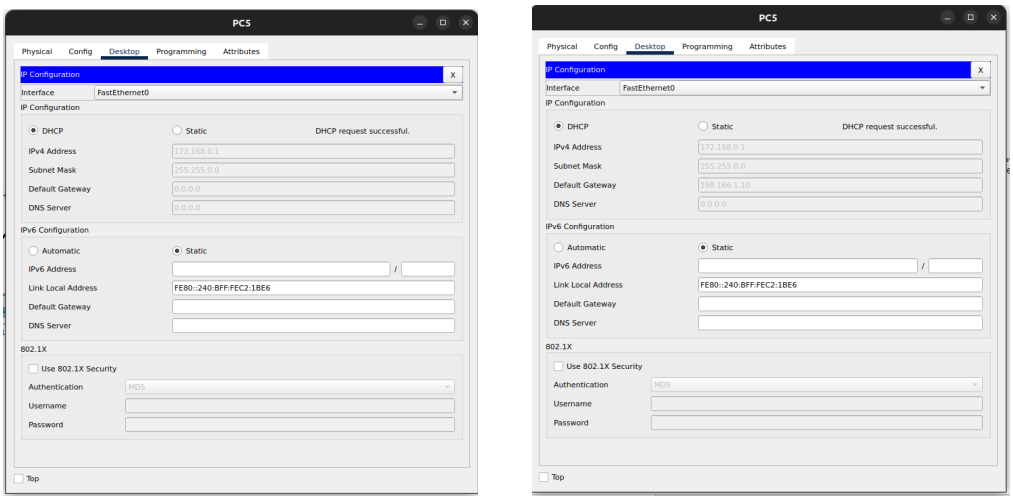
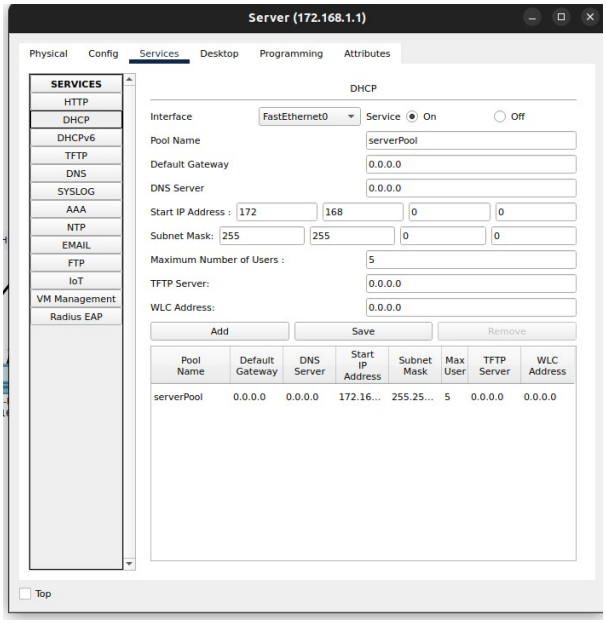
COMPUTER NETWORKS
LAB – 3

ROHAN G
CS22B1093

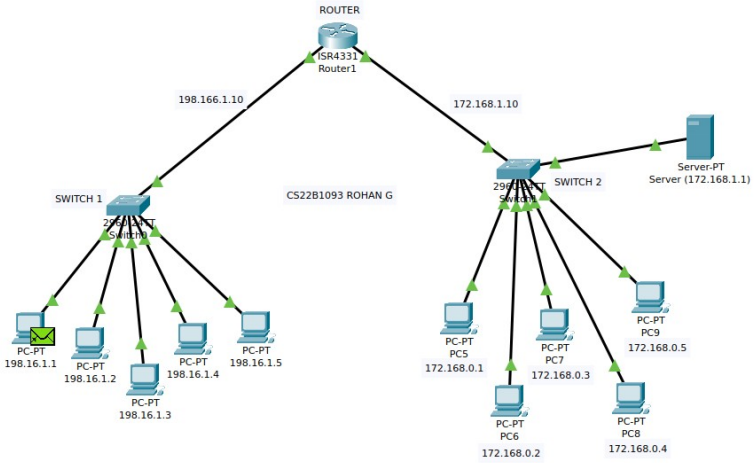
Q1)



PC getting assigned IP by DHCP server . Before and after configuring subnets.



