

# TRABAJO DE FIN DE CARRERA

**TÍTULO DEL TFC:** Evaluación de técnicas de captura y muestreo de tráfico en redes de ordenadores

**TITULACIÓN:** Ingeniería Técnica de Telecomunicaciones, especialidad  
Telemática

**AUTOR:** Xavier Bastías López

**DIRECTOR:** David Rincón Rivera

**FECHA:** 20 de diciembre de 2012

**Título:** Evaluación de técnicas de captura y muestreo de tráfico en redes de ordenadores

**Autor:** Xavier Bastías López

**Director:** David Rincón Rivera

**Fecha:** 20 de diciembre de 2012

## Resumen

En las redes de ordenadores (típicamente redes IP), el análisis de tráfico con fines de administración y control se realiza construyendo un registro de los flujos IP activos. Esta monitorización se realiza en ciertos nodos de la red y forma parte de las operaciones relacionadas con la gestión de la red.

Cisco Systems desarrolló NetFlow para la medición de tráfico mediante el análisis de los paquetes que circulan a través de dispositivos de red (routers y comutadores). Capturar todos los paquetes es una tarea que consume muchos recursos. Netflow permite el muestreo de paquetes, aportando información útil, pero consumiendo recursos moderados en el dispositivo. Sin embargo, la estimación del número de paquetes y del volumen en bytes de cada flujo se ve afectada por el muestreo.

En este trabajo se presentan diferentes escenarios, configuraciones y scripts, en los que se mezclarán conceptos de generación y captura de tráfico junto con la evaluación de hardware de alta precisión (DAG) y software de muestreo de tráfico IP (Netflow), así como un sistema de obtención y procesado para los datos, con el objetivo es realizar un estudio sobre el comportamiento de Sampled Netflow para evaluar el error cometido (al muestrear) en la estimación del volumen y de la cantidad de paquetes de cada flujo IP. No nos consta que un estudio experimental de estas características se haya realizado previamente.

Los resultados del montaje experimental confirman tanto la teoría como las simulaciones, aunque (debido a la modesta cantidad de flujos y de paquetes generados, por limitaciones de equipos y tiempo) son más precisas para las probabilidades más altas de muestreo, y no tanto para las más bajas.

**Title:** Evaluation of traffic capture and sampling techniques in computer networks

**Author:** Xavier Bastías López

**Director:** David Rincón Rivera

**Date:** December, 20th 2012

## Overview

In computer networks (typically IP-based), traffic analysis for control and administration purposes is accomplished by keeping a log of the active IP flows. This monitoring is performed in certain nodes of the network and is part of the operations related to the network management.

Cisco Systems developed NetFlow for traffic measurement and analysis of the packets that are routed through network devices (routers and switches). Capturing all packets is a task that consumes many resources. Netflow allows packet sampling, providing useful information, but consuming moderate resources in the device. However, the accuracy of the estimation of the number of packets and the volume (in bytes) of each flow is affected by sampling.

In this work we present different scenarios, configurations and scripts, that will be used for traffic generation and capture, together with the evaluation of high precision hardware (DAG) and an IP traffic sampling software (Netflow), as well as one system for obtaining and processing the information, with the objective to realize a study of Sampled Netflow's behavior and to evaluate the inaccuracy in the estimation of the volume and the quantity of packets of every IP flow. As far as we know, this is the first time that an experimental study of these characteristics has been carried out.

The results of the experimental testbed confirm the results of both the theory and simulations. However, due to the modest amount of flows and packets generated by limitations of equipment and time, the results are more precise for the higher sampling probabilities, and not so much for the lowest.

# ÍNDICE

|  |           |
|--|-----------|
| <b>INTRODUCCIÓN .....</b>  | <b>7</b>  |
| Objetivos.....   | 8         |
| <b>CAPÍTULO 1. NETFLOW .....</b>   | <b>9</b>  |
| 1.1. ¿Qué es Netflow?.....   | 9         |
| 1.2. Beneficios.....   | 9         |
| 1.3. Arquitectura .....  | 10        |
| 1.4. Definición de flujo y funcionamiento de Netflow.....  | 11        |
| 1.5. Versiones de Netflow .....  | 12        |
| 1.6. Netflow v5.....   | 13        |
| 1.7. Full Netflow.....   | 15        |
| 1.8. Sampled Netflow .....   | 16        |
| <b>CAPÍTULO 2. EQUIPOS Y SOFTWARE.....</b>   | <b>17</b> |
| 2.1. Cisco 3700.....   | 17        |
| 2.2. Switch Zyxel GS1100-16.....   | 18        |
| 2.3. Switch SMC Tigerswitch 8624T .....  | 18        |
| 2.4. DAG Card 4.3 GE.....  | 19        |
| 2.5. Servidores .....  | 20        |
| 2.6. Nfsen .....   | 21        |
| 2.7. Dagutils.....   | 22        |
| <b>CAPÍTULO 3. GENERADORES DE TRÁFICO .....</b>  | <b>23</b> |
| 3.1. Iperf/Jperf .....   | 23        |
| 3.1.1. Realización de pruebas.....   | 24        |
| 3.1.2. Resultados.....   | 25        |
| 3.2. MGEN .....  | 26        |
| 3.2.1. Resultados.....   | 26        |
| <b>CAPÍTULO 4. PRUEBAS DE MONITORIZACIÓN DE FLUJOS CON NETFLOW.....</b>                            | <b>28</b> |
| 4.1. Prueba 1: Envío de datos en dos sentidos .....  | 28        |
| 4.2. Prueba 2: Envío de dos flujos de datos en un mismo sentido.....                               | 32        |
| 4.3. Pruebas de consumo de CPU.....  | 34        |
| 4.3.1. Prueba de consumo de CPU sin Netflow.....   | 34        |
| 4.3.2. Prueba de consumo de CPU con Full Netflow.....  | 35        |
| <b>CAPÍTULO 5. GENERACIÓN CON MGEN, CAPTURA CON DAG Y MONITORIZACIÓN FULL/RANDOM NETFLOW .....</b> | <b>37</b> |
| 5.1. Realización de pruebas con un flujo y Full Netflow.....                                       | 37        |

|  |           |
|--|-----------|
| 5.2. Resultados.....   | 38        |
| 5.3. Realización de pruebas con un flujo y Random Netflow.....   | 39        |
| 5.4. Resultados.....   | 40        |
| 5.5. Comparativa final de muestreo .....   | 40        |
| 5.6. Generación de múltiples flujos con MGEN, captura con DAG y Full/Random Netflow ...                        | 41        |
| 5.6.1. Resultados pruebas iniciales.....   | 43        |
| 5.7. Realización de 10 pruebas.....  | 44        |
| 5.7.1. Resultados con Full Netflow.....  | 44        |
| 5.7.2. Resultados con Random Netflow .....   | 45        |
| <b>CAPÍTULO 6. GENERACIÓN Y CAPTURA CON DAG .....</b>  | <b>47</b> |
| 6.1. Creación de trazas .....  | 47        |
| 6.2. Resultados.....   | 49        |
| <b>CAPÍTULO 7. TCPREPLAY: EDICIÓN Y REPETICIÓN DE FICHEROS PCAP</b>  | <b>50</b> |
| 7.1. Introducción a tcpreplay .....  | 50        |
| 7.2. Creación, repetición y estudio del comportamiento de las trazas y Nfsen con $10^5$ y $10^6$ paquetes..... | 51        |
| 7.3. Resultados.....   | 51        |
| 7.3.1. Traza $10^5$ paquetes .....   | 51        |
| 7.3.2. Traza 1 millón .....  | 53        |
| <b>CAPÍTULO 8. EVALUACIÓN DE SAMPLED NETFLOW .....</b>   | <b>55</b> |
| 8.1. Introducción .....  | 55        |
| 8.2. Descripción de las pruebas .....  | 56        |
| 8.3. Automatización con cron .....   | 56        |
| 8.4. Procesado de archivos <i>nfcapd</i> .....   | 57        |
| 8.5. Cotejo de pruebas .....   | 58        |
| 8.6. Tratado de matrices y graficado.....  | 59        |
| 8.7. Diagramas de dispersión.....  | 61        |
| <b>CAPÍTULO 9. CONCLUSIONES Y LÍNEAS FUTURAS.....</b>  | <b>64</b> |
| <b>GLOSARIO .....</b>  | <b>66</b> |
| <b>BIBLIOGRAFÍA .....</b>  | <b>67</b> |
| <b>ANEXOS .....</b>  | <b>70</b> |
| I. Instalación de software .....   | 70        |
| II. Pruebas Iperf .....  | 75        |
| III. Pruebas MGEN .....  | 110       |
| IV. Configuración dispositivos.....  | 135       |
| V. Pruebas Random Netflow con MGEN, DAG y Nfsen.....   | 159       |
| VI. Pruebas iniciales de 6 flujos .....  | 203       |

|       |   |     |
|-------|---|-----|
| VII.  | 10 pruebas con Full/Random Netflow .....                        | 207 |
| VIII. | Creación de trazas básicas .....                                | 238 |
| IX.   | Tratado CESCA y comandos .....                                  | 248 |
| X.    | Traza 100000 y un millón de paquetes .....                      | 249 |
| XI.   | Sintaxis <i>cron</i> para las pruebas Full/Random Netflow ..... | 260 |
| XII.  | Procesado de archivo <i>nfcapd</i> .....                        | 263 |
| XIII. | Software java de comparación .....                              | 265 |
| XIV.  | Procesado de matrices en Matlab .....                           | 269 |
| XV.   | Diagramas de dispersión.....                                    | 274 |

## INTRODUCCIÓN

Las redes de ordenadores son fundamentales para la comunicación y la integración de servicios que se utilizan hoy en día. Dichos servicios están orientados al consumo y a la comodidad de nuestra sociedad: el hecho de poder comprar un billete de avión, consultar extractos bancarios, realizar videoconferencias o llamadas telefónicas comporta un consumo de tráfico transparente para el usuario pero no para el operador de red.

