



# INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

## ANALIZADOR DE TRAMAS. VERSIÓN IP

*Redes de Computadoras*

Autores:

Hernández Vergara, Eduardo  
Rojas Cruz, José Ángel

Profesora:

M. en C. Nidia Asunción Cortez Duarte

Mayo 2022

# Índice

<b>1. Solución</b>	<b>2</b>
1.1.Mapa de memoria . . . . .	2
1.2.Correr Programa . . . . .	2
<b>2. Código</b>	<b>4</b>

# 1. Solución

## 1.1. Mapa de memoria

Mapa de memoria Cabecera IP									
Cabecera Ethernet	T[0]								dest
	...								
	t[5]								ICMP
	t[6]								
	...								
Cabecera IP	t[11]	0	0	0					
	t[12]	0	0	0	0	1	0	0	0
	t[13]	0	0	0	0	0	0	0	0
	t[14]								version= (T[14]>>4)
Cabecera IP	t[15]			d	t	r	c		Tipo de servicio
	t[16]								Tamaño total
	t[17]								Identificador
	t[18]								Bandera
	t[19]								Desp. Fragmento
	t[20]								Tiempo de Vida
	t[21]								Protocolo
	t[22]								Checksum IP
	t[23]								Dirección IP Origen
	t[24]								Dirección IP Destino
Opciones Cabecera	t[25]								Opciones y relleno
	t[26]								
	t[27]								
	t[28]								
	t[29]								
	t[30]								
	t[31]								
	t[32]								
	t[33]								
	t[34]								
Opciones Cabecera	t[35]								
	t[36]								
	t[37]								
	...								
	...								
	...								
	...								
	...								
	...								
	t[73]								
Tam datos IP = Tamaño total - IHL									
Cabecera ICMP									

  

IP INICIA	T[14]
IP TERMINA	T[?]
La siguiente cabecera ICMP, UDP o TCP inicia T[14+IHL]	
T[14+IHL+1]	

Figura 2: Mapa de memoria

## 1.2. Correr Programa

```
Cabecera ethernet
MAC DESTINO 00: 1f: 45: 9d: 1e: a2
MAC ORIGEN 00: 23: 8b: 46: e9: ad
TIPO IP
Es de tipo IPv4
Es un ts normal
Tamaño de cabecera: 24 bytes
Tipo de servicio: 66
El tamaño total es de 32834
El id es de 1109
More Fragment
El desplazamiento del fragmento es de 5137
El tiempo de vida es 128
Es UDP
Datos: El checksum es: 27632
Direccion IP origen: 148.204.57.203
Direccion IP destino: 148.204.103.2
Opciones: AA BB CC DD 04 0C
Cabecera ethernet
MAC DESTINO 00: 1f: 45: 9d: 1e: a2
MAC ORIGEN 00: 23: 8b: 46: e9: ad
TIPO IP
Es de tipo IPv4
Es un ts normal
Tamaño de cabecera: 32 bytes
Tipo de servicio: 66
El tamaño total es de 32834
El id es de 1109
More Fragment
El desplazamiento del fragmento es de 5137
El tiempo de vida es 128
Es UDP
Datos: El checksum es: 27632
Direccion IP origen: 148.204.57.203
Direccion IP destino: 148.204.103.2
Opciones: AA BB CC DD EE FF AB CD anshii@anshii-X455LA:~/Documentos/RedesP/ip$
```

