Escuela Superior de Cómputo

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Práctica 2

Implementación del algoritmo Checksum para el encabezado IP, así como TCP y UDP

Unidad de aprendizaje: Redes de computadoras

Grupo: 2CM10

Alumnos(a):
Nicolás Sayago Abigail
Ramos Díaz Enrique

Profesor(a):
Moreno Cervantes Axel

05 de Marzo de 2018

Índice

1. Introducción

En este reporte se mostrará como obtener el checksum de la capa de Red y de Transporte, utilizaremos un método llamado Checksum. Se describirá la forma en la que funciona y los métodos que se utilizaron para implementar la solución.

2. Marco Teorico

2.1. Capa de Red

En esta capa se lleva a cabo el direccionamiento lógico que tiene carácter jerárquico, se selecciona la mejor ruta hacia el destino mediante el uso de tblas de enrutamiento a través del uso de protocolos de enrutamiento o por direccionamiento estático. Protocolos de capa de red: IP, IPX, RIP, IGRP, Apple Talk.

2.1.1. Funciones de la capa de red

La capa de red define cómo transportar el tráfico de datos entre dispositivos que no están conectados localmente en el mismo dominio de difusión, es decir, que pertenecen a diferentes redes. Para conseguir esta comunicación se necesita conocer las direcciones lógicas asociadas a cada puesto de origen y de destino y una ruta bien definida a través de la red apra alcanzar el destino deseado. La capa de red es independiente de la de enlace de datos y, por tanto, puede ser utilizada para conectividad de medios fisicos diferentes.

2.2. Capa de Transporte

Es la encargada de la comunicación confiable entre host, control de flujo y de la corrección de errores entre otras cosas. Los datos son divididos en segmentos identificados con un encabezado con un número de puerto que identifica la aplicación de origen. En esta capa funcionan protocolos como UDP y TCP, siendo este último uno de los más utilizados debido a su estabilidad y confiabilidad.

2.2.1. Funciones de la capa de transporte

Para conectar dos dispositivos remotos es necesario establecer una conexión. La capa de transporte establece las relgas para esta interconexión. Permite que las estaciones finales ensamblen y reensamblen múltiples segmentos del mismo flujo de datos. Esto se hace por medio de identificadores que en TCP/IP reciben el nombre de número de puerto. La capa cuatro permite además que las aplicaciones soliciten transporte fiable entre los sistemas. Asegura que los segmentos distribuidos serán confirmados al remitente. Coloca de nuevo los segmentos en su orden correcto en el receptor. Proporciona control de flujo regulando el tráfico de datos.

En la capa de transporte, los datos pueden ser transmitidos de forma fiable o no fiable. Para IP, el protocolo TCP (Protocolo de control de transporte) es fiable u orientado a conexión con un saludo previo de tres vías, mientras que UDP (Protocolo de datagrama de usuario) no es fiable, o no orientado a conexión, donde solo se establece un saludo de dos vías antes de enviar los datos.

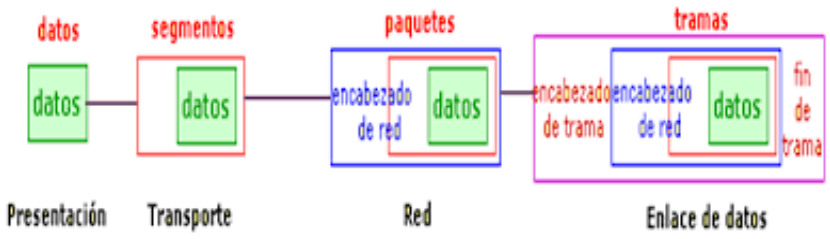
TCP utiliza una técnica llamada ventanas, donde se establece la cantidad de envío de paquetes antes de transmitir; mientras que en el windowing o de ventana deslizante, el flujo de envío de datos es negociado dinámicamente entre el emisor y receptor. En las ventanas deslizantes o windowing cada acuse de recibo (ACK) confirma la recepción y el envío siguiente.

