



# VIT<sup>®</sup>

## Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Name: Apurba Koirala

Reg no: 22BCE3799

Subject Code: BCSE308P

Course Title: Computer Networks Lab

Lab Slot: L31 + L32

Guided by: Dr. Arivoli A

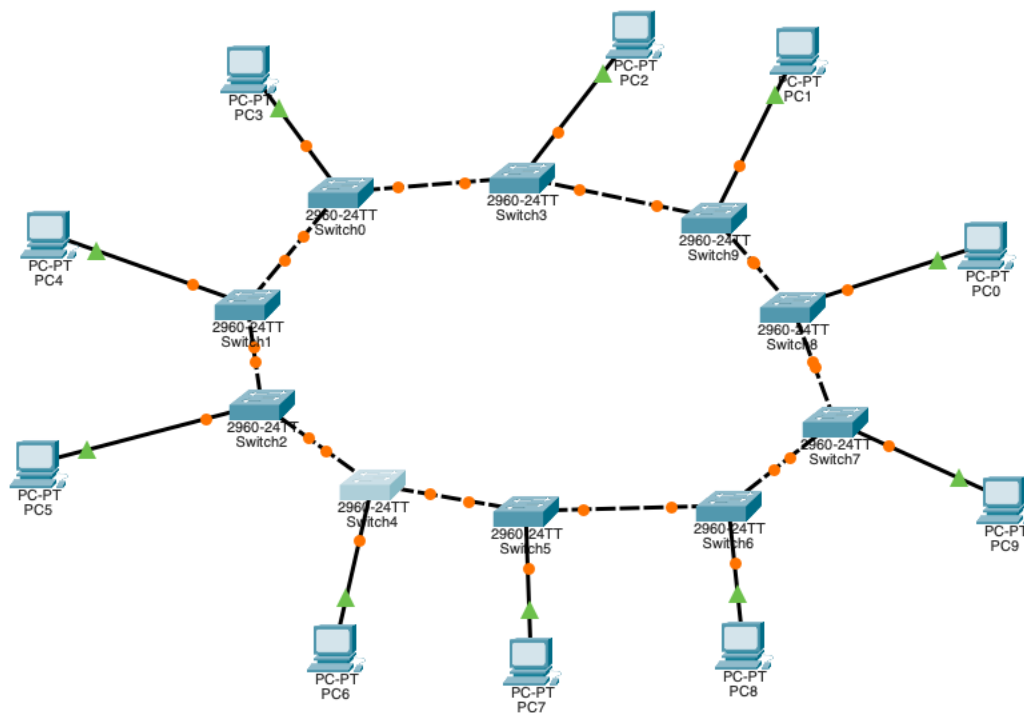
Lab Assessment 3

Question no. 1

1. Create a network with minimum of 10 devices using various topologies using cisco packet tracer tool and test the connectivity with the transmission of simple PDU.(Intermediate Device – Hub).

Solution;

RING TOPOLOGY:



PC0:

```

Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.10

Pinging 192.168.0.10 with 32 bytes of data:

Reply from 192.168.0.10: bytes=32 time<1ms TTL=128
Reply from 192.168.0.10: bytes=32 time<1ms TTL=128
Reply from 192.168.0.10: bytes=32 time<1ms TTL=128
Reply from 192.168.0.10: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

```

PC4:

```

Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.7

Pinging 192.168.0.7 with 32 bytes of data:

Reply from 192.168.0.7: bytes=32 time=1ms TTL=128
Reply from 192.168.0.7: bytes=32 time<1ms TTL=128
Reply from 192.168.0.7: bytes=32 time=1ms TTL=128
Reply from 192.168.0.7: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.7:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

```

PC6:

```




Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.1

Pinging 192.168.0.1 with 32 bytes of data:

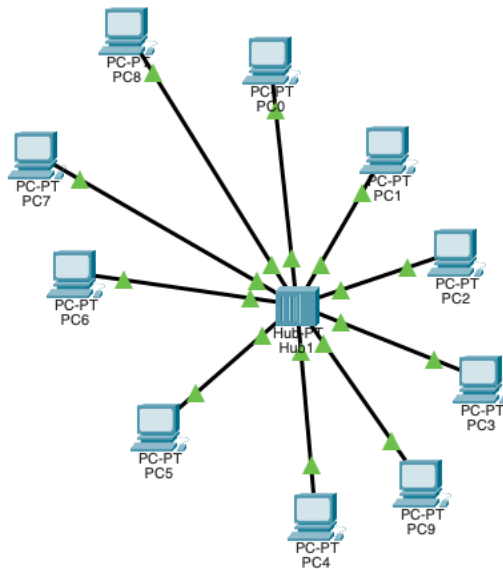
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

```

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC3	PC2	IC...		0.000	N	0	(...)	(delete)
	Successful	PC4	PC5	IC...		0.000	N	1	(...)	(delete)
	Successful	PC0	PC2	IC...		0.000	N	2	(...)	(delete)

## STAR TOPOLOGY:



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.9

Pinging 192.168.0.9 with 32 bytes of data:

Reply from 192.168.0.9: bytes=32 time=25ms TTL=128
Reply from 192.168.0.9: bytes=32 time=12ms TTL=128
Reply from 192.168.0.9: bytes=32 time=11ms TTL=128
Reply from 192.168.0.9: bytes=32 time=13ms TTL=128

Ping statistics for 192.168.0.9:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 11ms, Maximum = 25ms, Average = 15ms