En las redes de los operadores de telecomunicaciones es muy importante saber hacia dónde va el tráfico, que volumen comporta y de qué tipo es, por lo que es muy necesario analizarlo y estudiar su comportamiento.

Analizar el tráfico en tiempo real es una tarea que puede proporcionar mucha información acerca de los dispositivos conectados a la red y abre la puerta a tareas tan importantes como optimizaciones, o predicción de la futura carga de la red. Ver el tráfico como una colección de flujos es útil para implementar las políticas de ingeniería de tráfico (por ejemplo, el equilibrio de carga), para detectar anomalías en la red (es decir, ataques DoS o propagación de gusanos) o de la contabilidad, con el que los ISP pueden cobrar a sus clientes o validar acuerdos de interconexión con otros operadores.

En la actualidad los routers pueden realizar medidas de tráfico a nivel de flujo gracias al software como Netflow de Cisco o productos similares como Jflow. Todos estos productos se basan en el análisis detallado de los campos de los paquetes IP que el router commuta y encamina. Si este análisis se realiza para cada paquete (Full Netflow), se obtiene la mayor precisión posible en el análisis, pero por el contrario el rendimiento del router puede degradarse debido al consumo de recursos (CPU y memoria del router), por lo que es necesario implementar medidas menos costosas como el análisis con Sampled Netflow, en el que el análisis no es para cada paquete sino para un subconjunto de ellos en base a una probabilidad establecida. Este método reduce sustancialmente el consumo de recursos en los routers, pero otorga menos precisión en los análisis.

Netflow analiza los flujos que atraviesan una o varias interfaces relacionando bytes y paquetes en registros que él mismo crea proporcionando estadísticas de uso. Estas estadísticas pueden ser recuperadas a través de una interfaz de línea de comandos (CLI) o dando instrucciones al router para exportarlas a una colectora para un posterior análisis y almacenamiento.

## Objetivos

El objetivo principal de este proyecto es conocer a fondo la validez de los reportes que ofrece Netflow con varios tráficos, generados mediante tarjetas Ethernet comerciales y hardware de altas prestaciones (Endace DAG). Cómo objetivos específicos podemos destacar:

- a) Una valoración de dos de los generadores de tráfico más populares, Iperf y MGEN, nos hará ver sus diferencias y escoger uno como instrumento auxiliar para evaluar el comportamiento de con Netflow.
- b) Analizar el tráfico con una tarjeta DAG de alta precisión nos permitirá validar los datos ofrecidos por Netflow y analizar las limitaciones tanto del hardware como del software de generación de tráfico.
- c) Conocer los procedimientos de obtención y extracción de datos. Esto implica un trabajo en segundo plano con diferentes aplicaciones como Wireshark, Matlab, etc. que también iremos viendo a lo largo del proyecto.
- d) Finalmente, el montaje y la configuración del escenario de prueba también forma parte de este trabajo.

El resto de memoria está organizado como sigue:

En el capítulo 1 se presenta Netflow y sus aplicaciones. En el capítulo 2 se describe la electrónica de red que usaremos y el software necesario para llevar a cabo las pruebas. En el capítulo 3 realizaremos un estudio sobre generadores de tráfico. En el capítulo 4 mostraremos el funcionamiento de Netflow y se analizará su consumo de CPU. En el capítulo 5 realizaremos una batería de pruebas con Netflow, así como la generación de tráfico con MGEN. En el capítulo 6 veremos la generación de trazas con las aplicaciones de la tarjeta DAG. En el capítulo 7 analizaremos la calidad de TCPreplay, un programa de replicación de trazas. En el capítulo 8 se evaluará el efecto del muestreo sobre Netflow. Finalmente en el noveno y último capítulo analizaremos los resultados de la prueba anterior.

## CAPÍTULO 1. NETFLOW

En este capítulo describiremos Netflow, comentando sus características y versiones, modo de funcionamiento y alternativas.

### 1.1. ¿Qué es Netflow?

NetFlow [1] es una técnica de captura de flujos desarrollada por Cisco Systems para recolectar información de tráfico IP. También se le da el mismo nombre a un protocolo para transferir la información de la captura. Los routers, dependiendo de si está soportado por su sistema operativo IOS, pueden activar este servicio y adquirir datos de los diferentes flujos que atraviesan sus interfaces de red.

Otras compañías del sector ofrecen productos similares para sus routers, como por ejemplo: Jflow [2] o cflowd [3] para Juniper Networks, NetStream [4] para 3Com, etc.

Los dispositivos Cisco que soporten este software permiten generar registros de los flujos activos e inactivos. Estos son exportados desde el router (transportados sobre User Datagram Protocol (UDP) o Sequence Control Transmission Protocol (SCTP)) y recogidos hacia una base de datos a través de un colector NetFlow. Más adelante, con otras herramientas, se pueden visualizar los datos en modo texto o en modo gráfico.

### 1.2. Beneficios

Netflow aporta los siguientes beneficios:

Monitorización de la red.

Nos permite mediante el análisis de flujo visualizar patrones en el tráfico asociados a cada uno de los switches y routers de nuestra red y de esta manera anticiparnos a problemas que puedan surgir.

Monitorización de aplicaciones.

A los gestores de red les permite obtener una descripción detallada de las aplicaciones que están usándola. Por lo tanto podemos planificar los nuevos servicios y asignar recursos a las aplicaciones en la red.

Monitorización de usuarios.

Obtener información de los recursos que están utilizando los usuarios de nuestra red.

Planificación de la red.

Nos permite capturar información durante un largo periodo de tiempo, para posteriormente ser analizada y de esta manera anticiparnos a los crecimientos de la red, ya sea de dispositivos de enrutamiento, puertos o ancho de banda.

Análisis de seguridad.

Netflow permite identificar y clasificar los ataques de denegación de servicio (DoS), virus y gusanos en tiempo real. Podemos detectar anomalías en el tráfico de la red y posteriormente estudiar su comportamiento para futuros ataques.

Contabilidad y facturación.

Proporciona estadísticas muy detalladas (direcciones IP, número de paquetes y bytes, hora, tipo de servicio, aplicación, etc.) que nos permiten facturar como proveedor de servicio en base al tiempo utilizado del recurso, ancho de banda, calidad de servicio, uso de la aplicación, etc.

Almacenamiento de datos Netflow.

Podemos almacenar información para su posterior recuperación y análisis, y de esta manera por ejemplo observar que aplicaciones y servicios están siendo utilizadas por usuarios internos o externos para posteriormente realizar mejoras sobre los mismos.

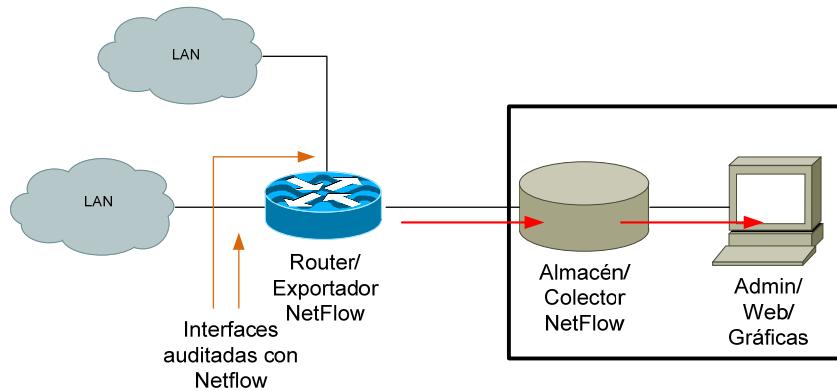
### 1.3. Arquitectura

Para hacer posible el análisis de flujos con Netflow se requieren tres componentes, tal y como se muestran en la Fig.1.1.

Exportador: Recopila información de los flujos, la analiza (flow cache) y la prepara para exportarla (flow records).

Colector: Escucha en un puerto UDP y guarda los flujos del colector.

Analizador: Filtra, muestra, analiza y/o grafica los datos obtenidos del colector.



**Fig. 1.1 Escenario básico de uso de Netflow**

#### 1.4. Definición de flujo y funcionamiento de Netflow

Un flujo se define como una secuencia unidireccional de paquetes IP que comparten ciertas características comunes, como por ejemplo:

- Dirección IP origen
- Dirección IP destino
- Puerto origen
- Puerto destino
- Tipo de protocolo a nivel 3
- Byte de ToS (Type of Service)
- Interfaz de entrada al router

NetFlow tiene una serie de componentes clave que intervienen en su funcionamiento, estos son:

##### Netflow cache

Es la memoria del colector Netflow en la que se almacenan los flujos de información. Esta memoria está limitada por la capacidad del propio dispositivo o también es configurable.

##### Netflow Flow Record

Es el registro de cada flujo detectado, en donde podemos encontrar toda la información del flujo, IP, protocolo, bytes, etc.

Flow export timers.

La exportación de los datos almacenados puede darse por varios motivos:

Inactividad de tráfico (15 segundos por defecto).

Expire el tiempo de actividad de los flujos (30min por defecto).