2.3. Proceso de encapsulación de los datos

El proceso desde que los datos son incorporados al ordenador hasta que se transmiten al medio se llama encapsulación. Estos datos son formateados, segmentados, identificados con el direccionamiento lógico y fisico para finalmente ser enviados al medio. A cada capa del modelo OSI le corresponde una **PDU** (Unidad de datos) siguiendo por lo tanto el siguiente orden de encapsulamiento:

- 1. Datos
- 2. Segmentos
- 3. Paquetes
- 4. Tramas
- 5. Bits

Debido a que posiblemente la cantidad de los datos sea deamasiada, la capa de transporte desde el origen se encarga de segmentarlos par así ser empaquetados debidamente, esta misma capa en el destino se encargará de reensamblar los datos y colocarlos en forma secuencial, ya que no siempre llegan a su destino en el orden en que han sido segmentados, así mmismo acorde al protocolo que se esté utilizando habrá o no corrección de errores. Estos segmentos son empaquetados (paquetes o datagramas) e identificados en la capa de red con la dirección lógica o IP correspondiente al origen y destino. Ocurre lo mismo con la dirección MAC en la capa de enlace de datos formándose las tramas o frames para ser transmitidos a través de alguna interfaz. Finalmente las tramas son enviadas al medio desde la capa física.



IMPORTANTE: El proceso inverso se realiza en el destino y se llama desencapsulación de datos.

2.4. Algoritmo de Suma de comprobación

Este algorimo permite verificar la integridad de un PDU y su cálculo es de la siguiente manera:

Envío:

- Ordenar los datos en palabras de 16 bits.
- Poner ceros en la posición de checksum y sumar (considerar acarreos).
- Sumar cualquier acarreo fuera de los 16 bits.
- Complementar a 1 el resultado.

3. Diseño de la solución

3.1. Checksum en la capa de Red

Queremos obtener un arreglo que será enviado al método Checksum, dicho arreglo debe estar compuesto de los datos a partir del Byte 15, por lo tanto primero calculamos el número de bytes que corresponden a los datos de la trama. Observamos la siguiente figura:

ETHERNET

MAC 0 (6 Bytes)	MAC 0 (6 Bytes)	TIPO (2 Bytes)	DATOS (46-1500 Bytes)	CRC (5 Bytes)

Sabemos que en el byte 15 se encuentra el número por el cual debe ser multiplicado 4, para así obtener la longitud de los datos que serán enviados al método Checksum. Para obtenerlo lo que hacemos es obtener el valor más hacia la derecha del byte 15, y hacemos un **AND** para obtener el número en cuestión. Mostramos un ejemplo para describir esto.

```
Trama:
20 16 D8 EB 60 CE 0C 96 BF 99 BC 93 08 00 45 70
00 28 8D DB 40 00 5A 06 6B 61 1F 0D 46 24 C0 A8
01 4A 01 BB 09 50 B4 88 AD D6 FA C8 9D 05 50 11
00 CE 82 A9 00 00
```

45 esta en el byte 15, pero solo queremos el número 5, entonces:

&	0100	0101
	0000	1111
	0000	0101

De tal forma que únicamente nos quedamos con el número que necesitamos, multiplicamos este número por 4, de tal forma que ahora tenemos el total de Bytes que serán sumados (llamamos n a tal número).

Creamos un arreglo de Bytes del tamaño del resultado anterior. Lo llenamos con los datos que empiezan desde el byte 15, hasta el byte $(15 + n)$.

Ahora simplemente enviamos los bytes de datos al método para realizar el checksum de la trama.

3.2. Checksum en la capa de Transporte

Similar al checksum de la capa de Red, debemos de obtener un arreglo de bytes que será enviado al método calculateChecksum() de la clase Checksum.

Aquí nos encontraremos con dos tipos de tramas que manejan protocolos distintos: protocolo TCP y protocolo UDP. Sin embargo, para ambos casos el arreglo de bytes será el mismo, exceptuando el byte que define el protocolo de la trama.

En las siguientes figuras podemos observar las cabeceras de las tramas de ambos protocolos:

UDP

+	Bits 0 - 15	16 - 31
0	Puerto origen	Puerto destino
32	Longitud del Mensaje	Suma de verificación
64	Datos	

TCP

Offsets Octeto		0								1								2								3							
Octeto	Bit	0	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
0	0	Puerto de origen																Puerto de destino														
4	32	Número de secuencia																														
8	64	Número de acuse de recibo (si ACK es establecido)																														
12	96	Longitud de Cabecera				Reservado			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Tamaño de Ventana														
16	128	Suma de verificación																Puntero urgente (si URG es establecido)														
20 ...	160 ...	Opciones (Si la Longitud de Cabecera > 5, relleno al final con "0" bytes si es necesario) ...																														

