Imagen que contiene Icono

Descripción generada automáticamenteDiagrama, Forma

Descripción generada automáticamente

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**

Texto

Descripción generada automáticamente

1. 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 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**

Texto

Descripción generada automáticamente

1. 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

Texto

Descripción generada automáticamente

**Checksum incorrecto Trama 1**

**0x6ebb**

**+5**

**---------**

**(0x6ec0)^-1 = 913f**

**Checksum incorrecto ☹**

**0x02cb**

**+5**

**---------**

**(0x02d0)^-1 = fd2f**

**Checksum = fd2f**

**0x4604**

**0x8042**

**0x0455**

**0x3411**

**0x8001**

**0x0000**

**0x94cc**

**0x39cb**

**0x94cc**

**0x6702**

**0xeede**

**0xcadb**

**-------**

**0x502cb**

**0x4604**

**0x8042**

**0x0455**

**0x3411**

**0x8001**

**0x6bf0**

**0x94cc**

**0x39cb**

**0x94cc**

**0x6702**

**0xeede**

**0xcadb**

**-------**

**0x56ebb**

**0x4604**

**0x8042**

**0x0455**

**0x3411**

**0x8001**

**0xfd2f**

**0x94cc**

**0x39cb**

**0x94cc**

**0x6702**

**0xeede**

**0xcadb**

**-------**

**0x5fffa**

**Comprobación:**

**0xfffa**

**+5**

**---------**

**(0xffff)^-1 = 0000**

**Checksum correcto 😊**

Texto

Descripción generada automáticamente

**Checksum correcto Trama 2**

**0x4706**

**0x8042**

**0x0455**

**0x3411**

**0x8011**

**0xa392**

**0x94cc**

**0x39cb**

**0x94cc**

**0x6702**

**0xaabb**

**0xccdd**

**0xeeff**

**0xabac**

**-------**

**0x6fff9**

**-------**

**0x5fffa**

**0x8fff9**

**+6**

**---------**

**(0xffff)^-1 = 0000**

**Checksum correcto 😊**

1. 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**

Texto

Descripción generada automáticamente

**Otras pruebas**

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

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

**Anexo. Código fuente del analizador completo con versión 3**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242 | *// Elaborado por: Hernández Rodríguez Armando Giovanni*  *// Analizador de Tramas Versión 3*  #include<stdio.h>  unsigned char i = 0x00; *// 1 byte*  unsigned char j = 0x00; *// 1 byte*  unsigned short int tot = 0x0000; *// 2 bytes*  unsigned int checksum = 0x00000000; *// 4 bytes*  unsigned char IHL = 0x00; *// 1 byte*  void analizaLLC(unsigned char T[]){  unsigned char SS[][5] = {"RR", "RNR", "REJ", "SREJ"};  unsigned char UC[][6] = {"UI", "SIM", "-", "SARM", "UP", "-", "-", "SABM", "DISC", "-", "-",  "SARME", "-", "-", "-", "SABME", "SNRM", "-", "-", "RSET", "-", "-", "-", "XID", "-", "-",  "-", "SNRME"}; *// comandos - p*  unsigned char UR[][6] = {"UI", "RIM", "-", "DM", "-", "-", "-", "-", "RD", "-", "-", "-", "UA",  "-", "-", "-", "-", "FRMR", "-", "-", "-", "-", "-", "XID"}; *// respuestas - f*  printf("\n\n..:::Cabecera LLC:::.."); *//En LLC solo hay SABME T-U 1byte , T-S y T-I 2bytes*  switch(T[16] & 3){ *//0000 0011 -> 3 xxxx xxxx & 0000 0011 -> {0, 1, 2, 3}*  case 0:  case 2: *//T-I*  printf("\nT-I, N(s)=%d, N(r)=%d",T[16]>>1, T[17]>>1);  if(T[17]&1){  if(T[15]&1){ printf(", -f\n"); }*//LSB SAPo*  else{ printf(", -p\n"); }  }  break;  case 1: *//T-S*  printf("\nT-S: %s, N(r)=%d", SS[(T[16]>>2) & 3], T[17]>>1);  if(T[17]&1){ *// p/f encendido?*  if(T[15]&1){ printf(", -f\n"); } *//LSB SAPo*  else{ printf(", -p\n"); }  }  break;  case 3: *//T-U MMMx MM11 T[16]*  if(T[16]&16){*// p-f = 1?