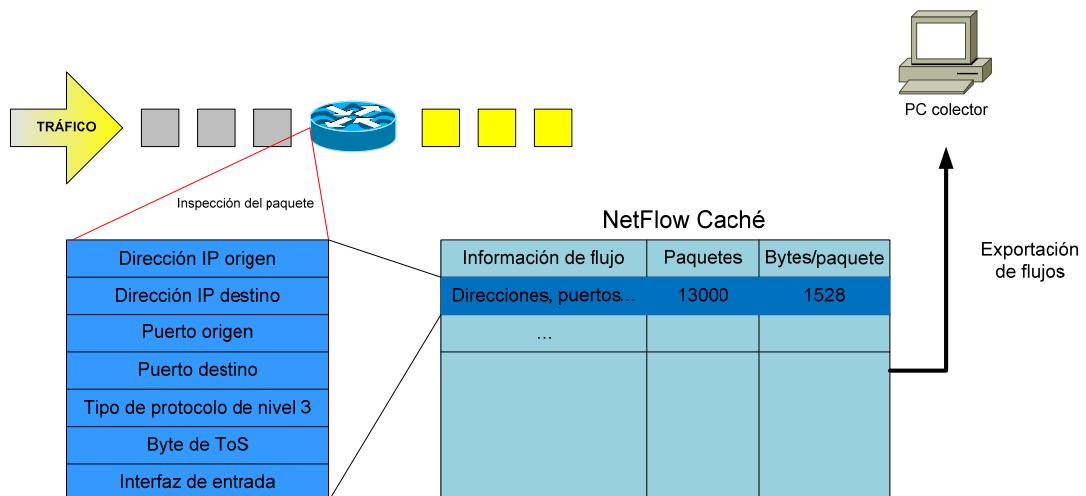
Que la cache se llene.

El protocolo de transporte nos indica que ha finalizado la conexión mediante flags TCP como FIN o RST.

Netflow export format.

Se exporta la información almacenada en la cache en un formato en concreto. Esta información se envía mediante paquetes UDP tamaño definido por el tamaño de la MTU de nuestra red (normalmente 1500 bytes para redes de área local Ethernet). Cada paquete puede contener entre 20 y 50 flujos, en función de la versión de Netflow utilizada.

En la Fig. 1.2 se muestra un ejemplo del comportamiento de los registros Netflow:



**Fig. 1.2 Funcionamiento registros Netflow**

## 1.5. Versiones de Netflow

Desde que se creó NetFlow en el año 1996 se han creado 9 versiones diferentes para intentar mejorar la anterior y adaptarse a las necesidades del momento.

**Tabla 1.1.** Versiones de Netflow

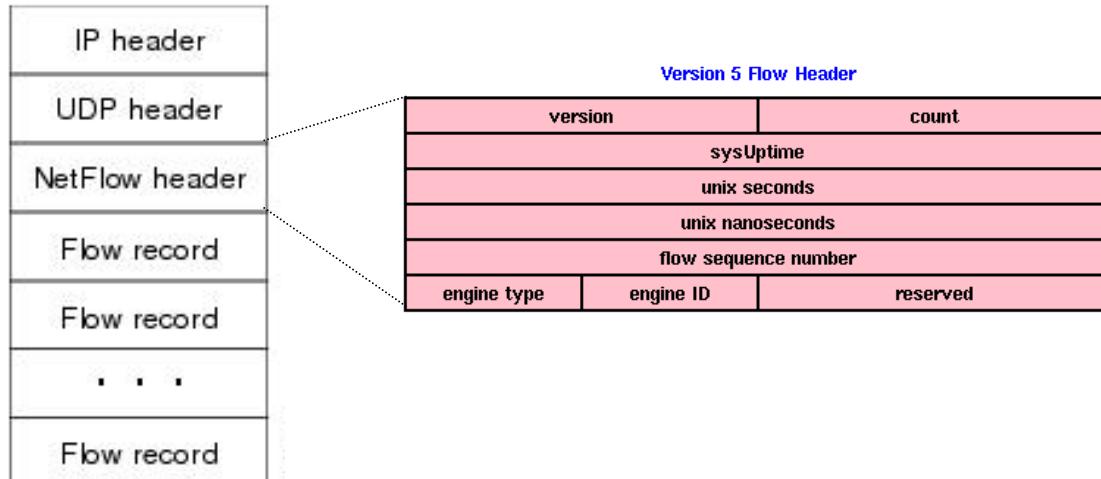
| Versión | Información   |
|---------|---|
| v1      | Primera implementación ya obsoleta y limitada a IPv4 sin máscaras     |
| v2      | Versión interna de Cisco nunca publicada                              |
| v3      | Versión interna de Cisco nunca publicada                              |
| v4      | Versión interna de Cisco nunca publicada                              |
| v5      | La versión más extendida y utilizada en numerosos routers, sólo IPv4  |
| v6      | Versión no admitida por Cisco   |
| v7      | Igual que v5 pero con un campo del router origen                      |
| v8      | Agregación de flujos sólo para información que ya está presente en v5 |
| v9      | Versión basada en plantillas. Orientado a flujos IPv6 y MPLS          |

## 1.6. Netflow v5

El datagrama de NetFlow a exportar se compone de una cabecera y una secuencia de registros de flujo.

- La cabecera contiene información como el número de secuencia, cantidad de registros y sysUpTime (tiempo desde el último reinicio), etc.
- El registro contiene información sobre el flujo como direcciones IP, puertos e información de encaminamiento.

La Fig.1.3 muestra el datagrama y la cabecera utilizada para el formato de exportación NetFlow versión 5.

**Fig. 1.3** Datagrama y cabecera Netflow v5 [13]

A continuación veremos la descripción de la cabecera (Tabla 1.2.) y el formato de los registros Netflow v5 (Tabla 1.3.) junto con dos casos concretos de capturas de una cabecera y un registro Netflow, en las figuras 1.4 y 1.5 respectivamente.

**Tabla 1.2.** Descripción de los campos de la cabecera

| Campo             | Descripción  |
|-------------------|--|
| version           | Versión de exportación de NetFlow  |
| count             | Número de flujos exportados en el paquete (1-30)   |
| sys_uptime        | Tiempo actual en ms desde que se reinició el router  |
| unix_secs         | Segundos transcurridos desde el 1 de enero de 1970   |
| unix_nsecs        | Nanosegundos residuales desde el 1 de enero de 1970  |
| flow_sequence     | Secuencia de contador de los flujos totales observados   |
| engine_type       | Tipo de motor de conmutación de flujos   |
| engine_id         | Número de ranura del motor de flujo de conmutación   |
| sampling_interval | Los dos primeros bits confirman si hay muestreo sampleado, los 14 bits restantes tienen el valor de la velocidad de muestreo |

```

Frame 49642: 450 bytes on wire (3600 bits), 450 bytes captured (3600 bits)
Ethernet II, Src: Cisco_5f:9c:e1 (00:07:b3:5f:9c:e1), Dst: Intel_cf:b7:8f (00:0e:0c:cf:b7:8f)
Internet Protocol Version 4, Src: 10.0.13.100 (10.0.13.100), Dst: 10.0.13.101 (10.0.13.101)
User Datagram Protocol, Src Port: 52780 (52780), Dst Port: 9910 (9910)
Cisco NetFlow/IPFIX
    Version: 5
    Count: 8
    SysUptime: 5481532
    Timestamp: May 10, 2012 17:49:21.927610110 Hora de verano romance
        CurrentSecs: 1336664961
        CurrentNSecs: 927610110
    FlowSequence: 264
    EngineType: RP (0)
    EngineId: 0
    00... .... .... .... = SamplingMode: No sampling mode configured (0)
    ..00 0000 0000 0000 = SampleRate: 0

```

**Fig. 1.4** Captura de la cabecera de exportación Netflow

**Tabla 1.3.** Formato de los registros Netflow v5

| Campo   | Descripción                                    |
|---------|--|
| srcaddr | Dirección IP origen                            |
| dstaddr | Dirección IP destino                           |
| nexthop | Dirección IP del siguiente salto (router)      |
| input   | Índice SNMP de la interfaz de entrada          |
| output  | Índice SNMP de la interfaz de salida           |
| dPkts   | Paquetes en el flujo                           |
| dOctets | Número total de bytes de capa 3 en el paquete  |
| first   | Tiempo de inicio de flujo                      |
| last    | Tiempo en que se ha recibido el último paquete |
| srcport | Puerto TCP/UDP origen                          |

|           |   |
|-----------|---|
| dstport   | Puerto TCP/UDP destino                                |
| pad1      | Bytes a cero sin uso                                  |
| tcp_flags | Indicadores TCP                                       |
| prot      | Tipo de protocolo IP (por ejemplo, TCP = 6; UDP = 17) |
| tos       | Tipo de servicio IP                                   |
| src_as    | Número de sistema autónomo o nodo origen              |
| dst_as    | Número de sistema autónomo o nodo destino             |
| src_mask  | Prefijo de máscara de subred origen                   |
| dst_mask  | Prefijo de máscara de subred destino                  |

```

    □ pdu 1/8
      SrcAddr: 10.0.13.101 (10.0.13.101)
      DstAddr: 84.88.39.235 (84.88.39.235)
      NextHop: 0.0.0.0 (0.0.0.0)
      InputInt: 3
      OutputInt: 0
      Packets: 4
      Octets: 260
      □ [Duration: 5.000000000 seconds]
        StartTime: 5460.900000000 seconds
        EndTime: 5465.900000000 seconds
        SrcPort: 56406
        DstPort: 53
        padding
        TCP Flags: 0x10
        Protocol: 17
        IP ToS: 0x00
        SrcAS: 0
        DstAS: 0
        SrcMask: 24 (prefix: 10.0.13.0/24)
        DstMask: 0 (prefix: 84.88.39.235/32)
        padding
  
```

**Fig. 1.5** Captura real de un registro Netflow

Como se puede observar en la captura de la Fig.1.5, el flujo con IP origen 10.0.13.101 y puerto 56406 tiene como destino la IP 84.88.39.235 con puerto 53, se trata de un flujo UDP (protocolo 17<sup>1</sup>) compuesto por 4 paquetes que transportan 260 bytes y su duración es de 5 segundos.

