Instituto Politécnico Nacional

Escuela Superior de Cómputo

PRÁCTICA NO. 2

Checksum

Redes de Computadora

Profesor: Axel Ernesto Moreno Cervantes

Grupo: 2CM10

Fecha: 02 / Abril /2018

Alumnos:

- Calva Hernández José Manuel 2017630201
- Ruíz López Luis Carlos 2014081397

Índice

Introducción

Una trama es la unidad de transmisión de las operaciones de la capa o nivel 2, enlace de datos, de una red. Cuando la capa de enlace de datos recibe un mensaje, le da formato para convertirlo en una trama de datos o paquete. Los campos que componen una trama ethernet son los siguientes:

- Preámbulo (Preamble).
  Campo de 7 bytes de longitud con una secuencia de bits utilizada para sincronizar y estabilizar el medio físico antes de iniciar la transmisión. Es una secuencia de unos y ceros. El patrón es el siguiente:
  10101010 10101010 10101010 10101010 10101010 10101010 10101010
- SFD (Start Frame Delimiter).
  Delimitador de inicio de trama. Campo de 1 byte de longitud que contiene la secuencia 10101011. Indica el inicio de una trama de datos.
- Dirección de destino (Destination Address).
  Campo de 6 bytes de longitud que contiene la dirección MAC a la que se envía la trama. El bit más a la izquierda del campo indica cuando la dirección es individual (indicado por un 0) o un grupo de direcciones (indicado por un 1). El segundo bit desde la izquierda indica cuando la dirección destino es globalmente administrada (indicado por un 0).
  La capa de enlace de datos del remitente añade la dirección de destino a la trama. La capa de enlace de datos del destinatario examina la dirección de destino para identificar los mensajes a recibir.
- Dirección de origen (Source Address).
  Campo de 6 bytes de longitud que contiene la dirección MAC del dispositivo que envía la trama. La dirección de origen es siempre una dirección individual y el bit más a la izquierda es siempre 0. Con ella el receptor conoce a quien debe dirigir las respuestas del mensaje.
- Tipo de protocolo o longitud.
  Campo de 2 bytes de longitud. Este campo es el que distingue a las tramas 802.3 de las tramas Ethernet. Valores para este campo iguales o menores de x05DC (1500 en decimal) indican que es una trama 802.3 y el valor representa la longitud del campo de datos.
  Valores para este campo iguales o mayores de x0600 indican que es una trama Ethernet y el valor representa el tipo de protocolo.
- Datos (Payload).
  Campo de 46 a 1500 bytes de longitud. Contiene los datos a transferir entre origen y destino. Si este campo fuera menor de 46 bytes se añade un campo de 'relleno' para mantener el tamaño mínimo de paquete.
- FCS (Frame Check Sequence). Secuencia de verificación de trama.
  Campo de 4 bytes de longitud que contiene un valor de para control de errores, CRC (Cyclical Redundancy Check). La verificación de redundancia cíclica (CRC), consiste en un valor calculado por el emisor que resume todos los datos de la trama. El receptor calcula nuevamente el valor y, si coincide con el de la trama, entiende que la trama se ha transmitido sin errores. El campo FCS es generado sobre los campos dirección de destino, la dirección de origen, el tipo/longitud y datos.

Desarrollo

La práctica consiste en capturar tramas que pasan a través de la tarjeta de red configurada en modo promiscuo. A partir de estas tramas analizamos si se trataba de una trama Ethernet, y posteriormente si el protocolo de Internet era IP. Para ello utilizamos la librería pcap y el código que nos fue proporcionado por el profesor.

Para esta práctica supusimos que la trama con la que se trataba era Ethernet, por lo cual no se revisó que el campo tipo/longitud fuera mayor o igual a 1500 bytes.

```
1.  int lon = packet.size();
2.  if (lon > 1500) {
3.      System.out.println("Trama Ethernet")
4.  }
```

Posteriormente, revisamos el byte 13 y 14 de la trama para verificar que el protocolo que se iba a utilizar era IPv4, es decir, revisamos que el valor fuera igual a 0x08 en el byte 13 y 0x00en el byte 14.

```
1.  byte[] trama = packet.getByteArray(0, lon);
2.  if (packet.size() > 12 && packet.getUByte(12) == 0x08 && packet.getUByte(13) == 00) {
3.      System.out.println("Protocolo IP");
4.  }
```

Una vez validado el tipo de protocolo como IPv4 continuamos a calcular el checksum para verificar posibles errores y dar por buena la trama.

