Práctica 6 – Programación: Procesado de tráfico

1. Introducción

El objetivo de esta práctica es realizar un programa que nos permita llevar a cabo unos análisis básicos del tráfico generado al navegar por la web.

Para llevar a cabo el análisis necesitaremos una muestra de tráfico. Podremos obtener una si capturamos con tshark y simultáneamente visitamos algunas páginas web con el navegador. Emplee por ejemplo las siguientes opciones, que le darán una salida de texto con una línea por cada paquete IP capturado:

```
$ tshark -i <interface> -f "ip" -l -T fields -E occurrence=f -E separator=, -e frame.number -e frame.time_epoch -e ip.src -e ip.dst -e ip.proto -e tcp.srcport -e tcp.dstport -e udp.srcport -e udp.dstport -e frame.len > captura.csv
```

Si está utilizando los escritorios virtualizados de la UPNA, en estas máquinas no tendrá la posibilidad de utilizar tshark, pero puede subir ficheros de texto generados en otra máquina. Simplemente debe arrastrarlos a la ventana del VDI y deberían subirse a `/home/alumno/thinclient_drives/GUACFS/`.

El fichero generado será un csv (*Comma Separated Values*) con las siguientes columnas:

1. Número del paquete en la captura
2. Fecha en la que se ha capturado el paquete, en formato unix epoch[1]
3. Dirección IP origen
4. Dirección IP destino
5. Valor del campo protocolo de la cabecera IP (indica el protocolo de la PDU que viaja en los datos del paquete IP).
6. Puerto origen, en caso de que el protocolo de transporte sea TCP.
7. Puerto destino, en caso de que el protocolo de transporte sea TCP.
8. Puerto origen, en caso de que el protocolo de transporte sea UDP.
9. Puerto destino, en caso de que el protocolo de transporte sea UDP.
10. Tamaño del paquete ethernet, en bytes (probablemente sin contar el CRC).

En los materiales de la asignatura encontrará un fichero de ejemplo de captura de tráfico con el que podrá validar los resultados de los distintos apartados.

Durante la práctica se realizarán todos los ejercicios en funciones separadas que se ejecutarán desde la función main, por lo que se irán añadiendo funcionalidades al programa a lo largo de la práctica, pero nunca eliminando. La ejecución del programa

[1] https://en.wikipedia.org/wiki/Unix_time

deberá ser:

```
$ java P6 <id_ejercicio> <captura>
```

2. Lectura del fichero de captura

En primer lugar, se debe leer el fichero de captura y almacenarlo en una lista de paquetes.

Para almacenar los paquetes en memoria se debe utilizar la siguiente clase:

```
class NetPacket {
    float time;
    String srcIP;
    String dstIP;
    int proto;
    Integer srcPort;
    Integer dstPort;
    int len;

    public NetPacket(float time, String srcIP, String dstIP, int proto,
Integer srcPort, Integer dstPort, int len) {
         this.time = time;
         this.srcIP = srcIP;
         this.dstIP = dstIP;
         this.proto = proto;
         this.srcPort = srcPort;
         this.dstPort = dstPort;
         this.len = len;
    }
}
```

Los campos `srcPort` y `dstPort` se definen como `Integer` en lugar de `int` para que cuando el protocolo no sea ni TCP (6) ni UDP (17), se pueda almacenar `null`.

En este apartado se pide completar y utilizar la siguiente función para leer el fichero de captura y almacenar los paquetes.

```
public static ArrayList<NetPacket> readFile(String filename) {
        ArrayList<NetPacket> pkts = new ArrayList<>();
        return pkts;
}
```

3. Número de hosts distintos

En este apartado se pide contar el número de hosts involucrados en las comunicaciones capturadas. Para ello, hay que contar el número de direcciones IP distintas que aparecen en la captura, considerando IPs tanto origen como destino.