*  if(T[15]&1){ printf("\nT-U: %s -f\n", UR[((T[16] >> 2) & 3) | ((T[16] >> 3) & 28)] ); }*//LSB SAPo*  else{ printf("\nT-U: %s -p\n", UC[((T[16] >> 2) & 3) | ((T[16] >> 3) & 28)] ); }  }  break;  }  }  void analizaARP(unsigned char t[]){  printf("\n\n..:::Cabecera ARP:::..");  printf("\n\nTipo de direccion de HW: ");  if(t[15] == 1){printf("Ethernet");}  else if(t[15] == 6){printf("IEEE 802");}  else if(t[15] == 15){printf("Frame Relay");}  else if(t[15] == 16){printf("ATM");}  else{printf("0x%.2x 0x%.2x", t[14], t[15]);}    *//Protocol Address Type*  printf("\nTipo de direccion de Protocolo:");  if(t[16] == 8){  printf(" IP");  }else{  printf(" 0x%.2x 0x%.2x", t[16], t[17]);  }  *//HW Add Len*  printf("\nLongitud direccion HW: %d bytes", t[18]);  *//Protocol Add Len*  printf("\nLongitud direccion de Protocolo: %d bytes", t[19]);  *//OPCODE*  printf("\nCodigo de operacion: ");  if(t[21] == 1){printf("ARP Request");}  else if(t[21] == 2){printf("ARP Reply");}  else if(t[21] == 3 || t[21] == 8){printf("Inverse ARP Request");}  else if(t[21] == 9 || t[21] == 4){printf("Inverse ARP Reply");}  else{printf("%d", (t[20] | t[21]));}    *//MAC Origen*  printf("\nMAC Origen: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x", t[22], t[23], t[24], t[25], t[26], t[27]);  *//IP Origen*  printf("\nIP Origen: %d.%d.%d.%d", t[28], t[29], t[30], t[31]);  *//MAC Destino*  printf("\nMAC Destino: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x", t[32], t[33], t[34], t[35], t[36], t[37]);  *//IP Destino*  printf("\nIP Destino: %d.%d.%d.%d\n", t[38], t[39], t[40], t[41]);  }  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];  }  *// 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 chechksum (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);  }  void analizaTrama(unsigned char t[]){  printf("\n-------------------------------------------------------------------------------------");  printf("\n\n\tTrama: %d\n", i+1);  printf("\n..:::Cabecera Ethernet:::..\n");  printf("\nMAC Destino: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x", t[0], t[1], t[2], t[3], t[4], t[5]);  printf("\nMAC Origen: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x", t[6], t[7], t[8], t[9], t[10], t[11]);  *// tot -> Tamaño o tipo*  tot = (t[12]<<8) | t[13]; *// 2bytes*  *// tot < 05dc (tamaño de cabecer LLC)*  if(tot<1500){  printf("\nTama%co de cabecera LLC: %d bytes\n", 164, tot);  analizaLLC(t);  }  else{*// tot = 0x0800 (tipo IP)*  if(tot == 2048){  printf("\nTipo IP\n");  analizaIP(t);  }*//tot = 0x0806 (tipo ARP)*  else if(tot == 2054){  printf("\nTipo ARP\n");  analizaARP(t);  }  else{  printf("\nTipo: %.2x.%.2x\n", t[12], t[13]);  }  }  }  int main(){  printf("\n\t<<<Escuela Superior de C%cmputo>>>\nElaborado por: Hern%cndez Rodr%cguez Armando Giovanni\n", 162, 160, 161);    unsigned char t[][256]=  { *// 16 columnas x fila*    {0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0x00,0x46,0x04,  0x80,0x42,0x04,0x55,0x34,0x11,0x80,0x01,0x6b,0xf0,0x94,0xcc,0x39,0xcb,0x94,0xcc,  0x67,0x02,0xee,0xde,0xca,0xdb,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,0x08,0x00,0x47,0x06,  0x80,0x42,0x04,0x55,0x34,0x11,0x80,0x11,0xa3,0x92,0x94,0xcc,0x39,0xcb,0x94,0xcc,  0x67,0x02,0xaa,0xbb,0xcc,0xdd,0xee,0xff,0xab,0xac,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,0x08,0x00,0x45,0x08,  0x80,0x42,0x04,0x55,0x34,0x11,0x80,0x11,0x95,0xa4,0x94,0xcc,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},  {0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0x00,0x48,0x02,  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 **solament**e de analizador IP y checksum

Texto

Descripción generada automáticamente