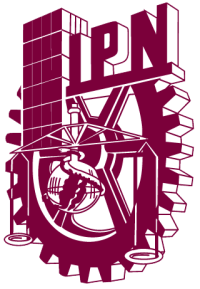
****

**Instituto Politécnico Nacional**

**Escuela Superior de Computo (ESCOM)**

**Nombres:** **Galindo Reyes Agustin**

**Pérez Leal Rodolfo**

**Grupo: 2CM8**

**Asignatura: Redes de Computadoras**

**Profesor: Axel Ernesto Moreno Cervantes**

**Practica 1: Checksum**

**Periodo: Agosto-Diciembre 2017**

**Introducción**

**El algoritmo de *checksum***

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.

*En aritmética complemento a uno, un entero negativo -x se representa como el complemento de x; es decir, cada bit de x es invertido. Cuando los números se adicionan, si se obtiene un acarreo (carry) en el bit más significativo, se debe incrementar el resultado. Por ejemplo, sumemos -5 y -3 en aritmética complemento a uno con enteros de 4 bits. En este caso +5 se representaría con 0101 y -5 con 1010;  +3 se representaría con 0011 y -3 con 1100. Al sumar 1010 y 1100, ignorando el acarreo (carry) que queda en el bit más significativo, tendremos como resultado 0110. En la aritmética complemento a uno, cuando una operación genera un acarreo (carry) en el bit más significativo, se debe incrementar el resultado; es decir que 0110 se convierte en 0111, que es la representación complemento a uno de -8 (obtenido de invertir los bits 1000).*

El uso del algoritmo de *checksum* de Internet en los headers de los protocolos se puede resumir en tres pasos simples.

1. Los octetos adyacentes que se deben verificar con al suma de chequeo deben ser acomodados para formar enteros de 16 bits, luego se calcula la suma complemento a uno de estos enteros (de 16 bits)
2. Para generar el *checksum*, el campo de *checksum* del header del PDU que será transmitido es puesto en cero, luego la suma complemento a uno es calculada sobre los octetos correspondientes y el complemento a uno de esta suma se coloca en el campo de *checksum*.
3. Para revisar el checksum, la suma es calculada sobre los mismo octetos, incluyendo el campo de checsum. Si el resultado es 16 bits con valor 1 (-0 en aritmética complemento a uno), el chequeo es correcto.

Como un ejemplo sencillo del cálculo del *checksum* supongamos que tenemos tres "palabras" de 16 bits

0110011001100110   
0101010101010101   
0000111100001111

La suma de las dos primeras palabras sería:

0110011001100110   
0101010101010101   
1011101110111011

Adicionando ahora la tercera "palabra" al resultado anterior tenemos

1011101110111011   
0000111100001111   
1100101011001010

La suma complemento a uno se obtiene convirtiendo todos los ceros en unos y todos los unos en ceros. De esta forma la suma complemento a uno de 1100101011001010 sería 0011010100110101. Que vendría a ser el *checksum*. Al llegar al receptor las cuatro palabra de 16 bits, incluyendo el *checksum* son sumados y el resultado debe ser 1111111111111111. Si uno de los bits es cero, un error ha sido detectado.

Dependiendo del protocolo, se deben seleccionar ciertos campos de los headers para realizar los cálculos del *checksum*. En IP el checksum se calcula sólo sobre los octetos que componen el header del datagrama ([RFC791](http://www.rfc-editor.org/rfc/rfc791.txt)), en UDP ([RFC768](http://www.rfc-editor.org/rfc/rfc768.txt)) se calcula sobre un seudo-header , el header UDP y los datos que transporta UDP y en TCP ([RFC793](http://www.rfc-editor.org/rfc/rfc793.txt)) se hace un cálculo similar que en UDP. Si desea conocer los detalles del cáculo del checksum puede ver los siguientes ejemplos.

* [Cálculo del campo de checksum en el header de IP](http://www.arcesio.net/checksum/checksumIP.html)
* [Cálculo del campo de checksum en el header de UDP](http://www.arcesio.net/checksum/checksumUDP.html)
* [Cálculo del campo de checksum en el header de TCP](http://www.arcesio.net/checksum/checksumTCP.html)

El [RFC1071](http://www.rfc-editor.org/rfc/rfc1071.txt), *Computing the Internet Checksum*, discute métodos para calcular de manera eficiente el *checksum*de Internet que se utiliza en los protocolos IP, UDP y TCP.

