Práctica 7 – Programando en Java (opción 4)

Objetivos

El objetivo de esta práctica es poner en práctica los conocimientos adquiridos a lo largo del curso sobre programación Java. La duración será de 2 horas y su valor es de 0,5 puntos

1- Especificación (0,375 puntos)

Realice un programa que lea la lista de los paquetes y muestre la cantidad total de bytes que ve en cada intervalo de tiempo de duración indicada:
Uso:
`java TimeThr <nombredelfichero> <duración del intervalo>`

Lee un fichero con el nombre indicado con líneas que contienen la información de una trama Ethernet observada en la red por orden de aparición. Cada línea tendrá el formato

<tiempo> <tamaño de la trama Ethernet> <nombre_protocolo> <desglose_del_frame> <MAC origen> <MAC destino>

El programa agrupará las tramas que aparezcan en cada intervalo de duración indicada mostrando para cada intervalo que pase: el tiempo final, la cantidad de bytes observada en el intervalo y la cantidad de bytes totales observada hasta ese momento.

NOTA: Para poder hacer los cálculos por intervalos, necesitamos saber, para cada línea, a qué intervalo se corresponde. Esto se puede hacer con dos variables: una que guarde la duración del intervalo que nos pasan y otra con el intervalo actual. Esta variable la inicializaremos a uno.

Si el valor del tiempo es menor que el intervaloActual * duracionIntervalo, sumamos en dos variables: una que lleve la cuenta total y otra que lleve la cuenta del intervalo.

Si el valor del tiempo es mayor, sumaremos uno al intervalo actual, imprimiremos el resultado del intervalo y reiniciaremos la variable sumaIntervalo.

Ejemplos:

$ cat captura_4.txt

```
0.000000000 812 TLSv1 eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.007136000 81 DNS eth:ip:udp:dns e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.025526000 120 DNS eth:ip:udp:dns bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
0.025886000 78 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.203658000 60 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
0.224625000 74 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
0.224679000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.227117000 140 SSL eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.426741000 66 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
0.432235000 1514 TLSv1 eth:ip:tcp:ssl bc:14:01:0b:5d:72 e0:ca:
94:69:d2:a0
0.433294000 1514 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
0.433323000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.433578000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.465280000 264 TLSv1 eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.700226000 66 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
0.858970000 125 TLSv1 eth:ip:tcp:ssl bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
0.859018000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
0.860302000 812 TLSv1 eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.062825000 66 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.187496000 871 TLSv1 eth:ip:tcp:ssl bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.187541000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.196062000 267 HTTP eth:ip:tcp:http:data-text-lines bc:14:01:0b:5d:72
e0:ca:94:69:d2:a0
1.196111000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.222449000 1372 TLSv1 eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:
72
1.229003000 78 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.427270000 60 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.439517000 74 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.439569000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.441012000 140 SSL eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.654139000 66 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.662969000 1514 TLSv1 eth:ip:tcp:ssl bc:14:01:0b:5d:72 e0:ca:
94:69:d2:a0
1.663811000 1514 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.663845000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.664073000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.698578000 264 TLSv1 eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.731015000 230 DB-LSP-DISC eth:ip:udp:db-lsp-disc e0:ca:94:69:d2:a0
ff:ff:ff:ff:ff:ff
1.731480000 230 DB-LSP-DISC eth:ip:udp:db-lsp-disc e0:ca:94:69:d2:a0
ff:ff:ff:ff:ff:ff
1.732081000 492 TLSv1 eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.915418000 125 TLSv1 eth:ip:tcp:ssl bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.915462000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
1.916645000 1372 TLSv1 eth:ip:tcp:ssl e0:ca:94:69:d2:a0 bc:14:01:0b:5d:
72
1.930572000 66 TCP eth:ip:tcp bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.966718000 615 TLSv1 eth:ip:tcp:ssl bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0
1.966760000 66 TCP eth:ip:tcp e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72
```

```
$ java TimeThr captura_4.txt 0.01

0.010000 893 893 714400.0
0.030000 198 1091 158400.0
0.210000 60 1151 48000.0
0.230000 280 1431 224000.0
0.430000 66 1497 52800.0
0.440000 3160 4657 2528000.0
0.470000 264 4921 211200.0
0.710000 66 4987 52800.0
0.860000 191 5178 152800.0
0.870000 812 5990 649600.0
1.070000 66 6056 52800.0
1.190000 937 6993 749600.0
1.200000 333 7326 266400.0
1.230000 1450 8776 1160000.0
1.430000 60 8836 48000.0
1.440000 140 8976 112000.0
1.450000 140 9116 112000.0
1.660000 66 9182 52800.0
1.670000 3160 12342 2528000.0
1.700000 264 12606 211200.0
1.740000 952 13558 761600.0
1.920000 1563 15121 1250400.0
1.940000 66 15187 52800.0
1.970000 681 15868 544800.0
```

