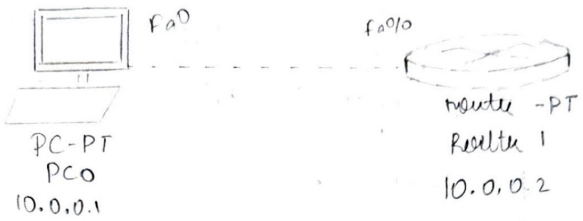
Week-4 18/8/23

Aim: To understand the operation of TELNET by accessing the router in server from a PC in IT office.

Topology:



Procedure:

Step 1: Create a topology given above.

Step 2: Set the IP address & gateway as 10.0.0.1 are 10.0.0.2 resp

Step 3: In router, in CLI,

```
router > enable
router# config t
router(config)# hostname r1
r1(config)# enable secret p1
r1(config)# interface fa0/0
r1(config-if)# ip address 10.0.0.2 255.0.0.0
r1(config-if)# no shut
r1(config-if)# line vty 0 5
r1(config-line)# login
r1(config-line)# password p0
r1(config-line)# exit
r1(config)# exit
```

Step 4: In command prompt of PC,

PC > ping 10.0.0.2

pinging 10.0.0.2 with 32 bytes of data:

reply from 10.0.0.2 : bytes = 32 time = 0ms TTL = 255
reply from 10.0.0.2 : bytes = 32 time = 0ms TTL = 255
reply from 10.0.0.2 : bytes = 32 time = 0ms TTL = 255
reply from 10.0.0.2 : bytes = 32 time = 0ms TTL = 255

ping statistics for 10.0.0.2:

packets : sent = 4, recieved = 4, lost = 0 (0% loss)

PC > telnet 10.0.0.2

trying 10.0.0.2 ... Open

User Access verification

password: (p0)

r1 > enable

password : (p1)

