parameters:
   - Prompt the user to enter the no. of vertices ('n') & the adjacent matrix representing the graph.
   - Read & store these values.
3. Initialize variables:
   - Create arrays to store the cost matrix ('cost'), distances from the start node ('distance'), predecessors of each node ('pred'), & visited status of each node ('visited')
   - Initialise variables 'count' to count the number of nodes visited so far, 'mindistance' to store the minimum distance, 'nextnode' to store the next node to visit, & loop counters.
4. Initialize cost matrix:
   - Initialize the cost matrix based on the adjacency matrix. If there's no direct edge betn nodes, set the dist. to infinity.
5. Initialize Distance vectors:
   - Initialize the distance vector ('distance') with the distances from the start node to all other nodes.
   - Initialize the predecessor vector ('pred') with the start node itself.
   - Mark the start node as visited.

6. Dijkstra's Algorithm loop:
- Enter a loop that continues until all nodes have been visited ('count < n-1').
- Find the node with min. distance that has not been visited ('nextnode')
- Mark 'nextnode' as visited.
- Update distances & predecessors for neighbouring nodes if a shorter path is found through 'next node'.

7. Output Shortest paths:
- For each node (excluding start node):
- Print the distance from the start node to current node ('distance[i]').
- Print shortest path from start node to the current node by tracing back through predecessors. ('pred').

8. End
- End of Dijkstra's algorithm.

Theory:
In link state routing algorithm, optimized routing tables are obtained by iteratively exchanging & updating information among routers in a network.

Exp- 4a - Analysis of Application Layer packet (DNS) using DNS commands.

1. nslookup:
Syntax: nslookup domain_name
Description: Retrieves DNS info, including IP addresses associated with a domain.

2. ipconfig:
a. Syntax: ipconfig /flushdns
Description: Flushes the DNS resolver cache on the local machine.

b. Syntax: ipconfig /displaydns.
Description: Displays contents of the DNS resolver cache, showing recently resolved domain names & their corresponding IP addresses.

c. Syntax: ipconfig /all.
Description: Displays detailed information about the network interfaces on the machine, including DNS server configurations.

3. netsh:
Syntax: netsh interface ipv4 show dnsservers
netsh interface ipv6 show dnsservers
Description: Displays the DNS server settings for all network interfaces.

4. sc query "dnscache".
Syntax: sc query "dnscache"
Description: Checks the status of the DNS client service.

5. sc stop "dnscache" / sc start "dnscache":
Syntax: sc stop "dnscache" or sc start "dnscache"
Description: Stops or starts the DNS client service.

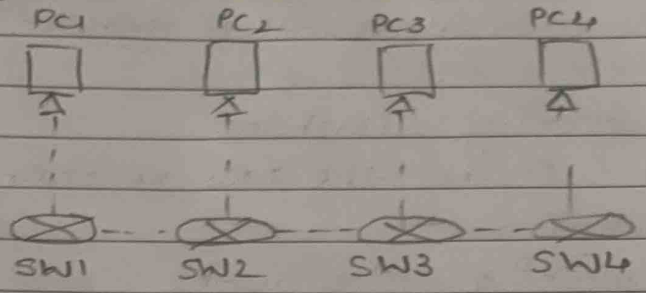
6. dnscmd:
Syntax: dnscmd /info.
Description: Displays info. about DNS server.

Rainbow

Exp 1c - Study & configure Bus, Mesh, Ring & Star network topologies.

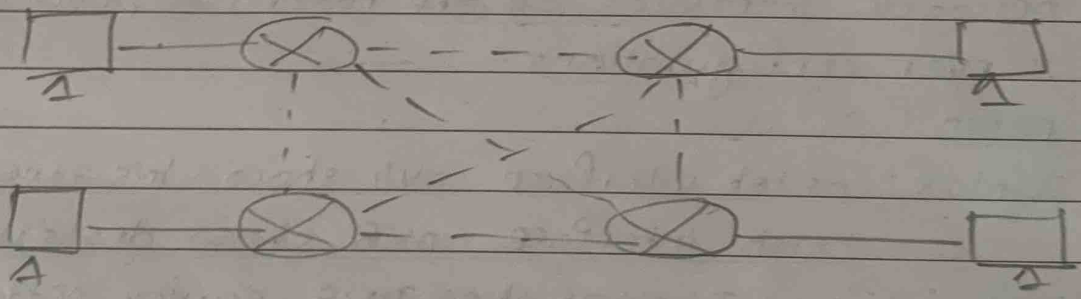
1. Bus Topology -

- It is multi-point where one cable acts as backbone to link all the devices in a network.