El procedimiento que seguimos fue el siguiente:

1. Calcular la longitud del encabezado IP.
2. Crear un nuevo arreglo de bytes para almacenar el encabezado.
3. Identificar IP de origen e IP destino.
4. Calcular el checksum utilizando el método proporcionado por el profesor.

```
1.  int lonIP = 4 * (packet.getUByte(14) & 0x0F);
2.  byte[] encabezadoIP = new byte[lonIP];
3.  System.arraycopy(trama, 14, encabezadoIP, 0, lonIP);
4.  System.out.printf("Checksum IP: %02X\n", calculateChecksum(encabezadoIP));
```

```
1.  public static long calculateChecksum(byte[] buf) {
2.      int length = buf.length;
3.      int i = 0;
4.      long sum = 0;
5.      long data; // Handle all pairs
6.      while (length > 1) { // Corrected to include @Andy's edits and various comments on Stack Overflow
7.          data = (((buf[i] << 8) & 0xFF00) | ((buf[i + 1]) & 0xFF));
8.          sum += data; // 1's complement carry bit correction in 16-bits (detecting sign extension)
9.          if ((sum & 0xFFFF0000) > 0) {
10.             sum = sum & 0xFFFF;
11.             sum += 1;
12.         }
13.         i += 2;
14.         length -= 2;
15.     } // Handle remaining byte in odd length buffers
```

```
    if (length > 0) { // Corrected to include @Andy's edits and various comments on Stack Overflow
        sum += (buf[i] << 8 & 0xFF00); // 1's complement carry bit correction in 16-bits (detecting sign extension)
        if ((sum & 0xFFFF0000) > 0) {
            sum = sum & 0xFFFF;
            sum += 1;
        }
    } // Final 1's complement value correction to 16-bits
    sum = ~sum;
    sum = sum & 0xFFFF;
    return sum;
}
```

Finalmente, procedimos a identificar el protocolo que se utilizó en la capa de transporte. Para esto revisamos el valor del byte 24 de la trama. Si el valor resultaba ser 0x06, sabíamos que se trataba de TCP. Por otro lado, si el valor era 0x11, el protocolo era UDP.

```
if (lon > 23 && trama[23] == 0x06) {
    System.out.println("Protocolo TCP");
} else if ((lon > 23) && (trama[23] == 0x11)) {
    System.out.println("Protocolo UDP");
}
```

Para ambos casos se requería calcular el valor del checksum. Sin embargo, para ambos había que calcular previamente un pseudo-header que se utiliza en el cálculo del checksum.

En cuanto al checksum de TCP, lo calculamos de la manera siguiente:

1. Calculamos la longitud del encabezado TCP.
2. Creamos un nuevo arreglo que almacenaba el encabezado TCP.
3. Creamos un arreglo que iba a contener la información correspondiente al pseudo-header.
4. Agregamos la IP de origen y destino al pseudo-header, en ese orden.
5. Agregamos la información correspondiente a los bytes 9 y 10. En el byte 9 se asigna el valor de 0 y en el byte 10 el valor de 0x06 correspondiente a que se trata de un protocolo TCP.
6. Calculamos la longitud del payload utilizando la longitud total menos la longitud del encabezado IP menos 14 que es la longitud del encabezado TCP.
7. Agregamos la información del payload a los bytes 11 y 12.
8. Creamos un nuevo arreglo que contendrá el payload de TCP y copiamos la información.
9. Creamos un arreglo auxiliar que contendrá: el pseudo-header, el encabezado TCP y el payload.
10. Calculamos el checksum utilizando el arreglo auxiliar.

```
int lonTCP = (((trama[16] << 8) & 0xFF00) | ((trama[17]) & 0xFF));
lonTCP -= lonIP;
byte[] encabezadoTCP = new byte[lonTCP];
System.arraycopy(trama, 14 + lonIP, encabezadoTCP, 0, lonTCP);
byte[] pseudoEncabezado = new byte[12]; // IP Origen + IP Destino + byte ceros + protocolo + longitud pdu transporte + pdu transporte
System.arraycopy(IP_origen, 0, pseudoEncabezado, 0, 4); // IP Origen
System.arraycopy(IP_destino, 0, pseudoEncabezado, 4, 4); // IP Destino
pseudoEncabezado[8] = 0x00; // byte ceros
pseudoEncabezado[9] = 0x06; // protocolo
int lonTCP_payload = lon - 14 - lonIP;
pseudoEncabezado[10] = (byte)((lonTCP_payload & 0xFF00) >> 8);
```

```
12. pseudoEncabezado[11] = (byte)(lonTCP_payload & 0xFF);
13. int len_payload = lon - 14 - lonIP - lonTCP;
14. byte[] payload = new byte[len_payload];
15. System.arraycopy(trama, 14 + lonIP + lonTCP, payload, 0, len_payload);
16. byte[] tmp = new byte[12 + lonTCP_payload];
17. System.arraycopy(pseudoEncabezado, 0, tmp, 0, 12);
18. System.arraycopy(encabezadoTCP, 0, tmp, 12, lonTCP);
19. System.arraycopy(payload, 0, tmp, 12 + lonTCP, len_payload);
20. System.out.printf("Checksum TCP: %02X\n", calculateChecksum(tmp));
```

