EXPLORE EXPERIENCE EMPOWER

P P SAVANI UNIVERSITY

School of **Engineering**

LABORATORY MANUAL

Department of Computer Science and Application

COMPUTER NETWORKS

(SSCS2010) | **B.Sc.IT**

Name of Student: **PUROHIT PARTHKUMAR ANILBHAI**

Enrollment No.: **23SS02IT157**

Academic Year: 2024-25

P P Savani School of Engineering

CERTIFICATE

This is to certify that

Mr./ Ms. **PUROHIT PARTHKUMAR ANILBHAI**

of Computer Science and Application having Enrollment No. **23SS02IT157**

has completed

his/her Term work in the subject of

COMPUTER NETWORKS

(SSCS2010).

Marks Obtained:_____out of _____

Sign of Faculty

Date:_____

Sign of Head of Department

Date: _____

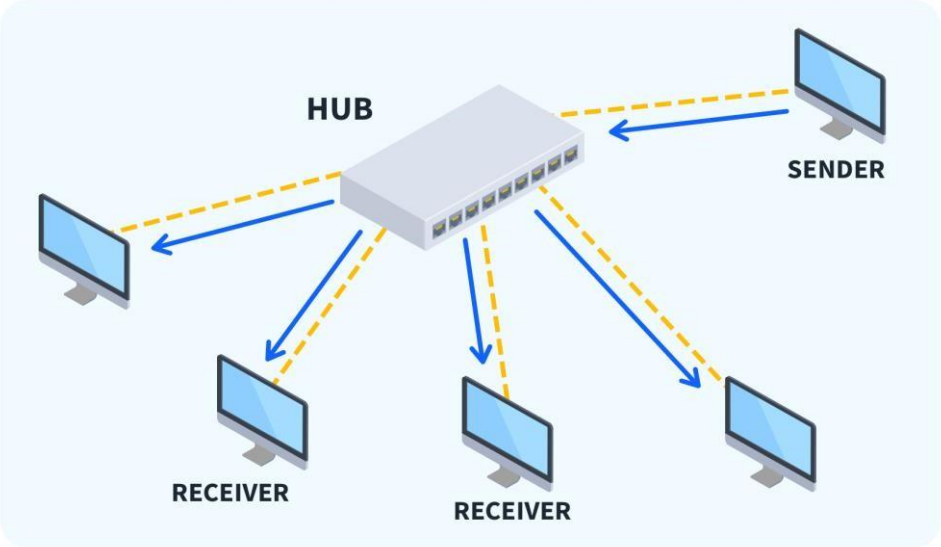
CONTENTS

Practical – 1

Aim: Study of different network devices in detail. (Repeater, Hubs, Switches, Bridge, Routers and Gateway).

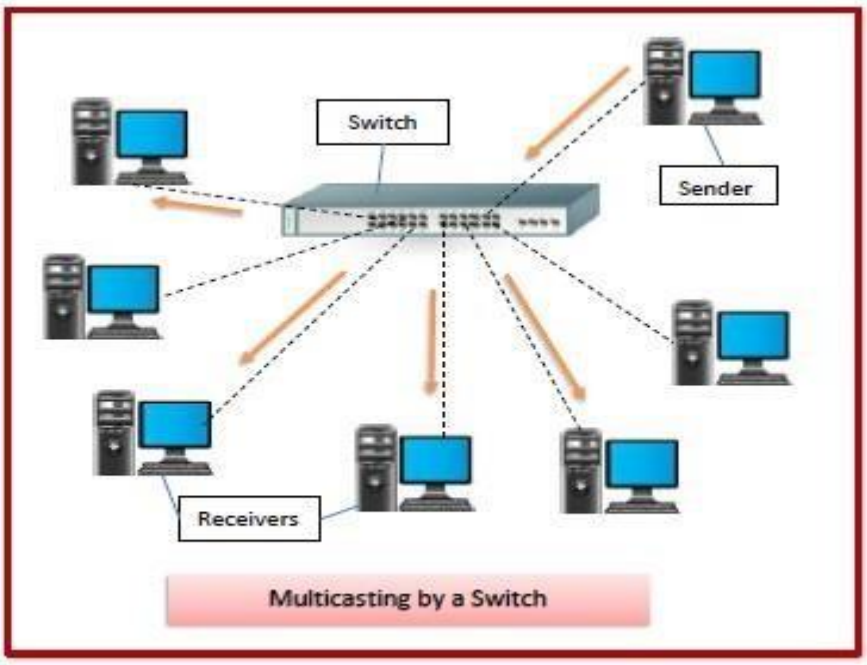
1. Hubs:

- Hub is a network device that is used to connect multiple computers in a network.
- All the information send to the hub is automatically send to each port to every device.
- Hubs can also be referred to as repeaters or concentrators, and they serve as the canter of a local area network (LAN).
- Hub generally used to connect computers in a LAN.
- Transmission mode of hub is half duplex.



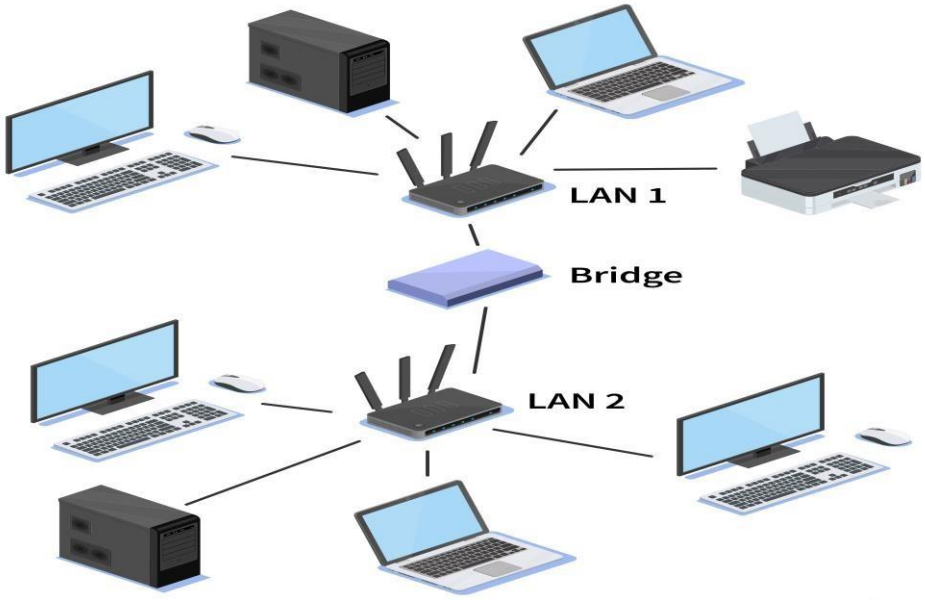
2. Switches:

Switch is a network device that connects multiple computer together in the network.
It is mainly used to send the private message as well as there is no wasting of data.
Switch can easily identifie that which device is connected with which port by using MAC address, that's why it delivered message on particular destination machine.
Switches refer to the networking devices that operate at an OSI model's layer 2 or data link layer.
They establish connections between networked devices and employ packet switching to transmit, receive, or forward data packets or frames over the network.



3. Bridge:

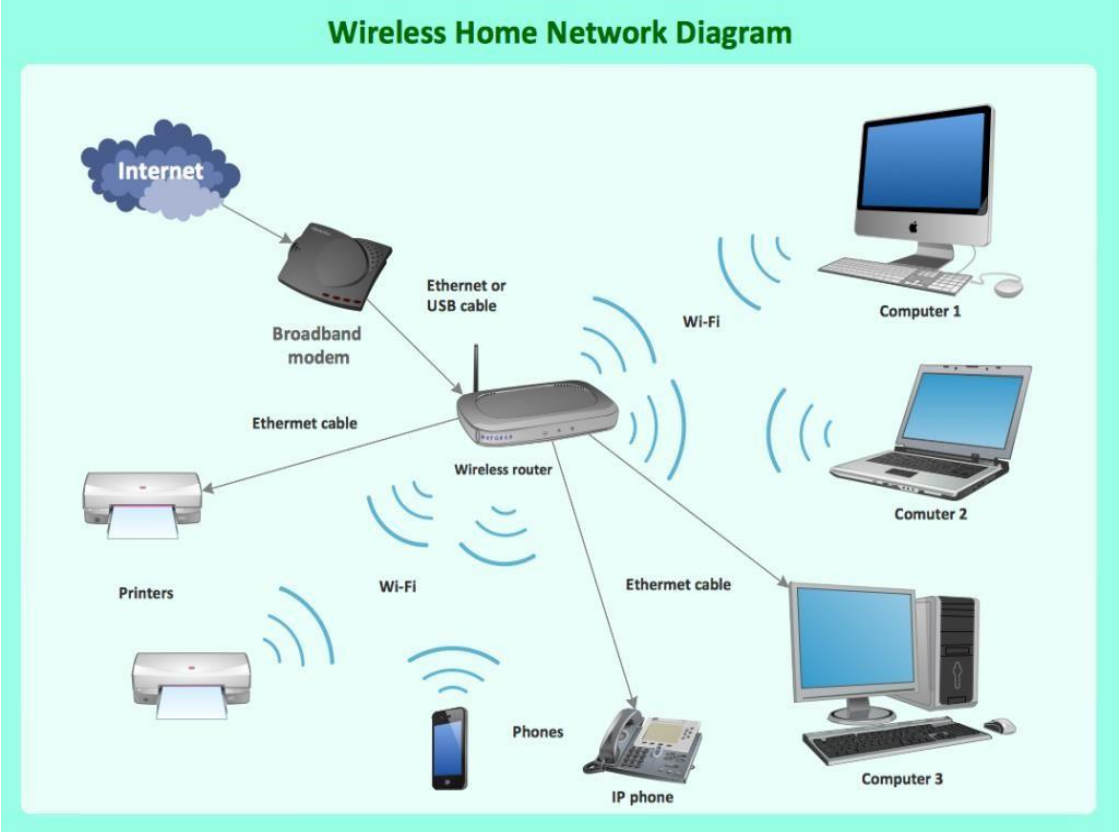
Bridge is a network devices that is used to separate LAN into no of section.
A bridge is a network device that connects multiple subnetworks to create a single network.
It provides interconnection with other computer networks that use the same protocol.
Through a bridge, multiple LANs can be connected to form a larger and extended LAN.



