



Instituto Politécnico Nacional
Escuela Superior de Cómputo



Redes de Computadoras

“Analizador de tramas”

Versión 3.- IP

Alumno:

Hernández Rodríguez Armando Giovanni

Profesora:

M. en C. NIDIA ASUNCIÓN CORTEZ DUARTE

Grupo: 2CM15

Entrega: 29 noviembre 2021

IP (Internet Protocol)

1. Una trama IP con opciones ICMP imprimir las opciones en hexadecimal

Trama 1

```
00 1F 45 9D 1E A2 00 23 8B 46 E9 AD 08 00 46 04
80 42 04 55 34 11 80 01 6B F0 94 CC 39 CB 94 CC
67 02 EE DE CA DB 04 0C 00 35 00 2E 85 7C E2 1A
01 00 00 01 00 00 00 00 00 00 03 77 77 77 03 69
73 63 05 65 73 63 6F 6D 03 69 70 6E 02 6D 78 00
0X00 1C 00 01
```

MAC DESTINO = 00 1F 45 9D 1E A2

MAC ORIGEN = 00 23 8B 46 E9 AD

TIPO = 08 00 (IP)

VERSIÓN = 4 (IPV4)

IHL = 6 (24 BYTES, 4 BYTES OPCIONES)

TIPO DE SERVICIO = 04 (r -> FIABILIDAD MÁXIMA)

TAMAÑO TOTAL = 80 42 (32834 BYTES)

IDENTIFICADOR = 04 55

BANDERA = 01 (M -> MORE FRAGMENT)

DESPLAZAMIENTO(OFFSET) = 1411 (5137*8 = 41096 BYTES)

TTL = 80 (128 SALTOS)

PROTOCOLO = 01 (ICMP)

CHECKSUM = 6B F0

IP ORIGEN = 94 CC 39 CB

IP DESTINO = 94 CC 67 02

OPCIONES = EE DE CA DB

```

-----
Trama: 1

...::Cabecera Ethernet:...

MAC Destino: 00:1f:45:9d:1e:a2
MAC Origen: 00:23:8b:46:e9:ad
Tipo IP

...::Cabecera IP:...

Opciones: ee de ca db
Checksum incorrecto :(
El checksum correcto es: fd2f
-----

```

2. Una trama IP de costo mínimo imprimir TTL

Trama 2

```

00 1F 45 9D 1E A2 00 23 8B 46 E9 AD 08 00 47 06
80 42 04 55 34 11 80 11 A3 92 94 CC 39 CB 94 CC
67 02 AA BB CC DD EE FF AB AC 04 0C 00 35 00 2E
85 7C E2 1A 01 00 00 01 00 00 00 00 00 03 77
77 77 03 69 73 63 05 65 73 63 6F 6D 03 69 70 6E
02 6D 78 00 00 1C 00 01

```

MAC DESTINO = 00 1F 45 9D 1E A2

MAC ORIGEN = 00 23 8B 46 E9 AD

TIPO = 08 00 (IP)

VERSIÓN = 4 (IPV4)

IHL = 7 (28 BYTES, 8 BYTES OPCIONES)

TIPO DE SERVICIO = 06 (FIABILIDAD MÁXIMA Y COSTO MÍNIMO)

TAMAÑO TOTAL = 80 42 (32834 BYTES)

IDENTIFICADOR = 04 55

BANDERA = 01 (M -> MORE FRAGMENT)

DESPLAZAMIENTO(OFFSET) = 1411 (5137*8 = 41096 bytes)

TTL = 80 (128 SALTOS)

PROTOCOLO = 11 (17->UDP)

CHECKSUM = A3 92

IP ORIGEN = 94 CC 39 CB

IP DESTINO = 94 CC 67 02

OPCIONES = EE DE CA DB

```

-----
      Trama: 2

.....Cabecera Ethernet:.....
MAC Destino: 00:1f:45:9d:1e:a2
MAC Origen: 00:23:8b:46:e9:ad
Tipo IP

.....Cabecera IP:.....

TTL: 128 saltos
Checksum correcto :)
-----

```

3. Verificar el checksum de las tramas IP, en caso de que esté correcto imprimir 😊 en caso de que sea incorrecto 😞 e imprimir el checksum correcto. Llamar a su función checksum.