C:\>ping 192.168.0.10

Pinging 192.168.0.10 with 32 bytes of data:

Reply from 192.168.0.10: bytes=32 time<1ms TTL=128
Reply from 192.168.0.10: bytes=32 time<1ms TTL=128
Reply from 192.168.0.10: bytes=32 time<1ms TTL=128
Reply from 192.168.0.10: bytes=32 time<1ms TTL=128




Ping statistics for 192.168.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.0.1

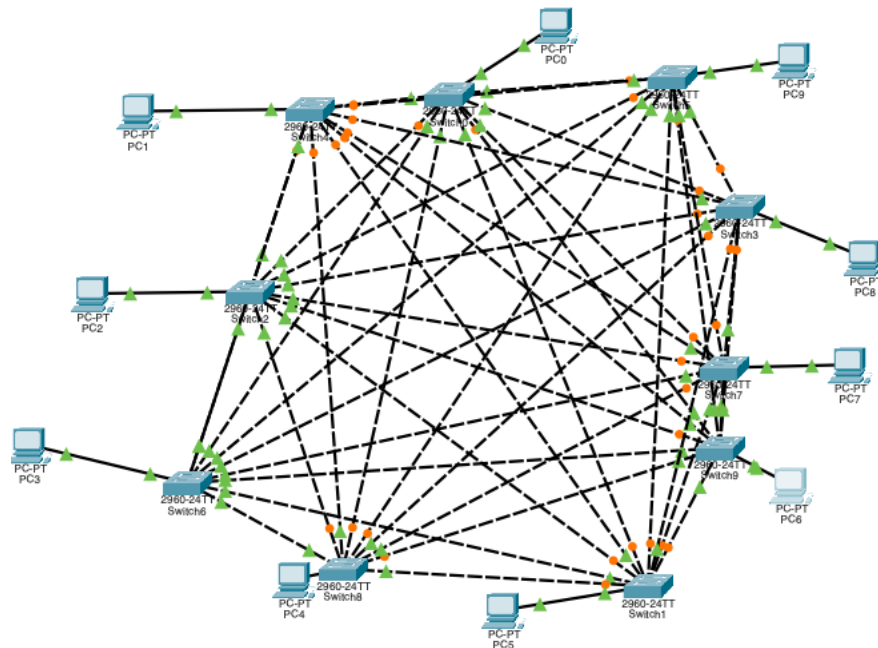
Pinging 192.168.0.1 with 32 bytes of data:

Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC7	PC1	IC...		0.000	N	0	(...)	(delete)
	Successful	PC6	PC3	IC...		0.000	N	1	(...)	(delete)
	Successful	PC5	PC0	IC...		0.000	N	2	(...)	(delete)

MESH TOPOLOGY:



```

Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.4

Pinging 192.168.0.4 with 32 bytes of data:

Reply from 192.168.0.4: bytes=32 time<1ms TTL=128
Reply from 192.168.0.4: bytes=32 time<1ms TTL=128
Reply from 192.168.0.4: bytes=32 time<1ms TTL=128
Reply from 192.168.0.4: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms





C:\>ping 192.168.0.7

Pinging 192.168.0.7 with 32 bytes of data:

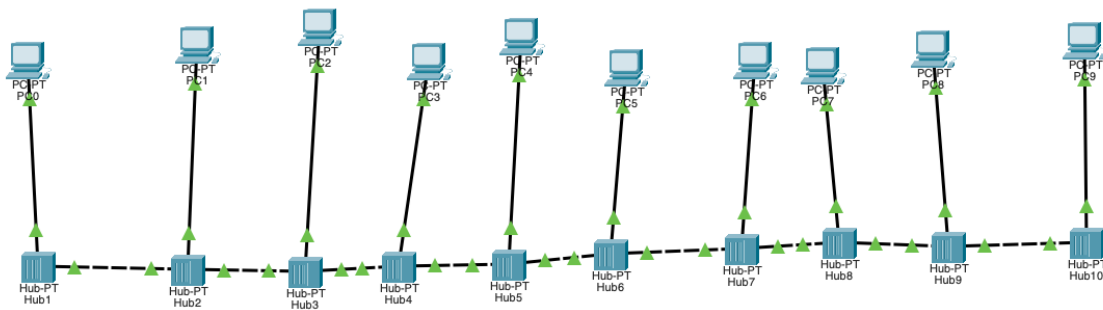
Reply from 192.168.0.7: bytes=32 time<1ms TTL=128
Reply from 192.168.0.7: bytes=32 time<1ms TTL=128
Reply from 192.168.0.7: bytes=32 time<1ms TTL=128
Reply from 192.168.0.7: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.7:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

```

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC1	PC9	IC...		0.000	N	0	(...)	(delete)
	Successful	PC2	PC6	IC...		0.000	N	1	(...)	(delete)
	Successful	PC3	PC4	IC...		0.000	N	2	(...)	(delete)
	Successful	PC5	PC7	IC...		0.000	N	3	(...)	(delete)