Para la realización de este apartado deberá completarse y utilizarse la siguiente función:

```
public static int countUniqueIPs(ArrayList<NetPacket> pkts) {}
```

El valor `id_ejercicio` (parámetro en la línea de ejecucuón) para este aparado será `unique_hosts`. La ejecución del programa, por tanto, deberá ser:

```
$ java P6 unique_hosts <captura>
```

La salida mostrará el número de hosts distintos. Por ejemplo:

```
$ java P6 unique_hosts captura.csv

48
```

Alternativamente, puede validar el funcionamiento de su programa comparando el resultado con:

```
$ awk -F "," '{print $3; print $4}' <captura> | sort | uniq | wc -l
```

Punto de control 1: Muestre al profesor el programa indicado. (20%)

4. Host con más paquetes recibidos

Se pide obtener el host que más paquetes ha recibido durante la captura. Para ello, deberá contar sobre la columna de dirección IP destino, cuántas veces aparece cada una de las direcciones IP. Finalmente, deberá seleccionar aquella dirección IP que aparece con mayor frecuencia.

Para la realización de este apartado deberá completarse y utilizarse la siguiente función:

```
public static Map.Entry<String, Integer>
topIpByReceivedPackets(ArrayList<NetPacket> pkts) {}
```

El valor `id_ejercicio` para este aparado será `top_received_pkts`. La ejecución del programa, por tanto, deberá ser:

```
$ java P6 top_received_pkts <captura>
```

La salida mostrará la dirección IP del host con más paquetes junto con el número de paquetes, separados por un espacio. Por ejemplo:

```
$ java P6 top_received_pkts captura.csv

192.168.1.90 2393
```

Alternativamente, puede validar el funcionamiento de su programa comparando el resultado con:

```
$ awk -F "," '{print $4}' <captura> | sort | uniq -c | sort -nr | head -n
1 | sed -r 's/ +([0-9]+) ([0-9\.]+)/\2 \1/g'
```

5. Host con más bytes enviados

Se pide obtener el host que más bytes ha enviado durante la captura. Para ello, deberá obtener la suma de la columna del tamaño del paquete para cada dirección IP origen. Finalmente, deberá seleccionar aquella dirección IP con mayor número de bytes acumulados.

Para la realización de este apartado deberá completarse y utilizarse la siguiente función:

```
public static Map.Entry<String, Integer>
topIpBySentBytes(ArrayList<NetPacket> pkts) {}
```

El valor `id_ejercicio` para este aparado será `top_sent_bytes`. La ejecución del programa, por tanto, deberá ser:

```
$ java P6 top_sent_bytes <captura>
```

La salida mostrará la dirección IP del host con más bytes junto con el número de bytes, separados por un espacio. Por ejemplo:

```
$ java P6 top_sent_bytes captura.csv

142.250.200.100 825890
```

Alternativamente, puede validar el funcionamiento de su programa comparando el resultado con:

```
$ awk -F "," '{bytes[$3] += $10} END{for (ip in bytes) print ip, bytes
[ip]}' <captura> | sort -nr -k 2 | head -n 1
```

Punto de control 3: Muestre al profesor el programa indicado. (40%)

6. Número de flujos de transporte

Se pide contar el número de conexiones (flujos a nivel de la capa de transporte) que se han establecido durante las comunicaciones. Una conexión se identifica por la combinación de 3 elementos: el protocolo de la capa de transporte y cada uno los extremos de la comunicación (compuestos cada uno por dirección IP y puerto), independientemente de si aparecen como origen o destino.

En el fichero de captura del que se dispone, el puerto a utilizar dependerá de si el protocolo de transporte es TCP (6) o UDP (17). Para este análisis ignoraremos el tráfico del resto de protocolos.