**Cálculo del *checksum* de UDP y TCP**

El campo de *checksum* de TCP y UDP cubre tanto la cabecera como los datos (El *checksum* de la cabecera IP sólo cubre la propia cabecera IP). Para realizar el cálculo del *checksum*, además de utilizar todo el paquete (TCP ó UDP) (cabecera + datos), se añade una pseudo-cabecera que incluye la dirección IP fuente, la dirección IP destino, el campo de protocolo de la cabecera de IP y la longitud del paquete (TCP ó UDP). Tal y como se indica en el RFC793, esta pseudo-cabecera proporciona protección adicional contra paquetes recibidos erróneamente por problemas de enrutado.

El *checksum* es el complemento a uno de 16 bits de la suma en complemento a uno de todas las palabras de 16 bits de la pseudo-cabecera, cabera y texto del paquete (TCP o UDP).

Si el paquete contiene un número impar de octetos, el último octeto del paquete se rellena con ceros por la derecha hasta obtener una palabra de 16 bits con propósito de calcular el *checksum*.

**Desarrollo**

Para esta práctica fue necesario instalar la librería Jnetpcap en Netbeans, para ello primero nos vamos a la sección de librerías, y le damos en agregar librería, en la configuración nos dirigimos a la ventana que dice Library Classpath, donde agregaremos el archivo jar de la librería, luego nos vamos a la sección de Sources donde agregaremos el zip que descargamos que dice jnetpcap-src, para finaliar nos vamos a la ventana que dice javadoc y agregamos el zip que dice jnetpcap-javadoc. Luego nos vamos a la sección de RUN donde nos vamos a customize, nos vamos a new configuration, e insertamos lo siguiente, **- Djava.library.path="g:\libs\jnetpcap-1.4.b0004-1",** dentro de la comillas pones la dirección donde se encuentre el jar de la librería.

Despues se nos dieron dos programas uno que se llama Checksum y otro llamado Captura, la idea es modificar el programa de Captura el cual nos da varias tramas detectadas, y convertir el programa de Checksum en un método que llamemos dentro de la Captura para que calcule el Checksum de las tramas encontradas por el programa.

En el programa de Captura vamos a cambiar algunas cosas para que de las tramas obtenidas calculemos un pseudo encabezado, y a partir de el poder calcular el checksum de la trama, y verificar que es correcto con el que tiene la trama.

En la siguiente parte del código vamos evaluando byte por byte de la cadena para ir formando nuestro pseudo encabezado, mandándolo a una cadena auxiliar, también creamos una cadena para el encabezado, cuando termine de evaluar tanto el encabezado como el pseudo encabezado, los concatenara y la cadena final será la que se mandara al checksum para evaluarse y comprobar con el que tenia la trama inicial.

public void nextPacket**(**PcapPacket packet**,** String user**)** **{**

//Variables

byte auxenc**[],** auxps**[];** //auxiliar para mandar al metodo calculateChecksum()

ArrayList**<**Byte**>** encabezado **=** new ArrayList**<>();** //encabezado IP

ArrayList **<**Byte**>** pseudo **=** new ArrayList**<>();** //pseudoencabezado (UDP/TCP)

String tipoProtocolo **=** null**;**

byte aux**;**

int protocolo **=** packet**.**getUByte**(**23**);** //leer el protocolo

int tipo **=** packet**.**getUByte**(**12**)\***256**+**packet**.**getUByte**(**13**);** //tipo de trama

long checksumEncabezado**,** checksumPseudo**;**

int tamaño **=** packet**.**getUByte**(**14**)** **&** 0x0F**;** //Obtener el tamaño

tamaño **\*=** 4**;** //tamaño en Bytes del encabezado

System**.**out**.**println**(**"---------------------------------------------------------------------------\n"**);**