DHCP Server Configuration

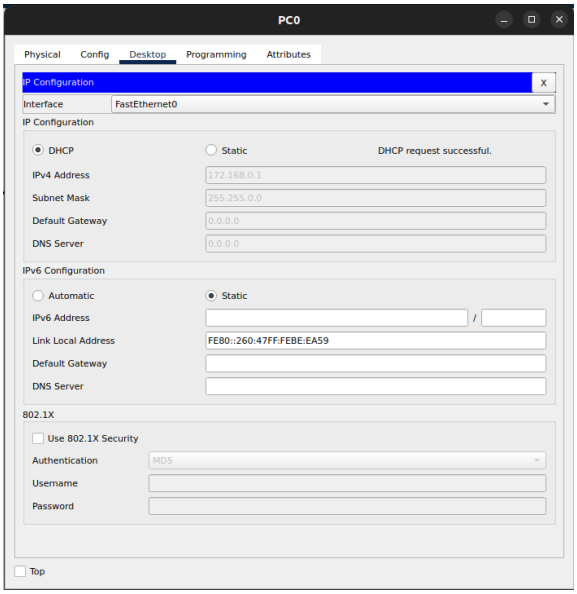


Entire Network , with one LAN PC's getting assigned IP's through DHCP server

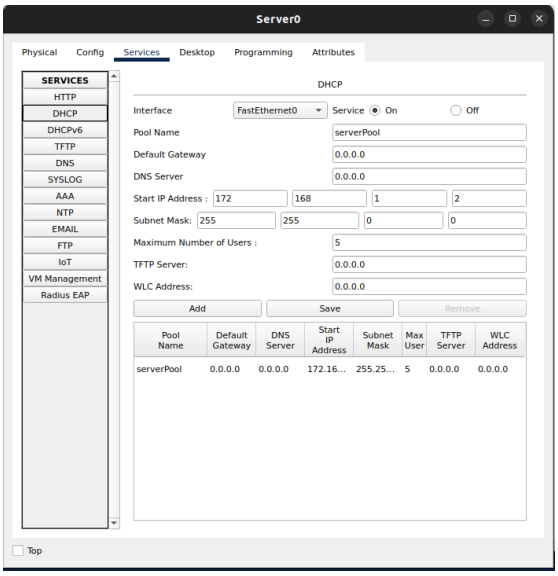
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	198....	PC5	IC...		0.000	N	0	(e...	(delete)
	Successful	198....	PC7	IC...		0.000	N	1	(e...	(delete)
	Successful	198....	PC7	IC...		0.000	N	2	(e...	(delete)

Packet transmission between different PC's

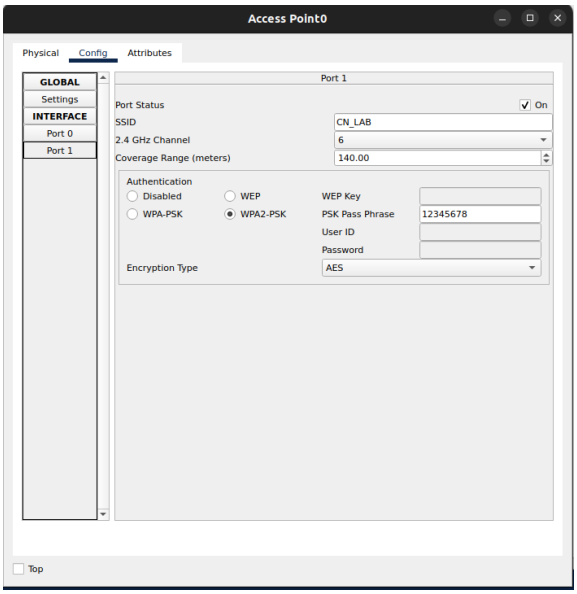
Q2)