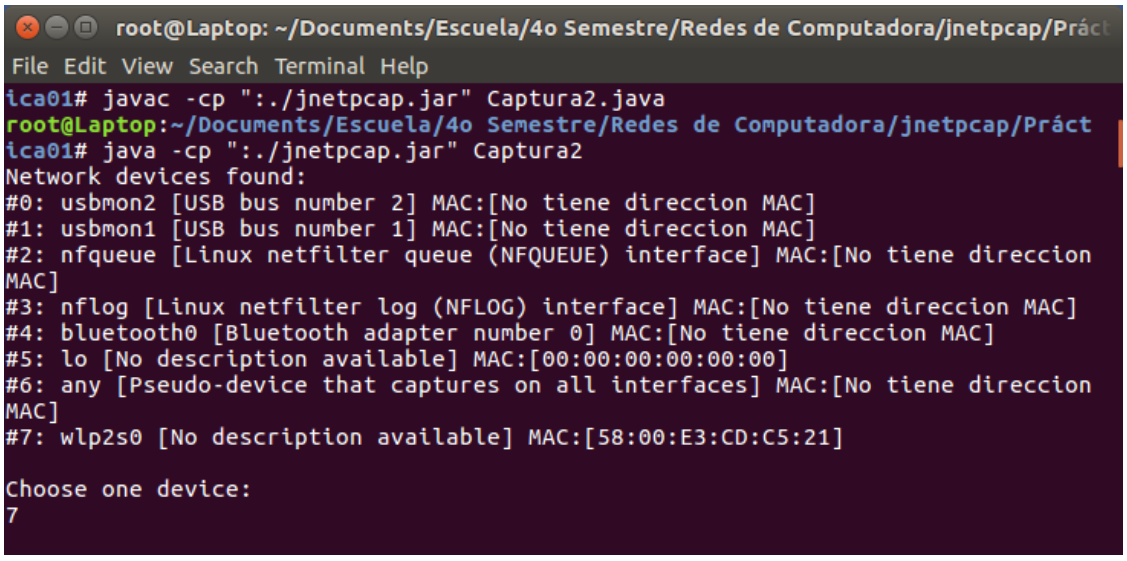
Por otra parte el checksum de UPD, lo calculamos de la manera siguiente:

1. Creamos un arreglo que contendrá el encabezado UDP y copiamos la información.
2. Creamos un arreglo de 12 bytes que nos servirá para crear el pseudo-header.
3. Copiamos el IP origen y el IP destino al pseudo-header.
4. Inicializamos los bytes 9 y 10. El 9 se inicializa a 0x00 y el 10 a 0x11 debido a que se trata del protocolo UDP.
5. Copiamos el encabezado UDP al pseudo-header.
6. Calculamos la longitud del payload: longitud de la trama menos longitud de IP menos longitud de UDP menos 14.
7. Creamos un arreglo de bytes que contendrá el payload y copiamos la información de la trama.
8. Creamos un arreglo temporal de bytes para realizar el cálculo del checksum. Agregamos: pseudo-header, encabezado UDP y el payload, en ese orden.
9. Realizamos el cálculo del checksum.