Por ejemplo, en el siguiente extracto de una captura se observan paquetes de 4 flujos diferentes:

329,1729379562.336972850,192.168.1.90,80.58.61.250,17,,,44111,53,86

340,1729379562.349534486,192.168.1.90,5.255.145.201,6,48312,80,,,74

341,1729379562.349576805,192.168.1.90,5.255.145.201,6,48318,80,,,74

345,1729379562.361851128,80.58.61.250,192.168.1.90,17,,,53,44111,185

347,1729379562.365954125,5.255.145.201,192.168.1.90,6,80,48312,,,74

348,1729379562.365989155,192.168.1.90,5.255.145.201,6,48312,80,,,66

349,1729379562.365954227,5.255.145.201,192.168.1.90,6,80,48318,,,74

350,1729379562.366007122,192.168.1.90,5.255.145.201,6,48318,80,,,66

357,1729379562.378454957,192.168.1.90,5.255.145.201,6,48320,80,,,74

359,1729379562.379803175,192.168.1.90,5.255.145.201,6,48318,80,,,505

360,1729379562.379905642,192.168.1.90,5.255.145.201,6,48312,80,,,505

369,1729379562.395324780,5.255.145.201,192.168.1.90,6,80,48320,,,74

En este ejemplo, los flujos serían los siguientes:

- 17, 192.168.1.90:44111, 80.58.61.250:53
- 6, 192.168.1.90:48312, 5.255.145.201:80
- 6, 192.168.1.90:48318, 5.255.145.201:80
- 6, 192.168.1.90:48320, 5.255.145.201:80

En este ejercicio se pide contar el número de combinaciones diferentes de estos 3 valores que aparecen en la captura.

Para la realización de este apartado deberá agrupar los paquetes en flujos utilizando la siguiente clase. Esta trae una función que permite generar un identificador del flujo que es independientes de cuál de los dos extremos del mismo es el origen del paquete:

```
class NetFlow {
        HashSet<String> flowID;
        ArrayList<NetPacket> pkts;

        public NetFlow(HashSet<String> flowID, ArrayList<NetPacket> pkts)
{
            this.flowID = flowID;
            this.pkts = pkts;
        }

        public static HashSet<String> flowID(NetPacket pkt) {
            return new HashSet<String>(Arrays.asList(new String[]{
                Integer.toString(pkt.proto),
                String.format("%s:%s", pkt.srcIP, pkt.srcPort),
                String.format("%s:%s", pkt.dstIP, pkt.dstPort),
            }));
        }
}
```

La agrupación de los paquetes en flujos deberá realizarse completando la siguiente función:

public static ArrayList<NetFlow> groupPacketsByFlow(ArrayList<NetPacket> pkts) {}

El valor `id_ejercicio` para este aparado será `num_flows`. La ejecución del programa, por tanto, deberá ser:

```
$ java P6 num_flows <captura>
```

La salida mostrará el número de flujos distintos. Por ejemplo:

```
$ java P6 num_flows captura.csv
201
```

6. Top 5 flujos de transporte por bytes.

Vamos a obtener los 5 flujos que más bytes han intercambiado durante la captura. Para ello, deberá obtener la suma del tamaño de todos los paquetes de cada uno y, posteriormente, seleccionar los 5 flujos con mayor volumen de bytes acumulados.

Para la realización de este apartado deberá completarse y utilizarse la siguiente función:

public static List<Map.Entry<HashSet<String>, Integer>>
top5FlowsByTotalBytes(ArrayList<NetFlow> flows) {}

El valor `id_ejercicio` para este aparado será `top5_flow_bytes`. La ejecución del programa, por tanto, deberá ser:

```
$ java P6 top5_flow_bytes <captura>
```

La salida mostrará los identificadores de cada flujo (protocolo, direcciones IP y puertos) y el número de bytes, para cada una de los 5 flujos con más bytes. Por ejemplo:

```
$ java P6 top5_flow_bytes captura.csv
udp 142.250.200.100:443 192.168.1.90:51821 866221
udp 192.168.1.90:59048 142.250.184.3:443 829779
tcp 192.168.1.90:60928 34.36.165.17:443 40441
tcp 192.168.1.90:51772 34.120.208.123:443 36750
udp 172.217.17.14:443 192.168.1.90:59954 33144
```

Punto de control 5: Muestre al profesor el programa indicado. (40%)