Redes de Computadoras
“Analizador V2 (ARP, LLC)”

Integrantes:

- Martínez Coronel Brayan Yosafat.
- Sánchez Méndez Edmundo Josue.

Fecha: 16 de enero de 2021

Profesora: Nidia Asunción Cortez Duarte

1.-Una trama IP con opciones -ICMP- Imprimir las opciones en hexadecimal

```
1.#include <stdio.h>
2.#define NO_TRAMAS 14
3.
4.unsigned char ss[][5] = {"RR", "RNR", "REJ", "SREJ"};
5.unsigned char uc[][6] = {"UI", "SIM", "-", "SARM", "UP", "-", "-", "SABM", "DIOC", "-", "-
", "SARME", "-", "-", "-", "SABME", "SNRN", "-", "-", "RSET", "-", "-", "-", "XID", "-", "-
", "-", "SNRNE"};
6.unsigned char ur[][6] = {"UI", "RIM", "-", "DM", "-", "-", "-", "-", "RD", "-", "-", "-
", "UA", "-", "-", "-", "-", "FRMR", "-", "-", "-", "-", "-", "XID", "-", "-", "-", "-"};
7.
8.void checksumIP(unsigned char *trama, unsigned char IHL)
9.{
10.    unsigned char i = 14;
11.    unsigned int valchecksum = 0x00;
12.    for (i; i < IHL + 14; i += 2)
13.    {
14.        valchecksum += (trama[i] << 8 | trama[i + 1]);
15.    }
16.    valchecksum = valchecksum -
 (trama[24] << 8 | trama[25]); //Ignorando trama[24] y trama[25]
17.    valchecksum = ~(valchecksum + (valchecksum >> 16)) & 0xFFFF;
18.    if (valchecksum == ((trama[24] << 8) | trama[25]))
19.    {
20.        printf("\n:D");
21.    }
22.    else
23.    {
24.        printf("\n:(\n");
25.        printf("Checksum calculado: %x\n", valchecksum);
26.    }
27.}
28.
29.void analizarARP(unsigned char *trama)
30.{
31.    printf("\n\t.::Cabezera ARP::.\n");
32.    printf("Tipo de hardware: %i ", ((trama[14] << 8) | trama[15]));
33.    switch (((trama[14] << 8) | trama[15]))
34.    {
35.    case 1:
36.        printf("Ethernet\n");
37.        break;
38.    case 6:
39.        printf("IEEE 802 (Token Ring)\n");
40.        break;
41.    case 15:
42.        printf("Frame Relay\n");
43.        break;
```

****

TRAMA: 14

****

CABECERA ETHERNET

MAC Destino: 00-1f-45-9d-1e-a2

MAC Origen: 00-23-8b-46-e9-ad

.::Cabezera IP::.

:(

Checksum calculado: cf31

.::ICMP::.

Opciones: 20 19 63 04 28 00 00 20 19 63 01 43

2.-Una trama IP de Costo mínimo Imprimir TTL

```
1.    case 16:
2.        printf("ATM\n");
3.        break;
4.    default:
5.        printf("-\n");
6.        break;
7.    }
8.    printf("Tipo de protocolo: %i ", (trama[16] << 8 | trama[17]));
9.    if ((trama[16] << 8 | trama[17]) == 2048)
10.    {
11.        printf("IPv4\n");
12.    }
13.    else
14.    {
15.        printf("-\n");
16.    }
17.    printf("Longitud de la direccion de hardware: %i bytes\n", trama[18]);
18.    printf("Longitud de la direccion de protocolo: %i bytes\n", trama[19]);
19.    printf("Operacion: %i ", ((trama[20] << 8) | trama[21]));
20.    switch (((trama[20] << 8) | trama[21]))
21.    {
22.    case 1:
23.        printf("ARP Request\n");
24.        break;
25.    case 2:
26.        printf("ARP Reply\n");
27.        break;
28.    case 3:
29.        printf("Inverse ARP Request\n");
30.        break;
31.    case 4:
32.        printf("Inverse ARP Reply\n");
33.        break;
34.    case 8:
35.        printf("Inverse ARP Request\n");
36.        break;
37.    case 9:
38.        printf("Inverse ARP Reply\n");
39.        break;
40.    default:
41.        printf("-\n");
42.        break;
43.    }
44.    unsigned char n = trama[18];
45.    unsigned char m = trama[19];
46.    unsigned char j = 22;
47.    printf("Direccion MAC origen: ");
```