Para ello se tomarán los ejemplos anteriores Trama 1 y Trama 2

```

-----
      Trama: 1


.....Cabecera Ethernet:.....
MAC Destino: 00:1f:45:9d:1e:a2
MAC Origen: 00:23:8b:46:e9:ad
Tipo IP

.....Cabecera IP:.....

Opciones: ee de ca db
Checksum incorrecto :(
El checksum correcto es: fd2f
-----

```

Checksum incorrecto Trama 1

0x4604		0x4604	
0x8042	0x6ebb	0x8042	0x02cb
0x0455	+5	0x0455	+5
	-----		-----
0x3411	$(0x6ec0)^{-1} = 913f$	0x3411	$(0x02d0)^{-1} = fd2f$
0x8001		0x8001	
0x6bf0	Checksum incorrecto 😞	0x0000	Checksum = fd2f
0x94cc		0x94cc	
0x39cb		0x39cb	
0x94cc		0x94cc	
0x6702		0x6702	
0xeede		0xeede	
0xcadb		0xcadb	
-----		-----	
0x56ebb	0x4604	0x502cb	
	0x8042		
	0x0455		
	0x3411		
	0x8001	0xffffa	
	0xfd2f	+5	

		$(0xffff)^{-1} = 0000$	
	Checksum correcto 😊		
	0x94cc		
	0x39cb		
	0x94cc		
	0x6702		
	0xeede		
	0xcadb		

	0x5fffa		

Comprobación:

```

-----
Trama: 2

...::Cabecera Ethernet:...

MAC Destino: 00:1f:45:9d:1e:a2
MAC Origen: 00:23:8b:46:e9:ad
Tipo IP

...::Cabecera IP:...

TTL: 128 saltos
Checksum correcto :)
-----

```

Checksum correcto Trama 2

0x4706

0x8042

0x0455

0x8fff9

0x3411

+6

0x8011

0xa392

(0xffff)^-1 = 0000

0x94cc

Checksum correcto 😊

0x39cb

0x94cc

0x6702

0xaabb

0xccdd

0xeeff

0xabac

0x6fff9

4. Una trama UDP cuyo encapsulado IP no tenía opciones, devolver el valor del offset en decimal

Trama 3

```

00 1F 45 9D 1E A2 00 23 8B 46 E9 AD 08 00 45 08
80 42 04 55 34 11 80 11 95 A4 94 CC 39 CB 94 CC
67 02 04 0C 00 35 00 2E 85 7C E2 1A 01 00 00 01
00 00 00 00 00 00 03 77 77 77 03 69 73 63 05 65
73 63 6F 6D 03 69 70 6E 02 6D 78 00 00 1C 00 01

```

MAC DESTINO = 00 1F 45 9D 1E A2

MAC ORIGEN = 00 23 8B 46 E9 AD

TIPO = 08 00 (IP)

VERSIÓN = 4 (IPV4)

IHL = 5 (20 BYTES, 0 BYTES OPCIONES)

TIPO DE SERVICIO = 08 (MÁXIMO RENDIMIENTO)

TAMAÑO TOTAL = 80 42 (32834 BYTES)

IDENTIFICADOR = 04 55

BANDERA = 01 (M -> MORE FRAGMENT)

DESPLAZAMIENTO(OFFSET) = 1411 (5137*8 = 41096 bytes)

TTL = 80 (128 SALTOS)

PROTOCOLO = 11 (17->UDP)

CHECKSUM = 95 A4

IP ORIGEN = 94 CC 39 CB

IP DESTINO = 94 CC 67 02

SIN OPCIONES

```
-----  
Trama: 3  
  
.....Cabecera Ethernet:.....  
  
MAC Destino: 00:1f:45:9d:1e:a2  
MAC Origen: 00:23:8b:46:e9:ad  
Tipo IP  
  
.....Cabecera IP:.....  
  
Offset(desplazamiento): 41096 bytes  
Checksum incorrecto :(  
El checksum correcto es: b7d6  
-----
```

Otras pruebas

Trama de costo mínimo, con opciones, ICMP. Por lo tanto se imprime TTL y opciones

```
-----  
Trama: 4  
  
.....Cabecera Ethernet:.....  
  
MAC Destino: 00:1f:45:9d:1e:a2  
MAC Origen: 00:23:8b:46:e9:ad  
Tipo IP  
  
.....Cabecera IP:.....  
  
Opciones: aa bb cc dd ae be cc ae eb ec de cd  
TTL: 136 saltos  
Checksum incorrecto :(  
El checksum correcto es: ef2a  
-----
```