4. Routers:

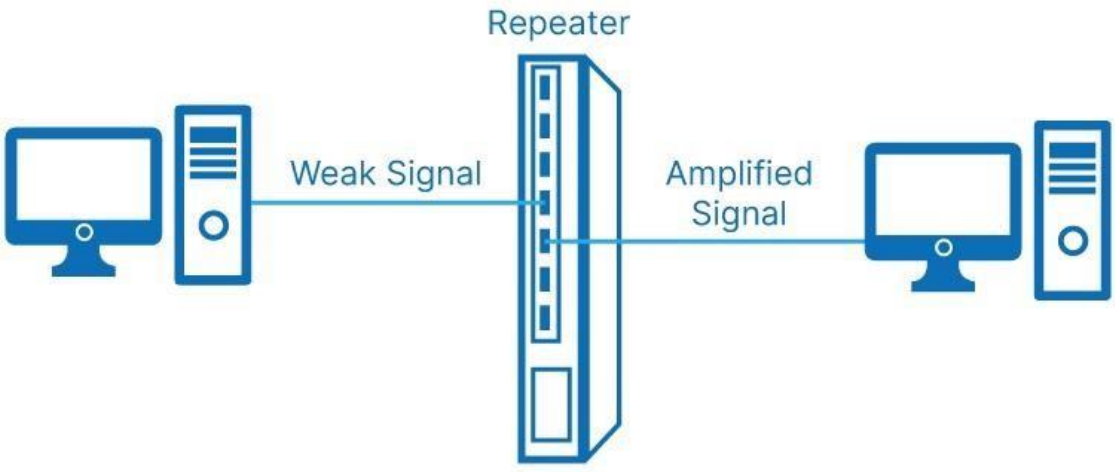
Router is a network device which works as a traffic controller.
A man wrok of router is to choose a congestion free path through which the data packet will travel.
router receive data packets to the sender analyse and forward those data packets then giving to receiver.
A router is a device that connects two or more packet-switched networks or subnetworks.

It serves two primary functions: managing traffic between these networks by forwarding data packets to their intended IP addresses, and allowing multiple devices to use the same Internet connection.



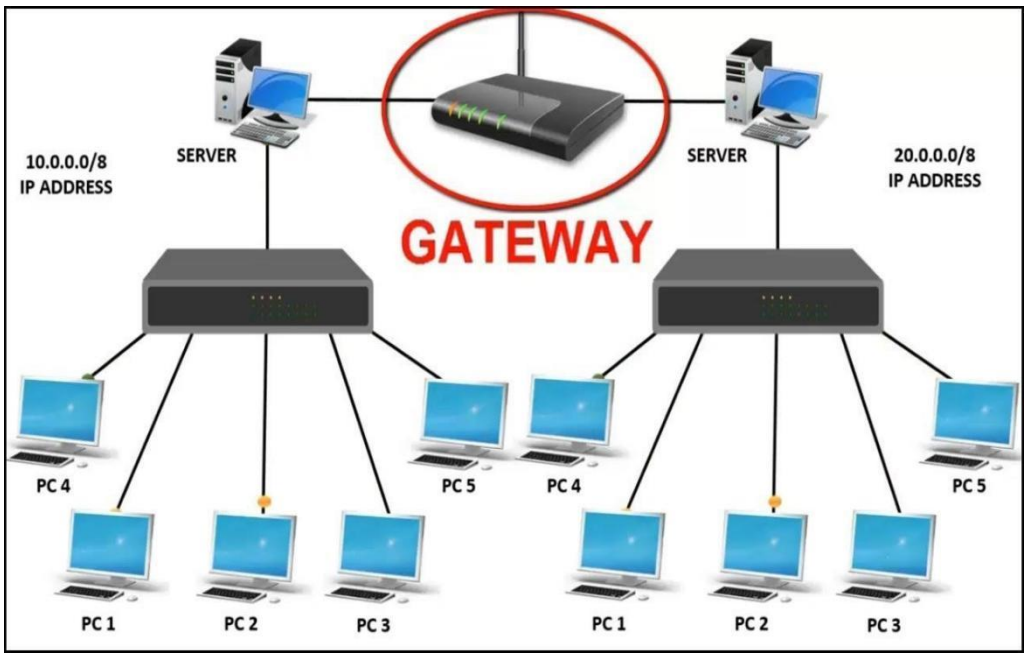
5. Repeater:

Repeater is a network device through which we can "boostup the weak signals".
When the single travels in the network after travelling some distance the intensity of the signal becomes low.
In order to regenerate the weak signal we should use repeater device.
Repeater is a type of network node that amplifies incoming signals and rebroadcasts them over a wider area or higher network layer than the original signal.
In computer networks, a repeater is used to increase the network's reach, restore a damaged or weak signal, or provide access to inaccessible nodes.



6. Gateway:

Gateway is a hardwave device that is used to connect two dissimilar type of network.
It allow us to send & receive data through the internet event it is LAN network.
A computer that sits between different networks or applications. The gateway converts information, data or other communications from one protocol or format to another.
A router may perform some of the functions of a gateway.

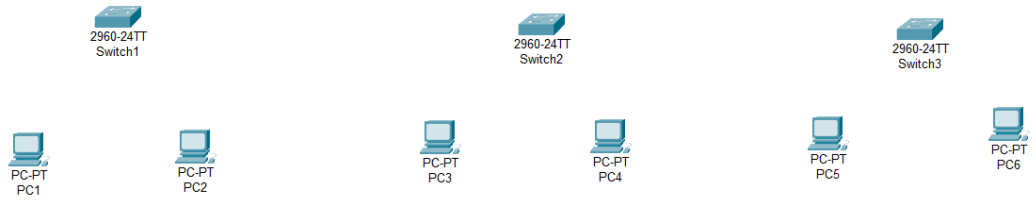


Practical – 1.1

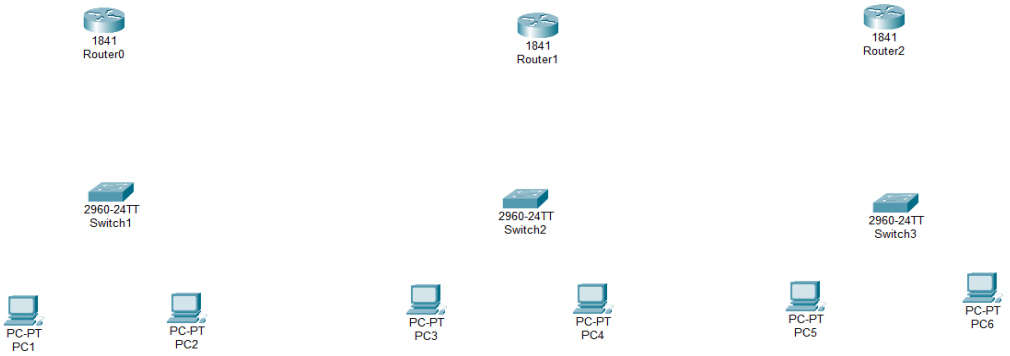
Aim:Implement static routing in cisco packet tracer.

Step 1: Select Six pc.

Step 2: Select Three Switches (No. Of switch 2960).

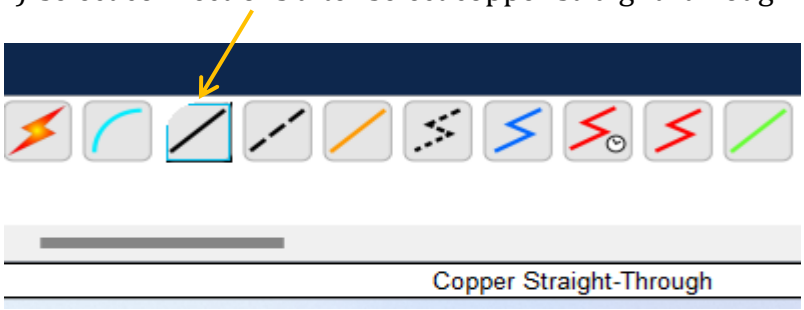


Step 3: select Three Router (no. Of router is 1841).