****

TRAMA: 11

****

CABECERA ETHERNET

MAC Destino: 02-ff-53-c3-e9-ab

MAC Origen: 00-ff-66-7f-d4-3c

.::Cabezera IP::.

:(

Checksum calculado: 4b72

TTL= 128 saltos

3.-Verificar el checksum de las tramas IP, en caso de que este correcto imprimir :) en caso de que sea incorrecto ☹ e imprimir el checksum correcto.

```
    for (j; j < 22 + n; j++)
    {
        printf("%.2x-", trama[j]);
    }
    printf("\b\t\n");
    printf("Direccion IP origen: ");
    for (j; j < 22 + n + m; j++)
    {
        printf("%i.", trama[j]);
     }
     printf("\b\t\n");
     printf("Direccion MAC destino: ");
     for (j; j < 22 + 2 * n + m; j++)
     {
         printf("%.2x-", trama[j]);
     }
     printf("\b\t\n");
     printf("Direccion IP destino: ");
     for (j; j < 22 + 2 * n + 2 * m; j++)
     {
         printf("%i.", trama[j]);
     }
     printf("\b\t");
}

void analizarLLC(unsigned char *trama)
{

    printf("\n\t.::CABECERA LLC::.\n");
    switch (trama[16] & 3)
    {
    case 0: // T-I
        printf("T-I, N(s)= %d, N(r)=%d, ", trama[16] >> 1, trama[17] >> 1);
        if (trama[17] & 1) //PF=1
        {
            if (trama[15] & 1) //LSB_SAPo=1
            {
                printf("F\n");
            }
            else
            {
                printf("P\n");
            }
        }
        break;
```

```
******************************

TRAMA: 2

******************************
      CABECERA ETHERNET
MAC Destino: 00-1f-45-9d-1e-a2
MAC Origen: 00-23-8b-46-e9-ad

      .::Cabezera IP::.

:(
Checksum calculado: 3f45
******************************

TRAMA: 9

******************************
      CABECERA ETHERNET
MAC Destino: 00-1f-45-9d-1e-a2
MAC Origen: 00-23-8b-46-e9-ad

      .::Cabezera IP::.

:(
Checksum calculado: b7d6

      .::UDP::.
Offset= 41096 bytes
******************************

TRAMA: 10

******************************
      CABECERA ETHERNET
MAC Destino: 00-1f-45-9d-1e-a2
MAC Origen: 00-23-8b-46-e9-ad

      .::Cabezera IP::.

:(
Checksum calculado: 3f3d
```

```
******************************

TRAMA: 11

******************************
      CABECERA ETHERNET
MAC Destino: 02-ff-53-c3-e9-ab
MAC Origen: 00-ff-66-7f-d4-3c

      .::Cabezera IP::.

:(
Checksum calculado: 4b72
TTL= 128 saltos
******************************

TRAMA: 14

******************************
      CABECERA ETHERNET
MAC Destino: 00-1f-45-9d-1e-a2
MAC Origen: 00-23-8b-46-e9-ad

      .::Cabezera IP::.

:(
Checksum calculado: cf31

      .::ICMP::.
Opciones: 20 19 63 04 28 00 00 20 19 63 01 43
******************************
```