- Use less cabling compared to other topologies.
- A fault or break in the cable stops all the transmission.

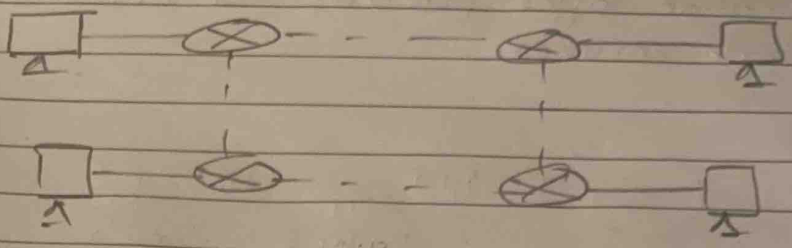
2. Mesh Topology -



- Every device has a dedicated point-to-point link to every other device.
- Because of point-to-point link fault identification is easy.
- Cabling req. is high.

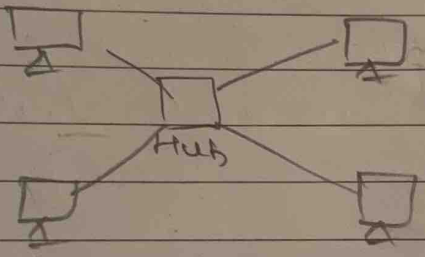
Rainbow

3. Ring Topology :-



- Each device has a dedicated point-to-point connection with only two devices on either side of it.
- Fault isolation easy.
- A break in the link can disable the entire network.

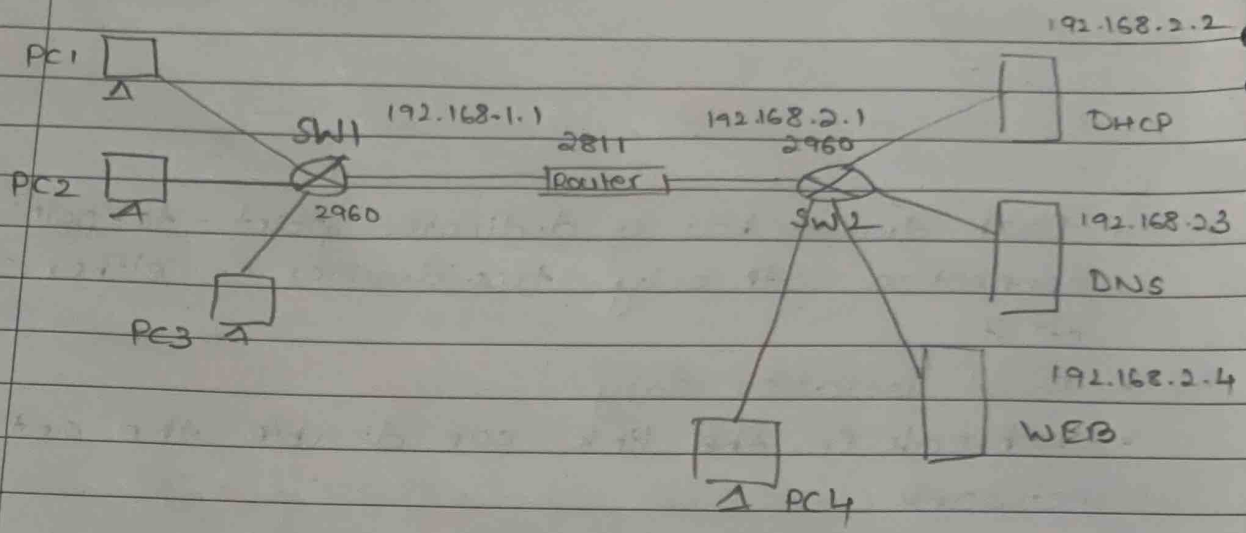
4. Star Topology



- Each device has dedicated point-to-point link only to a central hub (controller)
- If one device has to send data to another, it has to send it through the hub.
- Less cabling.
- Whole dependancy on a single hub ; if it goes down whole system is dead.

Rainbow

Exp-1-d - To build a basic network with application layer protocols for enhanced network services.



Components -

1. 2811 router.
2. 2960 switch x2.
3. Servers x3.
4. PC x4.

Procedure -

1. Configure DHCP server.

⇒ go to desktop → IP config

IP = 192.168.2.2.

D.G = 192.168.2.1.