A este arreglo de bytes le llamaremos pseudo encabezado y debe estar conformador por los siguientes campos:

Pseudoencabezado para el Checksum de la capa de transporte:

IP Origen (4 bytes)	IP Destino (4 bytes)	1 byte (cero)	Protocolo	Longitud PDU de Transporte (2 bytes)	PDU Transporte

3.2.1. IP Origen

Inicia en el byte 27 del PDU y termina en el 30.

```
Trama:
70 8B CD BD 24 78 E0 88 5D 41 10 D6 08 00 45 00
00 4A 8A CD 40 00 2F 06 52 63 42 E4 21 96 0A 00
00 04 01 BB D6 81 BD 0C F3 12 AB 63 58 38 50 18
05 70 EE 65 00 00 17 03 03 00 1D E4 F5 51 DB 2A
10 D0 74 21 E7 E5 A6 DD C4 71 65 7A 61 D1 BB 48
0B A3 93 DB BD 5F 01 61
```

3.2.2. IP Destino

Enseguida de la IP Origen. Inicia en el byte 31 del PDU y termina en el 34.

```
Trama:
70 8B CD BD 24 78 E0 88 5D 41 10 D6 08 00 45 00
00 4A 8A CD 40 00 2F 06 52 63 42 E4 21 96 0A 00
00 04 01 BB D6 81 BD 0C F3 12 AB 63 58 38 50 18
05 70 EE 65 00 00 17 03 03 00 1D E4 F5 51 DB 2A
10 D0 74 21 E7 E5 A6 DD C4 71 65 7A 61 D1 BB 48
0B A3 93 DB BD 5F 01 61
```

3.2.3. 1 byte

Solo se agrega el byte 0x00 al arreglo o pseudo encabezado.

3.2.4. Protocolo

Lo ubicamos en el byte 24 de la trama. El protocolo UDP se representa por el byte 0x11 y el protocolo TCP se representa por el byte 0x06.

```
Trama:
70 8B CD BD 24 78 E0 88 5D 41 10 D6 08 00 45 00
00 4A 8A CD 40 00 2F 06 52 63 42 E4 21 96 0A 00
00 04 01 BB D6 81 BD 0C F3 12 AB 63 58 38 50 18
05 70 EE 65 00 00 17 03 03 00 1D E4 F5 51 DB 2A
10 D0 74 21 E7 E5 A6 DD C4 71 65 7A 61 D1 BB 48
0B A3 93 DB BD 5F 01 61
```

3.2.5. Longitud PDU de Transporte

Para hallar esta longitud, primero debemos ordenar los bytes 17 y 18 de la trama en palabras de 16 bits, hallar su valor en decimal, y luego debemos restarle el **valor decimal de la longitud del Checksum de la capa de red**, previamente calculado (valor más a la derecha del byte 15 de la trama multiplicado por 4). Este resultado debe ser agregado como byte (base octal).

```
Trama:
70 8B CD BD 24 78 E0 88 5D 41 10 D6 08 00 45 00
00 4A 8A CD 40 00 2F 06 52 63 42 E4 21 96 0A 00
00 04 01 BB D6 81 BD 0C F3 12 AB 63 58 38 50 18
05 70 EE 65 00 00 17 03 03 00 1D E4 F5 51 DB 2A
10 D0 74 21 E7 E5 A6 DD C4 71 65 7A 61 D1 BB 48
0B A3 93 DB BD 5F 01 61
```

3.2.6. PDU Transporte

Son todos los bytes restantes que inician en la posición 35 de la trama hasta el fin de ésta.