## 1.7. Full Netflow

Full Netflow es el sistema de captura IP más preciso, ya que detecta todos los paquetes que atraviesa una interfaz. Esto hace que sea un sistema muy fiable pero muy costoso a nivel de recursos y memoria. En condiciones de altas velocidades y/o gran cantidad de flujos el router puede colapsarse y no responder con normalidad.

---

<sup>1</sup> Se puede consultar la asignación de números a los protocolos de la familia TCP/IP en [http://en.wikipedia.org/wiki/List\\_of\\_IP\\_protocol\\_numbers](http://en.wikipedia.org/wiki/List_of_IP_protocol_numbers)

## 1.8. Sampled Netflow

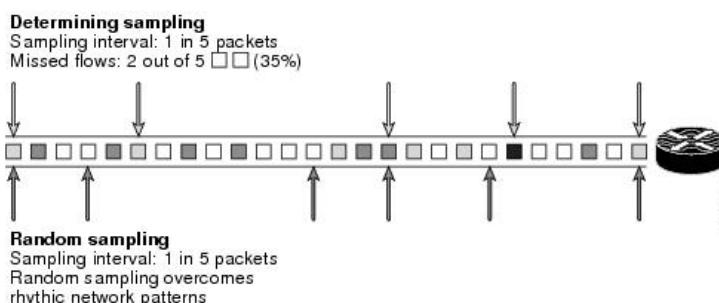
Sampled NetFlow, a diferencia del anterior, captura solo ciertos paquetes (por ejemplo, uno de cada 100, o de cada 1000). En una interfaz, Sampled NetFlow permite recoger estadísticas de NetFlow para un subconjunto de entradas de tráfico. La frecuencia de muestreo puede variar entre 1 y 1/65535, es decir, hasta 1 paquete muestreado por cada 65535 transmitidos. De esta manera se reduce significativamente la utilización de la CPU, ya que no se ha de analizar todo el tráfico que llega y se reduce el volumen de exportación, por lo que envía una aproximación del tráfico cursado.

El muestreo es útil para ciertas tareas de planificación de la capacidad de una red, ya que no es necesario analizar todos los flujos para comprender el comportamiento de la red, pero la calidad de la aproximación depende tanto de la frecuencia de muestreo escogida, como de la manera en que realiza la elección de los paquetes.

Hay 3 tipos de muestreo utilizado en las plataformas de Cisco: muestreo determinista, período de muestreo y toma de muestras al azar (Random Netflow). El muestreo determinista, cuando la frecuencia de muestreo es 1/N, selecciona todos los paquetes múltiples de N (0, N, 2N ...). El muestreo al azar consiste en seleccionar cada paquete con una probabilidad de 1/N, con lo cual, en media, se muestreará 1 paquete de cada N, pero en posiciones no determinadas. El método de tiempo de muestreo seleccionará un paquete para muestrear cada N milisegundos.

La IOS que tiene el router Cisco 3700 que utilizaremos en la pruebas, sólo soporta uno de los tres muestreos aleatorios: Random Netflow. Por tanto sólo podremos utilizar Full Netflow y Random Netflow.

En la Fig. 1.6. se muestra el funcionamiento de los modelos aleatorio y determinista de Sampled NetFlow. En la parte superior, el modelo determinista elige el primer paquete de cada 5 paquetes que pasan por la interfaz. En la parte inferior el modelo aleatorio elige en media un paquete de cada 5 que pasan, sin importar el orden.



**Fig. 1.6.** Modelo determinista (Sampled Netflow) y modelo aleatorio (Random Netflow) [1]

## CAPÍTULO 2. EQUIPOS Y SOFTWARE

En este capítulo hablaremos de la electrónica de red y el software utilizado para el desarrollo del proyecto.

### 2.1. Cisco 3700

El Cisco 3700 es un router de gama media-alta que soporta múltiples servicios, protocolos y procesos. En nuestro caso corre una versión de IOS 12.3(11)T3, la cual soporta Full Netflow y Random Netflow.

El router incluye, de serie, dos interfaces FastEthernet a 100Mbps (FastEthernet0/0 y FastEthernet 0/1), un puerto de consola y un puerto auxiliar. Si es necesario, tiene bahías para insertar diferentes tipos de módulos: por defecto viene con tres bahías para WIC's y tres más para agregar módulos más grandes, tipo switch Ethernet, ATM o interfaces serie para Frame Relay.



**Fig. 2.1** Parte trasera router Cisco 3700

**Tabla 2.1.** Especificaciones técnicas Cisco 3700

|              |   |
|--------------|---|
| Procesador   | 240-MHz PMC-Sierra RM7061A RISC processor |
| SDRAM        | 128-256MB                                 |
| NVRAM        | 56KB                                      |
| CompactFlash | 32, 64 o 128MB                            |
| BootROM      | 512KB                                     |
| Tamaño       | 2U (dos unidades de rack de 21")          |

## 2.2. Switch Zyxel GS1100-16

El Zyxel GS1100 es un conmutador Ethernet no gestionable. Soporta múltiples estándares IEEE, tales como: 802.3, 802.3u, 802.3ab, 802.3az, 802.3x, 802.3z y 802.1p. Tiene 16 puertos 1000BASE-T lo que nos permite trabajar a altas tasas de transmisión. Su tabla de direcciones MAC puede llegar hasta 8000 registros.

Por otro lado, tiene características “green” a favor del medio ambiente como la detección de cualquier enlace inactivo para el ajuste dinámico de potencia de salida, o el uso de energía de acuerdo con la longitud de los cables Ethernet conectados.



**Fig. 2.2** Parte frontal del Zyxel GS1100-16

Este dispositivo se utilizará para realizar las pruebas del [capítulo 3](#), en el que tomaremos contacto con los diferentes generadores de flujo.

## 2.3. Switch SMC Tigerswitch 8624T

El switch<sup>2</sup> SMC Tigerswitch 8624T es un dispositivo de muy altas prestaciones, es gestionable mediante puerto serie (RS-232) y mediante una dirección IP de gestión. Soporta múltiples estándares IEEE, tales como: 802.3, 802.3u, 802.3ab, 802.3az, 802.3x, 802.3z, 802.1p, 802.1Q, 802.1d, etc. Tiene 24 puertos 1000BASE-T lo que nos permite trabajar a altas tasas de transmisión y 4 puertos SFP a 1000BASE-SX/LX para conectar enlaces de subida (uplink) con fibra.

Una de las características más importantes de este switch es la capacidad de *Port Mirroring* que consiste en replicar el/los sentidos de un puerto a otro. Así podremos capturar tráfico de diferentes fuentes, redirigirlo a un puerto de salida concreto y analizarlo externamente. En capítulos posteriores se verá su funcionamiento y su configuración.

---

<sup>2</sup> Agradecemos a la Fundació i2Cat la cesión del switch para la realización de este proyecto.



**Fig. 2.3 Parte frontal del SMC Tigerswitch 8624T**

Este switch lo utilizaremos para realizar el escenario completo y realizar todas las pruebas del [capítulo 5](#) en adelante.

## 2.4. DAG Card 4.3 GE

Las tarjetas pasivas de monitorización Endace DAG [5] son una de las herramientas físicas de análisis de red. A diferencia de las tarjetas NIC habituales que fácilmente que pueden verse sobrecargadas con el tráfico de una red (incluso si es pequeña), las tarjetas DAG permiten una monitorización eficiente del tráfico de red. Para ello, utilizan la memoria del PC evitando interrupciones y liberando así la carga a la CPU. Además debido a su diseño ofrecen una gran precisión en el marcado temporal (los timestamps asociados a la llegada de cada paquete).

Por otro lado, la capacidad de la tarjeta para capturar a tasa máxima va ligada a las capacidades del PC donde esta está instalada. Es muy importante la velocidad del disco duro, así como tener en cuenta que al estar conectadas al bus PCI-X solo pueden trabajar hasta 4 Gbps. Además la tarjeta dispone de un buffer FIFO donde almacenar unos cuantos milisegundos de tráfico que le permiten soportar ráfagas de paquetes pequeños a tasa máxima.

El reloj interno utilizado por las tarjetas para realizar las capturas ofrece la posibilidad de almacenar el timestamp con una precisión superior que las herramientas que utilicen la librería *libpcap*. Las DAG nos ofrecen una resolución de hasta 15 nanosegundos frente a 1 milisecondo ofrecido por *libpcap*. A diferencia de *libpcap* estas tarjetas realizan el timestamp al principio del paquete y no al final de la recepción del mismo (eliminando así el tiempo de procesado del paquete).

Por otro lado, a pesar de estar orientadas a la monitorización y como se verá más adelante en el [capítulo 6](#), pueden llegar a generar tráfico.

Como se puede ver en la Fig. 2.4., la tarjeta dispone de dos puertos ópticos 1000BASE-SX para realizar monitorización con fibra. En nuestro caso hemos utilizado transceivers SFP que nos permitirán convertirlos a 1000BASE-T sobre par trenzado con conectores RJ-45.