DNS = 192.168.2.3.

2. Configure DNS server -

IP = 192.168.2.3

D.G = 192.168.2.1

DNS = 192.168.2.3.

Rainbow

3. Configure web server
IP = 192.168.2.4
DG = 192.168.2.1
DNS = 192.168.2.3.

- Services tab for DHCP: -
1. Create two pool names for two different n/w.
a. Server Pool.
- D.G - 192.168.2.1
- DNS - 192.168.2.3
- Start IP address 192 168 2 10
- Max users = 50
- Save.

b. Server Pool1
- D.G - 192.168.1.1.
- DNS - 192.168.2.3
- Start IP address 192 168 1 10
- Max users = 50
- ~~Save~~ Add
- Turn DHCP 'ON'.

- Services tab for DNS
1. Give domain name - aaryan.com
2. Give address of web server - 192.168.2.4
3. add.
4. turn on.

- Services tab for HTTP:
1. Both HTTP & HTTPS should be on.
2. change/edit the index as per choice.
3. Save.

Rainbow

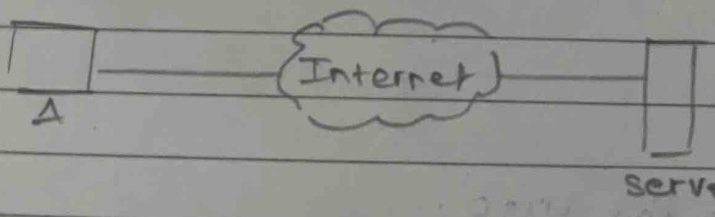
4. Click on the PC in the network along with the servers, Desktop - ipconfig - select DHCP.

5. Router
   - config router f 0/0
     IP address 192.168.1.1
     Turn on.
   - config router f 0/1.
     IP address 192.168.2.1.
     Turn on.

6. CLI
   int fa 0/0 [left side network]
   ip helper-address 192.168.2.2. (DHCP server IP)
7. Turn DHCP 'ON' for all 3 PC's on left side n/w.

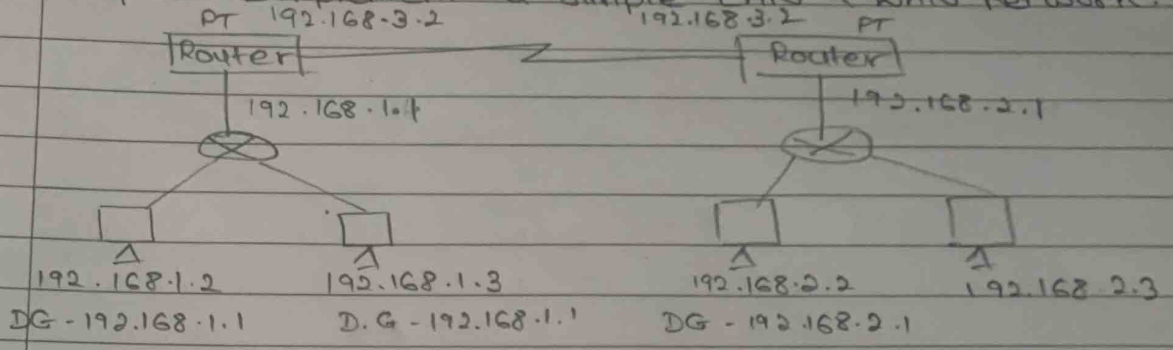
Theory:-

1. Every computer on a network is called a host or end device. Servers are computers that provide information to end devices.
2. Clients are computers that sends requests to the servers to retrieve info.
   - webpage from a web server.
   - email from an email server.



- Email server : Runs email server software. Clients use client software to access email.
- Web server : Runs web server software. Clients use browser software to access webpages.
- File server : Stores corporate & user files. The client devices access these files.

Exp Id - Implement a simple LAN & WAN network.



Components -

1. Router PT x 2. 2. Switch PT x 2 3. PC x6.

Procedure

1. Configure the 2 LAN networks separately with the given IP's & D.G.
2. Configure both routers's fast ethernet port 0/0 with the respective D.G. Turn on.
3. Connect both routers wing serial DTE.
4. Configure serial port configuration:

- R1 - 192.168.3.2 R2 - 192.168.3.3
- Then, go to static.

network - 192.168.2.0	n/w - 192.168.1.0
Subnet mask - 255.255.255.0	Subnet mask - 255.255.255.0
Next hop - 192.168.3.3	Next hop - 192.168.3.2

- Add.