Trama de costo mínimo, UDP, sin opciones. Por lo tanto se imprime TTL y offset

```
-----  
Trama: 5  
  
.....Cabecera Ethernet:.....  
  
MAC Destino: 00:1f:45:9d:1e:a2  
MAC Origen: 00:23:8b:46:e9:ad  
Tipo IP  
  
.....Cabecera IP:.....  
  
TTL: 160 saltos  
Offset(desplazamiento): 41112 bytes  
Checksum incorrecto :(  
El checksum correcto es: 97da  
-----
```


Anexo. Código fuente del analizador completo con versión 3

```

1 // Elaborado por: Hernández Rodríguez Armando Giovanni
2 // Analizador de Tramas Versión 3
3 #include<stdio.h>
4
5 unsigned char i = 0x00; // 1 byte
6 unsigned char j = 0x00; // 1 byte
7 unsigned short int tot = 0x0000; // 2 bytes
8 unsigned int checksum = 0x00000000; // 4 bytes
9 unsigned char IHL = 0x00; // 1 byte
10
11 void analizaLLC(unsigned char T[]){
12     unsigned char SS[][5] = {"RR", "RNR", "REJ", "SREJ"};
13     unsigned char UC[][6] = {"UI", "SIM", "-", "SARM", "UP", "-", "-", "SABM", "DISC",
14 "-", "-",
15 "SARME", "-", "-", "-", "SABME", "SNRM", "-", "-", "RSET", "-", "-", "-", "XID",
16 "-", "-",
17 "-", "SNRME"}; // comandos - p
18     unsigned char UR[][6] = {"UI", "RIM", "-", "DM", "-", "-", "-", "-", "RD", "-", "-",
19 "-", "UA",
20 "-", "-", "-", "-", "FRMR", "-", "-", "-", "-", "-", "XID"}; // respuestas - f
21     printf("\n\n...::Cabecera LLC:..."); //En LLC solo hay SABME T-U 1byte , T-S y T-I
22 2bytes
23     switch(T[16] & 3){ //0000 0011 -> 3   xxxx xxxx & 0000 0011 -> {0, 1, 2, 3}
24         case 0:
25             case 2: //T-I
26                 printf("\nT-I, N(s)=%d, N(r)=%d",T[16]>>1, T[17]>>1);
27                 if(T[17]&1){
28                     if(T[15]&1){ printf(", -f\n"); } //LSB SAPO
29                     else{ printf(", -p\n"); }
30                 }
31                 break;
32
33             case 1: //T-S
34                 printf("\nT-S: %s, N(r)=%d", SS[(T[16]>>2) & 3], T[17]>>1);
35                 if(T[17]&1){ // p/f encendido?
36                     if(T[15]&1){ printf(", -f\n"); } //LSB SAPO
37                     else{ printf(", -p\n"); }
38                 }
39                 break;
40
41             case 3: //T-U   MMMx MM11 T[16]
42                 if(T[16]&16){ // p-f = 1?
43                     if(T[15]&1){ printf("\nT-U: %s -f\n", UR[((T[16] >> 2) & 3) | ((T[16] >> 3)
44 & 28)] ); } //LSB SAPO
45                     else{ printf("\nT-U: %s -p\n", UC[((T[16] >> 2) & 3) | ((T[16] >> 3) & 28)]
46 ); }
47                 }
48                 break;
49
50     }