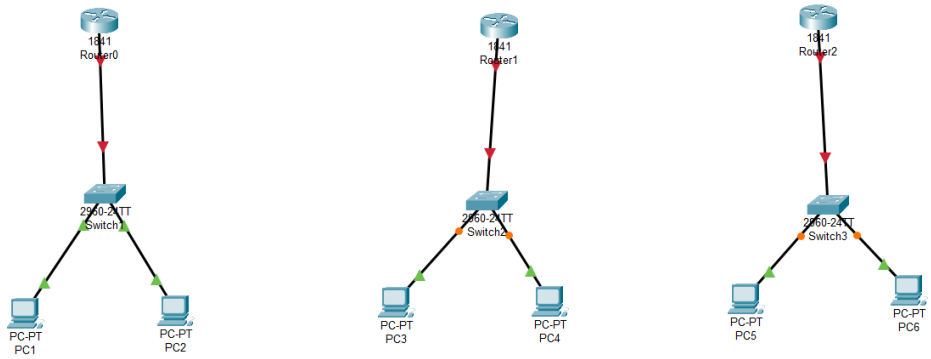


Step 4: The most common type of cable used to connect a PC and a switch is a straight-through Ethernet cable. Sama as straight-through used to connect switch and Router.(Below image to select)

1) select connections after select copper straight -through

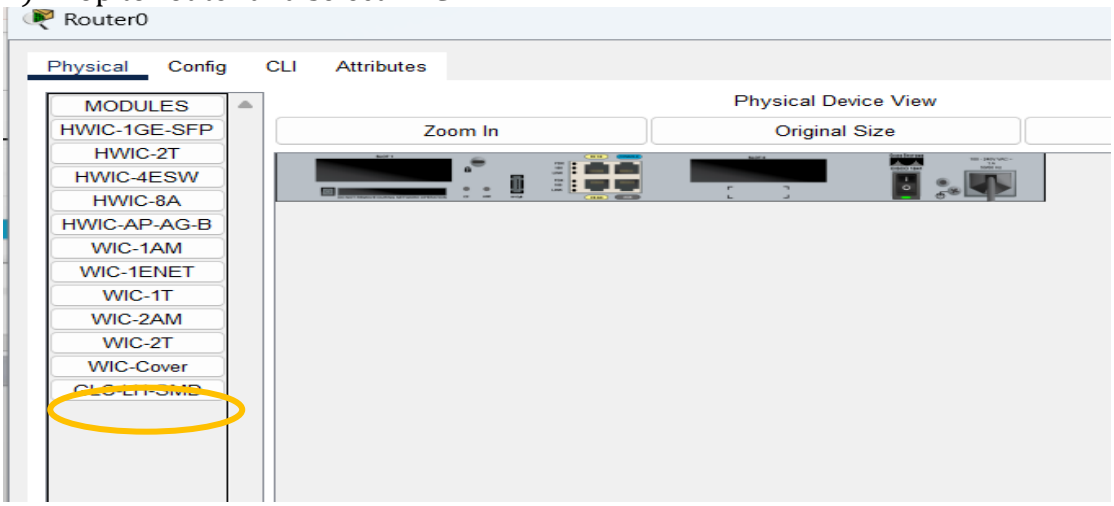


Step 5: After drop the pc and select last option FastEthernet . Same process follow to connect switch to router.

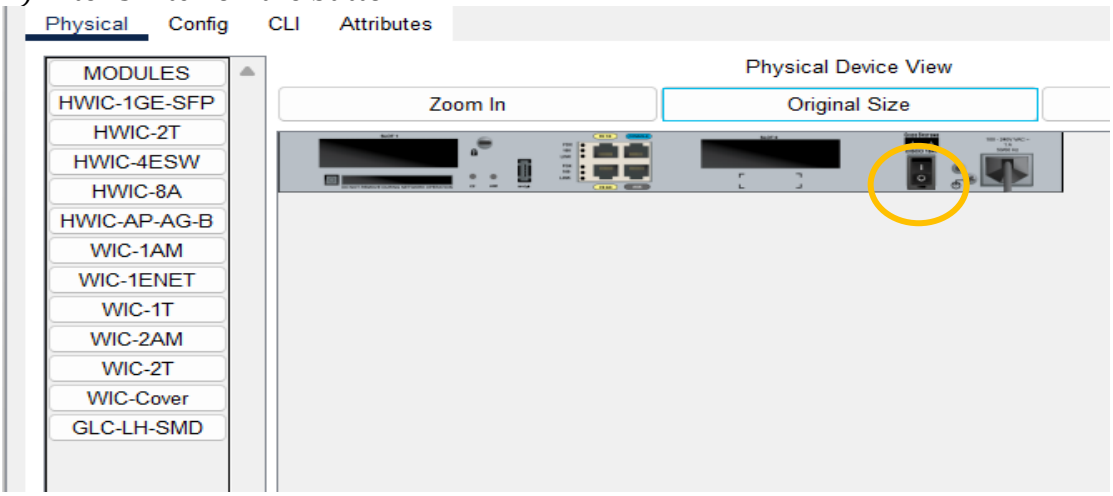


Step 6: How to add extra port in router.

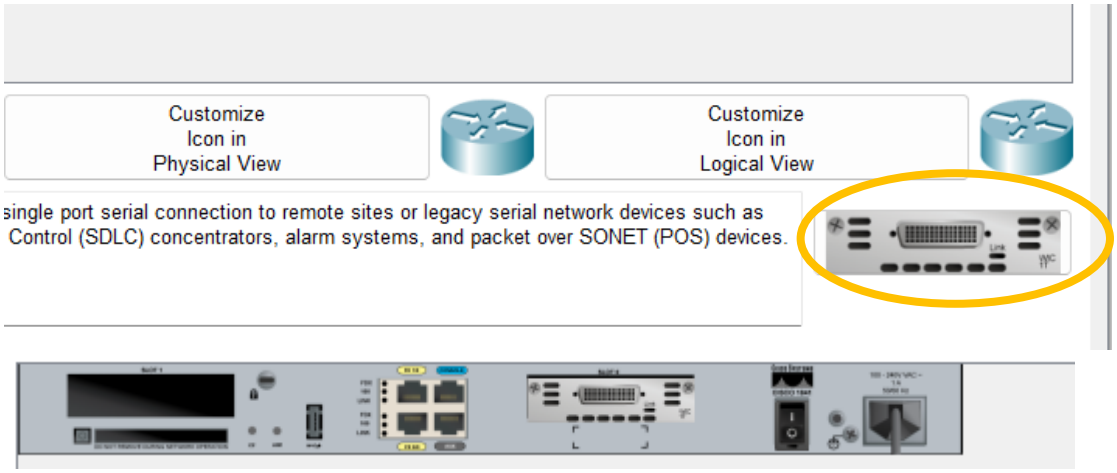
1) Drop to router and select WIC-1T.



2) After switch off the button.



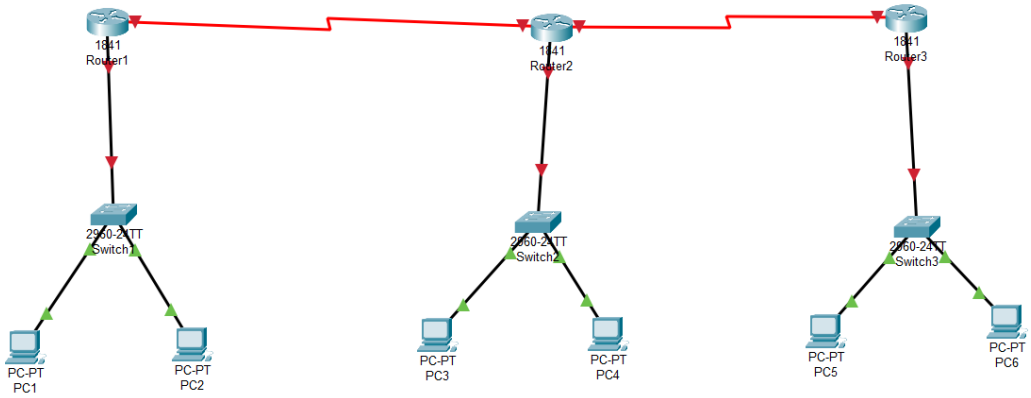
3) Click WIC-1T. and in bottom part one part to select and move empty slot.



4) After on the button.

- Same process between router 2 and 3 . but in router 2 select two port.