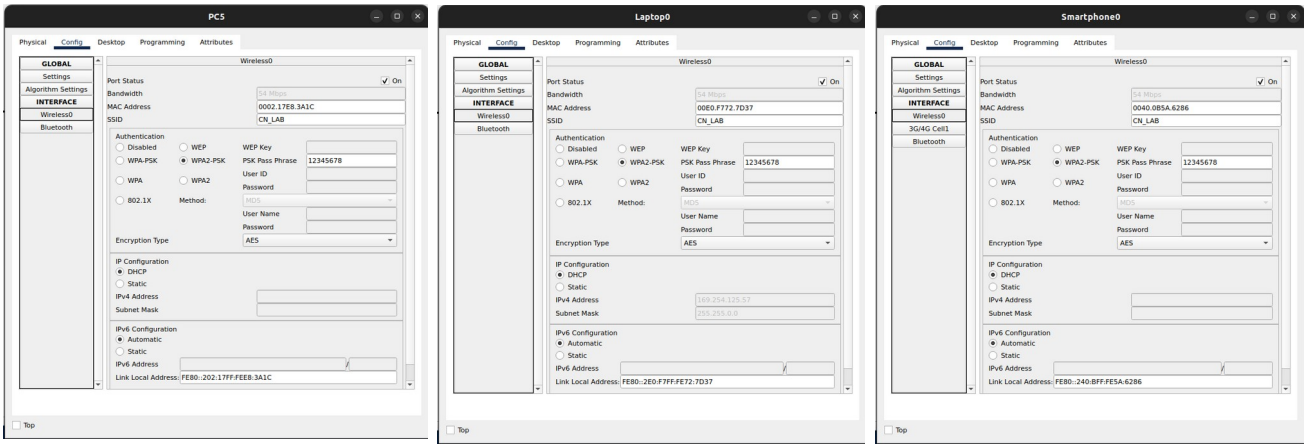
Wired LAN PC's IP Configuration through DHCP



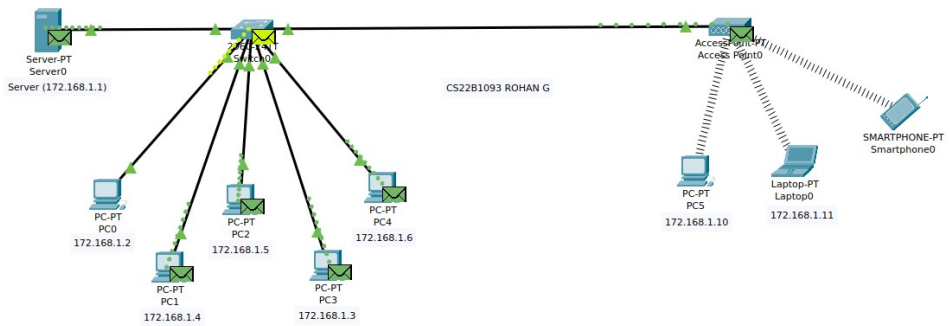
DHCP Server Configuration



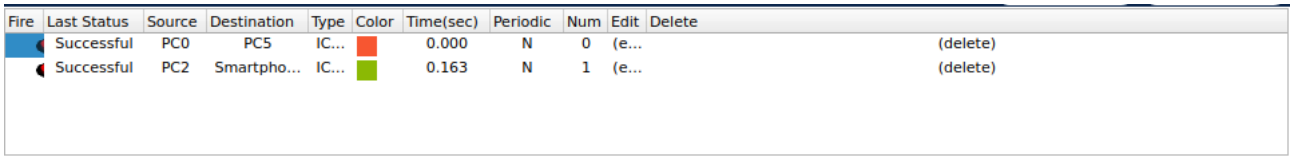
Setting the Access Point to WPA2-PSK



Wireless Configuration of 3 different devices connected to the Wireless LAN



Entire Network , with wired LAN network being assigned IP’s through DHCP server.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC5	IC...		0.000	N	0	(e...	(delete)
	Successful	PC2	Smartpho...	IC...		0.163	N	1	(e...	(delete)