System**.**out**.**printf**(**"Received packet at %s caplen=%-4d len=%-4d %s\n"**,**

new Date**(**packet**.**getCaptureHeader**().**timestampInMillis**()),**

packet**.**getCaptureHeader**().**caplen**(),** // Length actually captured

packet**.**getCaptureHeader**().**wirelen**(),** // Original length

user // User supplied object

**);**

/\*\*\*\*\*\*Desencapsulado\*\*\*\*\*\*\*\*/

**if(**tipo **==** 2048**){** //si el encabeado es tipo IP

**for(**int i**=**0**;**i**<**packet**.**size**();**i**++){** //recorrer el paquete

aux **=** **(**byte**)** packet**.**getUByte**(**i**);**

System**.**out**.**printf**(**"%02X "**,**aux**);**

**if(**i**%**16**==**15**)**

System**.**out**.**println**(**""**);**

**if(**i**>**13 **&&** i**<**14**+**tamaño**){** //Desde donde empieza el encabezado IP(byte 14) hasta el tamaño del encabezado

encabezado**.**add**(**aux**);**

**if(**i **>** 25**)** //Si es mayor a 25 (IP origen e IP destino)

pseudo**.**add**(**aux**);**

**}**

**}** //fin for

int longitudpseudo **=** packet**.**getUByte**(**16**)\***256 **+** packet**.**getUByte**(**17**);** //Campo longitud del pseudoencabezado

longitudpseudo **-=** tamaño**;** //valor decimal de la longitud del pseudoencabezado

//-----Agregar los bytes al Psuedo-encabezado

pseudo**.**add**((**byte**)**0x00**);** //Byte zero

pseudo**.**add**((**byte**)**protocolo**);** //Byte Protocolo

**if(**protocolo **==** 17**){** //Si es protocolo UDP

tipoProtocolo **=** "UDP"**;**

pseudo**.**add**((**byte**)**packet**.**getUByte**(**18**+**tamaño**));**

pseudo**.**add**((**byte**)**packet**.**getUByte**(**19**+**tamaño**));**

**}**//fin protocolo UDP

**else** **if(**protocolo **==** 06**){** //Protocolo TCP

tipoProtocolo **=** "TCP"**;**

//-------Campo longitud--------------------

pseudo**.**add**((**byte**)**packet**.**getUByte**(**16**));** //primer byte del campo longitud

pseudo**.**add**((**byte**)(**packet**.**getUByte**(**17**)-**tamaño**));** //segundo byte del campo longitud - longitud(IHL)

**}**//fin protocolo tcp

//--------------Recorrer el encabezado TCP/UDP

**for** **(**int i **=** 14**+**tamaño**;** i**<** longitudpseudo**+**14**+**tamaño**;** i**++)**

pseudo**.**add**((**byte**)**packet**.**getUByte**(**i**));**

//------------Llamar a la funcion para calcular el checksum---------

auxenc **=** new byte**[**encabezado**.**size**()];**

auxps **=** new byte**[**pseudo**.**size**()];**

//-----------pasar los bytes al arreglo

**for(**int i**=**0**;** i**<**encabezado**.**size**();** i**++)** //para el arreglo del encabezado

auxenc**[**i**]** **=** encabezado**.**get**(**i**);**

**for(**int i**=**0**;** i**<**pseudo**.**size**();** i**++)** //para el arreglo del pseudoencabezado

auxps**[**i**]** **=** pseudo**.**get**(**i**);**

checksumEncabezado **=** Checksum**.**calculateChecksum**(**auxenc**);**

checksumPseudo **=** Checksum**.**calculateChecksum**(**auxps**);**

System**.**out**.**printf**(**"\n\nTrama IP\nProtocolo = %02x - %s \nIHL: %d\nLongitud del Pseudoencabezado: %d\n"**,**protocolo**,** tipoProtocolo**,** tamaño**,** longitudpseudo**);**

//--------------------Verificar si es velido el checksum

System**.**out**.**printf**(**"Checksum del encabezado IP: %02X\n"**,** checksumEncabezado**);**

**if(**checksumEncabezado **==** 0**)** //Encabezado

System**.**out**.**printf**(**"Checksum del encabezado Valido\n"**);**

**else**

System**.**out**.**printf**(**"Checksum del encabezado NO Valido\n"**);**

System**.**out**.**printf**(**"Checksum del protocolo (pseudoencabezado): %02X\n"**,** checksumPseudo**);**