BUS TOPOLOGY:



```

Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.0.5

Pinging 192.168.0.5 with 32 bytes of data:

Reply from 192.168.0.5: bytes=32 time<1ms TTL=128
Reply from 192.168.0.5: bytes=32 time<1ms TTL=128
Reply from 192.168.0.5: bytes=32 time<1ms TTL=128
Reply from 192.168.0.5: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ping 192.168.0.9

Pinging 192.168.0.9 with 32 bytes of data:

Reply from 192.168.0.9: bytes=32 time=1ms TTL=128
Reply from 192.168.0.9: bytes=32 time<1ms TTL=128
Reply from 192.168.0.9: bytes=32 time<1ms TTL=128
Reply from 192.168.0.9: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.9:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

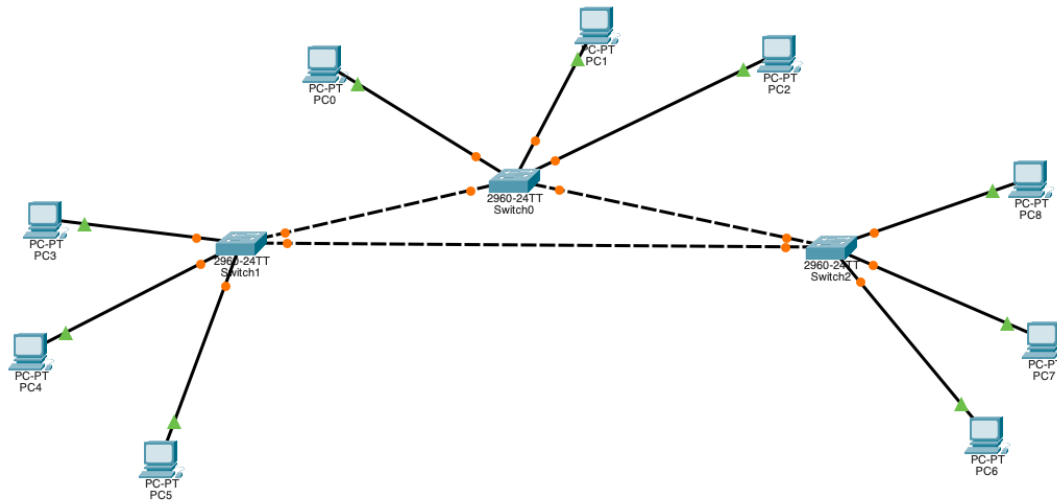
```

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC1	PC4	IC...	Green	0.000	N	0	(...)	(delete)
	Successful	PC0	PC9	IC...	Purple	0.000	N	1	(...)	(delete)
	Successful	PC4	PC7	IC...	Yellow	0.000	N	2	(...)	(delete)

Question no. 2:

2. Create a network with clustering using 3 switches and 9 PC's and test the connectivity with transmission of simple PDU. (Intermediate device – Switch)

Solution;



Transmission using simple PDU results:

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC8	IC...		0.000	N	0	(...	(delete)
	Successful	PC3	PC6	IC...		0.000	N	1	(...	(delete)
	Successful	PC3	PC5	IC...		0.000	N	2	(...	(delete)
	Successful	PC4	PC1	IC...		0.000	N	3	(...	(delete)

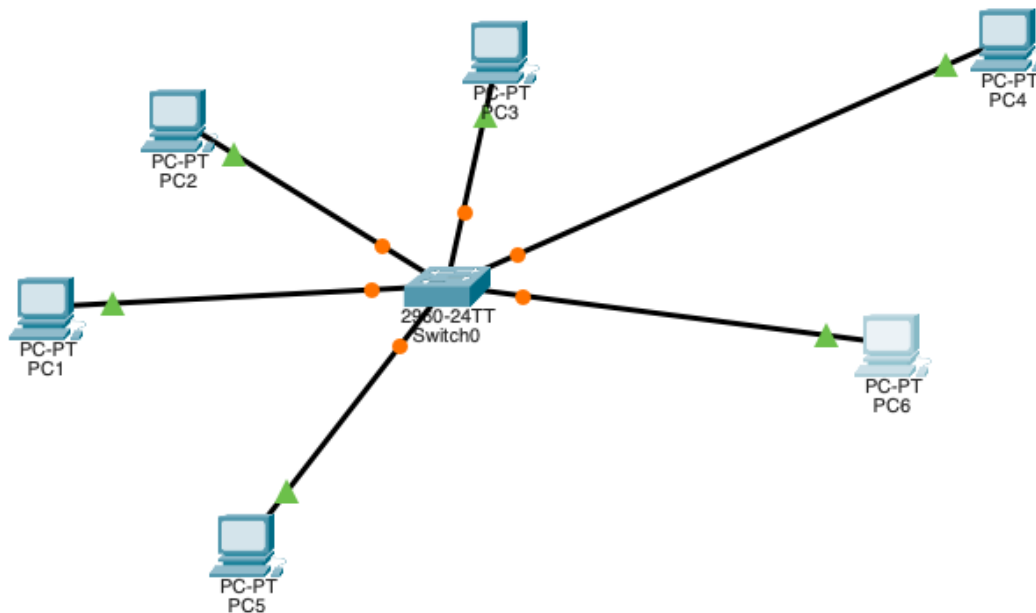


Question no. 3

3. Six devices connected with one switch using CISCO packet tracer simulator and execute the command using command prompt.

- IP address configuration for all PCs.
- Ping command for all the devices network (Minimum 3 commands)
- Show the message trasmitted from  
PC1 to PC3  
PC6 to PC5  
PC4 to PC2  
Upload the screenshots for all the above.