And final output is:

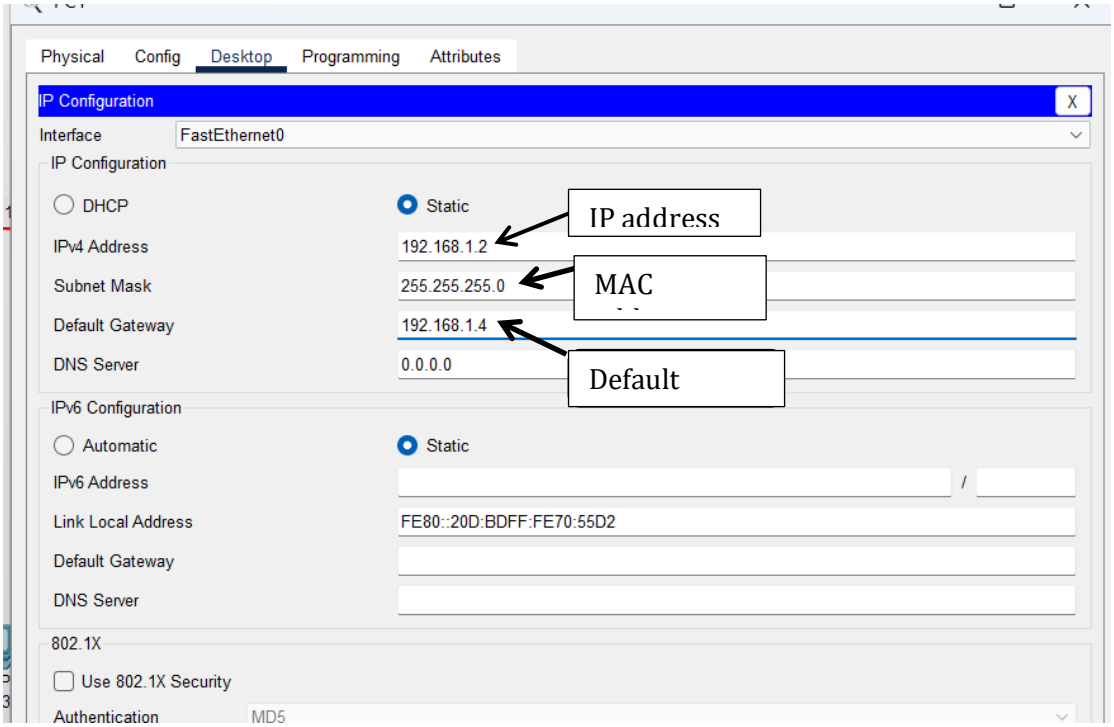


<u>Give IP Address following this note:</u>

- PC1 : 192.168.1.2
- PC2 : 192.168.1.3
- PC1 and PC2 default ip address: 192.168.1.4

- PC3 : 192.168.2.2
- PC4 : 192.168.2.3
- PC1 and PC2 default ip address: 192.168.2.4
- PC5 : 192.168.3.2
- PC6 : 192.168.3.3
- PC5 and PC6 default ip address: 192.168.3.4

<u>Step 7:</u> How to give IP address to pc.
Click pc **->** go 3rd option Desktop **->** go 1st option IP configuration

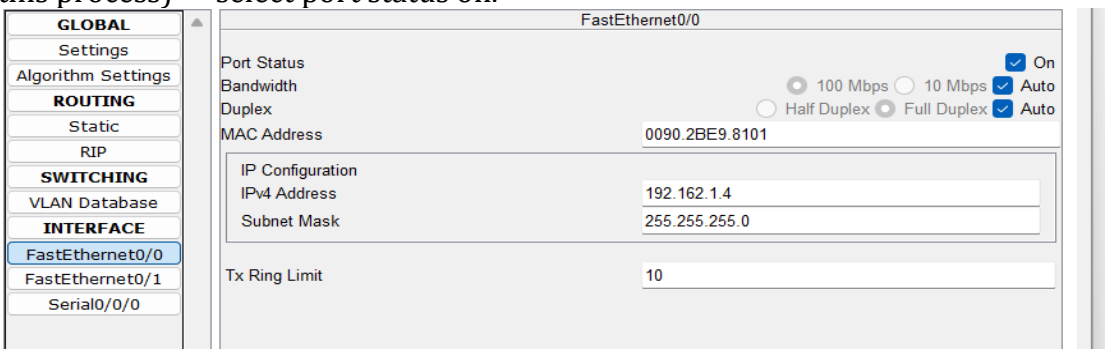


Same process follow to give IP address all PC.

Step 6: Give IP address to Router

- **We give default IP address**

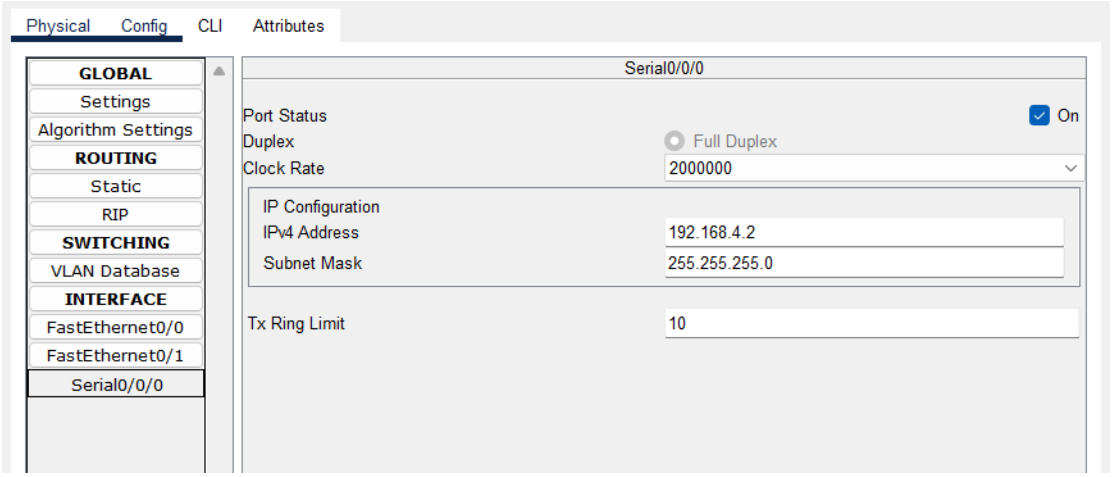
Click router -> 2nd option config -> FastEthernet0/0 -> option IP configuration(after complete this process) -> select port status on.



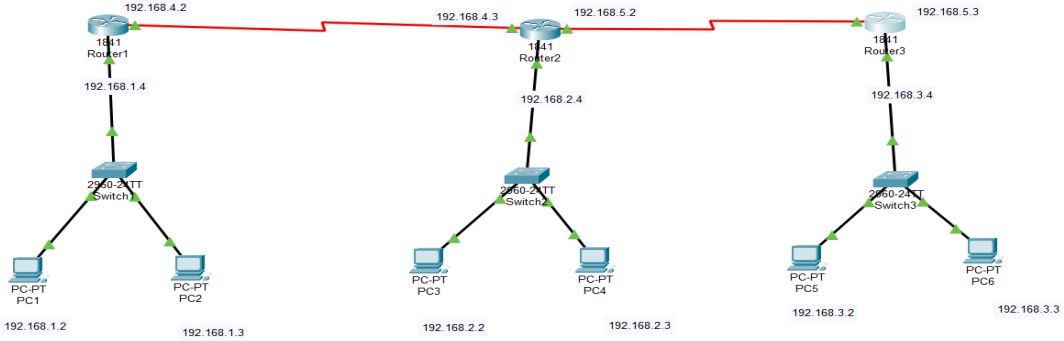
Same process follow to give all router default address.

Step 7: give serial IP address to router

Click router -> 2nd option config -> serial 0/0/0 (only give 2nd serial 0/1/0 for 2nd router) -> option IP configuration(after complete this process) -> select port status on.

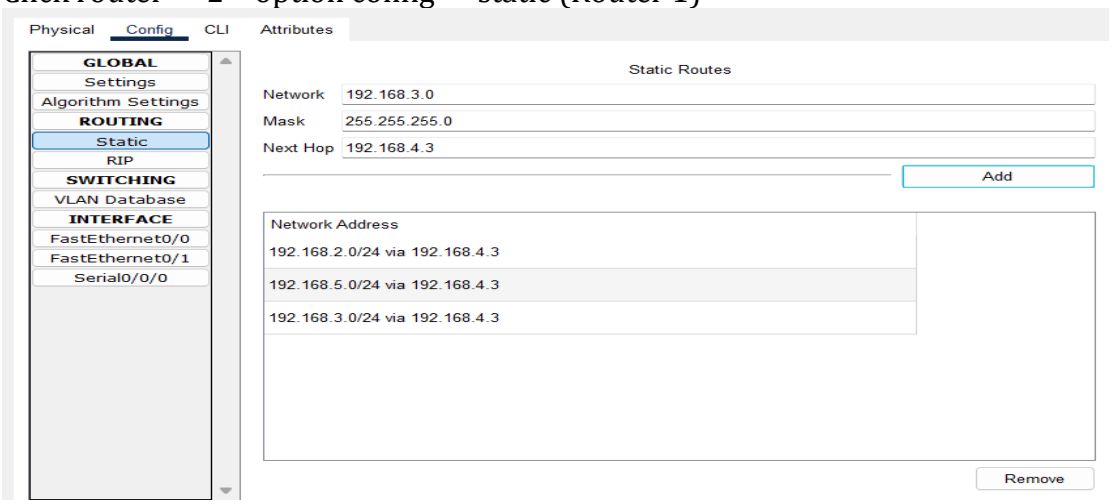


Same process follow to give all router serial address.