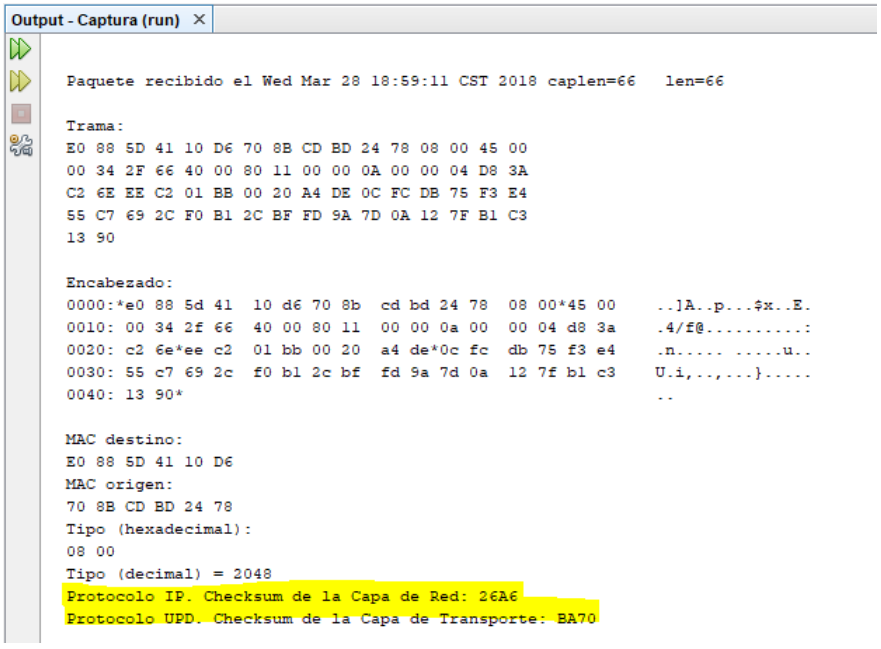
4. Implementación de la solución

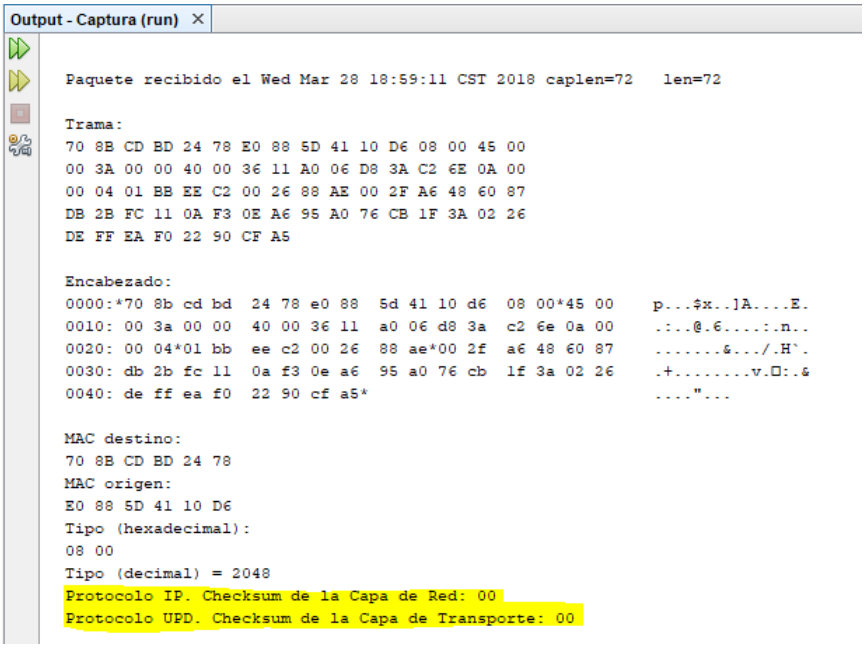
```
//------------------CHECKSUM CAPA DE RED-----------------------------

if(tipo==2048)
{ // Verifica que el protocolo de la capa de red sea IP: 2048 - decimal

//Calculamos el numero de bytes que corresponden a los datos de la trama
//Se apartan las posiciones 0-3 del arreglo de transporte para los bytes extras
int n=packet.getUByte(14)&0x0F, bytesr=0;

//Creamos un arreglo para los bytes de datos
byte[] encabezado = new byte[n*4];

//Ingresamos los bytes de datos al arreglo
for (int i=14; i<14+(n*4); i++)
{
    encabezado[bytesr]=(byte)packet.getUByte(i);
    bytesr++;
}

//Enviamos los bytes de datos para realizar el checksum de la trama
long chkr = Checksum.calculateChecksum(encabezado);
System.out.printf("Protocolo IP. Checksum de la Capa de Red: %02X\n",chkr);

//--------------------CHECKSUM TRANSPORTE------------------

// Longitud que tendra el pseudoencabezado:
// IPOrigen + IPDestino + 00 + Protocolo + LongitudPDUTransporte + PDUTransporte
int ltrans=(packet.size()-26)+4;

// Longitud de la trama menos longitud del arreglo del checksum de red IP
// (Esta es la LongitudPDUTransporte mencionada arriba)
int longitud=((packet.getUByte(16)==0)?packet.getUByte(17):
    (packet.getUByte(16)*256)+packet.getUByte(17))-(n*4);

// Se apartan las posiciones 0-3 del arreglo de transporte para:
// 00 + Protocolo + LongitudPDUTransporte
int bytest=0;
byte[] pseudoenc = new byte[ltrans]; //Pseudoencabezado

//Ingresamos los campos: IPOrigen + IPDestino + PDUTransporte
for (int i=26; i<34; i++)
{
    pseudoenc[bytest]=(byte)packet.getUByte(i);
    bytest++;
}

bytest=0;

//Posicion 8: 0x00
pseudoenc[8]=0x00;

//Posicion 10 y 11: Longitud PDUTransporte
```