La tarjeta DAG ha sido estudiada con anterioridad en dos TFC [6] [7], donde se describe su configuración, y se documenta su uso a partir de diversos experimentos.



**Fig. 2.4 Tarjeta Endace DAG 4.3GE**

## 2.5. Servidores

Para completar el escenario es necesario de disponer de ordenadores con rapidez y cierta potencia de procesado. Los dos PC del laboratorio (servgenmonI y servgenmonII) tienen las siguientes características:

- Placa base Supermicro X6DHE-XG2: Chipset Intel E7520
- Microprocesador Xeon 3.0 GHz Nocona
- Memoria Ram 2x 1024 Mbytes DDR2 ECC
- Tarjeta de red Intel PRO/1000Mbps
- Tarjeta gráfica Nvidia PCI Express de 258 Mbytes
- Endace Dag 4.3GE PCI-X 133MHz

Los dos PC tienen instalado SUSE Linux 11.2 de 64 bits aunque con diferente kernel, servgenmonI utiliza kernel con versión 2.4.27-speakup y servgenmonII utiliza la 2.6.32. Esto es debido a que el software de la tarjeta DAG sólo soporta la versión 2.4.27 o inferior.

Para el uso del colector de datos Netflow (Nfsen) se utilizará un tercer equipo (RadPC) menos potente, con las siguientes características:

Microprocesador Intel Pentium IV a 2.4 GHz

Memoria RAM de 256Mbytes

Tarjeta gráfica nVidia GeForce MX440

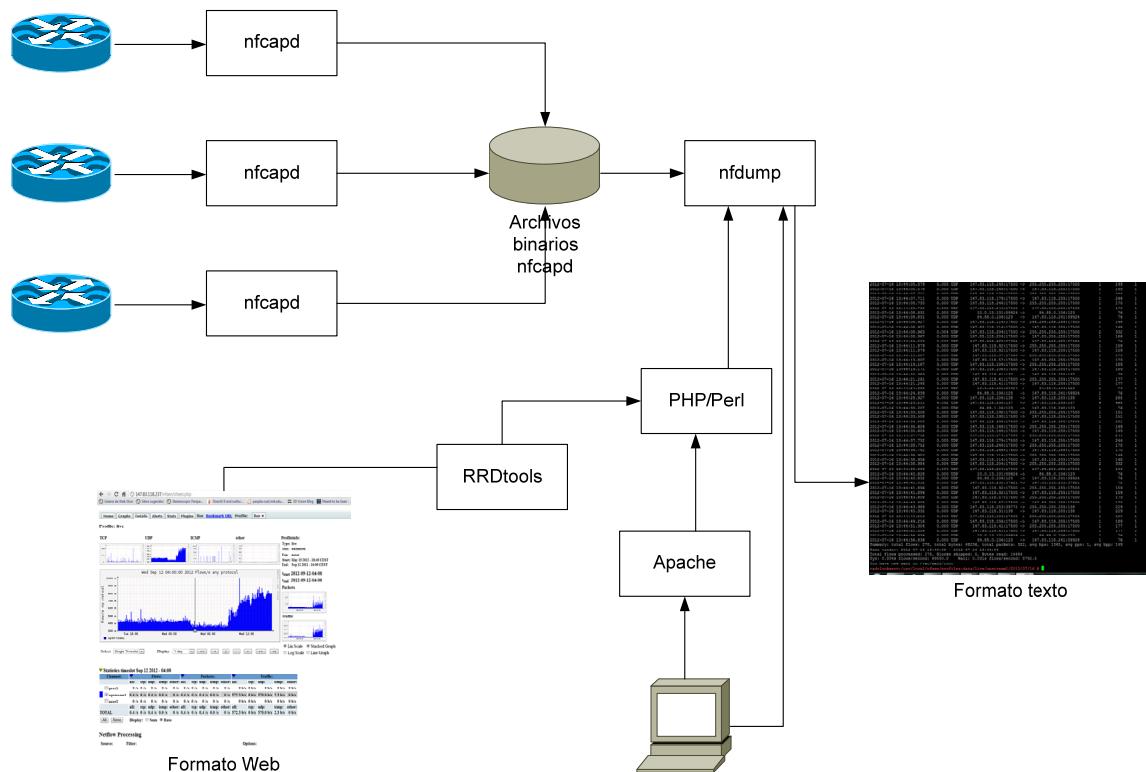
Tarjeta de red Realtek 8169 1000 Mbps

SUSE Linux 2.6.32

## 2.6. Nfsen

Nfsen es un front end gráfico que trabaja conjuntamente con la suite nfdump (es decir, nfcapd, nfprofile,etc.) y que nos permite recolectar, almacenar, procesar y visualizar los registros que envían los routers que usan Netflow.

El proceso de tratamiento de registros y archivos se ilustra en la Fig. 2.5:



**Fig. 2.5 Funcionamiento Nfsen**

En la Fig. 2.5 se muestra como los routers Netflow envían sus registros al colector Nfsen, el cual tiene un proceso *nfcapd* abierto para cada router. Por defecto y en intervalos de 5 minutos estos procesos vuelcan sus contenidos a un archivo binario con formato:

*nfcapd.AAAAMMDDHHMM*

donde *AAAA* es el año, *MM* el mes, *DD* el día, *HH* la hora y *MM* los minutos.

A partir de aquí *nfdump* los trata de la manera que quiera el usuario, en modo texto o en modo gráfico vía web.

La instalación de Nfsen se describe en el [Anexo I](#).

## 2.7. Dagutils

*Dagutils* es un conjunto de aplicaciones para la instalación, configuración y funcionamiento de la DAG. La instalación y la configuración la hemos realizado en función de los TFC de Iván Pérez [6] y Jordi Soler [7]. Se podrán encontrar más detalles en el [Anexo I](#).

La suite incorpora muchas herramientas, pero las más importantes para el funcionamiento y el uso de la DAG son:

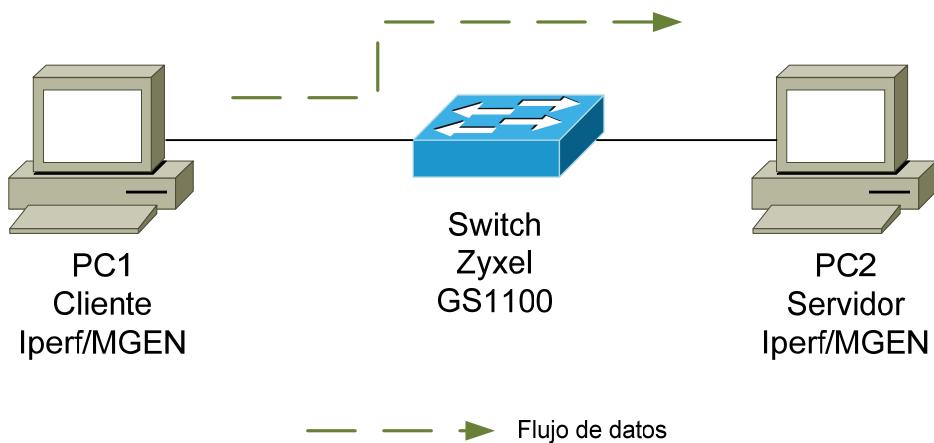
- *daggen*: Este software nos permitirá generar archivos ERF o PCAP en base a un script programado con anterioridad.
- *dagsnap*: Con esta utilidad podremos escuchar el canal y recibir todos los paquetes o tramas.
- *dagconvert*: Esta utilidad nos permite realizar conversiones de archivos PCAP o ERF.
- *dagclock*: Esta aplicación permite que la DAG se pueda sincronizar con el reloj interno del PC o con servidores externos NTP.

## CAPÍTULO 3. GENERADORES DE TRÁFICO

En cualquier escenario en redes IP existen flujos de tráfico, ya sean internos sobre la propia red o externos hacia el resto de Internet. Estos flujos generan varios elementos a analizar para obtener un buen rendimiento de la red. Es muy importante tener en cuenta que los flujos ocupan capacidad, producen retardos a otros flujos, así como conocer la distribución de su espaciado, ya que de eso depende el comportamiento de las colas de los routers (una ráfaga de paquetes, por ejemplo, puede provocar pérdidas por desbordamiento de las colas).

Los generadores de flujo nos proporcionan la capacidad de simular diferentes tipos de tráfico con multitud de características para reproducir escenarios realistas en un laboratorio. Por ello vamos a dedicar algunas páginas a probar el comportamiento de los dos generadores de tráfico más populares.

El escenario que vamos a emplear para realizar las pruebas son ordenadores diferentes a los comentados en el apartado [2.5](#).



**Fig. 3.1** Escenario de pruebas Iperf

### 3.1. Iperf/Jperf

La herramienta Iperf [8] es una de las más conocidas en la red por su sencillez y su frontend gráfico, y está disponible tanto para sistemas operativos Windows como Linux.

Iperf está diseñado para medir el rendimiento del ancho de banda vía TCP y UDP. Con iperf podemos saber cuántos paquetes se están perdiendo, cuando se producen cortes en los diferentes escenarios estudiados, jitter, etc.

Para cada simulación con Iperf se configura un nodo como servidor y otro como cliente. De esta forma todos los nodos se estarán enviando mensajes de información entre ellos. Se pueden hacer diversas mediciones con Iperf. Todo depende de los parámetros que se utilicen, como por ejemplo, la tasa que se desea alcanzar, la cantidad de payload,etc.