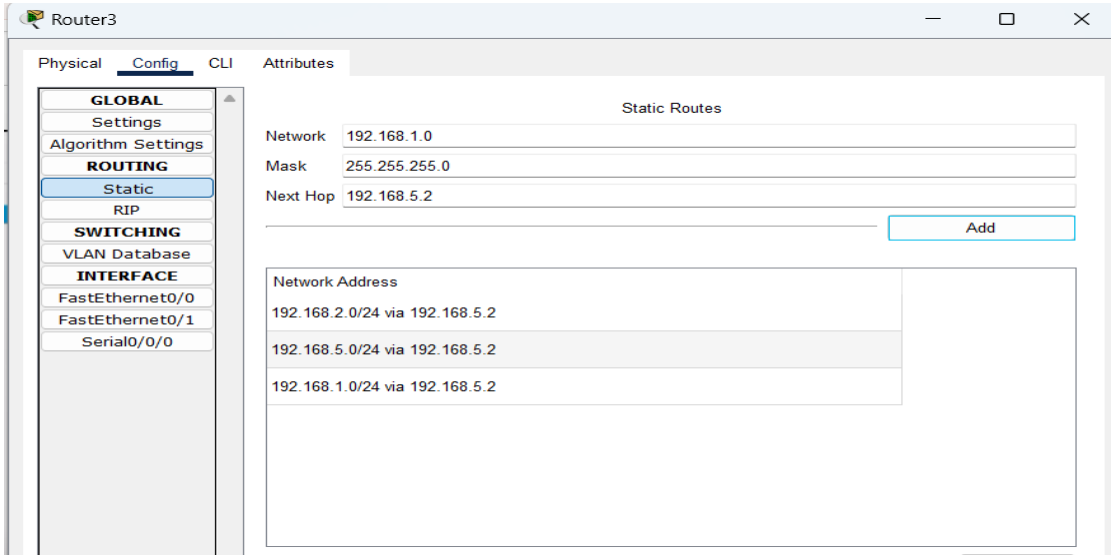


NOW apply static in router:(Process for only router 1/3)

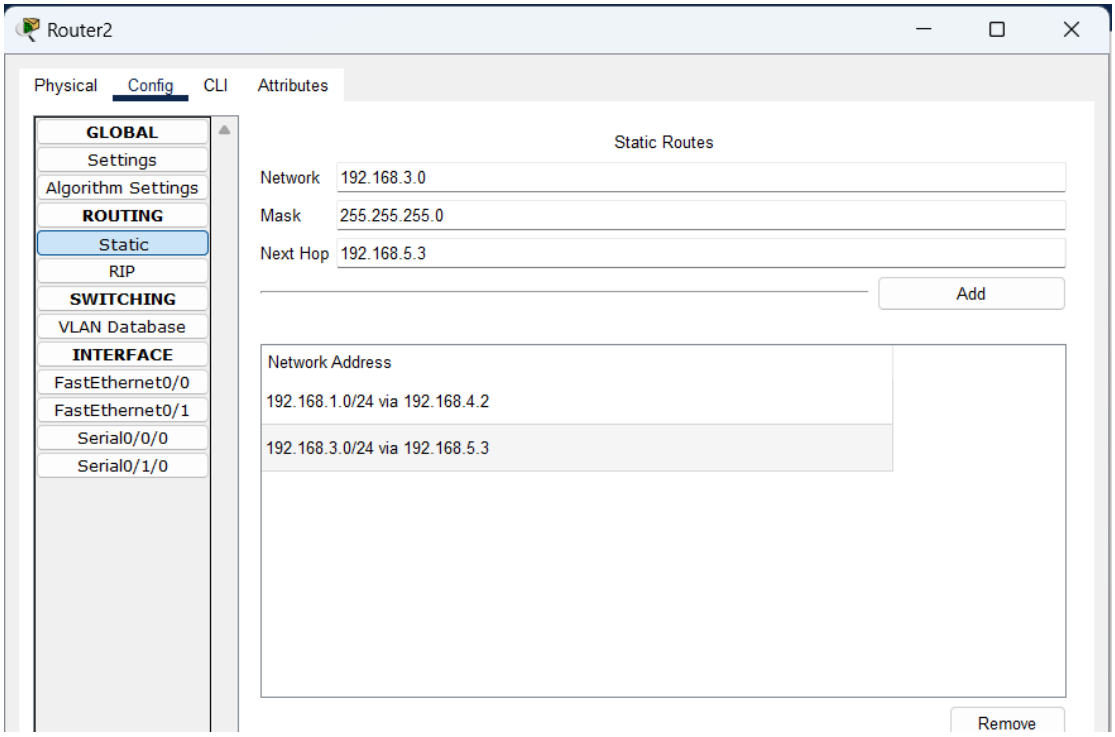
Click router -> 2nd option config -> static (Router 1)



Router 3:

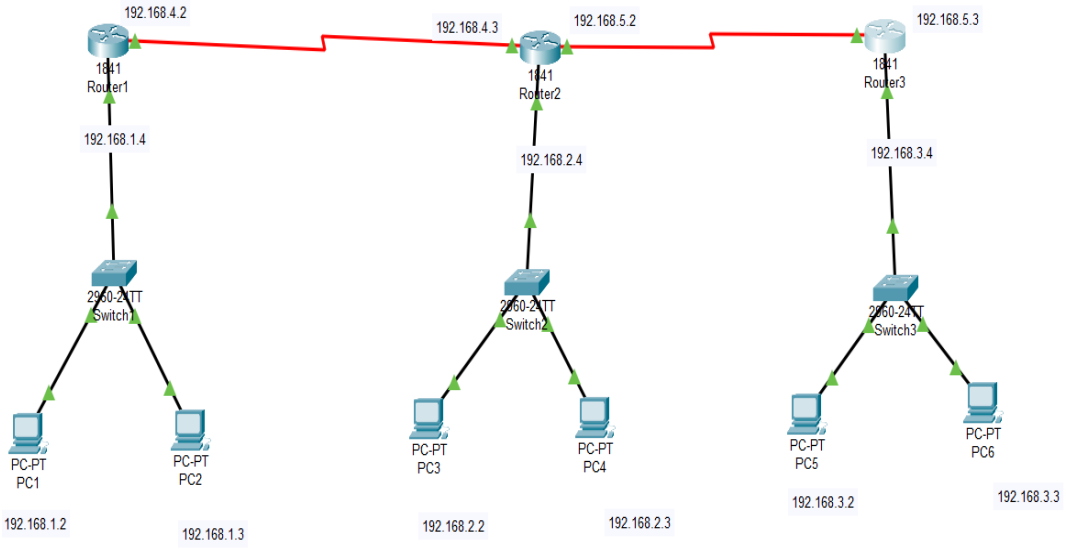


<u>Click router -> 2nd option config -> static (Router 2)</u>



- First time it will unsuccessful, try again it will be successful
- After these click on simulation and click on the play button…

Final OUTPUT


192.168.4.2
192.168.4.3
192.168.5.2
192.168.5.3
1841
Router1
1841
Router2
1841
Router3
192.168.1.4
192.168.2.4
192.168.3.4
2960-24TT
Switch1
2960-24TT
Switch2
2960-24TT
Switch3
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3
PC-PT
PC4
PC-PT
PC5
PC-PT
PC6
192.168.1.2
192.168.1.3
192.168.2.2
192.168.2.3
192.168.3.2
192.168.3.3

Practical – 2

Aim: Implementation Flow Control Algorithms: LRC.

Concept:
LRC (even parity)

- **Sender (original Data) + LRC**

11001100	**11100011**	**10101010**	**11100111**	**01100010**

1 1 0 0 1 1 0 0
1 1 1 0 0 0 1 1
1 0 1 0 1 0 1 0
1 1 1 0 0 1 1 1
---
0 1 1 0 0 0 1 0

Receiver
1 1 0 0 1 1 0 0
1 1 1 0 0 0 1 1
1 0 1 0 1 0 1 0
1 1 1 0 0 1 1 1
0 1 1 0 0 0 1 0
---
0 0 0 0 0 0 0 0 **-> All bits are 0 means data is accepted**

Code:

1)

```
#include <stdio.h>
#include <string.h>

void calculateLRC(char data[][5], int rows, int cols, char *lrc) {
   for (int i = 0; i < cols; i++) {
      int count = 0;
      for (int j = 0; j < rows; j++) {
         if (data[j][i] == '1') {
            count++;
         }
      }
      lrc[i] = (count % 2 == 0) ? '0' : '1';
   }
   lrc[cols] = '\0';
}

int main() {
   char data[3][5] = {"1010", "1100", "0111"};
   int rows = 3, cols = 4;
   char lrc[5];
```

```
    calculateLRC(data, rows, cols, lrc);

    printf("Original Data Block:\n");
    for (int i = 0; i < rows; i++) {
        printf("%s\n", data[i]);
    }
    printf("LRC: %s\n", lrc);

    // Checking the LRC
    char received_block[4][5] = {"1010", "1100", "0111", "0001"}; // Example received block
    calculateLRC(received_block, 3, cols, lrc);
    if (strcmp(lrc, received_block[3]) == 0) {
        printf("Data block is valid.\n");
    } else {
        printf("Data block is invalid.\n");
    }

    return 0;
}
```

OUTPUT:

- **Data is valid**

```
/tmp/Myx1M1brIa.o
Original Data Block:
1010
1100
0111
LRC: 0001
Data block is valid.
```

2)

```
#include <stdio.h>
#include <string.h>

void calculateLRC(char data[][5], int rows, int cols, char *lrc) {
    for (int i = 0; i < cols; i++) {
        int count = 0;
        for (int j = 0; j < rows; j++) {
            if (data[j][i] == '1') {
                count++;
            }
        }
        lrc[i] = (count % 2 == 0) ? '0' : '1';
    }
    lrc[cols] = '\0';
}
```

```
int main() {
    char data[3][5] = {"1010", "1100", "0111"};
    int rows = 3, cols = 4;
    char lrc[5];

    calculateLRC(data, rows, cols, lrc);

    printf("Original Data Block:\n");
    for (int i = 0; i < rows; i++) {
        printf("%s\n", data[i]);
    }
    printf("LRC: %s\n", lrc);

    // Checking the LRC
    char received_block[4][5] = {"1010", "1100", "0111", "0001"}; // Example received block
    calculateLRC(received_block, 3, cols, lrc);
    if (strcmp(lrc, received_block[3]) == 0) {
        printf("Data block is valid.\n");
    } else {
        printf("Data block is invalid.\n");
    }

    return 0;
}
```