```

```

51 }
52
53 void analizaARP(unsigned char t[]){
54     printf("\n\n...:Cabecera ARP:...");
55
56     printf("\n\nTipo de direccion de HW: ");
57     if(t[15] == 1){printf("Ethernet");}
58     else if(t[15] == 6){printf("IEEE 802");}
59     else if(t[15] == 15){printf("Frame Relay");}
60     else if(t[15] == 16){printf("ATM");}
61     else{printf("0x%.2x 0x%.2x", t[14], t[15]);}
62
63     //Protocol Address Type
64     printf("\n\nTipo de direccion de Protocolo:");
65     if(t[16] == 8){
66         printf(" IP");
67     }else{
68         printf(" 0x%.2x 0x%.2x", t[16], t[17]);
69     }
70     //HW Add Len
71     printf("\n\nLongitud direccion HW: %d bytes", t[18]);
72
73     //Protocol Add Len
74     printf("\n\nLongitud direccion de Protocolo: %d bytes", t[19]);
75
76     //OPCODE
77     printf("\n\nCodigo de operacion: ");
78     if(t[21] == 1){printf("ARP Request");}
79     else if(t[21] == 2){printf("ARP Reply");}
80     else if(t[21] == 3 || t[21] == 8){printf("Inverse ARP Request");}
81     else if(t[21] == 9 || t[21] == 4){printf("Inverse ARP Reply");}
82     else{printf("%d", (t[20] | t[21]));}
83
84     //MAC Origen
85     printf("\n\nMAC Origen: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x", t[22], t[23], t[24], t[25],
86 t[26], t[27]);
87
88     //IP Origen
89     printf("\n\nIP Origen: %d.%d.%d.%d", t[28], t[29], t[30], t[31]);
90
91     //MAC Destino
92     printf("\n\nMAC Destino: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x", t[32], t[33], t[34],
93 t[35], t[36], t[37]);
94
95     //IP Destino
96     printf("\n\nIP Destino: %d.%d.%d.%d\n", t[38], t[39], t[40], t[41]);
97 }
98
99 unsigned short int getChecksum(unsigned char frame[], unsigned char IHL){
100     j = 0x00;
101     checksum = 0x00000000;
102
103     // Suma hexadecimal con bloques de 2bytes

```

```

104     for(j=0; j<IHL; j++){
105         // Suma la concatenacion de 1 byte y el byte siguiente de una trama
106         checksum += (frame[j]<<8) | frame[++j];
107     }
108     // Suma hexadecimal con el acarreo
109     checksum += checksum>>16;
110     // Complementar a uno el resultado
111     checksum = ~(checksum);
112
113     return checksum;
114 }
115
116 void correctChecksum(unsigned char frame[], unsigned char IHL){
117     checksum = 0x00000000;
118     checksum = getChecksum(frame, IHL);
119
120     /* Si el checksum dado es igual a 0 entonces el checksum es correcto
121        sino el checksum es incorrecto y se procede a calcular el valor del checksum
122        correcto,
123        poniendo en 0 el apartado de checksum (t[10], t[11])
124        */
125     if((checksum == 0)){
126         printf("\nChecksum correcto :)\n");
127     }else{
128         // Obtener el checksum correspondiente a una trama para ello se asigna un
129         valor de 0 al checksum
130         frame[10]= 0x00;
131         frame[11] = 0x00;
132         checksum = getChecksum(frame, IHL);
133         printf("\nChecksum incorrecto :(");
134         printf("\nEl checksum correcto es: %.4x\n", checksum);
135     }
136 }
137
138 void analizaIP(unsigned char t[]){
139     IHL = (t[14]&15)*4; // Tamaño cabecera min ->20 bytes (5 palabras), max ->60
140     bytes(15 palabras)
141     unsigned char frame[IHL-1];
142     printf("\n\n...::Cabecera IP:... \n");
143     //printf("\nIHL: %d bytes", IHL);
144     //1.
145     // Cabecera IP tiene opciones
146     if(IHL>20){
147         // Protocolo sig ICMP
148         if(t[23] == 1){
149             printf("\nOpciones: ");
150             for(j=34; j<14+IHL; j++){
151                 printf("%.2x ", t[j]);
152             }
153         }
154     }
155     // 2. Trama de costo mínimo 'c'
156