**if(**checksumPseudo **==** 0**)**

System**.**out**.**printf**(**"Checksum del protocolo Valido\n"**);**

**else**

System**.**out**.**printf**(**"Checksum del protocolo NO Valido\n"**);**

**}** //fin if tipo

// System.out.println("\n\nEncabezado: "+ packet.toHexdump());

**}**

**};**

**Pruebas**

run:

Network devices found:

#0: \Device\NPF\_{943451B1-27C9-427D-92CD-9989FCAF982C} [Realtek PCIe GBE Family Controller] MAC:[C8:5B:76:99:9A:73]

#1: \Device\NPF\_{CABED956-05D0-4A37-BA7D-2C1F3F57529E} [Microsoft] MAC:[3E:24:31:87:46:94]

Choosing 'Microsoft' on your behalf:

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:54 CDT 2017 caplen=103 len=103 jNetPcap rocks!

A4 71 74 F1 7C 49 3E 24 31 87 46 94 08 00 45 00

00 59 07 AE 00 00 80 11 98 06 C0 A8 01 45 5E F5

79 FD D6 02 0D D8 00 45 CA 85 00 01 00 00 DA F7

96 B3 97 91 E7 4A 00 60 00 00 00 00 08 3A FF FE

80 00 00 00 00 00 00 00 00 FF FF FF FF FF FE FF

02 00 00 00 00 00 00 00 00 00 00 00 00 00 02 85

00 7D 38 00 00 00 00

Trama IP

Protocolo = 11 - UDP

IHL: 20

Longitud del Pseudoencabezado: 69

Checksum del encabezado IP: 00

Checksum del encabezado Valido

Checksum del protocolo (pseudoencabezado): 00

Checksum del protocolo Valido

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:54 CDT 2017 caplen=151 len=151 jNetPcap rocks!

3E 24 31 87 46 94 A4 71 74 F1 7C 49 08 00 45 00

00 89 11 1F 00 00 76 11 98 65 5E F5 79 FD C0 A8

01 45 0D D8 D6 02 00 75 93 D6 00 01 00 00 DA F7

96 B3 97 91 E7 4A 00 00 00 29 FD 44 21 E1 17 60

00 00 00 00 30 3A FF FE 80 00 00 00 00 00 00 80

00 F2 27 A1 0A 86 02 FE 80 00 00 00 00 00 00 00

00 FF FF FF FF FF FE 86 00 64 9D 00 00 00 00 00

00 3A 98 00 00 07 D0 03 04 40 40 FF FF FF FF FF

FF FF FF 00 00 00 00 20 01 00 00 5E F5 79 FD FF

00 00 00 00 20 01 00

Trama IP

Protocolo = 11 - UDP

IHL: 20

Longitud del Pseudoencabezado: 117

Checksum del encabezado IP: 00

Checksum del encabezado Valido

Checksum del protocolo (pseudoencabezado): 00

Checksum del protocolo Valido

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:54 CDT 2017 caplen=126 len=126 jNetPcap rocks!

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:57 CDT 2017 caplen=42 len=42 jNetPcap rocks!

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:57 CDT 2017 caplen=42 len=42 jNetPcap rocks!

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:58 CDT 2017 caplen=100 len=100 jNetPcap rocks!

A4 71 74 F1 7C 49 3E 24 31 87 46 94 08 00 45 00

00 56 34 85 40 00 80 06 6A C6 C0 A8 01 45 D8 3A

C1 2E D4 EA 01 BB C7 72 D7 40 E9 AE E1 0C 50 18

00 42 0E 86 00 00 17 03 03 00 29 3A A6 E5 AB B3

21 6E A8 16 44 EE 4A 39 9B 55 75 A0 88 AD 97 AE

FC E2 80 C2 D7 5E A1 24 C8 4B 52 E6 3C 64 95 43

03 A5 FE ED

Trama IP

Protocolo = 06 - TCP

IHL: 20

Longitud del Pseudoencabezado: 66

Checksum del encabezado IP: 00

Checksum del encabezado Valido

Checksum del protocolo (pseudoencabezado): 00

Checksum del protocolo Valido

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:58 CDT 2017 caplen=100 len=100 jNetPcap rocks!