Figura 3: Imprimir las Tramas

## 2. Código

Listing 1: Analizador de tramas en C

```
1 #include<stdio.h>
2 #include <stdlib.h>
3 void analizarTrama(unsigned char *T);
4 void leerTrama(unsigned char *);
5 void analizaARP(unsigned char *);
6 void analizaIp(unsigned char *);
7 void decimal(char c, unsigned char *);
8 int main() {
9     printf("Integrantes: \n");
10    printf("Hernandez Vergara Eduardo\nRojas Cruz Jose Angel");
11    FILE * ptr;
12    char c;
13    unsigned char hex = 0, ct = 0, p = 0, n = 0, i = 0;
14    unsigned char nbytet[36];
15    ptr = fopen("tramasIP.txt", "r");
16    while(c != EOF) {
17        c = fgetc(ptr);
18        if(c == '}{') {
19            nbytet[p] = ct;
20            ct = 0;
21            p++;
22        }
23        if(hex == 2) {
24            hex = 0;
25            ct++; //Contar cantidad de bytes
26        }
27        if(hex == 1) {
28            hex++;
29        }
30
31        if(c == 'x')
32            hex = 1;
33    }
34    fclose(ptr);
35    p = 0;
36    c = 0;
37    ptr = fopen("tramasIP.txt", "r");
38    unsigned char *trama = (unsigned char*) (malloc(sizeof(char)*nbytet[0]));
39    while(c != EOF) {
40        c = fgetc(ptr);
41        if(c == '}{') {
42            leerTrama(trama);
43            free(trama);
44            i++;
45            trama = (unsigned char*) (malloc(sizeof(char)*nbytet[i]));
46            p = 0;
47        }
48        if(hex == 2) {
49            decimal(c, &n);
50            trama[p] = n;
51            n = 0;
52            hex = 0;
53            p++;
54        }
55        if(hex == 1) {
56            decimal(c, &n);
```

```

57         n *= 16;
58         hex++;
59     }
60
61     if(c == 'x')
62         hex = 1;
63 }
64 fclose(ptr);
65
66 return 0;
67 }
68
69 void leerTrama(unsigned char * T){
70     printf("\nCabecera ethernet \n");
71     unsigned short tot = T[12] << 8 | T[13];
72     printf("MAC DESTINO %.2x: %.2x: %.2x: %.2x: %.2x: %.2x\n", T[0], T[1], T
73         [2], T[3], T[4], T[5]);
74     printf("MAC ORIGEN %.2x: %.2x: %.2x: %.2x: %.2x: %.2x\n", T[6], T[7], T
75         [8], T[9], T[10], T[11]);
76     if (tot < 1500){
77         printf("Tamano de la cabecera LLC: %d bytes \n", tot);
78         analizarTrama(T);
79     }else{
80         if (tot == 2048){
81             printf("TIPO IP\n"); // analiza IP
82             analizaIp(T);
83         }else if (tot == 2054){
84             printf("TIPO ARP\n"); // analiza ARP
85             analizaARP(T);
86         }else{
87             printf("TIPO: %.2x%.2x", T[12], T[13]);
88         }
89     }
90 }
91
92 void analizarTrama(unsigned char *T){
93     char ss[][5] = {"RR", "RNR", "REJ", "SREJ"};
94     char uc[][5] = {"UI", "SIM", "-", "SARM", "UP", "-", "-", "SABM", "DISC",
95         "-", "-", "SARME", "--", "-", "-", "SABME", "SNRM", "--", "-", "RSET", "-
96         ",
97         "-", "-", "XID", "-", "-", "-", "SNRME"};
98     char ur[][5] = {"UI", "RIM", "-", "DM", "-", "-", "-", "-", "RD",
99         "-", "-", "--", "UA", "-", "-", "--", "--", "FRMR", "-", "--", "-",
100         "-", "-", "XID", "-", "-", "-", "--"};
101     printf("TIPO: %.2x %.2x\n", T[16], T[17]);
102     switch (T[16]&3){
103     case 0:
104         if (T[17]&1){
105             if (T[15]&1){
106                 printf("TIPO: T-I. N(s) = %d, N(r)=%d l-f\n", T[16]>>1, T
107                     [17]>>1);
108             }else{
109                 printf("TIPO: T-I. N(s) = %d, N(r)=%d l-p\n", T[16]>>1, T
110                     [17]>>1);
111             }
112         }else{
113             if (T[15]&1){
114                 printf("TIPO: T-I. N(s) = %d, N(r)=%d 0-f\n", T[16]>>1, T
115                     [17]>>1);
116             }else{

```

```

111         printf("TIPO: T-I. N(s) = %d, N(r)=%d 0-p\n", T[16]>>1, T
           [17]>>1);
112     }
113 }
114 break;
115 case 1:
116     printf("t-S, S = %s\n", ss[(T[16]>>2)&3]);
117     if (T[17]&1){
118         if (T[15]&1){
119             printf("TIPO: T-S. N(s) = -, N(r)=%d 1-f\n", T[17]>>1);
120         }
121         else{
122             printf("TIPO: T-S. N(s) = -, N(r)=%d 1-p\n", T[17]>>1);
123         }
124     }else{
125         if (T[15]&1){
126             printf("TIPO: T-S. N(s) = -, N(r)=%d 0-f\n", T[17]>>1);
127         }
128         else{
129             printf("TIPO: T-S. N(s) = -, N(r)=%d 0-p\n", T[17]>>1);
130         }
131     }
132     break;
133 case 2:
134     if (T[17]&1){
135         if (T[15]&1){
136             printf("TIPO: T-I. N(s) = %d, N(r)=%d 1-f\n", T[16]>>1, T
               [17]>>1);
137         }else{
138             printf("TIPO: T-I. N(s) = %d, N(r)=%d 1-p\n", T[16]>>1, T
               [17]>>1);
139         }
140     }else{
141         if (T[15]&1){
142             printf("TIPO: T-I. N(s) = %d, N(r)=%d 0-f\n", T[16]>>1, T
               [17]>>1);
143         }else{
144             printf("TIPO: T-I. N(s) = %d, N(r)=%d 0-p\n", T[16]>>1, T
               [17]>>1);
145         }
146     }
147     break;
148 case 3:
149     if (T[16]&16){
150         if (T[15]&1){
151             printf("T-U %s 1-f\n", ur[(T[16]>>2&3) | (T[16]>>3&28)]);
152         }else{
153             printf("T-U %s 1-p\n", uc[(T[16]>>2&3) | (T[16]>>3&28)]);
154         }
155     }else{
156         if (T[15]&1){
157             printf("T-U %s 0-f\n", ur[(T[16]>>2&3) | (T[16]>>3&28)]);
158         }else{
159             printf("T-U %s 0-p\n", uc[(T[16]>>2&3) | (T[16]>>3&28)]);
160         }
161     }
162     break;
163 }
164 }
165 void analizaARP(unsigned char *T){