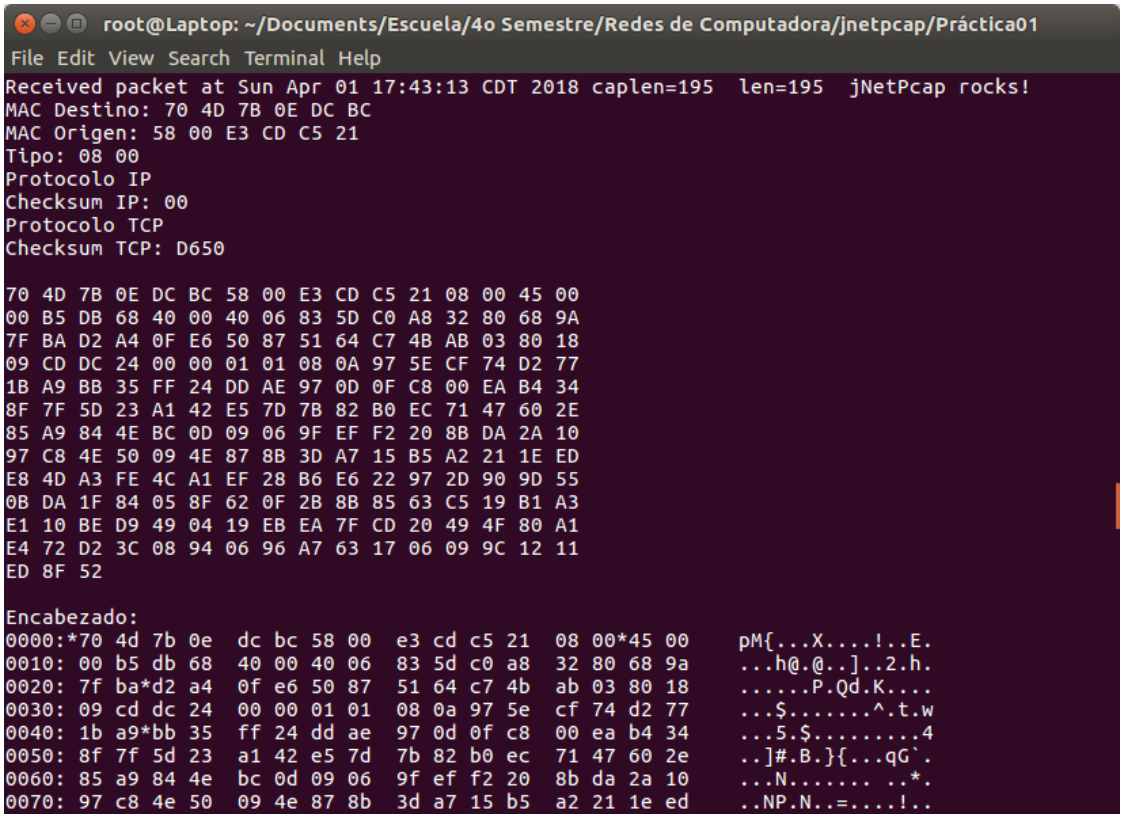
```
1.  int lonUDP = 8;
2.  byte[] encabezadoUDP = new byte[lonUDP];
3.  System.arraycopy(trama, 14 + lonIP, encabezadoUDP, 0, lonUDP);
4.  byte[] pseudoEncabezado = new byte[12];
5.  System.arraycopy(IP_origen, 0, pseudoEncabezado, 0, 4);
6.  System.arraycopy(IP_destino, 0, pseudoEncabezado, 4, 4);
7.  pseudoEncabezado[8] = 0x00;
8.  pseudoEncabezado[9] = 0x11;
9.  System.arraycopy(encabezadoUDP, 4, pseudoEncabezado, 10, 2);
10. int len_payload = lon - 14 - lonIP - lonUDP;
11. byte[] payload = new byte[len_payload];
12. System.arraycopy(trama, 14 + lonIP + lonUDP, payload, 0, len_payload);
13. byte[] tmp = new byte[12 + lonUDP + len_payload];
14. System.arraycopy(pseudoEncabezado, 0, tmp, 0, 12);
15. System.arraycopy(encabezadoUDP, 0, tmp, 12, lonUDP);
16. System.arraycopy(payload, 0, tmp, 12 + lonUDP, len_payload);
17. System.out.printf("Checksum UDP: %02X\n", calculateChecksum(tmp));
```