4.-Una trama UDP cuyo encapsulado IP no tenia opciones—devolver el valor Offset en decimal.

```
1.    case 1: // T-S
2.        printf("T-S, %s, N(r)=%d, ", ss[(trama[16] >> 2) & 3], trama[17] >> 1);
3.        if (trama[17] & 1) //PF=1
4.        {
5.            if (trama[15] & 1) //LSB_SAPo=1
6.            {
7.                printf("F\n");
8.            }
9.            else
10.            {
11.                printf("P\n");
12.            }
13.        }
14.        break;
15.    case 2: // T-I
16.        printf("T-I, N(s)= %d, N(r)=%d, ", trama[16] >> 1, trama[17] >> 1);
17.        if (trama[17] & 1) //PF=1
18.        {
19.            if (trama[15] & 1)
20.            {
21.                printf("F\n");
22.            }
23.            else
24.            {
25.                printf("P\n");
26.            }
27.        }
28.        break;
29.    case 3:                 // T-U
30.        if (trama[16] & 16) //PF=1
31.        {
32.            if (trama[15] & 1) //LSB_SAPo=1
33.            {
34.                printf("T-
U %s F\n", ur[(((trama[16] >> 2) & 3) | ((trama[16] >> 3) & 28))]);
35.            }
36.            else
37.            {
38.                printf("T-
U %s P\n", uc[(((trama[16] >> 2) & 3) | ((trama[16] >> 3) & 28))]);
39.            }
40.        }
41.        break;
42.    default:
43.        printf("%d", trama[16] & 3);
44.        break;
45.    }
46.}
```

****

TRAMA: 9

****

CABECERA ETHERNET

MAC Destino: 00-1f-45-9d-1e-a2

MAC Origen: 00-23-8b-46-e9-ad

.::Cabezera IP::.

:(

Checksum calculado: b7d6

.::UDP::.