Packet Transmission between the different devices in the network

Q3)

```
//CS22B1093
//ROHAN G

#include <stdio.h>
#include <stdlib.h>

#define MAX_INPUT 50
#define MAX 200

char *string_to_stuffed(char *arr)
{
   char *stuffed = (char *)malloc(sizeof(char) * MAX);
   int count = 0;
   int i = 0, j = 0;

   stuffed[j++] = '0';
   for (int k = 0; k < 6; k++) {
      stuffed[j++] = '1';
   }
   stuffed[j++] = '0';

   for (int z = 0; arr[z] != '\0'; z++) {
      if (arr[z] == '1') {
         count++;
         stuffed[j++] = '1';
         if (count == 5) {
            stuffed[j++] = '0';
            count = 0;
         }
      } else {
         stuffed[j++] = '0';
         count = 0;
      }
   }

   stuffed[j++] = '0';
   for (int k = 0; k < 6; k++) {
      stuffed[j++] = '1';
   }
   stuffed[j++] = '0';

   stuffed[j] = '\0';
   printf("Stuffed string: %s\n", stuffed);
   return stuffed;
```

```
}

char *stuffed_to_string(char *stuffed)
{
    char *output = (char *)malloc(sizeof(char) * MAX_INPUT);
    int count = 0;
    int j = 0;

    int i = 8;

    while (stuffed[i] != '\0') {
        if (stuffed[i] == '1') {
            count++;
            output[j++] = '1';
            if (count == 5) {
                i++;
                count = 0;
            }
        } else {
            output[j++] = '0';
            count = 0;
        }
        i++;
    }

    output[j] = '\0';

    output[j - 7] = '\0';

    printf("Unstuffed string: %s\n", output);
    return output;
}

int main()
{
    char arr[MAX_INPUT];
    printf("Enter the string: ");
    scanf("%s", arr);

    printf("Input string: %s\n", arr);

    char *stuffed = string_to_stuffed(arr);
    char *output = stuffed_to_string(stuffed);

    free(stuffed);
    free(output);
    return 0;
}
```

T1:

```
[~/sem5/cn/lab3]
rzeta ./a
Enter the string: 10000101
Input string: 10000101
Stuffed string: 011111101000010101111110
Unstuffed string: 10000101
```

T2:

```
[~/sem5/cn/lab3]
rzeta ./a
Enter the string: 111111111
Input string: 111111111
Stuffed string: 0111111011111011110111111(
Unstuffed string: 111111111
```

T3:

```
[~/sem5/cn/lab3]
rzeta ./a
Enter the string: 11111111111111111
Input string: 1111111111111111
Stuffed string: 011111101111101111101111101101111110
Unstuffed string: 1111111111111111
```

T4:

```
[~/sem5/cn/lab3]
rzeta ./a
Enter the string: 1000011111
Input string: 1000011111
Stuffed string: 011111101000011111001111110
Unstuffed string: 1000011111
```

T5:

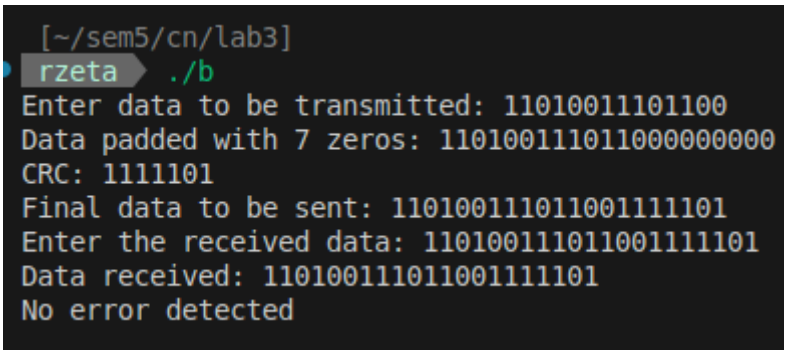
```
[~/sem5/cn/lab3]
rzeta ./a
Enter the string: 111111111111111111
Input string: 111111111111111111
Stuffed string: 0111111011111011111011111011101111110
Unstuffed string: 111111111111111111
```