3E 24 31 87 46 94 A4 71 74 F1 7C 49 08 00 45 70

00 56 61 B3 00 00 38 06 C5 28 D8 3A C1 2E C0 A8

01 45 01 BB D4 EA E9 AE E1 0C C7 72 D7 6E 50 18

01 06 90 BB 00 00 17 03 03 00 29 00 00 00 00 00

00 00 16 1B 7A E3 9F 33 87 6E 95 CC BC BA 25 22

F4 3D 69 C6 6D 85 A1 94 38 DF 2E A3 21 3C 29 A1

47 76 A9 01

Trama IP

Protocolo = 06 - TCP

IHL: 20

Longitud del Pseudoencabezado: 66

Checksum del encabezado IP: 00

Checksum del encabezado Valido

Checksum del protocolo (pseudoencabezado): 00

Checksum del protocolo Valido

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:58 CDT 2017 caplen=54 len=54 jNetPcap rocks!

A4 71 74 F1 7C 49 3E 24 31 87 46 94 08 00 45 00

00 28 34 86 40 00 80 06 6A F3 C0 A8 01 45 D8 3A

C1 2E D4 EA 01 BB C7 72 D7 6E E9 AE E1 3A 50 10

00 42 13 CB 00 00

Trama IP

Protocolo = 06 - TCP

IHL: 20

Longitud del Pseudoencabezado: 20

Checksum del encabezado IP: 00

Checksum del encabezado Valido

Checksum del protocolo (pseudoencabezado): 00

Checksum del protocolo Valido

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:58 CDT 2017 caplen=126 len=126 jNetPcap rocks!

---------------------------------------------------------------------------

Received packet at Fri Sep 22 19:36:58 CDT 2017 caplen=112 len=112 jNetPcap rocks!

A4 71 74 F1 7C 49 3E 24 31 87 46 94 08 00 45 00

00 62 78 5E 40 00 80 06 4B F0 C0 A8 01 45 D0 4E

A4 0B C6 F0 69 8D 4A 65 C7 DB 57 27 C5 92 50 18

7F EB 34 9C 00 00 17 03 03 00 35 80 7D 30 D7 26

2D 08 BA D4 7D 6E A8 AE 8A C1 E7 09 EB E9 00 CC

A7 3B 47 2C AD B3 42 CA 7F EA 14 C0 67 57 AE 35

D7 3A 79 11 22 6D 6D 1C 46 6B 22 5E DD 18 A6 1E

Trama IP

Protocolo = 06 - TCP

IHL: 20

Longitud del Pseudoencabezado: 78

Checksum del encabezado IP: 00

Checksum del encabezado Valido

Checksum del protocolo (pseudoencabezado): 00

Checksum del protocolo Valido

BUILD SUCCESSFUL (total time: 11 seconds)

**Conclusiones**

**Galindo Reyes Agustin:** Con esta practica aprendimos la manera correcta de calcular el checksum de una trama y dependiendo del protocolo UDP/TCP, sabremos como proceder. Para poder calcular el checksum debimos aprender a sacar encabezados de la trama y de ahí poder definir un pseudo encabezado que seria la parte fundamental para poder hacer el respectivo calculo. Cabe destacar que se debe saber como dividir la trama y diferenciar los tipos de encabezados que tiene y saber en que bytes se encuentra cada cosa desde la MAC de origen, la longitud, el IHL para poder deducir bien nuestro pseudo encabezado.

**Pérez Leal Rodolfo:** Con la práctica entendí que por cada encabezado agregado en cada capa del modelo TCP existe un checksum por lo cual al usar el método del checksum para verificarlo se debe de además de usar el encabezado IP se debe calcular un pseudo encabezado correspondiente al protocolo TCP o UDP, para esto se tuvo que dividir la trama original dependiendo del tipo de protocolo que esta ocupara. Lo cual fue complicado pues se tenía que diferenciar que campo se encontraba en los diferentes bytes, asi como lo que este campo significaba para de esta forma poder ocuparlo a la hora de construir el pseudo encabezado, donde un campo importate es el de IHL que permite saber de que tamaño será el encabezado que utiliza el checksum.