Offset= 41096 bytes

```
1.void analizarIP(unsigned char *trama)
2.{
3.    unsigned char IHL = ((trama[14] & 15) * 4);
4.    printf("\n\t.::Cabezera IP::.\n");
5.    checksumIP(trama, IHL);
6.    if (trama[15] & 2)
7.    { //Es de costo minimo?
8.        printf("TTL= %d saltos", trama[22]);
9.    }
10.    if (IHL > 20)
11.    {
12.        if (trama[23] == 1)
13.        {
14.            printf("\n\t.::ICMP::.\n");
15.            unsigned char i = 34;
16.            printf("Opciones: ");
17.            for (i; i < 14 + IHL; i++)
18.            {
19.                printf("%.2X ", trama[i]);
20.            }
21.
22.        } // es ICMP ?
23.    }
24.    else
25.    {
26.        if (trama[23] == 17)
27.        {
28.            printf("\n\t.::UDP::.\n");
29.            printf("Offset= %d bytes", (((trama[20] & 31) << 8) | trama[21]) * 8);
30.        } // es UDP ?
31.    }
32.}
33.void analizartrama(unsigned char *trama)
34.{
35.    unsigned short int ToT = (trama[12] << 8) | trama[13];
36.    printf("\n*****************************\n\tCABECERA ETHERNET\n");
37.    printf("MAC Destino: %.2x-%.2x-%.2x-%.2x-%.2x-%.2x-
%.2x\n", trama[0], trama[1], trama[2], trama[3], trama[4], trama[5]);
38.    printf("MAC Origen: %.2x-%.2x-%.2x-%.2x-%.2x-%.2x-
%.2x\n", trama[6], trama[7], trama[8], trama[9], trama[10], trama[11]);
39.    if (ToT < 1500)
40.    {
41.        printf("Tamanio: %d bytes", ToT);
42.        analizarLLC(trama);
43.    }
44.    else if (ToT == 2048)
45.    {
46.        analizarIP(trama);
47.    }
1. else if (ToT == 2054)
2.    {
3.        analizarARP(trama);
4.    }
5.    else
6.    {
7.        printf("TIPO OTRO\n");
8.    }
9.    printf("\n*****************************\n");
10.}
11.
12.void main()
13.{
14.    unsigned char i = 0;
15.    unsigned char trama[][200] =
16.        {
17.            {//t1
18.             0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x08,
 0x06, 0x00, 0x10,
19.             0x08, 0x00, 0x06, 0x04, 0x00, 0x04, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x94,
 0xcc, 0x39, 0xcb,
20.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x94, 0xcc, 0x39, 0xfe},
21.            {//t2
22.             0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x08,
 0x00, 0x46, 0x00,
23.             0x80, 0x42, 0x04, 0x55, 0x34, 0x11, 0x80, 0x11, 0x6b, 0xf0, 0x94, 0xcc, 0x39,
 0xcb, 0x94, 0xcc,
24.             0x67, 0x02, 0xaa, 0xbb, 0xcc, 0xdd, 0x04, 0x0c, 0x00, 0x35, 0x00, 0x2e, 0x85,
 0x7c, 0xe2, 0x1a,
25.             0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x77, 0x77,
 0x77, 0x03, 0x69,
26.             0x73, 0x63, 0x05, 0x65, 0x73, 0x63, 0x6f, 0x6d, 0x03, 0x69, 0x70, 0x6e, 0x02,
 0x6d, 0x78, 0x00,
27.             0x00, 0x1c, 0x00, 0x01},
28.            {//t3
29.             0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00,
 0x04, 0xf0, 0xf1,
30.             0x09, 0x8d, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00,
31.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00,
32.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x7c, 0x9b, 0x6d},
```

```
1.            {//t4
2.             0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x80,
0x35, 0x00, 0x01,
3.             0x08, 0x00, 0x06, 0x04, 0x00, 0x03, 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x94,
0xcc, 0x3a, 0xe1,
4.             0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, 0x39, 0xcb, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
5.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xd8,
0xee, 0xdf, 0xb0},
6.            {//t5
7.             0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00,
0x03, 0xf0, 0xf0,
8.             0x53, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
9.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
10.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x43,
 0x05, 0x90, 0x6d},
11.            {//T6
12.             0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00,
 0x12, 0xf0, 0xf0,
13.             0x0a, 0x0b, 0x0e, 0x00, 0xff, 0xef, 0x14, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00,
 0x00, 0x7f, 0x23,
14.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00,
15.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
 0x99, 0x98, 0x6d},
16.            {//t7
17.             0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00,
 0x03, 0xf0, 0xf1,
18.             0x53, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00,
19.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00,
20.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x43,
 0x05, 0x90, 0x6d},
21.            {//t8
22.             0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00,
 0x03, 0xf0, 0xf0,
23.             0x43, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00,
24.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00,
25.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x43,
 0x05, 0x90, 0x6d},
```

```
1.            {//t9
2.             0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x08,
0x00, 0x45, 0x08,
3.             0x80, 0x42, 0x04, 0x55, 0x34, 0x11, 0x80, 0x11, 0x6b, 0xf0, 0x94, 0xcc, 0x39,
0xcb, 0x94, 0xcc,
4.             0x67, 0x02, 0xaa, 0xbb, 0xcc, 0xdd, 0x04, 0x0c, 0x00, 0x35, 0x00, 0x2e, 0x85,
0x7c, 0xe2, 0x1a,
5.             0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x77, 0x77,
0x77, 0x03, 0x69,
6.             0x73, 0x63, 0x05, 0x65, 0x73, 0x63, 0x6f, 0x6d, 0x03, 0x69, 0x70, 0x6e, 0x02,
0x6d, 0x78, 0x00,
7.             0x00, 0x1c, 0x00, 0x01},
8.            {//t10
9.             0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x08,
0x00, 0x46, 0x08,
10.             0x80, 0x42, 0x04, 0x55, 0x34, 0x11, 0x80, 0x11, 0x6b, 0xf0, 0x94, 0xcc, 0x39,
 0xcb, 0x94, 0xcc,
11.             0x67, 0x02, 0xaa, 0xbb, 0xcc, 0xdd, 0x04, 0x0c, 0x00, 0x35, 0x00, 0x2e, 0x85,
 0x7c, 0xe2, 0x1a,
12.             0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x77, 0x77,
 0x77, 0x03, 0x69,
13.             0x73, 0x63, 0x05, 0x65, 0x73, 0x63, 0x6f, 0x6d, 0x03, 0x69, 0x70, 0x6e, 0x02,
 0x6d, 0x78, 0x00,
14.             0x00, 0x1c, 0x00, 0x01},
15.            {//t11
16.             0x02, 0xff, 0x53, 0xc3, 0xe9, 0xab, 0x00, 0xff, 0x66, 0x7f, 0xd4, 0x3c, 0x08,
 0x00, 0x45, 0x02,
17.             0x00, 0x30, 0x2c, 0x00, 0x40, 0x00, 0x80, 0x06, 0x4b, 0x74, 0xc0, 0xa8, 0x01,
 0x02, 0xc0, 0xa8,
18.             0x01, 0x01, 0x04, 0x03, 0x00, 0x15, 0x00, 0x3b, 0xcf, 0x44, 0x00, 0x00, 0x00,
 0x00, 0x70, 0x02,
19.             0x20, 0x00, 0x0c, 0x34, 0x00, 0x00, 0x02, 0x04, 0x05, 0xb4, 0x01, 0x01, 0x04,
 0x02},
20.            {//t12 SOLICITUD ARP
21.             0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x08,
 0x06, 0x00, 0x01,
22.             0x08, 0x00, 0x06, 0x04, 0x00, 0x01, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x01,
 0x00, 0x00, 0x01,
23.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00,
24.             0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00},
```

```
{//t13 RESPUESTA ARP
            0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x08,
0x06, 0x00, 0x01,
            0x08, 0x00, 0x06, 0x04, 0x00, 0x02, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x01,
0x00, 0x00, 0x02,
            0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00},
           {//t9
            0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x08,
0x00, 0x48, 0x10,
            0x00, 0x3c, 0x04, 0x57, 0x00, 0x00, 0x80, 0x01, 0x98, 0x25, 0x94, 0xcc, 0x39,
0xcb, 0x94, 0xcc,
            0x3a, 0xe1, 0x20, 0x19, 0x63, 0x04, 0x28, 0x00, 0x00, 0x20, 0x19, 0x63, 0x01,
0x43, 0x64, 0x65,
             0x67, 0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, 0x70, 0x71, 0x72, 0x73,
 0x74, 0x75, 0x76,
             0x77, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69}};
    printf("Por:\n");
    printf("\t- Martinez Coronel Brayan Yosafat.\n");
    printf("\t- Sanchez Mendez Edmundo Josue.\n");
    for (i; i < NO_TRAMAS; i++)
    {
        printf("\nTRAMA: %i\n", i + 1);
        analizartrama(trama[i]);
    }
}
```