2- Mejorando el programa (0,125 puntos)

Queremos que el programa sea capaz de filtrar sólo los paquetes enviados o recibidos por diferentes máquinas. Para ello queremos tener un filtro que nos permita centrarnos en algunos paquetes. En el caso en que indiquemos filtros el programa esperará que las líneas de entrada tengan cuatro valores.

Uso:

java TimeThr <nombrefichero> <duración del intervalo> <mac_destino> <protocolo>

El programa hace lo mismo que el anterior, pero solo cuenta los paquetes si tienen el destino y protocolo indicados. El parámetro destino puede ser una dirección MAC o bien la palabra “any”, para indicar que nos da igual cualquier destino o cualquier origen.

Además, el programa deberá ser capaz de filtrar los resultados por protocolos, siguiendo el mismo principio. Si el parámetro es “TCP”, buscaremos sólo paquetes TCP, mientras que si es “any” nos valdrá cualquier protocolo.

La salida será igual que en el primer programa pero teniendo en cuenta solo los paquetes que pidamos. Para ello, recibiremos como entrada una cadena que especificará la mac origen y el protocolo buscado.

Ejemplos:

$ java TimeThr captura_4.txt 0.01 bc:14:01:0b:5d:72 TCP

```
0.010000 0 0 0.0
0.030000 78 78 62400.0
0.230000 66 144 52800.0
0.440000 132 276 105600.0
0.860000 66 342 52800.0
1.190000 66 408 52800.0
1.200000 66 474 52800.0
1.230000 78 552 62400.0
1.440000 66 618 52800.0
1.670000 132 750 105600.0
1.920000 66 816 52800.0
1.970000 66 882 52800.0
```

$ java TimeThr captura_4.txt 0.01 any TCP

```
0.010000 0 0 0.0
0.030000 78 78 62400.0
0.210000 60 138 48000.0
0.230000 140 278 112000.0
0.430000 66 344 52800.0
0.440000 1646 1990 1316800.0
0.710000 66 2056 52800.0
0.860000 66 2122 52800.0
1.070000 66 2188 52800.0
1.190000 66 2254 52800.0
1.200000 66 2320 52800.0
1.230000 78 2398 62400.0
1.430000 60 2458 48000.0
1.440000 140 2598 112000.0
1.660000 66 2664 52800.0
1.670000 1646 4310 1316800.0
1.920000 66 4376 52800.0
1.940000 66 4442 52800.0
1.970000 66 4508 52800.0
```

```
$ java TimeThr captura_4.txt 0.01 e0:ca:94:69:d2:a0 any

0.010000 0 0 0.0
0.030000 120 120 96000.0
0.210000 60 180 48000.0
0.230000 74 254 59200.0
0.430000 66 320 52800.0
0.440000 3028 3348 2422400.0
0.710000 66 3414 52800.0
0.860000 125 3539 100000.0
1.070000 66 3605 52800.0
1.190000 871 4476 696800.0
1.200000 267 4743 213600.0
1.430000 60 4803 48000.0
1.440000 74 4877 59200.0
1.660000 66 4943 52800.0
1.670000 3028 7971 2422400.0
1.920000 125 8096 100000.0
1.940000 66 8162 52800.0
1.970000 615 8777 492000.0
```

Ayuda

Comparar una cadena con otra:

```
public boolean equals(Object anObject)
```

Trocear una línea en las columnas que lo componen:

```
Método nextToken() de la clase StringTokenizer()

StringTokenizer st = new StringTokenizer("this is a test");
st.nextToken();

Operador y lógico: &&


Ejemplo:

String ip = st.nextToken();
String protocolo = st.nextToken();
if(ip.equals("10.1.1.1") && protocolo.equals("TCP")){
     ...
}
```