- **Data is not valid**

```
/tmp/iVfrNZm8Gl.o
Original Data Block:
1010
1100
0111
LRC: 0001
Data block is invalid.
```

Practical – 3

Aim:Implementation Flow Control Algorithms: VRC.(Parity check)

Parity check code in c

```
#include<stdio.h>
#include<string.h>
//create a function for parity check
char calculateVRC(char *data, int even_parity){
   int count=0;
   for(int i=0; i<strlen(data); i++){
      if(data[i]=='1'){
         count++;
      }
   }
   if(even_parity){
      return (count%2==0)?'0':'1';
   }
   else{
      return (count%2==0)?'1':'0';
   }
}
int main(){
   char data[]= "1010101";
   int even_parity=1;//set to 0 for odd parity
   char parity_bit=calculateVRC(data,even_parity);
   printf("Original data:%s\n",data);
   printf("Parity Bit:%c\n",parity_bit);
   printf("Encoded data:%s%c\n",data,parity_bit);
   //checking the VRC
   char received_data[]="10101010"; //example received data
   parity_bit = calculateVRC(received_data, even_parity);
   if(parity_bit=='0'){
      printf("Data is valid\n");
   }
   else{
      printf("Data is not valid");
   }
   return 0;
}
```

OUTPUT:

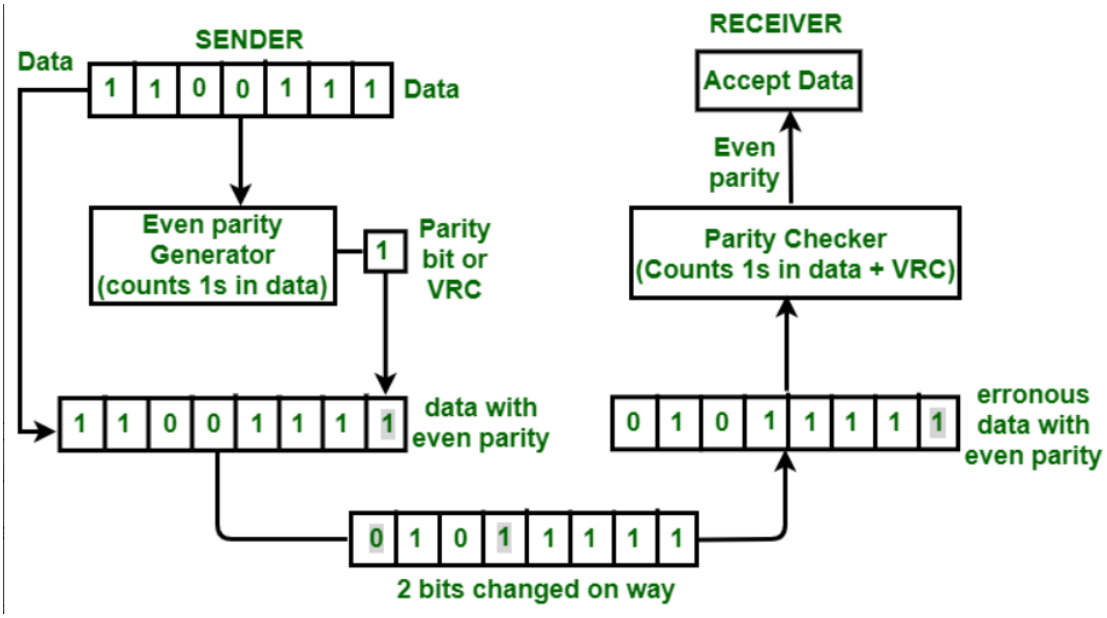
Even parity

```
/tmp/nrQRm02gGV.o
Original data:1010101
Parity Bit:0
Encoded data:10101010
Data is valid
```

Odd Parity:

```
/tmp/haQeo8K4V4.o
Original data:1010101
Parity Bit:1
Encoded data:10101011
Data is not valid
```

Concept:



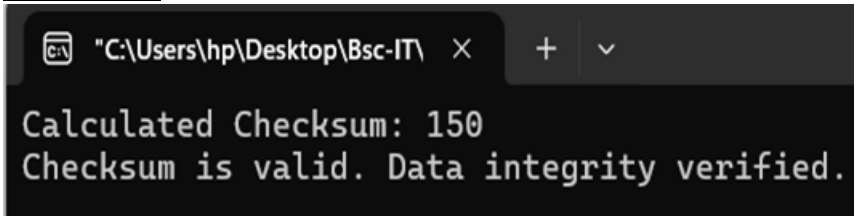
Practical – 4

Aim:Implementation Flow Control Algorithms: Checksum.

Code:

```
#include <stdio.h>
#include <stdint.h>
// Function to calculate checksum
uint8_t calculate_checksum(const uint8_t *data, size_t length) {
uint16_t sum = 0;
// Iterate over each byte of data
for (size_t i = 0; i < length; i++) {
sum += data[i];
}
// Return the checksum as a single byte
return (uint8_t)(sum % 256);
}
// Function to validate the checksum
int validate_checksum(const uint8_t *data, size_t length, uint8_t checksum) {
return calculate_checksum(data, length) == checksum;
}
int main() {
// Example data
uint8_t data[] = {10, 20, 30, 40, 50};
size_t length = sizeof(data) / sizeof(data[0]);
// Calculate the checksum
uint8_t checksum = calculate_checksum(data, length);
printf("Calculated Checksum: %u\n", checksum);
// Simulate data transmission (for example)
// Validate the checksum on the receiver's end
if (validate_checksum(data, length, checksum)) {
printf("Checksum is valid. Data integrity verified.\n");
} else {
printf("Checksum is invalid. Data integrity compromised.\n");
}
return 0;
}
```

OUTPUT:



Practical – 5

Aim:Implementation Flow Control Algorithms: CRC.

```
#include <stdio.h>
#include <string.h>
void xor(char *a, char *b, int len) {
   for (int i = 0; i < len; i++) {
      a[i] = (a[i] == b[i]) ? '0' : '1';
   }
}
void crc(char *data, char *generator, char *remainder) {
   int data_len = strlen(data);
   int gen_len = strlen(generator);
   char temp[data_len + gen_len + 1]; // +1 for null character

   strcpy(temp, data);
   strcat(temp, "000"); // append zeros

   for (int i = 0; i <= data_len - 1; i++) {
      if (temp[i] == '1') {
         xor(temp + i, generator, gen_len);
      }
   }

   strncpy(remainder, temp + data_len, gen_len - 1);
   remainder[gen_len - 1] = '\0'; // null-terminate the string
}
    int main() {
   char data[] = "100100";
   char generator[] = "1101";
   char remainder[4];

   crc(data, generator, remainder);
   printf("Original Data: %s\n", data);
   printf("CRC Remainder: %s\n", remainder);

   // Append the CRC to the Original data
   int data_len = strlen(data);
   char encoded_data[data_len + strlen(generator) - 1 + 1]; // +1 for null character
   strcpy(encoded_data, data);
   strcat(encoded_data, remainder);
   printf("Encoded data: %s\n", encoded_data);

   // Checking the CRC
   crc(encoded_data, generator, remainder);
   if (strcmp(remainder, "000") == 0) {
      printf("Data is valid.\n");
   } else {
      printf("Data is not valid\n");
```

```
    }
    return 0;
}
```

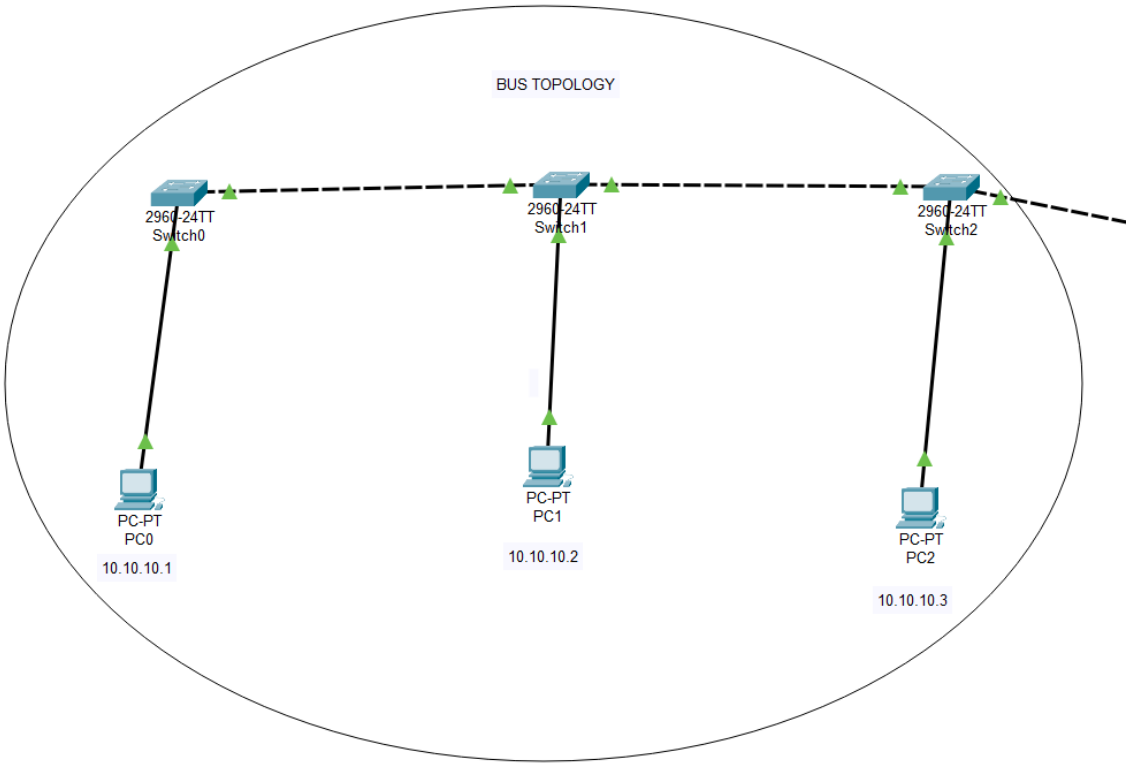
OUTPUT:

```
Original Data: 100100
CRC Remainder: 001
Encoded data: 100100001
Data is valid.
```