La captura de datos se realizará mediante la herramienta tcpdump, un software libre que permite capturar todos los paquetes de la interfaz (gracias a la librería pcap) y que no exige tantos recursos de CPU como Wireshark.

### 3.1.1. Realización de pruebas

Se van a realizar pruebas de generación de tráfico, variando la tasa o velocidad y el tamaño de los paquetes. Un tercer parámetro implícito de mayor importancia para el rendimiento del generador de flujo, es la cadencia de generación. Estos tres parámetros se relacionan con la siguiente ecuación:

$$\# \frac{\text{paquetes}}{\text{seg}} = \frac{BW \text{ (bps)}}{\text{tamaño paquete (bits)}} \quad (3.1)$$

El periodo entre paquetes (en segundos) es la inversa (Delta time) de la cadencia.

En nuestras pruebas hemos usado redes Gigabit Ethernet, con una MTU de 1500 bytes. Se ha creado una batería de pruebas con cuatro tamaños de paquetes diferentes: tamaño mínimo, es decir 64 bytes, tamaño medio 300 bytes, tamaño medio-alto 1000 bytes y tamaño alto 1400 bytes. No hemos sobrepasado la MTU; de este modo se evita la fragmentación de paquetes. Todos estos payloads son útiles, es decir, sin cabeceras, por lo que es necesario sumar 42 bytes de las diferentes cabeceras: 20 bytes de IP, 14 de Ethernet y 8 de UDP.

Las pruebas a realizar se basan en utilizar los diferentes tamaños de paquete y diferentes velocidades, resultando la siguiente batería de pruebas:

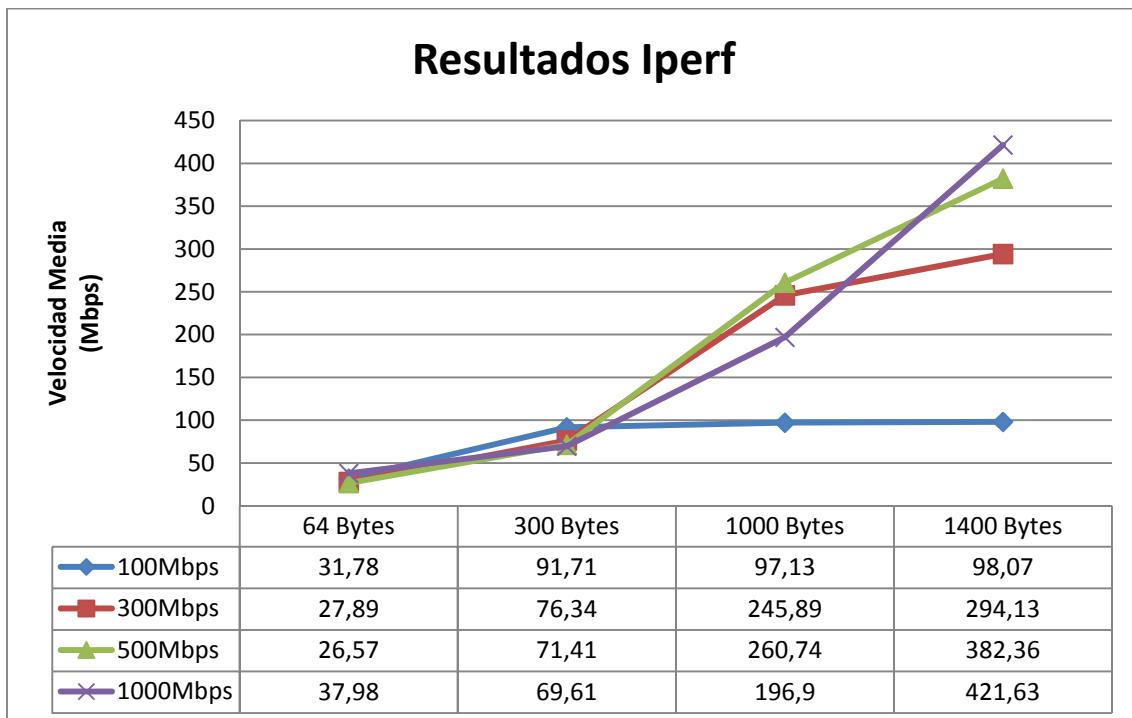
- ✓ Prueba a 100 Mbps con 64, 300, 1000 y 1400 bytes
- ✓ Prueba a 300 Mbps y 64, 300, 1000 y 1400 bytes
- ✓ Prueba a 500 Mbps y 64, 300, 1000 y 1400 bytes
- ✓ Prueba a 1000 Mbps y 64, 300, 1000 y 1400 bytes

Con este estudio podremos observar el comportamiento de Iperf con diferentes velocidades y tamaños de paquete. En el [Anexo II](#) podemos encontrar las diferentes pruebas, los comandos y las gráficas resultantes.

### 3.1.2. Resultados

La Fig. 3.2 muestra la velocidad alcanzada para cada tipo de prueba. Como se puede comprobar, Iperf trabaja bien a bajas velocidades y a mayor payload ya que se generan menos paquetes por segundo. Los relojes internos y la inestabilidad del programa también provocan que no se emitan correctamente las tramas a alta velocidad.

No hemos realizado pruebas inferiores a 100 Mbps ya que el objetivo es congestionar el canal con altas velocidades.



**Fig. 3.2** Velocidad alcanzada para cada prueba en Mbits/s

Con paquetes pequeños de 64 bytes se obtienen los peores resultados, ya que la cadencia de paquetes es muy alta y los PC no pueden alcanzarla. En cambio, con paquetes de 1000 y 1400 bytes los resultados son óptimos para 100Mbits/s, alcanzando la tasa deseada.

Por otro lado, con tasas elevadas de 500 Mbits/s y 1000Mbits/s los PC tampoco generan correctamente una cadencia adecuada para la velocidad que se desea alcanzar.

Los mejores resultados se obtienen con paquetes de 1400 bytes para las dos velocidades más bajas testeadas, es decir, 100 y 300 Mbits/s.

Cabe destacar que en algunas pruebas, no solo no alcanzamos la velocidad deseada sino que hay discrepancias entre la velocidad obtenida con Matlab a

partir de la medias de las capturas realizadas y el report que ofrece Iperf en la finalización de las transmisiones.

### 3.2. MGEN

MGEN [9] es una herramienta de generación de tráfico y flujos de datos en redes informáticas, tanto para sistemas operativos Windows como Linux.

Como en Iperf, cada simulación se configura un nodo como servidor y otro como cliente, aunque este software permite la conectividad multicast entre varios ordenadores.

Esta aplicación requiere que las simulaciones se configuren en un script, tanto para el servidor como para el cliente, lo que conlleva una flexibilidad más alta que Iperf en cuanto a cantidad y tiempo de generación de flujos. De esta manera todas los parámetros de puertos, direcciones IP, velocidad, etc. van incluidas con un orden y sintaxis específicos.

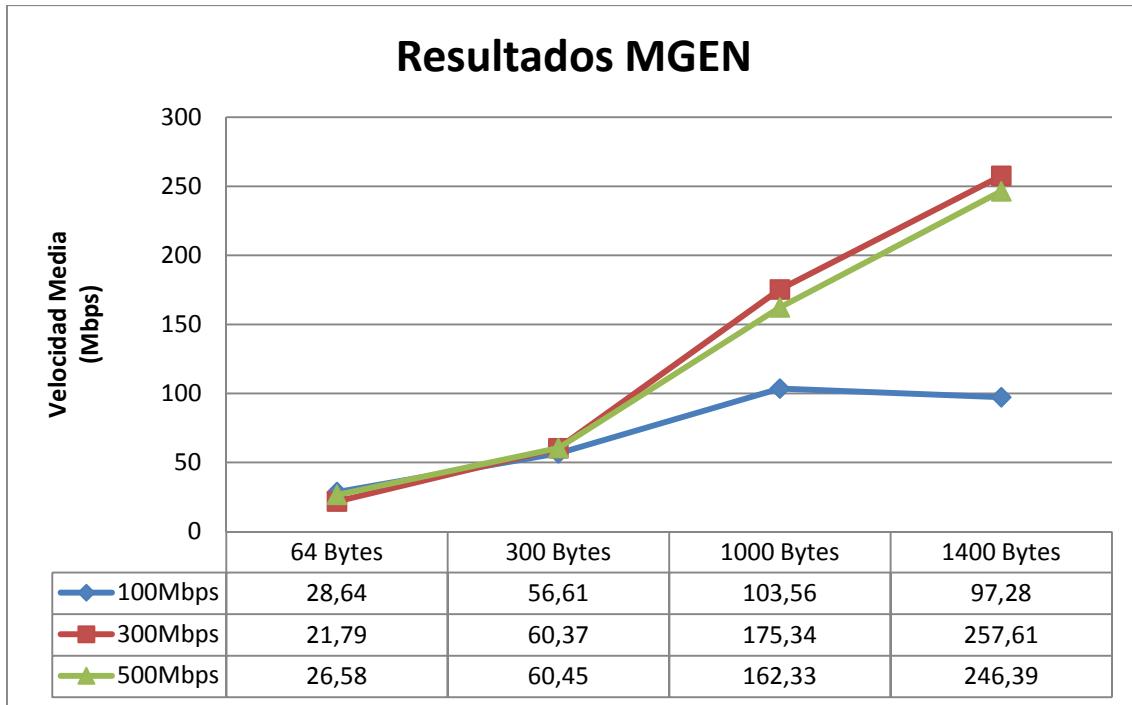
El escenario que vamos a emplear para realizar las pruebas es el mismo que el de la Fig. 3.1. Las pruebas a realizar son las mismas pruebas que se realizaron con Iperf:

- ✓ Prueba a 100 Mbits/s con 64, 300, 1000 y 1400 bytes
- ✓ Prueba a 300 Mbits/s y 64, 300, 1000 y 1400 bytes
- ✓ Prueba a 500 Mbits/s y 64, 300, 1000 y 1400 bytes

En el [Anexo III](#) podemos encontrar las diferentes pruebas, los comandos y las gráficas resultantes.