Capturas

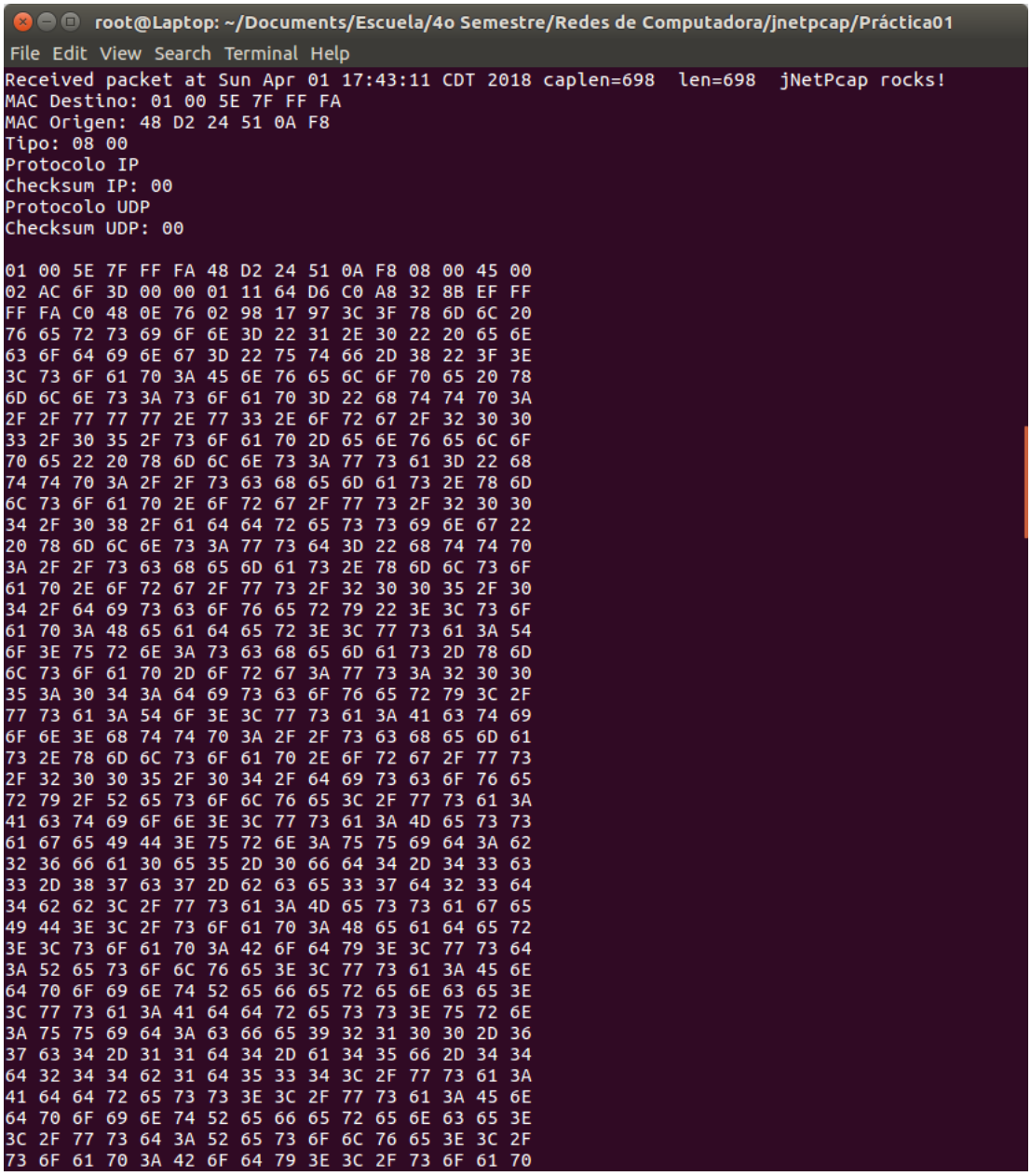
Compilación y ejecución del programa:



Verificación de Checksum en tramas de tipo TCP:



Verificación de Checksum en tramas de tipo UDP:



Conclusiones

- Calva Hernández José Manuel: La finalidad de la práctica se cumplió, ya que se pudo implementar el método de checksum que nos fue proporcionado por el profesor para la verificación de la integridad de las tramas, y acorde a esto, nos dimos cuenta de que muchas tramas TCP llegaban incompletas a la hora de verificarlas ya que el checksum era distinto de 0. Sin embargo, tanto las tramas UDP como el checksum en el encabezado IP tendían a estar correctos.
- Ruíz López Luis Carlos: Esta práctica nos mostró como identificar el campo checksum de las tramas, pero teniendo en cuenta el tipo de trama que se tiene diferenciándolas entre si es IP, TCP o UDP con lo visto en clase y este padre poder verlo gráficamente.