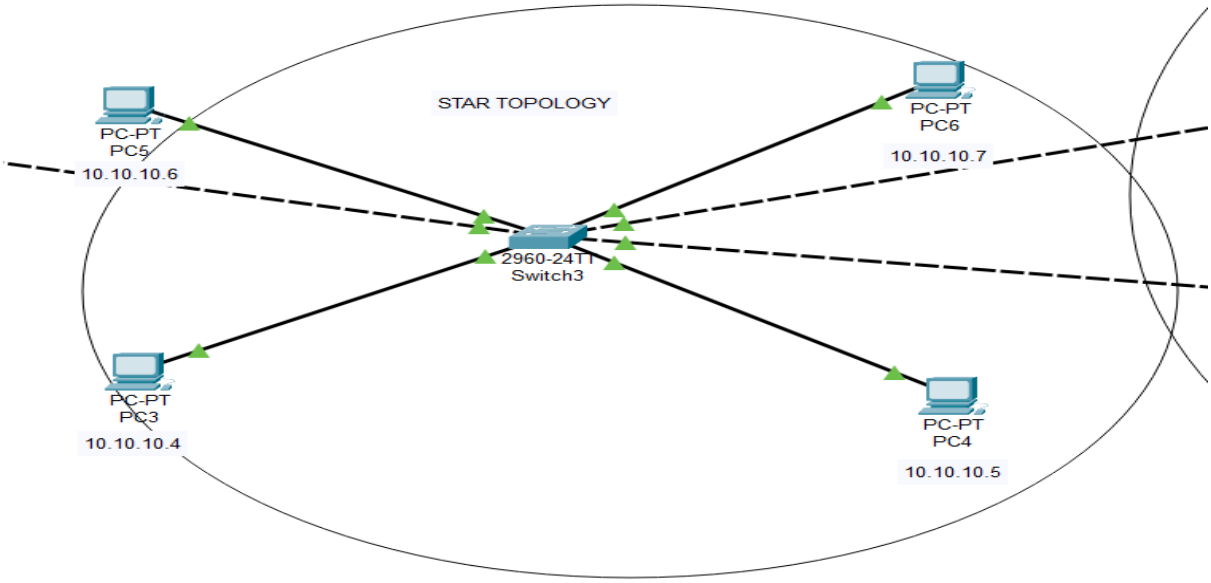
Practical – 6

Aim:Implement different LAN topologies using Network Simulator.