r1 # show ip route

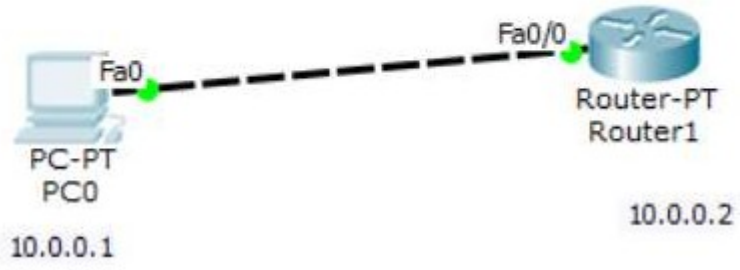
codes : C - connected

C 10.0.0.0/8 is directly connected, FastEthernet 0/0

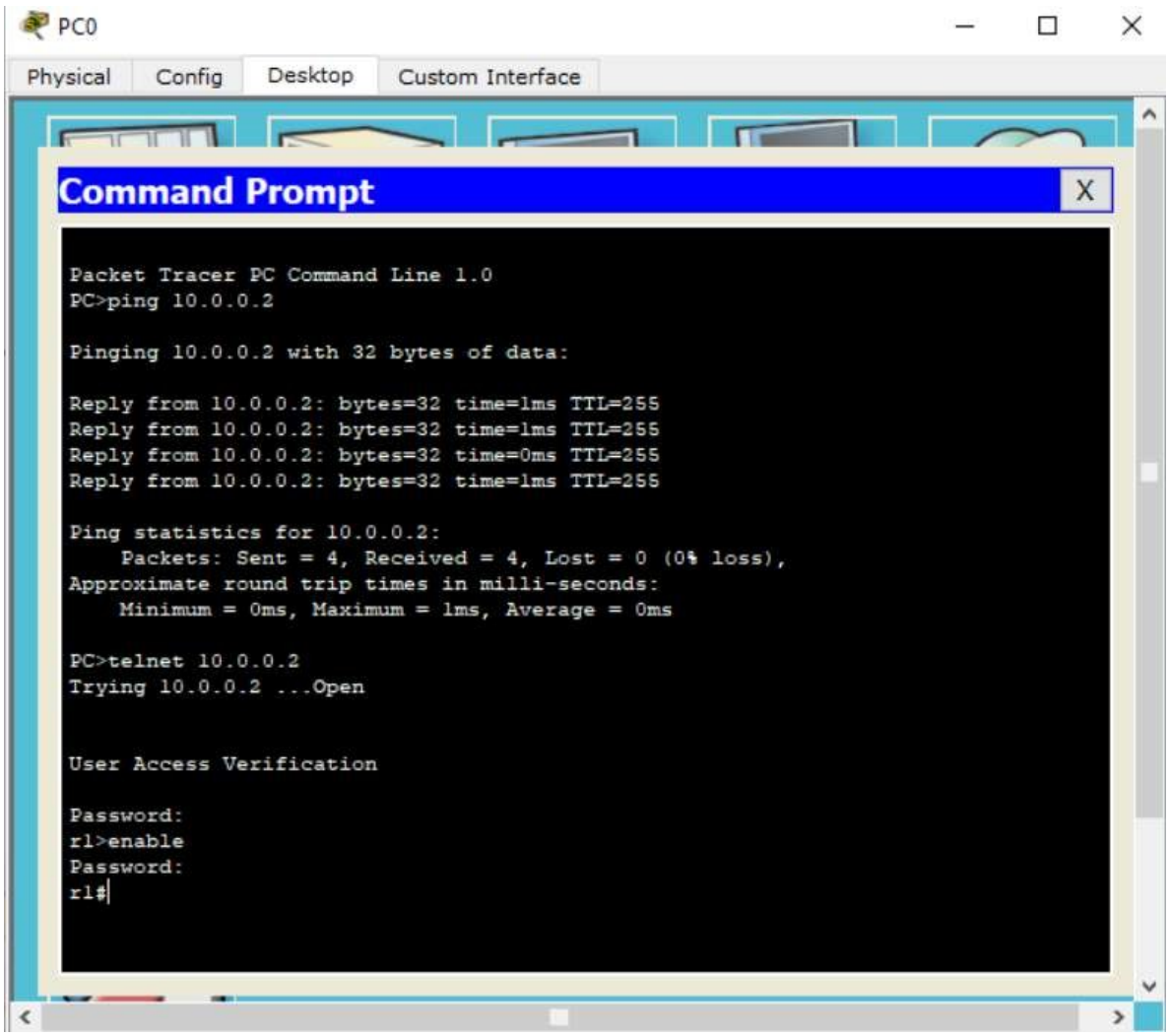
6/10

9/9/23

Topology:



Output:



Cycle-2

1) Write a program for error detecting code using CRC-CCITT

```
#include <stdio.h>
char m[50], g[50], r[50], q[50], temp[50]
void callxors(int);
void crc(int);
void calram();
void shiftl();
void main() {
int n, i=0;
char ch, flag=0;
printf("Enter frame bits:");
while((ch=getc(stdin))!='\n')
  m[i++]=ch;
  n=i;
  for(i=0;i<16;i++)
    m[n++]='0';
    m[n]='\0';
  printf("message after appending 16 zeros: %s", m);
  for(i=0; i<=16; i++)
    g[i]='0';
    g[0]=g[4]=g[11]=g[16]='1'; g[17]='\0';
  printf("\n quotient: %s", q);
  callxors(n);
  printf("\n transmitted frame: %s", m);
  printf("\n Enter transmitted frame:");
```

```
Scanf("%s",m);
printf("crc checking\n");
crc(n);
printf("\n\n Last remainder:%s",r);
for(i=0;i<16;i++){
    if(r[i]=='0')
     flag=1;
    else
      continue;
    if(flag==1)
      printf("errors during transmission");
    else
      printf("\n Recieved frame is correct");
  }
void crc(int n){
    int i,j;
    for(i=0;i<n;i++)
      r[i]=m[i]
    for(i=0;i<n-16;)i++)
      { if(r[0]=='1')
        {
        q[i]=1;
         xor();
        }
      else
         q[i]=0;
          shift();
       }
    r[16]=m[17+i];
    r[17]='\0';
```

```
void calram(){
  int i,j;
  for(i=1;i<16;i++)
   r[i-1]=((int)temp[i]-48)^((int)g[i]-48+48;
}
void shiftl(){
   int i;
   for(i=1;i<=16;i++)
     r[i-1]=r[i];
}
void calcrams(int n){
   int i,k=0;
   for(i=n-16;i<n;i++)
   m[i]=((int)m[i]-48)^((int)r[k++]-48)+48;
   m[i]='\0';
}
```

Output

Enter frame bits: 1011

message after appending 16 zeros: 1011 0000 0000 0000

generator: 10001 00000 00001

quotient: 1001

Transmitted frame: 1011 1011 0001 0110 1011

Enter transmitted frame: 1011 1011 0001 0110 1011

Last remainder: 0000 0000 0000 0000

Recieved frame is correct.