Solution;



- IP address configuration for all PCs

PC1:

IP Configuration

☐ DHCP ☒ Static

IPv4 Address

Subnet Mask

Default Gateway

DNS Server

PC2:

IP Configuration

☐ DHCP ☒ Static

IPv4 Address

Subnet Mask

Default Gateway

DNS Server

PC3:

IP Configuration

☐ DHCP ☒ Static

IPv4 Address

Subnet Mask

Default Gateway

DNS Server

PC4:

IP Configuration

☐ DHCP ☒ Static

IPv4 Address

Subnet Mask

Default Gateway

DNS Server

PC5:

IP Configuration

☐ DHCP ☒ Static

IPv4 Address: 192.168.0.3

Subnet Mask: 255.255.255.0

Default Gateway: 0.0.0.0

DNS Server: 0.0.0.0

PC6;

IP Configuration

☐ DHCP ☒ Static

IPv4 Address: 192.168.0.4

Subnet Mask: 255.255.255.0

Default Gateway: 0.0.0.0

DNS Server: 0.0.0.0

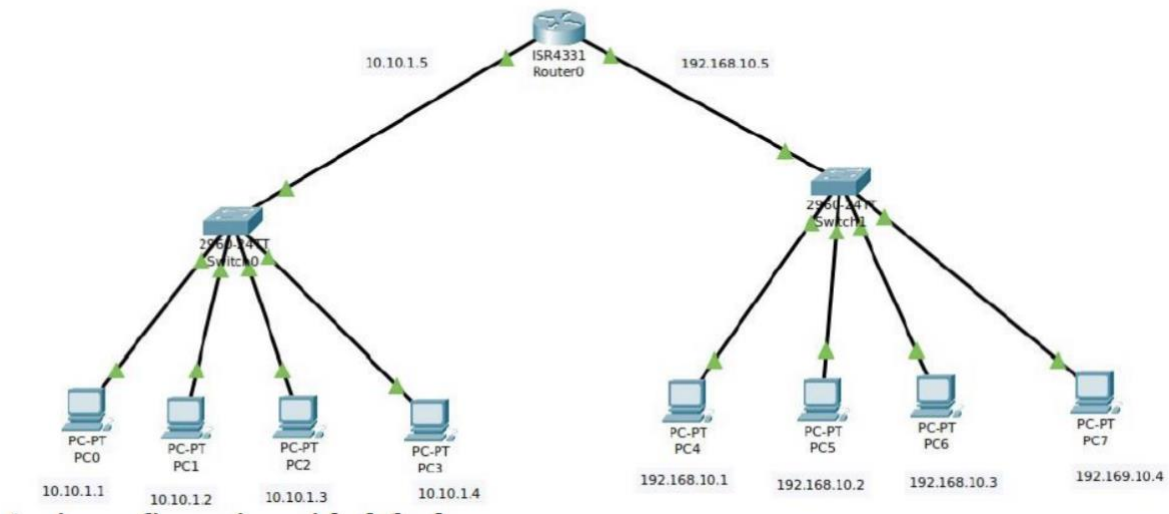
Question no. 4

4. Implement an IP configuration using Cisco Packet tracer for the given scenario.  
Show the output ( Screen Shot) of the following:

Create a two LAN using CISCO packet tracer simulator. Four devices connected with one switch of each LAN and establish the connection between two LAN (Different Class IP) two switch and one Router using CISCO Packet tracer simulator and command Prompt.

- a. IP address configuration with default gateway for all PCs.
- b. IP Configuration of Router
- c. Ping command for both the network 1 and 2 (Minimum 5 commands)

d. Show the message trasmitted from  
 PC0 to PC3  
 PC5 to PC7  
 PC6 to PC3  
 PC1 to PC5



Solution;

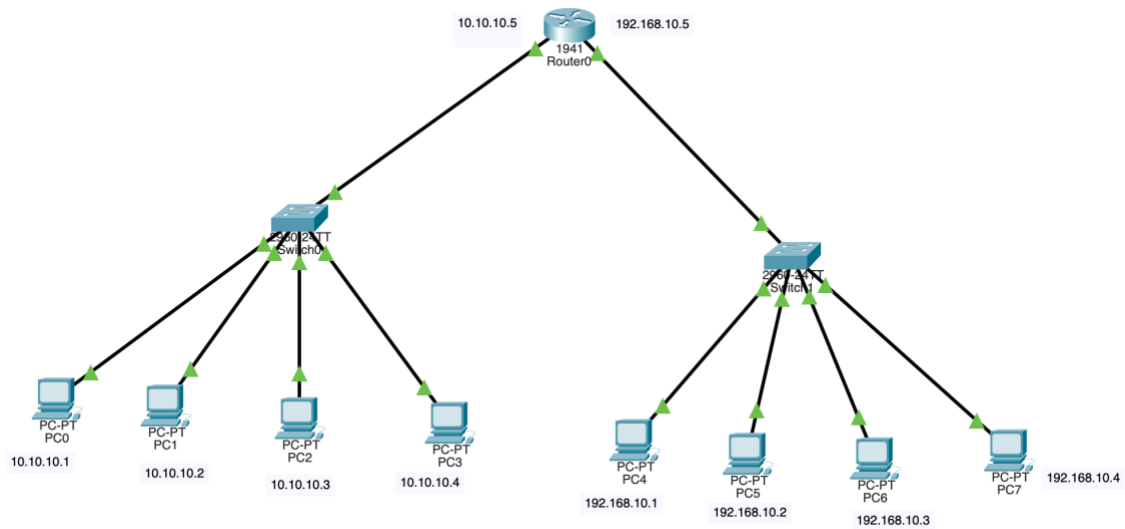


FIG 1

a. IP Configuration and Default Gateway for PCS:

Pc0:

IP Configuration	
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	10.10.10.1
Subnet Mask	255.0.0.0
Default Gateway	10.10.10.5

Pc1:

IP Configuration		
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static	Another device has attempted to use this IP address.
IPv4 Address	10.10.10.2	
Subnet Mask	255.0.0.0	
Default Gateway	10.10.10.5	

Pc2:

IP Configuration		
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static	This address is already used in the network.
IPv4 Address	10.10.10.3	
Subnet Mask	255.0.0.0	
Default Gateway	10.10.10.5	

Pc3:

IP Configuration		
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static	This address is already used in the network.
IPv4 Address	10.10.10.4	
Subnet Mask	255.0.0.0	
Default Gateway	10.10.10.5	

Pc4:

IP Configuration	
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	192.168.10.1
Subnet Mask	255.255.255.0
Default Gateway	192.168.10.5

Pc5:

IP Configuration	
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	192.168.10.2
Subnet Mask	255.255.255.0
Default Gateway	192.168.10.5

Pc6:

IP Configuration	
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	192.168.10.3
Subnet Mask	255.255.255.0
Default Gateway	192.168.10.5

Pc7:

IP Configuration	
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	192.168.10.3
Subnet Mask	255.255.255.0
Default Gateway	192.168.10.5

b. IP Configuration of Router

MAC Address	0002.16A0.1101
IP Configuration	
IPv4 Address	10.10.10.5
Subnet Mask	255.0.0.0
Tx Ring Limit	10
MAC Address	0002.16A0.1102
IP Configuration	
IPv4 Address	192.168.10.5
Subnet Mask	255.255.255.0
Tx Ring Limit	10

c. Ping command for both network 1 and 2  
Network 1

```
C:\>ping 192.168.10.4
```

```
Pinging 192.168.10.4 with 32 bytes of data:
```

```
Reply from 192.168.10.4: bytes=32 time<1ms TTL=127
```

```
Reply from 192.168.10.4: bytes=32 time<1ms TTL=127
```

```
Reply from 192.168.10.4: bytes=32 time<1ms TTL=127
```

```
Reply from 192.168.10.4: bytes=32 time<1ms TTL=127
```

```
Ping statistics for 192.168.10.4:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

```
Approximate round trip times in milli-seconds:
```

```
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

```
C:\>ping 10.10.10.4
```

```
Pinging 10.10.10.4 with 32 bytes of data:
```

```
Reply from 10.10.10.4: bytes=32 time<1ms TTL=128
```

```
Reply from 10.10.10.4: bytes=32 time<1ms TTL=128
```

```
Reply from 10.10.10.4: bytes=32 time<1ms TTL=128
```

```
Reply from 10.10.10.4: bytes=32 time<1ms TTL=128
```

```
Ping statistics for 10.10.10.4:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

```
Approximate round trip times in milli-seconds:
```

```
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

```
C:\>ping 192.168.10.2
```

```
Pinging 192.168.10.2 with 32 bytes of data:
```

```
Reply from 192.168.10.2: bytes=32 time<1ms TTL=127
```

```
Reply from 192.168.10.2: bytes=32 time<1ms TTL=127
```

```
Reply from 192.168.10.2: bytes=32 time<1ms TTL=127
```

```
Reply from 192.168.10.2: bytes=32 time<1ms TTL=127
```

```
Ping statistics for 192.168.10.2:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

```
Approximate round trip times in milli-seconds:
```

```
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

```
C:\>ping 192.168.10.1

Pinging 192.168.10.1 with 32 bytes of data:

Request timed out.
Reply from 192.168.10.1: bytes=32 time<1ms TTL=127
Reply from 192.168.10.1: bytes=32 time=1ms TTL=127
Reply from 192.168.10.1: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.10.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

```
C:\>ping 10.10.10.1

Pinging 10.10.10.1 with 32 bytes of data:

Reply from 10.10.10.1: bytes=32 time=17ms TTL=128
Reply from 10.10.10.1: bytes=32 time=24ms TTL=128
Reply from 10.10.10.1: bytes=32 time=16ms TTL=128
Reply from 10.10.10.1: bytes=32 time=23ms TTL=128

Ping statistics for 10.10.10.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 16ms, Maximum = 24ms, Average = 20ms
```

## Network 2

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.10.10.1

Pinging 10.10.10.1 with 32 bytes of data:

Reply from 10.10.10.1: bytes=32 time<1ms TTL=127
Reply from 10.10.10.1: bytes=32 time<1ms TTL=127
Reply from 10.10.10.1: bytes=32 time<1ms TTL=127
Reply from 10.10.10.1: bytes=32 time<1ms TTL=127

Ping statistics for 10.10.10.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```



```
C:\>ping 10.10.10.2
```

```
Pinging 10.10.10.2 with 32 bytes of data:
```

```
Reply from 10.10.10.2: bytes=32 time<1ms TTL=127  
Reply from 10.10.10.2: bytes=32 time<1ms TTL=127  
Reply from 10.10.10.2: bytes=32 time<1ms TTL=127  
Reply from 10.10.10.2: bytes=32 time<1ms TTL=127
```

```
Ping statistics for 10.10.10.2:
```

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