Q4)

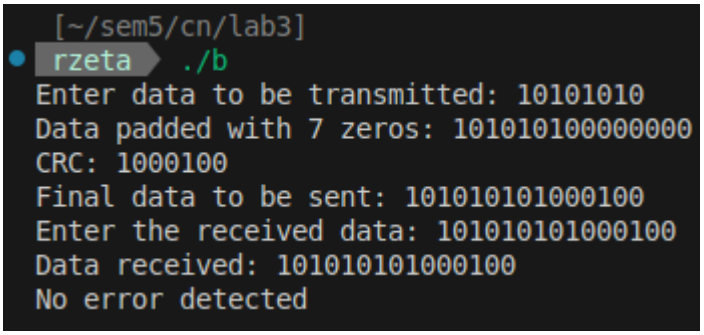
```
//CS22B1093
//ROHAN G

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

#define DIVISOR "10000111"
#define DIVISOR_LENGTH 8

char dataToSend[28];
char crcValue[28];
char generatorPolynomial[DIVISOR_LENGTH + 1] = DIVISOR;

int originalDataLength, i, j;

void performXOR();
void computeCRC();
void verifyReceivedData();

void performXOR() {
   for (j = 1; j < DIVISOR_LENGTH; j++)
      crcValue[j] = ((crcValue[j] == generatorPolynomial[j]) ? '0' : '1');
}

void computeCRC() {
   for (i = 0; i < DIVISOR_LENGTH; i++)
      crcValue[i] = dataToSend[i];

   do {
      if (crcValue[0] == '1')
         performXOR();

      for (j = 0; j < DIVISOR_LENGTH - 1; j++)
         crcValue[j] = crcValue[j + 1];

      crcValue[j] = dataToSend[i++];
   } while (i <= originalDataLength + DIVISOR_LENGTH - 1);
}

void verifyReceivedData() {
   printf("\nEnter the received data: ");
   scanf("%s", dataToSend);
   printf("Data received: %s", dataToSend);
   computeCRC();

   for (i = 0; (i < DIVISOR_LENGTH - 1) && (crcValue[i] != '1'); i++);

   if (i < DIVISOR_LENGTH - 1)
      printf("\nError detected\n\n");
   else
      printf("\nNo error detected\n\n");
```

```
}

int main() {
    printf("Enter data to be transmitted: ");
    scanf("%s", dataToSend);

    originalDataLength = strlen(dataToSend);

    for (i = originalDataLength; i < originalDataLength + DIVISOR_LENGTH - 1; i++)
        dataToSend[i] = '0';
    dataToSend[i] = '\0';

    printf("Data padded with %d zeros: %s", DIVISOR_LENGTH - 1, dataToSend);

    computeCRC();
    printf("\nCRC: %s", crcValue);

    for (i = originalDataLength; i < originalDataLength + DIVISOR_LENGTH - 1; i++)
        dataToSend[i] = crcValue[i - originalDataLength];
    dataToSend[i] = '\0';

    printf("\nFinal data to be sent: %s", dataToSend);

    verifyReceivedData();

    return 0;
}
```

T1:



T2:



T3:

```
[~/sem5/cn/lab3]
rzeta ./b
Enter data to be transmitted: 1111
Data padded with 7 zeros: 11110000000
CRC: 0101101
Final data to be sent: 11110101101
Enter the received data: 1110101101
Data received: 1110101101
Error detected
```

Error as received data is not same as sent data

T4:

```
[~/sem5/cn/lab3]
rzeta ./b
Enter data to be transmitted: 00000000
Data padded with 7 zeros: 00000000000000
CRC: 0000000
Final data to be sent: 00000000000000
Enter the received data: 00000000000000
Data received: 00000000000000
No error detected
```

T5:

```
[~/sem5/cn/lab3]
rzeta ./b
Enter data to be transmitted: 11111111
Data padded with 7 zeros: 11111111000000
CRC: 1100110
Final data to be sent: 11111111100110
Enter the received data: 11111111100110
Data received: 11111111100110
Error detected
```