```

```

166     if(T[14]<<8 | (T[15] == 1)){
167         printf("TIPO: ARP\n");
168     }else if(T[14]<<8 | (T[15] == 6)){
169         printf("IEEE 80.2 LAN\n");
170     }else{
171         printf("Otro: %d\n", (T[14]<<8 | T[15]));
172     }
173     // Tipo de direccion de Protocolo
174     if (T[16]<<8 | (T[17] == 0x0806)){
175         printf("TIPO: IPv4\n");
176     }else{
177         printf("TIPO: %.2x, %.2x\n", T[16], T[17]);
178     }
179     //Tamano de la MAC
180     printf("Tamano MAC: %d bytes\n", T[18]);
181     //Tamano de la Direccion IP
182     printf("Tamano IP: %d bytes\n", T[19]);
183     //Op Code
184     if (T[20]<<8 | (T[21] == 1)){
185         printf("Op Code: ARP Request\n");
186     }else if (T[20]<<8 | (T[21] == 2)){
187         printf("Op Code: ARP Reply\n");
188     }else{
189         printf("Otro: %d\n", (T[20]<<8 | T[21]));
190     }
191     printf("Direccion MAC origen: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x\n", T[22], T
        [23], T[24], T[25], T[26], T[27]);
192     printf("Direccion IP origen: %d.%d.%d.%d\n", T[28], T[29], T[30], T[31]);
193     printf("Direccion MAC destino: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x\n", T[32], T
        [33], T[34], T[35], T[36], T[37]);
194     printf("Direccion IP destino: %d.%d.%d.%d\n", T[38], T[39], T[40], T[41])
        ;
195
196 }
197 void analizaIp(unsigned char *T){
198     printf("Es de tipo IPv%d\n", T[14]>>4);
199     if(T[15]&2){
200         printf("Es un ts Costo minimo\n");
201     }else if(T[15]&4){
202         printf("Es un ts fiabilidad \n");
203     }else if(T[15]&8){
204         printf("Es un ts maximo rendimiento\n");
205     }else if(T[15]&16){
206         printf("Es un ts retardo minimo\n");
207     }
208     else{
209         printf("Es un ts normal\n");
210     }
211     //Internet Header Length
212     printf("Tamano de cabecera: %d bytes\n", (T[14]&15) * 4);
213     //Tipo de servicio
214     printf("Tipo de servicio: %d\n", T[17]);
215     printf("El tamano total es de %d\n", T[16]<<8 | T[17]);
216     printf("El id es de %d\n", T[18]<<8 | T[19]);
217     if(T[20]&64){
218         printf("Dont fragment\n");
219     }else if(T[20]&32){
220         printf("More Fragment\n");
221     }
222     printf("El desplazamiento del fragmento es de %d\n", (T[20]&31)<<8 | T[21])

```



```

;
223 printf("El tiempo de vida es %d\n", T[22]);
224 if(T[23] == 1){
225     printf("Es ICMP\n");
226     //ICMP(T);
227 }else if(T[23] == 6){
228     printf("Es TCP\n");
229     //TCP
230 }else if(T[23] == 17){
231     printf("Es UDP\n");
232 }else{
233     printf("Es otro\n");
234 }
235 //datos
236 printf("Datos: ");
237 printf("El checksum es: %d\n", T[24]<<8 | T[25]);
238 printf("Direccion IP origen: %d.%d.%d.%d\n", T[26], T[27], T[28], T[29]);
239 printf("Direccion IP destino: %d.%d.%d.%d\n", T[30], T[31], T[32], T[33]);
240 //opciones IHL
241 printf("Opciones: ");
242 if ((T[14]&15) * 4 > 20 && (T[14]&15) * 4 < 60 ){
243     int l;
244     for (l = 34; l < (T[14]&15) + 34; l++){
245         printf("%.2X ", T[l]);
246     }
247 }else{
248     printf("No hay opciones\n");
249 }
250
251 }
252 void decimal(char c, unsigned char *n){
253     switch(c){
254         case '1':
255             *n +=1;
256             break;
257         case '2':
258             *n +=2;
259             break;
260         case '3':
261             *n +=3;
262             break;
263         case '4':
264             *n +=4;
265             break;
266         case '5':
267             *n +=5;
268             break;
269         case '6':
270             *n +=6;
271             break;
272         case '7':
273             *n +=7;
274             break;
275         case '8':
276             *n +=8;
277             break;
278         case '9':
279             *n +=9;
280             break;
281         case 'a':

```

```
282     *n +=10;
283     break;
284     case 'b':
285         *n +=11;
286         break;
287     case 'c':
288         *n +=12;
289         break;
290     case 'd':
291         *n +=13;
292         break;
293     case 'e':
294         *n +=14;
295         break;
296     case 'f':
297         *n +=15;
298         break;
299     default:
300         *n += 0;
301 }
302 }
```

---