```
C:\>ping 192.168.10.3
```

```
Pinging 192.168.10.3 with 32 bytes of data:
```

```
Reply from 192.168.10.3: bytes=32 time<1ms TTL=128  
Reply from 192.168.10.3: bytes=32 time<1ms TTL=128  
Reply from 192.168.10.3: bytes=32 time=22ms TTL=128  
Reply from 192.168.10.3: bytes=32 time<1ms TTL=128
```

```
Ping statistics for 192.168.10.3:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
    Minimum = 0ms, Maximum = 22ms, Average = 5ms
```

```
C:\>ping 10.10.10.4
```

```
Pinging 10.10.10.4 with 32 bytes of data:
```

```
Reply from 10.10.10.4: bytes=32 time<1ms TTL=127  
Reply from 10.10.10.4: bytes=32 time<1ms TTL=127  
Reply from 10.10.10.4: bytes=32 time<1ms TTL=127  
Reply from 10.10.10.4: bytes=32 time<1ms TTL=127
```

```
Ping statistics for 10.10.10.4:
```

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

```

C:\>ping 192.168.10.2

Pinging 192.168.10.2 with 32 bytes of data:

Reply from 192.168.10.2: bytes=32 time<1ms TTL=128
Reply from 192.168.10.2: bytes=32 time<1ms TTL=128
Reply from 192.168.10.2: bytes=32 time<1ms TTL=128
Reply from 192.168.10.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

```

- d. . Show the message transmitted from
- PC0 to PC3
  - PC5 to PC7
  - PC6 to PC3
  - PC1 to PC5

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC3	IC...		0.000	N	0	(...	(delete)
	Successful	PC5	PC7	IC...		0.000	N	1	(...	(delete)
	Successful	PC6	PC3	IC...		0.000	N	2	(...	(delete)
	Successful	PC1	PC5	IC...		0.000	N	3	(...	(delete)

## FLOW CONTROL

### Exercise: 1

Implement the Stop and Wait Protocol ARQ for noisy channel and STOP and Wait protocol for Noiseless channel

### SAMPLE OUTPUT

Sender has to send frames: 1 2 3 4 5 6 7

Sender	Receiver
--------	----------

Sending Frame: 1	Received Frame: 1 Acknowledgement: 1
------------------	--------------------------------------

Sending Frame: 2	--- Timeout
------------------	-------------

Sending Frame: 2	Received Frame: 2 Acknowledgement: 2
------------------	--------------------------------------

Sending Frame: 3	Received Frame: 3 Acknowledgement: 3
------------------	--------------------------------------

Sending Frame: 4	Received Frame: 4 Acknowledgement: 4
------------------	--------------------------------------

Sending Frame: 5	Received Frame: 5 Acknowledgement: 5
------------------	--------------------------------------

Sending Frame: 6	Received Frame: 6 Acknowledgement: 6
------------------	--------------------------------------

Sending Frame: 7	Received Frame: 7 Timeout
------------------	---------------------------

Sending Frame: 7	Received Frame: 7 Duplicate, Discard Acknowledgement: 7
------------------	--

Solution;

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

```
#include <unistd.h>
```

```
void send_frame(int frame) {
```

```
    printf("Sending Frame: %d\n", frame);
```

```
    sleep(1);  
  
}
```

```
void receive_frame(int frame) {  
  
    printf("Received Frame: %d\n", frame);  
  
    sleep(1);  
  
}
```

```
void send_acknowledgement(int ack) {  
  
    printf("Acknowledgement: %d\n", ack);  
  
    sleep(1);  
  
}
```

```
void stop_and_wait_protocol(int frames[], int len) {  
  
    int i = 0;  
  
    while (i < len) {  
  
        int frame = frames[i];  
  
        send_frame(frame);  
  
        receive_frame(frame);  
  
        i++;  
    }  
}
```

```
send_acknowledgement(frame);
```

```
i++;
```

```
if (i < len) {
```

```
    send_frame(frames[i]);
```

```
    receive_frame(frames[i]);
```

```
    send_acknowledgement(frames[i]);
```

```
    i++;
```

```
} else {
```

```
    break;
```

```
}
```

```
}
```

```
}
```

```
int main() {
```

```
    int frames_to_send[] = {1, 2, 3, 4, 5, 6, 7};
```

```
    int len = sizeof(frames_to_send) / sizeof(frames_to_send[0]);
```

```
stop_and_wait_protocol(frames_to_send, len);
```

```
return 0;
```

```
}
```

```
Sending Frame: 1
Received Frame: 1
Acknowledgement: 1
Sending Frame: 2
Received Frame: 2
Acknowledgement: 2
Sending Frame: 3
Received Frame: 3
Acknowledgement: 3
Sending Frame: 4
Received Frame: 4
Acknowledgement: 4
Sending Frame: 5
Received Frame: 5
Acknowledgement: 5
Sending Frame: 6
Received Frame: 6
Acknowledgement: 6
Sending Frame: 7
Received Frame: 7
Acknowledgement: 7
```

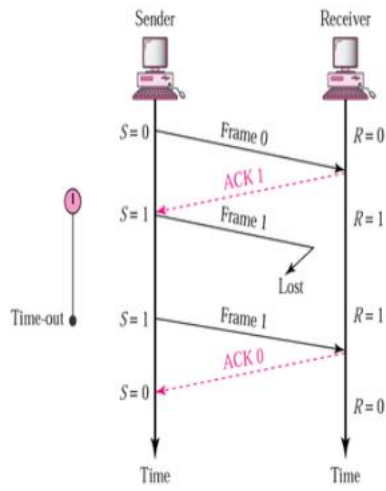
---

### Exercise 2: Implementation of Stop and Wait ARQ

Case 1: Frame lost

Case 2 – ACK lost

Case 1: Frames lost:



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

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