```

```

157         if(t[15]&2){printf("\nTTL: %d saltos", t[22]);} //tiempo de vida -> TTL máximo
158         número de enrutadores que un paquete puede atravesar.
159
160         //4.
161         // Cabecera IP no tiene opciones
162         if(IHL == 20){
163             // Protocolo sig UDP
164             if(t[23] == 17){
165                 // Se puede tener hasta (2^13)*8 = 65536 bytes de desplazamiento
166                 printf("\nOffset(desplazamiento): %d bytes",
167 ((t[20]&31)<<8) | t[21])*8);
168             }
169         }
170
171         //3. Verificacion del checksum
172         for(j=0; j<IHL; j++){
173             // t[14] a t[14+IHL-1] considerando las opciones
174             frame[j] = t[14+j];
175             //printf("\nframe[%d]=%.2x\n", j, t[14+j]);
176         }
177         correctChecksum(frame, IHL);
178     }
179
180 void analizaTrama(unsigned char t[]){
181     printf("\n-----
182 -----");
183     printf("\n\n\tTrama: %d\n", i+1);
184     printf("\n...:Cabecera Ethernet:... \n");
185     printf("\nMAC Destino: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x", t[0], t[1], t[2], t[3],
186 t[4], t[5]);
187     printf("\nMAC Origen: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x", t[6], t[7], t[8], t[9],
188 t[10], t[11]);
189
190     // tot -> Tamaño o tipo
191     tot = (t[12]<<8) | t[13]; // 2bytes
192
193     // tot < 05dc (tamaño de cabecer LLC)
194     if(tot<1500){
195         printf("\nTamaño de cabecera LLC: %d bytes\n", 164, tot);
196         analizaLLC(t);
197     }
198     else{// tot = 0x0800 (tipo IP)
199         if(tot == 2048){
200             printf("\nTipo IP\n");
201             analizaIP(t);
202         }//tot = 0x0806 (tipo ARP)
203         else if(tot == 2054){
204             printf("\nTipo ARP\n");
205             analizaARP(t);
206         }
207         else{
208             printf("\nTipo: %.2x:%.2x\n", t[12], t[13]);
209         }

```

```

210     }
211
212 }
213
214 int main() {
215
216     printf("\n\t<<Escuela Superior de C%mputo>>>\nElaborado por: Hern%cndez Rodr%cguez
217 Armando Giovanni\n", 162, 160, 161);
218
219     unsigned char t[][256]=
220 { // 16 columnas x fila
221
222 {0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0x00,0x46,0x04,
223
224 0x80,0x42,0x04,0x55,0x34,0x11,0x80,0x01,0x6b,0xf0,0x94,0xcc,0x39,0xcb,0x94,0xcc,
225
226 0x67,0x02,0xee,0xde,0xca,0xdb,0x04,0x0c,0x00,0x35,0x00,0x2e,0x85,0x7c,0xe2,0x1a,
227
228 0x01,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x77,0x77,0x77,0x03,0x69,
229
230 0x73,0x63,0x05,0x65,0x73,0x63,0x6f,0x6d,0x03,0x69,0x70,0x6e,0x02,0x6d,0x78,0x00,
231
232 0x00,0x1c,0x00,0x01},
233
234
235 {0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0x00,0x47,0x06,
236
237 0x80,0x42,0x04,0x55,0x34,0x11,0x80,0x11,0xa3,0x92,0x94,0xcc,0x39,0xcb,0x94,0xcc,
238
239 0x67,0x02,0xaa,0xbb,0xcc,0xdd,0xee,0xff,0xab,0xac,0x04,0x0c,0x00,0x35,0x00,0x2e,
240
241 0x85,0x7c,0xe2,0x1a,0x01,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x77,
242
243 0x77,0x77,0x03,0x69,0x73,0x63,0x05,0x65,0x73,0x63,0x6f,0x6d,0x03,0x69,0x70,0x6e,
244
245 0x02,0x6d,0x78,0x00,0x00,0x1c,0x00,0x01},
246
247
248 {0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0x00,0x45,0x08,
249
250 0x80,0x42,0x04,0x55,0x34,0x11,0x80,0x11,0x95,0xa4,0x94,0xcc,0x39,0xcb,0x94,0xcc,
251
252 0x67,0x02,0x04,0x0c,0x00,0x35,0x00,0x2e,0x85,0x7c,0xe2,0x1a,0x01,0x00,0x00,0x01,
253
254 0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x77,0x77,0x77,0x03,0x69,0x73,0x63,0x05,0x65,
255
256 0x73,0x63,0x6f,0x6d,0x03,0x69,0x70,0x6e,0x02,0x6d,0x78,0x00,0x00,0x1c,0x00,0x01},
257
258
259 {0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0x00,0x48,0x02,
260
261 0x80,0x42,0x04,0x55,0x34,0x11,0x88,0x01,0x3f,0x53,0x94,0xcc,0x39,0xcb,0x94,0xcc,