Error as received data is not same as sent data

Q5)

```
//CS22B1093
//ROHAN G

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

#define POLY_LENGTH strlen(generatorPolynomial)

char dataToSend[28];
char crcValue[28];
char generatorPolynomial[10];

int originalDataLength, i, j;

void performXOR();
void computeCRC();
void verifyReceivedData();

void performXOR() {
   for (j = 1; j < POLY_LENGTH; j++)
      crcValue[j] = ((crcValue[j] == generatorPolynomial[j]) ? '0' : '1');
}

void computeCRC() {
   for (i = 0; i < POLY_LENGTH; i++)
      crcValue[i] = dataToSend[i];

   do {
      if (crcValue[0] == '1')
         performXOR();

      for (j = 0; j < POLY_LENGTH - 1; j++)
         crcValue[j] = crcValue[j + 1];

      crcValue[j] = dataToSend[i++];
   } while (i <= originalDataLength + POLY_LENGTH - 1);
}

void verifyReceivedData() {
   printf("\nEnter the received data: ");
   scanf("%s", dataToSend);
   printf("Data received: %s", dataToSend);
   computeCRC();

   for (i = 0; (i < POLY_LENGTH - 1) && (crcValue[i] != '1'); i++);

   if (i < POLY_LENGTH - 1)
      printf("\nError detected\n\n");
   else
      printf("\nNo error detected\n\n");
}
```

```
int main() {
    printf("Enter data to be transmitted: ");
    scanf("%s", dataToSend);
    printf("\nEnter the Generator Polynomial: ");
    scanf("%s", generatorPolynomial);

    originalDataLength = strlen(dataToSend);

    for (i = originalDataLength; i < originalDataLength + POLY_LENGTH - 1; i++)
        dataToSend[i] = '0';
    dataToSend[i] = '\0';

    printf("Data padded with %ld zeros: %s", POLY_LENGTH - 1, dataToSend);

    computeCRC();
    printf("\nCRC : %s", crcValue);

    for (i = originalDataLength; i < originalDataLength + POLY_LENGTH - 1; i++)
        dataToSend[i] = crcValue[i - originalDataLength];
    dataToSend[i] = '\0';

    printf("\nFinal data to be sent: %s", dataToSend);

    verifyReceivedData();

    return 0;
}
```

T1:

```
[~/sem5/cn/lab3]
rzeta ./c
Enter data to be transmitted: 11010011101100

Enter the Generator Polynomial: 10011011
Data padded with 7 zeros: 11010011011000000000
CRC : 0100110
Final data to be sent: 11010011011000100110
Enter the received data: 11010011011000100110
Data received: 11010011011000100110
No error detected
```

T2:

```
[~/sem5/cn/lab3]
rzeta ./c
Enter data to be transmitted: 10101010

Enter the Generator Polynomial: 10011011
Data padded with 7 zeros: 10101010000000
CRC : 0111100
Final data to be sent: 10101010111100
Enter the received data: 10101010111100
Data received: 10101010111100
No error detected
```

T3:

```
[~/sem5/cn/lab3]
rzeta ./c
Enter data to be transmitted: 1111

Enter the Generator Polynomial: 10011011
Data padded with 7 zeros: 11110000000
CRC : 0000010
Final data to be sent: 11110000010
Enter the received data: 11110000010
Data received: 11110000010
No error detected
```

T4:

```
[~/sem5/cn/lab3]
rzeta ./c
Enter data to be transmitted: 00000000

Enter the Generator Polynomial: 10011011
Data padded with 7 zeros: 00000000000000
CRC : 0000000
Final data to be sent: 00000000000000
Enter the received data: 00000000000000
Data received: 00000000000000
No error detected
```

T5:

```
[~/sem5/cn/lab3]
rzeta ./c
Enter data to be transmitted: 11111111

Enter the Generator Polynomial: 10011011
Data padded with 7 zeros: 11111110000000
CRC : 0100010
Final data to be sent: 11111110100010
Enter the received data: 11111110100010
Data received: 11111110100010
Error detected
```

Error as received data is not same as sent data