```
#include <time.h>
```

```
void send_frame(int frame_number) {
```

```
    printf("Sending Frame %d\n", frame_number);
```

```
    sleep(1);
```

```
}
```

```
int receive_ack(int expected_ack) {
```

```
int ack = rand() % 2; // Ensures random ack is 0 or 1

sleep(1);

if (ack == expected_ack) {

    printf("ACK %d received\n", ack);

    return 1;

} else {

    printf("ACK %d received, expected ACK %d\n", ack, expected_ack);

    return 0;

}

}
```

```
void stop_and_wait_arq(int total_frames) {
```

```
    int frame_number = 0;
```

```
    int expected_ack = 0;
```

```
    int frames_sent = 0;
```

```
    while (frames_sent < total_frames) {
```

```
        send_frame(frame_number);
```



```
if (receive_ack(expected_ack)) {  
  
    frame_number = (frame_number + 1) % 2;  
  
    expected_ack = (expected_ack + 1) % 2;  
  
    frames_sent++;  
  
} else {  
  
    printf("Frame %d lost. Retransmitting Frame %d\n", frame_number, frame_number);  
  
}  
  
}  
  
}
```

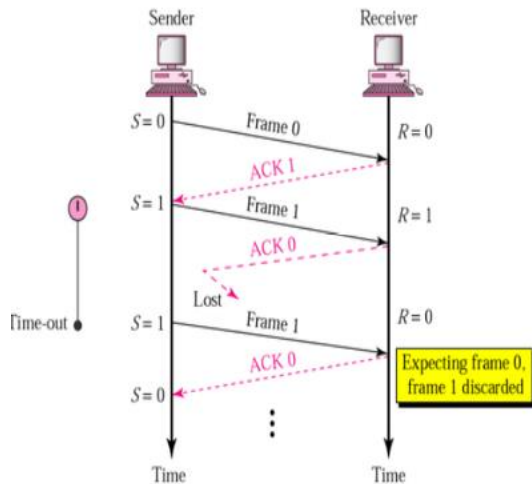
```
int main() {  
  
    srand(time(0));  
  
  
  
    int total_frames = 7; // Total number of frames to send  
  
    stop_and_wait_arq(total_frames);  
  
  
    return 0;  
  
}
```

```

Sending Frame 0
ACK 1 received, expected ACK 0
Frame 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 0 received
Sending Frame 1
ACK 0 received, expected ACK 1
Frame 1 lost. Retransmitting Frame 1
Sending Frame 1
ACK 0 received, expected ACK 1
Frame 1 lost. Retransmitting Frame 1
Sending Frame 1
ACK 0 received, expected ACK 1
Frame 1 lost. Retransmitting Frame 1
Sending Frame 1
ACK 1 received
Sending Frame 0
ACK 1 received, expected ACK 0
Frame 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 1 received, expected ACK 0
Frame 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 0 received
Sending Frame 1
ACK 0 received, expected ACK 1
Frame 1 lost. Retransmitting Frame 1
Sending Frame 1
ACK 0 received, expected ACK 1
Frame 1 lost. Retransmitting Frame 1
Sending Frame 1
ACK 1 received
Sending Frame 0
ACK 0 received
Sending Frame 1
ACK 1 received
Sending Frame 0
ACK 0 received

```

### Case 2: ACK Lost



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

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

```
#include <time.h>
```

```
void send_frame(int frame_number) {
```

```
    printf("Sending Frame %d\n", frame_number);
```

```
    sleep(1);
```

```
}
```

```
int receive_ack(int expected_ack) {
```

```
    int ack = rand() % 2;
```

```
    sleep(1);
```

```
    if (ack == expected_ack) {
```

```
        printf("ACK %d received\n", ack);
```

```
        return 1;
```

```
    } else {
```

```
        printf("ACK %d lost\n", ack);
```

```
        return 0;
```

```
}  
  
}
```

```
void stop_and_wait_arq(int total_frames) {
```

```
    int frame_number = 0;
```

```
    int expected_ack = 0;
```

```
    int frames_sent = 0;
```

```
    while (frames_sent < total_frames) {
```

```
        send_frame(frame_number);
```

```
        if (receive_ack(expected_ack)) {
```

```
            frame_number = (frame_number + 1) % 2;
```

```
            expected_ack = (expected_ack + 1) % 2;
```

```
            frames_sent++;
```

```
        } else {
```

```
            printf("ACK %d lost. Retransmitting Frame %d\n", expected_ack, frame_number);
```