#### 3.2.1. Resultados

La Fig. 3.3 muestra la velocidad alcanzada para cada tipo de prueba. Al igual que sucedía con Iperf, MGEN trabaja bien a bajas velocidades y cuando el payload es mayor. Cuanto mayor es la velocidad peor es la periodicidad conseguida con MGEN encontrándonos con medias muy desviadas de las teóricas.



**Fig. 3.3 Resultados MGEN**

Con paquetes de 64 bytes, al igual que ofrecía Iperf, hemos obtenido los peores resultados con todas las velocidades. Las velocidades medias alcanzadas por MGEN, tampoco alcanzan las velocidades deseadas excepto para los 100Mbps, en los que cumple la periodicidad en los últimos dos tamaños de paquete (1000 y 1400 bytes), obteniendo así un buenos resultados.

No hemos realizado pruebas inferiores a 100 Mbits/s ya que el objetivo es congestionar el canal con altas velocidades. Por otro lado, tampoco las hemos realizado a 1000Mbps ya que generaban muchos paquetes y no era posible capturarlos todos. Además, al tener interfaces FastEthernet en el router nos hemos decantado por descartarlas. En siguientes baterías de pruebas se configurarán diferentes flujos y velocidades.

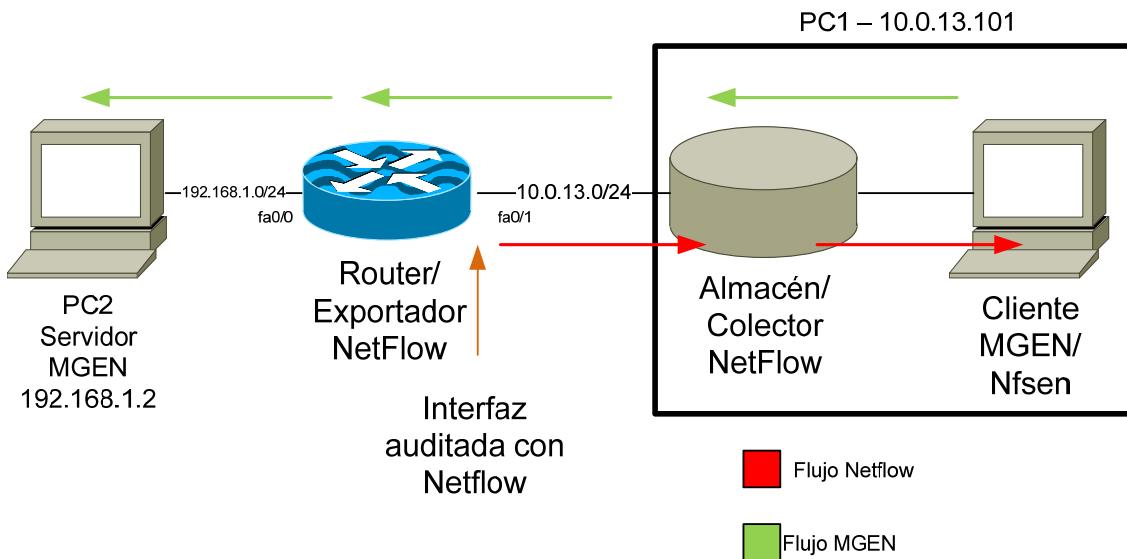
## CAPÍTULO 4. PRUEBAS DE MONITORIZACIÓN DE FLUJOS CON NETFLOW

La siguiente batería de pruebas se basa en observar, capturar y analizar los diferentes paquetes que envía Netflow desde el router hacia el colector Nfsen. Se analizarán dos pruebas, una en que se emitirán flujos en los dos sentidos y otra en que emitiremos varios flujos en el mismo sentido para comprobar el comportamiento de Netflow. También se hacen pruebas para comprobar el consumo de CPU.

### 4.1. Prueba 1: Envío de datos en dos sentidos

El escenario a utilizar son dos PC y un router Cisco 3700. Uno de los PC hará de colector Nfsen y cliente MGEN. Estas primeras pruebas se realizaron con ordenadores diferentes a los que se han comentado en el apartado [2.5](#)). El PC2 (un portátil) simplemente recibirá los datos como servidor.

En esta prueba vamos a emitir un flujo UDP del PC1 al PC2 y capturar con Wireshark el reporte que el router con Netflow envía al PC1. Por otro lado, comprobaremos que emitiendo en sentido contrario (del PC2 al PC1), Netflow no detecta el flujo.



**Fig. 4.1** Escenario primera prueba Netflow

En primera instancia con versión 5 de Netflow configurada, sólo es posible tener una interfaz escuchando paquetes, en este caso la FastEthernet0/1, con lo que si enviamos emitimos un flujo UDP del PC1 al PC2, el router colectará y enviará el report por defecto a los 15 segundos de la inactividad de dicho flujo.

A continuación se detalla la configuración del router Cisco 3700:

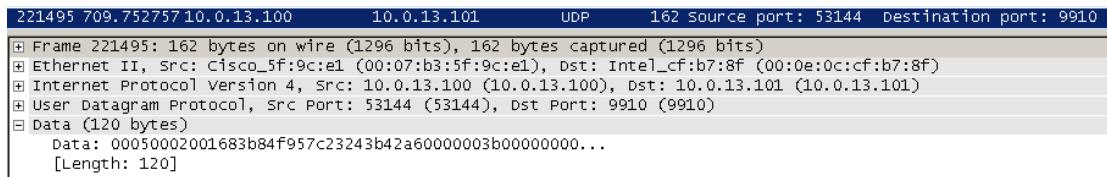
```
Habilitar el registro NetFlow en esta interfaz y configurar IP
ip address x.x.x.x y.y.y.y
ip route cache flow

Cisco3700(config)#int fa0/1
Cisco3700(config-if)#ip address 10.0.13.100 255.255.255.0
Cisco3700(config-if)#ip route-cache flow

Configuración Netflow en el router:
Configuración de la interfaz origen de NetFlow
ip flow-export source <interfaz>
Configuración de la versión de NetFlow
ip flow-export version <version>
Configuración de la IP/puerto destino del colector NetFlow a donde
van a ir los paquetes UDP/SCTP
ip flow-export destination <dirección IP> <puerto>

Cisco3700(config)#ip flow-export source FastEthernet0/1
Cisco3700(config)#ip flow-export version 5
Cisco3700(config)#ip flow-export destination 10.0.13.101 9910
```

Los paquetes Netflow se envían sobre el protocolo UDP, con lo que si se utiliza Wireshark, éste solo captura un datagrama con datos hexadecimales sin estructura, tal como se muestra en la Fig. 4.2.



**Fig. 4.2** Captura datagrama UDP Netflow sin decodificar

Pero es posible decodificar el mensaje para obtener todo el registro Netflow detallado, para eso es necesario aplicar un filtro (cflow) que incorpora el propio Wireshark. Con clic de botón derecho del ratón sobre el datagrama, *decode as* y búsqueda del filtro cflow, se interpretan todos los mensajes de este tipo, tal como muestra la Fig. 4.3.

```

221495 709.752757 10.0.13.100      10.0.13.101      CFLOW    162 total: 2 (v5) flows
  SrcMask: 24 (prefix: 10.0.13.0/24)
  DstMask: 24 (prefix: 10.0.13.0/24)
  padding
  pdu 2/2
    SrcAddr: 10.0.13.101 (10.0.13.101)
    DstAddr: 192.168.0.2 (192.168.0.2)
    NextHop: 192.168.0.2 (192.168.0.2)
    InputInt: 3
    OutputInt: 2
    Packets: 124957
    Octets: 40985896
  [duration: 10.000000000 seconds]
    SrcPort: 54489
    DstPort: 5001
    padding
    TCP Flags: 0x10
    Protocol: 17
    IP TOS: 0x00
    SrcAS: 0
    DstAS: 0
    SrcMask: 24 (prefix: 10.0.13.0/24)
    DstMask: 24 (prefix: 192.168.0.0/24)
    padding

```

**Fig. 4.3 Captura datagrama UDP Netflow decodificada**

Una vez decodificado, ya podemos interpretar correctamente el contenido de los registros Netflow, con los flujos detectados y sus parámetros internos.

