Práctica 7 – Programando en Java (opción 3)

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> <MAC origen> <MAC destino> <desglose_del_frame> <nombre_protocolo>

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_3.txt

```
0.000000000 66 bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0 eth:ip:tcp TCP
0.321012000 66 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp TCP
0.590781000 477 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa eth:ip:udp:http SSDP
0.600547000 549 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa eth:ip:udp:http SSDP
0.610379000 486 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa eth:ip:udp:http SSDP
0.615168000  515  00:21:5d:81:6d:96  33:33:00:00:00:0c  eth:ipv6:udp:http
SSDP
0.620974000 531 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa eth:ip:udp:http SSDP
0.626081000  560  00:21:5d:81:6d:96  33:33:00:00:00:0c  eth:ipv6:udp:http
SSDP
0.630536000 486 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa eth:ip:udp:http SSDP
0.635238000  515  00:21:5d:81:6d:96  33:33:00:00:00:0c  eth:ipv6:udp:http
SSDP
0.640535000 525 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa eth:ip:udp:http SSDP
0.645782000  554  00:21:5d:81:6d:96  33:33:00:00:00:0c  eth:ipv6:udp:http
SSDP
0.656351000  586  00:21:5d:81:6d:96  33:33:00:00:00:0c  eth:ipv6:udp:http
SSDP
0.666819000  515  00:21:5d:81:6d:96  33:33:00:00:00:0c  eth:ipv6:udp:http
SSDP
0.671897000 545 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa eth:ip:udp:http SSDP
0.679224000  574  00:21:5d:81:6d:96  33:33:00:00:00:0c  eth:ipv6:udp:http
SSDP
0.693778000 539 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa eth:ip:udp:http SSDP
0.698987000  568  00:21:5d:81:6d:96  33:33:00:00:00:0c  eth:ipv6:udp:http
SSDP
0.918165000  245  bc:14:01:0b:5d:72  e0:ca:94:69:d2:a0
eth:ip:tcp:http:data-text-lines HTTP
0.918206000 66 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp TCP
0.920351000 561 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp:http HTTP
0.997228000  84  00:21:5d:81:6d:96  33:33:00:01:00:03  eth:ipv6:udp:dns
LLMNR
0.998555000 64 00:21:5d:81:6d:96 01:00:5e:00:00:fc eth:ip:udp:dns LLMNR
1.099663000  84  00:21:5d:81:6d:96  33:33:00:01:00:03  eth:ipv6:udp:dns
LLMNR
1.100832000 64 00:21:5d:81:6d:96 01:00:5e:00:00:fc eth:ip:udp:dns LLMNR
1.111161000 66 bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0 eth:ip:tcp TCP
1.665976000 812 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp:ssl TLSv1
1.667813000 80 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:udp:dns DNS
1.685991000 103 bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0 eth:ip:tcp:ssl TLSv1
1.686037000 66 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp TCP
1.687323000 66 bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0 eth:ip:tcp TCP
1.687352000 66 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp TCP
1.690863000 96 bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0 eth:ip:udp:dns DNS
1.691555000 78 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp TCP
1.860375000 60 bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0 eth:ip:tcp TCP
1.893860000 74 bc:14:01:0b:5d:72 e0:ca:94:69:d2:a0 eth:ip:tcp TCP
1.893913000 66 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp TCP
1.896241000 140 e0:ca:94:69:d2:a0 bc:14:01:0b:5d:72 eth:ip:tcp:ssl SSL
```

```
$ java TimeThr captura_3.txt 0.01

0.010000 66 66 52800.0
0.330000 66 132 52800.0
0.600000 477 609 381600.0
0.610000 549 1158 439200.0
0.620000 1001 2159 800800.0
0.630000 1091 3250 872800.0
0.640000 1001 4251 800800.0
0.650000 1079 5330 863200.0
0.660000 586 5916 468800.0
0.670000 515 6431 412000.0
0.680000 1119 7550 895200.0
0.700000 1107 8657 885600.0
0.920000 311 8968 248800.0
0.930000 561 9529 448800.0
1.000000 148 9677 118400.0
1.100000 84 9761 67200.0
1.110000 64 9825 51200.0
1.120000 66 9891 52800.0
1.670000 892 10783 713600.0
1.690000 301 11084 240800.0
1.700000 174 11258 139200.0
1.870000 60 11318 48000.0
1.900000 280 11598 224000.0
```

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_origen> <mac_destino>

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

Ejemplos

$java TimeThr captura_3.txt 0.01 00:21:5d:81:6d:96 01:00:5e:7f:ff:fa

```
0.010000 0 0 0.0
0.600000 477 477 381600.0
0.610000 549 1026 439200.0
0.620000 486 1512 388800.0
0.630000 531 2043 424800.0
0.640000 486 2529 388800.0
0.650000 525 3054 420000.0
0.680000 545 3599 436000.0
0.700000 539 4138 431200.0
3.670000 486 4624 388800.0
```

$java TimeThr captura_3.txt 0.01 00:21:5d:81:6d:96 any

```
0.010000 0 0 0.0
0.600000 477 477 381600.0
0.610000 549 1026 439200.0
0.620000 1001 2027 800800.0
0.630000 1091 3118 872800.0
0.640000 1001 4119 800800.0
0.650000 1079 5198 863200.0
0.660000 586 5784 468800.0
0.670000 515 6299 412000.0
0.680000 1119 7418 895200.0
0.700000 1107 8525 885600.0
1.000000 148 8673 118400.0
1.100000 84 8757 67200.0
1.110000 64 8821 51200.0
```

```
$java TimeThr captura_3.txt 0.01 any 33:33:00:00:00:0c

0.010000 0 0 0.0
0.620000 515 515 412000.0
0.630000 560 1075 448000.0
0.640000 515 1590 412000.0
0.650000 554 2144 443200.0
0.660000 586 2730 468800.0
0.670000 515 3245 412000.0
0.680000 574 3819 459200.0
0.700000 568 4387 454400.0
```

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”)){
     ...
}
```