```
        }
```

```
    }
```

```
}
```

```
int main() {
```

```
    srand(time(0));
```

```
    int total_frames = 7; // Total number of frames to send
```

```
    stop_and_wait_arq(total_frames);
```

```
    return 0;
```

```
}
```

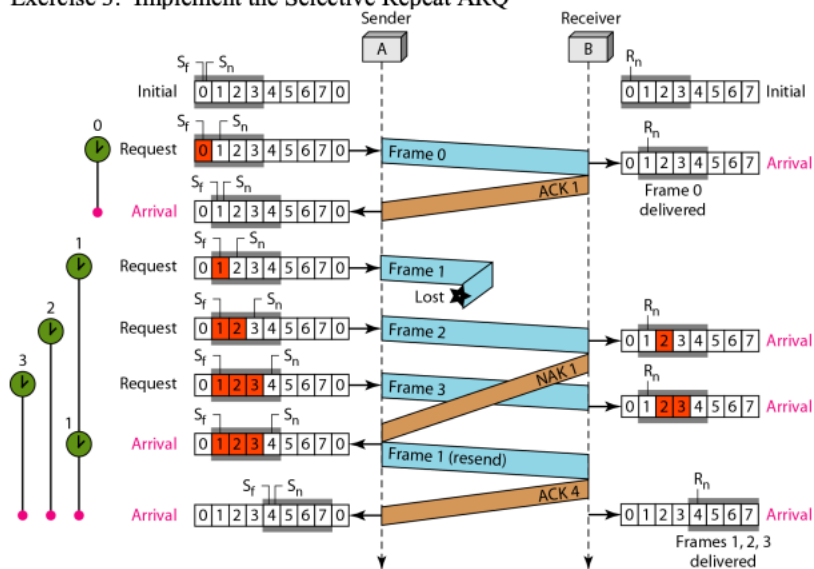
```

Sending Frame 0
ACK 1 lost
ACK 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 1 lost
ACK 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 1 lost
ACK 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 1 lost
ACK 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 1 lost
ACK 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 0 received
Sending Frame 1
ACK 1 received
Sending Frame 0
ACK 0 received
Sending Frame 1
ACK 1 received
Sending Frame 0
ACK 0 received
Sending Frame 1
ACK 0 lost
ACK 1 lost. Retransmitting Frame 1
Sending Frame 1
ACK 1 received
Sending Frame 0
ACK 1 lost
ACK 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 1 lost
ACK 0 lost. Retransmitting Frame 0
Sending Frame 0
ACK 0 received

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

```

### Exercise 3: Implement the Selective Repeat ARQ



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

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

```
#include <time.h>
```

```
void send_frame(int frame_number) {
```

```
    printf("Sending Frame %d\n", frame_number);
```

```
    sleep(1);
```

```
}
```

```
int receive_ack(int expected_ack) {
```

```
    int ack = (rand() % 2 == 0) ? expected_ack : -1; // Randomly generate ACK or NAK
```

```
    sleep(1);
```

```
    if (ack == expected_ack) {
```

```
        printf("ACK %d\n", ack);
```

```
        return 1;
```

```
    } else {
```

```
        printf("NAK %d\n", expected_ack);
```

```
        return 0;

    }

}
```

```
void selective_repeat_arq() {

    int window_size = 3;

    int base = 0;

    int next_sequence_number = 0;

    int frames[] = {0, 1, 2, 3, 4, 5, 6, 7};

    int total_frames = sizeof(frames) / sizeof(frames[0]);

    while (base < total_frames) {

        if (next_sequence_number < base + window_size && next_sequence_number < total_frames) {

            int frame_number = frames[next_sequence_number];

            send_frame(frame_number);

            next_sequence_number++;

        }

        if (receive_ack(base)) {
```



```

        base++;

    } else {

        printf("Resending Frame %d\n", frames[base]);

    }

}

}

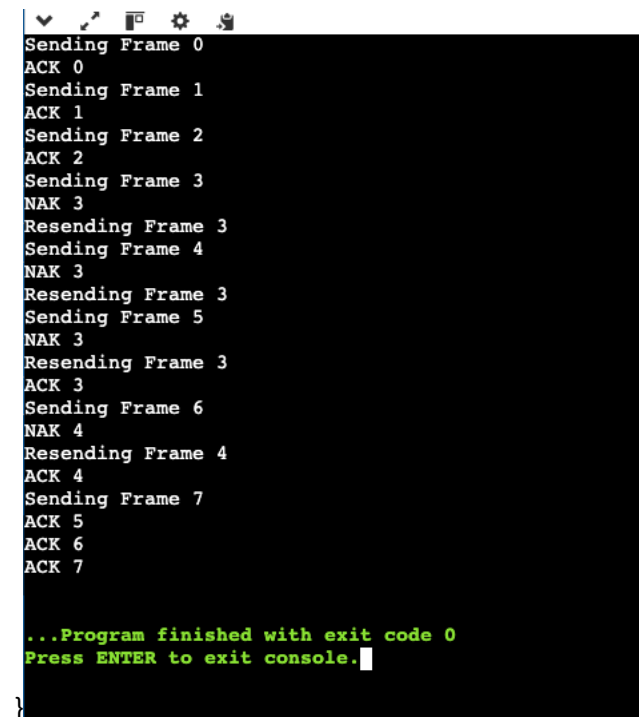
int main() {

    srand(time(0)); // Seed the random number generator

    selective_repeat_arq();

    return 0;

```



```

Sending Frame 0
ACK 0
Sending Frame 1
ACK 1
Sending Frame 2
ACK 2
Sending Frame 3
NAK 3
Resending Frame 3
Sending Frame 4
NAK 3
Resending Frame 3
Sending Frame 5
NAK 3
Resending Frame 3
ACK 3
Sending Frame 6
NAK 4
Resending Frame 4
ACK 4
Sending Frame 7
ACK 5
ACK 6
ACK 7

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

```