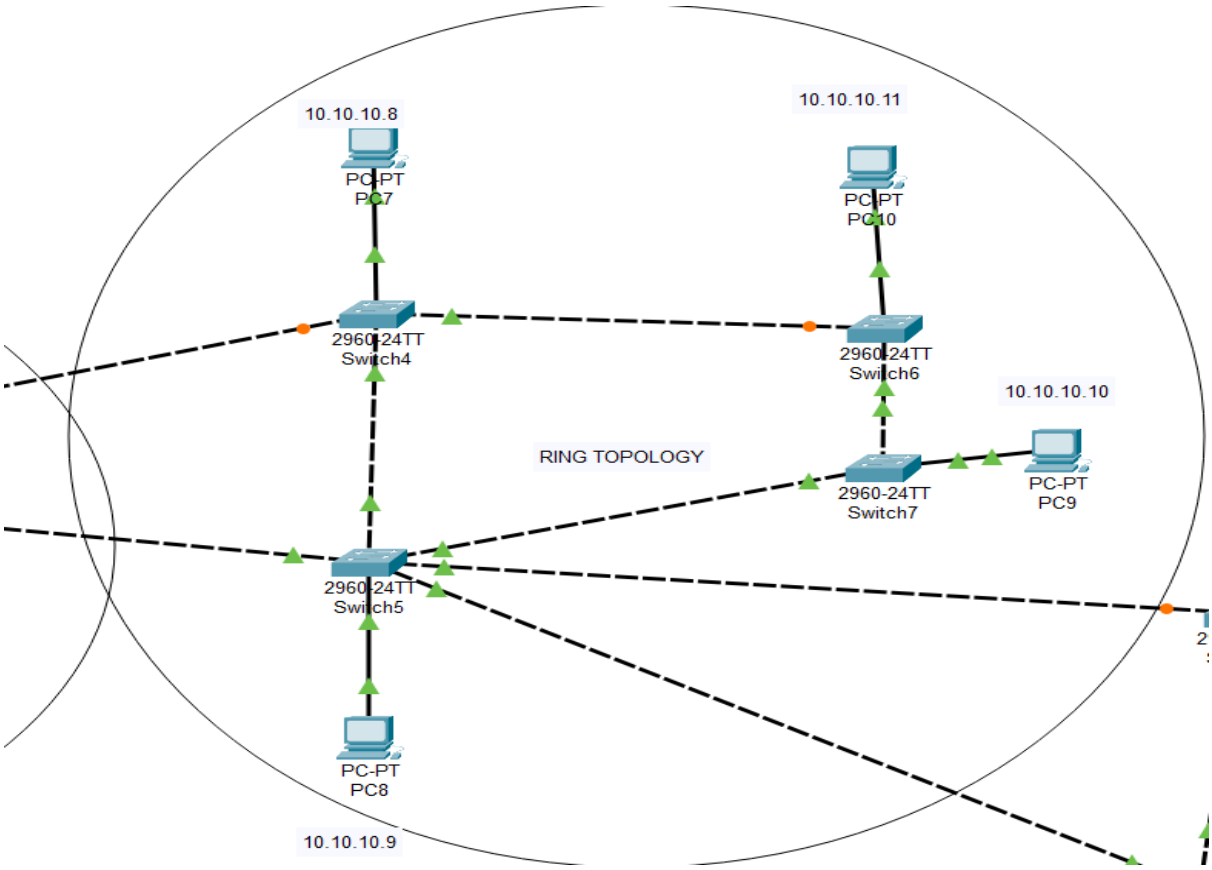
1. Bus Topology



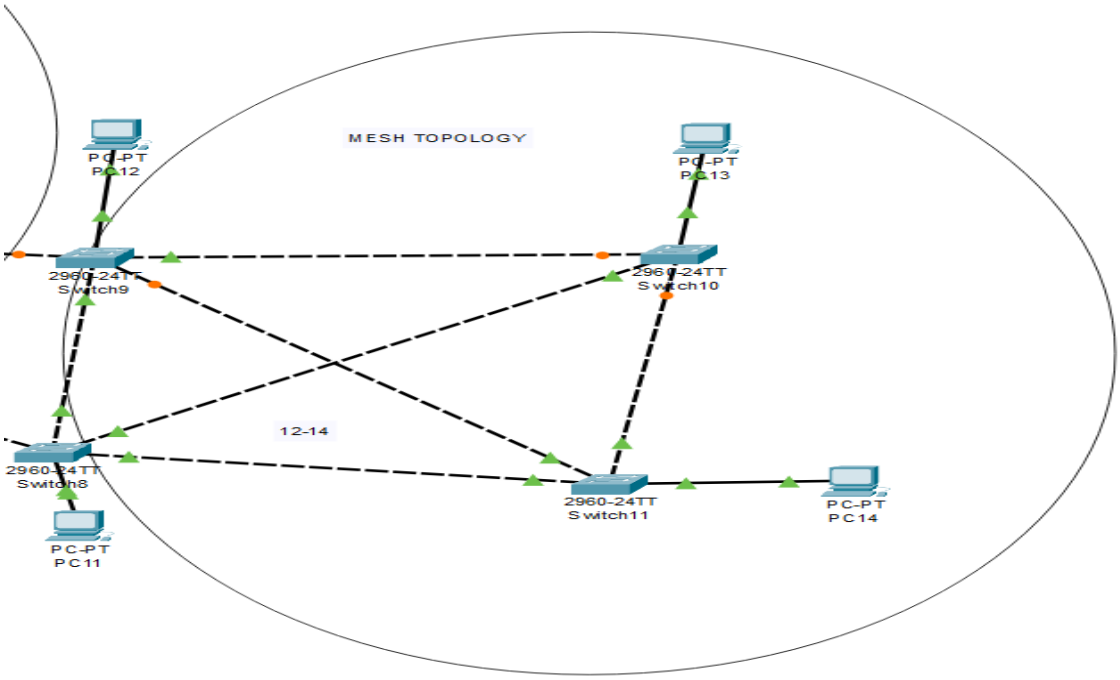
2. Star Topology



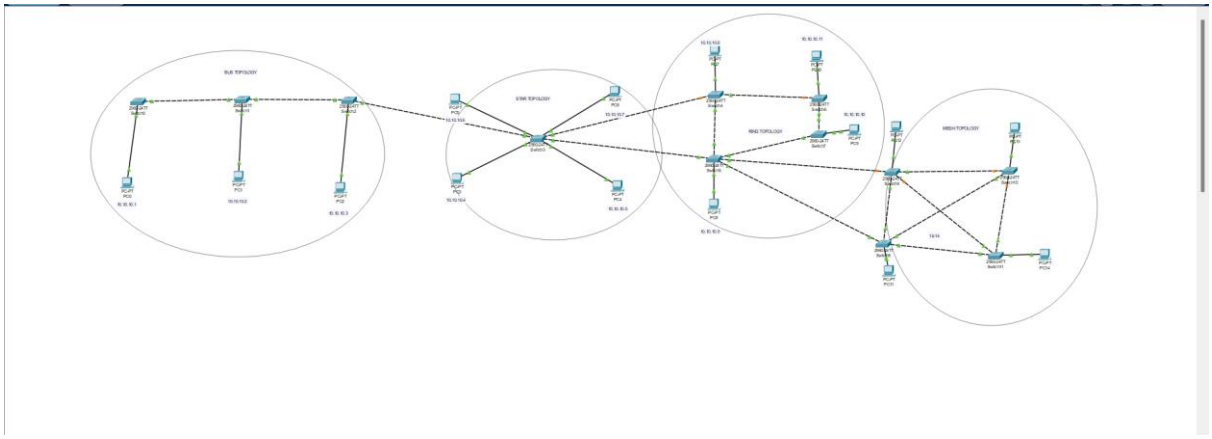
3.Ring Topology



4.Mesh Topology



All in One:



Practical – 7

Aim:Implement Packet Generation having information of packet number (2-dig), Total no of packets & data itself in the packet.

Code:

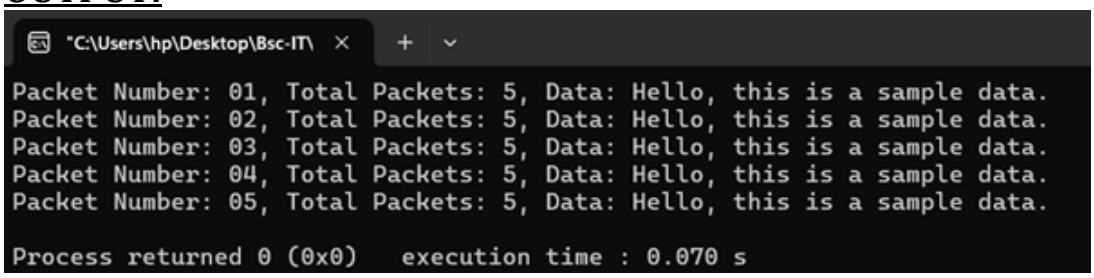
```
#include <stdio.h>
#include <stdint.h>
#include <string.h>

#define MAX_DATA_LENGTH 100
#define MAX_PACKETS 255

// 2 digits for packet number (00-99)
// Structure to represent a packet
typedef struct {
uint8_t packet_number;
// 2-digit packet number
uint8_t total_packets;
// Total number of packets
char data[MAX_DATA_LENGTH];
// Actual data in the packet
} Packet;
// Function to generate packets
void generate_packets(const char *data, uint8_t total_packets) {
size_t data_length = strlen(data);
if (total_packets > MAX_PACKETS) {
printf("Error:Total packets exceed the maximum limitof %d\n",MAX_PACKETS);
return;
}
for (uint8_t i = 0; i < total_packets; i++) {
Packet packet;
packet.packet_number = i + 1;
// Packet number starts from 1
packet.total_packets = total_packets;
// Copy data to the packet
strncpy(packet.data, data, MAX_DATA_LENGTH - 1);
packet.data[MAX_DATA_LENGTH - 1] = '\0';
// Ensure null termination
// Print the generated packet information
printf("Packet Number: %02d, Total Packets: %d, Data: %s\n",
packet.packet_number, packet.total_packets, packet.data);
}
```

```
}
int main() {
const char *data = "Hello, this is a sample data.";
uint8_t total_packets = 5;
// Example total packets to generate
// Generate packets with the specified data
generate_packets(data, total_packets);
return 0;
}
```

OUTPUT:



Practical – 8

Aim:Implement CSMA/CD between two machines

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h> // For sleep function

#define MAX_ATTEMPTS 5
#define BACKOFF_TIME(base) (rand() % (base))

// Function to simulate sending data
void send_data(int machine_id) {
  printf("Machine %d is trying to send data...\n", machine_id);
}

// Function to simulate collision detection
int detect_collision(int machine_id) {
  // Randomly simulate a collision (50% chance)
  return rand() % 2;
}

// Main CSMA/CD function
void csma_cd(int machine_id) {
  for (int attempt = 1; attempt <= MAX_ATTEMPTS; attempt++) {
    send_data(machine_id);

    // Check for collision
    if (detect_collision(machine_id)) {
      printf("Collision detected for Machine %d! Attempt %d\n", machine_id, attempt);
      int backoff_time = BACKOFF_TIME(2 << attempt);
      printf("Machine %d will back off for %d seconds.\n", machine_id, backoff_time);
      sleep(backoff_time); // Simulate backoff time
    } else {
      printf("Machine %d successfully sent data!\n", machine_id);
      return; // Data sent successfully
    }
  }
  printf("Machine %d failed to send data after %d attempts.\n", machine_id,
MAX_ATTEMPTS);
}

int main() {
  srand(time(NULL)); // Seed random number generator

  // Simulate two machines trying to send data
  printf("Starting CSMA/CD simulation...\n");
  csma_cd(1);
```

```
  csma_cd(2);

  return 0;
}
```

OUTPUT:

```
/tmp/fpMwNy2rAs.o
Starting CSMA/CD simulation...
Machine 1 is trying to send data...
Collision detected for Machine 1! Attempt 1
Machine 1 will back off for 0 seconds.
Machine 1 is trying to send data...
Collision detected for Machine 1! Attempt 2
Machine 1 will back off for 7 seconds.
Machine 1 is trying to send data...
Collision detected for Machine 1! Attempt 3
Machine 1 will back off for 0 seconds.
Machine 1 is trying to send data...
Collision detected for Machine 1! Attempt 4
Machine 1 will back off for 14 seconds.
Machine 1 is trying to send data...
Collision detected for Machine 1! Attempt 5
Machine 1 will back off for 11 seconds.
Machine 1 failed to send data after 5 attempts.
Machine 2 is trying to send data...
Collision detected for Machine 2! Attempt 1
Machine 2 will back off for 0 seconds.
Machine 2 is trying to send data...
Machine 2 successfully sent data!
```

Practical – 9

Aim:Implement Token Ring Between 3 Machines.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>//For sleep function
#include<pthread.h>

#define NUM_MACHINES 3
#define TOKEN_PASSED 1
#define TOKEN_NOT_PASSED 0

typedef struct{
int id;
int has_token;
}Machine;
//Global declaration of machines
Machine machines[NUM_MACHINES];
//Function to simulate a machine sending data
void*machine_send(void*arg){
Machine*machine=(Machine*)arg;
while(1){
//Wait until the machine has the token
while (!machine->has_token) {
usleep(1000000); // Sleep for a short time
}
// Simulate sending data
printf("Machine %d is sending data...\n", machine->id);
sleep(1); // Simulate time taken to send data
// Pass the token to the next machine
int next_machine_id = (machine->id % NUM_MACHINES);
// Corrected calculation for
next machine
printf("Machine %d is passing the token to Machine %d...\n", machine->id,
machines[next_machine_id].id);
machine->has_token = TOKEN_NOT_PASSED;
// Current machine releases the token
machines[next_machine_id].has_token = TOKEN_PASSED;
```

```
// Next machine gets the token
}
}
int main() {
pthread_t threads[NUM_MACHINES];
// Initialize machines
for (int i = 0; i < NUM_MACHINES; i++) {
machines[i].id = i + 1; // Assign machine ID (1-based index)
machines[i].has_token = (i == 0) ? TOKEN_PASSED : TOKEN_NOT_PASSED;
// Only the  first machine starts with the token
}
// Create threads for each machine
for (int i = 0; i < NUM_MACHINES; i++) {
pthread_create(&threads[i], NULL, machine_send, (void *)&machines[i]);
}
// Join threads (this will never be reached in this infinite loop)
for (int i = 0; i < NUM_MACHINES; i++) {
pthread_join(threads[i], NULL);
}
return 0;
}
```

OUTPUT:

```
Machine 1 is sending data...
Machine 1 is passing the token to Machine 2...
Machine 2 is sending data...
Machine 2 is passing the token to Machine 3...
Machine 3 is sending data...
Machine 3 is passing the token to Machine 1...
...
```