Conclusiones:

- Martínez Coronel Brayan Yosafat:

El protocolo IP me parece muy versátil, ya que considera que pueda tener opciones que ni siquiera están definidas como tal, pero no me gusta tanto que tenga que usar cosas que son un poco claras, como el hecho de que tenga que multiplicar por ocho el desplazamiento, eso se siente raro, quizá porque me gustó mucho LLC porque es muy claro lo que trata de indicar con los apartados. Sin embargo, me imagino que el hecho de que se use tanto es porque tiene demasiada versatilidad, cosa que supongo se ver aún más reflejada con TCP.

- Sánchez Méndez Edmundo Josue:

En conclusión del tema del protocolo IP, es que sin duda es un protocolo complicado en cierta manera ya que por ejemplo hay algunas opciones que no tiene definidas y no es muy claro como los anteriores protocolos vistos en clase. Pero sin duda alguna lo anteriormente mencionado nos proporciona versatilidad a comparación de otros protocolos, aunque para la programación no fue complicada en pareja la sentimos al mismo nivel que los protocolos anteriormente programados, solo que había dos casos que no se cumplían con ninguna trama, así que las tuvimos que modificar para poder entrar en los casos y sin duda alguna el caso numero 1 me pareció muy interesante ya que en la parte de opciones y relleno puse la boleta de mi pareja y la mía, así que pensé que ahí se podrían poner cualquier tipo de dato o información que queramos.