6/10

9/9/23

Output:

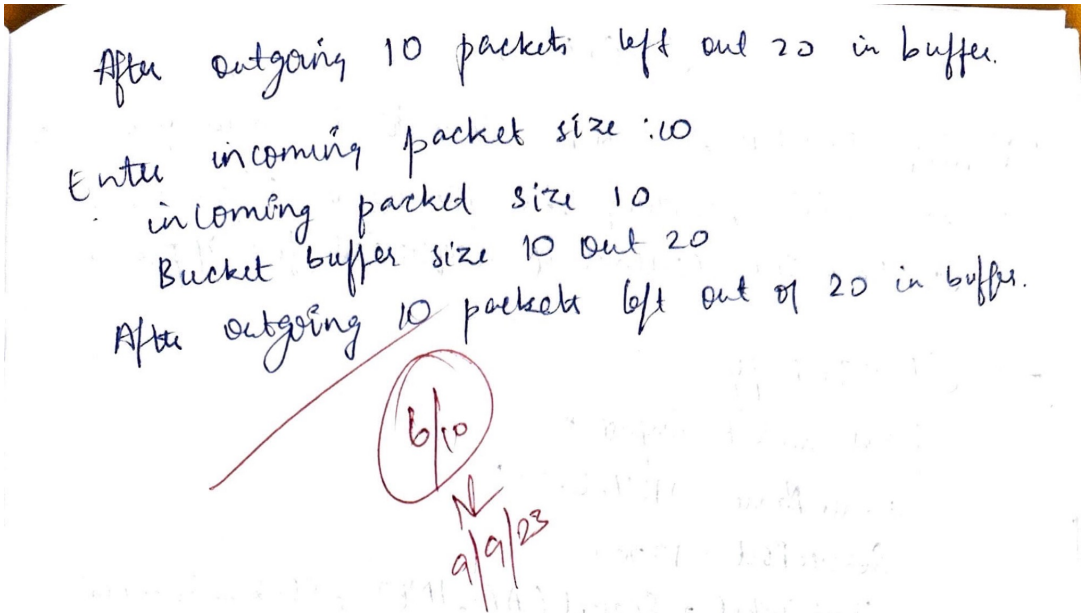
```
Enter the frame bits:1011
Message after appending 16 zeros:1011000000000000000
generator:10001000000100001


quotient:1011
transmitted frame:10111011000101101011
Enter transmitted frame:10111011000101101011
CRC checking


last remainder:0000000000000000

Received frame is correct
Process returned 0 (0x0)   execution time : 14.468 s
Press any key to continue.
```

2) Write a program for congestion control using leaky bucket algorithm

→
```
#include <stdio.h>
int main(){
int incoming, outgoing, bucket_size, n, store=0;
printf("Enter bucket_size, Outgoing_rate and no. of
  inputs: ");
Scanf("%d%d%d",&bucketsize,&outgoing,&n);
while(n!=0){
printf("Enter the incoming packet size:");
Scanf("%d",&incoming);
if(incoming <= (bucket_size - store)){
    store+=incoming;
printf("Bucket bucket size %d out of %d \n",store,
    bucket_size);
store=bucket_size;
}
store = store - outgoing;
}
store = store printf("After outgoing %d packets left
  out %d in buffer \n", store, bucket_size);
  n--;
}
return 0;
}
```

Output: Enter bucket, outgoing rate & no of input
20 10 2

Enter incoming packet size: 30
incoming packet size: 30
dropped 10 no of packets
Bucket buffer size 0 out 20



Output:

```
Enter bucket size, outgoing rate and no of inputs: 20 10 2
Enter the incoming packet size : 30
Incoming packet size 30
Dropped 10 no of packets
Bucket buffer size 0 out of 20
After outgoing 10 packets left out of 20 in buffer
Enter the incoming packet size : 10
Incoming packet size 10
Bucket buffer size 20 out of 20
After outgoing 10 packets left out of 20 in buffer

Process returned 0 (0x0)   execution time : 22.003 s
Press any key to continue.
```