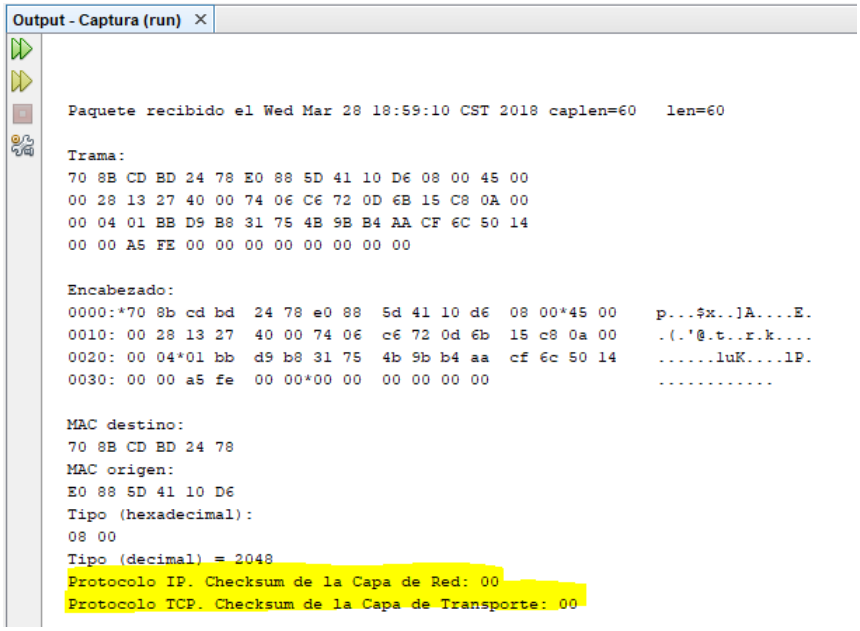
```
53  //Se divide la longitud del PDU (hexadecimal) en 2 bytes
54  int byte_long1=longitud&0xFF00, byte_long2=longitud&0x00FF;
55
56  pseudoenc[10]=(byte)byte_long1;
57  pseudoenc[11]=(byte)byte_long2;
58
59
60  if((byte)packet.getUByte(23)==0x11) //Revisa si es UDP: Byte 23: 11
61  {
62      //Posicion 9: Protocolo
63      pseudoenc[9]=0x11; // UDP: 11
64      System.out.print("Protocolo UPD. ");
65      bytest=12;
66
67      //Ingresamos los campos: PDUTransporte
68      for (int i=34; i<packet.size(); i++)
69      {
70          pseudoenc[bytest]=(byte)packet.getUByte(i);
71          bytest++;
72      }
73
74      //Enviamos los bytes de datos para realizar el checksum de la capa de transporte
75      long chkt = Checksum.calculateChecksum(pseudoenc);
76      System.out.printf("Checksum de la Capa de Transporte: %02X\n",chkt);
77  }
78  else if ((byte)packet.getUByte(23)==0x06) //Revisa si es TCP: Byte 23: 06
79  {
80      //Posicion 9: Protocolo
81      pseudoenc[9]=0x06; // TCP: 06
82      System.out.print("Protocolo TCP. ");
83      bytest=12;
84
85      //Ingresamos los campos: PDUTransporte
86      for (int i=34; i<packet.size(); i++)
87      {
88          pseudoenc[bytest]=(byte)packet.getUByte(i);
89          bytest++;
90      }
91
92      //Enviamos los bytes de datos para realizar el checksum de la capa de transporte
93      long chkt = Checksum.calculateChecksum(pseudoenc);
94      System.out.printf("Checksum de la Capa de Transporte: %02X\n",chkt);
95  }
96  }
```