```

```

0x67,0x02,0xaa,0xbb,0xcc,0xdd,0xae,0xbe,0xcc,0xae,0xeb,0xec,0xde,0xcd,0x04,0x0c,
0x00,0x35,0x00,0x2e,0x85,0x7c,0xe2,0x1a,0x01,0x00,0x00,0x01,0x00,0x00,0x00,0x00,
0x00,0x00,0x03,0x77,0x77,0x77,0x03,0x69,0x73,0x63,0x05,0x65,0x73,0x63,0x6f,0x6d,
0x03,0x69,0x70,0x6e,0x02,0x6d,0x78,0x00,0x00,0x1c,0x00,0x01},

{0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0x00,0x45,0x02,
0x80,0x42,0x04,0x55,0x34,0x13,0xa0,0x11,0x97,0xbc,0x94,0xcc,0x39,0xcb,0x94,0xcc,
0x67,0x02,0x04,0x0c,0x00,0x35,0x00,0x2e,0x85,0x7c,0xe2,0x1a,0x01,0x00,0x00,0x01,
0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x77,0x77,0x77,0x03,0x69,0x73,0x63,0x05,0x65,
0x73,0x63,0x6f,0x6d,0x03,0x69,0x70,0x6e,0x02,0x6d,0x78,0x00,0x00,0x1c,0x00,0x01}
};

    for(i=0; i<5; i++){
        analizaTrama(t[i]);
    }
    return 0;
}

```

Código solamente de analizador IP y checksum

```

unsigned short int getChecksum(unsigned char frame[], unsigned char IHL){
    j = 0x00;
    checksum = 0x00000000;

    // Suma hexadecimal con bloques de 2bytes
    for(j=0; j<IHL; j++){
        // Suma la concatenacion de 1 byte y el byte siguiente de una trama
        checksum += (frame[j]<<8) | frame[j+1];
    }
    // Suma hexadecimal con el acarreo
    checksum += checksum>>16;
    // Complementar a uno el resultado
    checksum = ~(checksum);

    return checksum;
}

void correctChecksum(unsigned char frame[], unsigned char IHL){
    checksum = 0x00000000;
    checksum = getChecksum(frame, IHL);

    /* Si el checksum dado es igual a 0 entonces el checksum es correcto
    sino el checksum es incorrecto y se procede a calcular el valor del checksum correcto,
    poniendo en 0 el apartado de checksum (t[10], t[11])
    */
    if((checksum == 0)){
        printf("\nChecksum correcto :\n");
    }else{
        // Obtener el checksum correspondiente a una trama para ello se asigna un valor de 0 al
        checksum
        frame[10] = 0x00;
        frame[11] = 0x00;
        checksum = getChecksum(frame, IHL);
        printf("\nChecksum incorrecto :");
        printf("\nEl checksum correcto es: %.4x\n", checksum);
    }
}

void analizaIP(unsigned char t[]){
    IHL = (t[14]&15)*4; // Tamaño cabecera min ->20 bytes (5 palabras), max ->60 bytes(15 palabras)
    unsigned char frame[IHL-1];
    printf("\n\n...:Cabecera IP::...\n");
    //printf("\nIHL: %d bytes", IHL);
    //1.
    // Cabecera IP tiene opciones
    if(IHL>20){
        // Protocolo sig ICMP
        if(t[23] == 1){
            printf("\nOpciones: ");
            for(j=34; j<14+IHL; j++){
                printf("%.2x ", t[j]);
            }
        }
    }
    // 2. Trama de costo mínimo 'c'
    if(t[15]&2){printf("\nTTL: %d saltos", t[22]);} //tiempo de vida -> TTL máximo número de
    enrutadores que un paquete puede atravesar.

    //4.
    // Cabecera IP no tiene opciones
    if(IHL == 20){
        // Protocolo sig UDP
        if(t[23] == 17){
            // Se puede tener hasta (2^13)*8 = 65536 bytes de desplazamiento
            printf("\nOffset(desplazamiento): %d bytes", (((t[20]&31)<<8)|t[21])*8);
        }
    }

    //3. Verificacion del checksum
    for(j=0; j<IHL; j++){
        // t[14] a t[14+IHL-1] considerando las opciones
        frame[j] = t[14+j];
        //printf("\nframe[%d]=%.2x\n", j, t[14+j]);
    }
    correctChecksum(frame, IHL);
}

```