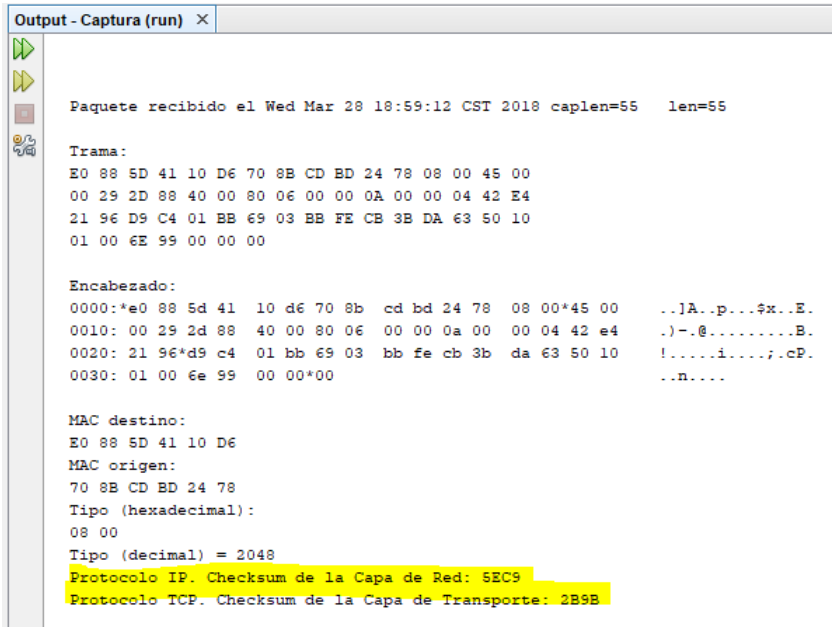
5. Funcionamiento

El programa muestra el valor del Checksum tanto de la capa de Red como de la capa de Transporte. Aquellas tramas que no son IP, se omiten.









6. Conclusiones

- Nicolás Sayago Abigail Al finalizar está práctica pude ver como funciona el algoritmo de Checksum y su implementación a un lenguaje de programación Su funcionamiento ayuda a ver los errores que pueden tener las tramas. También pude ver como trabajan las capas de red y de transporte.

- Ramos Diaz Enrique
  La idea en la que se basa la suma de chequeo de Internet es muy sencilla: se suman todas las

palabras de 16 bits que conforman el mensaje y se transmite, junto con el mensaje, el resultado de dicha suma (este resultado recibe el nombre de checksum). Al llegar el mensaje a su destino, el receptor realiza el mismo cálculo sobre los datos recibidos y compara el resultado con el checksum recibido. Si cualquiera de los datos transmitidos, incluyendo el mismo checksum, esta corrupto, el resultado no concordará y el receptor sabrá que ha ocurrido un error.

El checksum se realiza de la siguiente manera: los datos que serán procesados (el mensaje) son acomodados como una secuencias de enteros de 16 bits. Estos enteros se suman utilizando aritmética complemento a uno para 16 bits y, para generar el checksum, se toma el complemento a uno para 16 bits del resultado.

Referencias

[1] E. Ariganello, *Redes Cisco. Guía de estudio para la certificación CCNA 640-802, 2da Edición*, 2011.