En la Fig. 4.4, correspondiente a la interfaz del router, se aprecia como sólo registra un flujo desde 10.0.13.101 (PC1) con destino 192.168.0.2 (PC2)

```

Cisco3700#sh ip cache flow
IP packet size distribution (418230 total packets):
  1-32   64   96   128   160   192   224   256   288   320   352   384   416   448   480
  .000 .006 .001 .001 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000
  512   544   576   1024   1536   2048   2560   3072   3584   4096   4608
  .000 .000 .000 .000 .990 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
  1 active, 4095 inactive, 209 added
  4785 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 17416 bytes
  1 active, 1023 inactive, 209 added, 209 added to flow
  0 alloc failures, 0 force free
  1 chunk, 1 chunk added
  last clearing of statistics never
Protocol      Total    Flows   /Sec   Packets  Bytes   /Pkt   /Sec   Active(Sec)  Idle(Sec)
-----      Flows   /Sec   /Flow   /Pkt   /Sec   /Flow   /Flow
TCP-Telnet      84     0.0      34     40     0.6     11.0    15.1
UDP-other       91     0.0     4254    1496    86.2      5.8    15.3
ICMP          33     0.0      17     100     0.1      3.0    15.1
Total:        208     0.0     1878    1483    86.9      7.5    15.2

SrcIf      SrcIPaddress      DstIf      DstIPaddress      Pr SrcP DstP   Pkts
Fa0/1      10.0.13.101      Fa0/0      192.168.0.2      11 AF50 1389    27K
Cisco3700#

```

**Fig. 4.4 Captura Telnet flujo PC1->PC2**

En cambio, si se realiza la prueba inversa, es decir, si envía del PC2 al PC1, el router no colectará los datos, ya que su origen no es la interfaz configurada:

```
Cisco3700#sh ip cache flow
IP packet size distribution (467158 total packets):
 1-32   64   96   128   160   192   224   256   288   320   352   384   416   448   480
 .000 .006 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

 512   544   576   1024  1536  2048  2560  3072  3584  4096  4608
 .000 .000 .000 .000 .991 .000 .000 .000 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
 0 active, 4096 inactive, 209 added
 4820 ager polls, 0 flow alloc failures
 Active flows timeout in 30 minutes
 Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 17416 bytes
 0 active, 1024 inactive, 209 added, 209 added to flow
 0 alloc failures, 0 force free
 1 chunk, 1 chunk added
 last clearing of statistics never
Protocol      Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
-----        Flows     /Sec    /Flow  /Pkt   /Sec    /Flow     /Flow
TCP-Telnet      84      0.0     34     40     0.6    11.0    15.1
UDP-other       92      0.0    5039   1496    97.2    6.0    15.3
ICMP           33      0.0     17    100     0.1    3.0    15.1
Total:          209     0.0    2235   1486    97.9    7.6    15.2
SrcIf      SrcIPAddress      DstIf      DstIPAddress      Pr SrcP DstP   Pkts
Cisco3700#
```

**Fig. 4.5 Captura Telnet flujo PC2->PC1**

Si por otro lado también se requiere registrar la actividad de la otra interfaz (fa0/0), se debe configurar con la línea de habilitación de cache.

```
Habilitar el registro NetFlow en esta interfaz y configurar IP
ip address x.x.x.x y.y.y.y
ip route-cache flow

Cisco3700(config)#int fa0/0
Cisco3700(config)#ip address 192.168.0.1 255.255.255.0
Cisco3700(config-if)#ip route-cache flow
```

En la captura mostrada en la Fig. 4.6 se observa como en este caso sí se registra el flujo originado desde el PC2 hacia el router:

```
Cisco3700#sh ip cache flow
IP packet size distribution (491731 total packets):
 1-32   64   96   128   160   192   224   256   288   320   352   384   416   448   480
 .000 .006 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

 512   544   576   1024  1536  2048  2560  3072  3584  4096  4608
 .000 .000 .000 .000 .991 .000 .000 .000 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
 1 active, 4095 inactive, 214 added
 4925 ager polls, 0 flow alloc failures
 Active flows timeout in 30 minutes
 Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 17416 bytes
 1 active, 1023 inactive, 214 added, 214 added to flow
 0 alloc failures, 0 force free
 1 chunk, 1 chunk added
 last clearing of statistics never
Protocol      Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
-----        Flows     /Sec    /Flow  /Pkt   /Sec    /Flow     /Flow
TCP-Telnet      85      0.0     34     40     0.5    10.9    14.9
UDP-other       95      0.0    5138   1496    98.6    5.9    15.3
ICMP           33      0.0     17    100     0.1    3.0    15.1
Total:          213     0.0    2308   1486    99.3    7.5    15.1
SrcIf      SrcIPAddress      DstIf      DstIPAddress      Pr SrcP DstP   Pkts
Fa0/0      192.168.0.2      Local      192.168.0.1      06 1C47 0017    85
Cisco3700#
```

**Fig. 4.6 Captura Telnet flujo PC2->PC1**

## 4.2. Prueba 2: Envío de dos flujos de datos en un mismo sentido

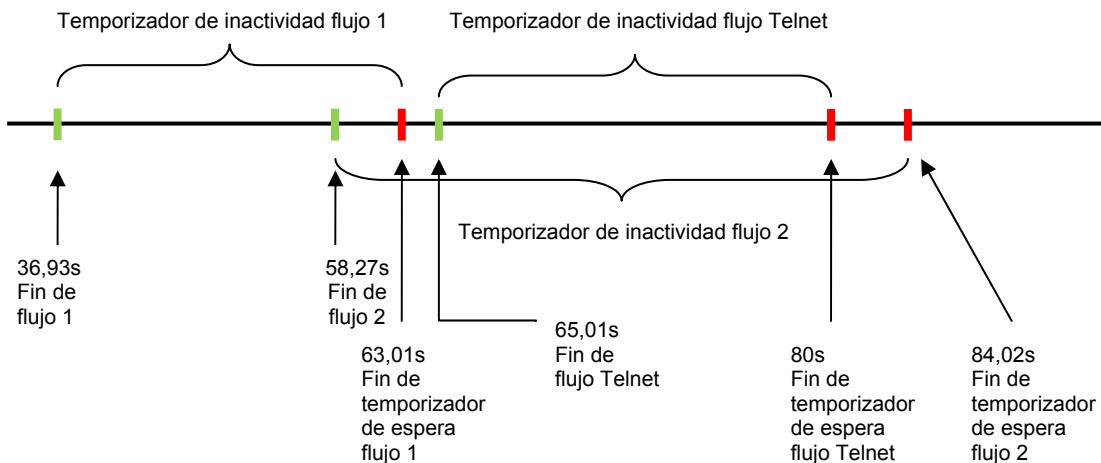
Es importante saber si Netflow genera un report para cada flujo o uno para varios juntos. Para eso vamos a describir una prueba simple que consiste en generar dos flujos iperf en el mismo sentido, detener uno y segundos más tarde detener el otro.

Se genera un primer flujo con duración de 10 segundos y que finaliza en el segundo 36,93 de la captura, por otro lado se inicia otro flujo en el instante 28,27 segundos con duración de 30 segundos y finalización en 58,27 segundos. Adicionalmente se generará un flujo Telnet conforme a los comandos enviados al router para la revisión de los flujos.

A partir del instante de finalización de cada flujo se pone en marcha el temporizador de inactividad, que por defecto son 15 segundos en todos los routers (aunque cabe destacar que en esta prueba usamos un timeout de 30 segundos).

Netflow es capaz de agregar diferentes flujos detectados en el mismo registro, pero sólo en el caso de que el temporizador de inactividad expire y se encuentre dentro del temporizador de inactividad de otro flujo.

En la Fig. 4.7 se muestra una representación temporal del experimento. Se puede observar como el temporizador de Telnet finaliza dentro del temporizador del flujo 2, con lo que se enviarán juntos en el mismo datagrama con fin en el instante 84,02 segundos. De este modo, Netflow agrupa los flujos en función del temporizador de inactividad.



**Fig. 4.7** Línea temporal de los temporizadores de inactividad

En la Fig.4.8 y Fig. 4.9 se puede observar la captura del datagrama Netflow del flujo 1 y la captura del datagrama Netflow del flujo 2 y Telnet respectivamente.

```

190613 63.010024 10.0.13.100 10.0.13.101 CFLOW 114 total: 1 (v5) flow
└ Frame 190613: 114 bytes on wire (912 bits), 114 bytes captured (912 bits)
  └ Ethernet II, Src: Cisco_Sf:9c:e1 (00:07:b3:5f:9c:e1), Dst: Intel_Cf:b7:8f (00:0e:0c:cf:b7:8f)
  └ Internet Protocol Version 4, Src: 10.0.13.100 (10.0.13.100), Dst: 10.0.13.101 (10.0.13.101)
  └ User Datagram Protocol, Src Port: 57083 (57083), Dst Port: 9910 (9910)
  └ Cisco NetFlow/IPFIX
    Version: 5
    Count: 1
    Sysuptime: 8678376
    └ Timestamp: Apr 25, 2012 18:28:24.691904526 Hora de verano romance
      CurrentSecs: 1335371304
      CurrentNSecs: 691904526
      FlowSequence: 267
      EngineType: RP (0)
      EngineId: 0
      00... .... .... = samplingMode: No sampling mode configured (0)
      ..00 0000 0000 0000 = Samplerate: 0
    └ pdu 1/1
      SrcAddr: 10.0.13.101 (10.0.13.101)
      DstAddr: 192.168.0.2 (192.168.0.2)
      NextHop: 192.168.0.2 (192.168.0.2)
      InputInt: 3
      OutputInt: 2
      Packets: 4263
      Octets: 6385974
      └ [Duration: 12.256000000 seconds]
        StartTime: 8651.044000000 seconds
        Endtime: 8663.300000000 seconds
        SrcPort: 41880
        DstPort: 5001
        padding
        TCP Flags: 0x10
        Protocol: 17
        IP TOS: 0x00
        SrcAS: 0
        DstAS: 0
        SrcMask: 0 (prefix: 10.0.13.101/32)
        DstMask: 24 (prefix: 192.168.0.0/24)
        padding

```

**Fig. 4.8 Captura datagrama Netflow con flujo 1**

```

190629 84.024342 10.0.13.100 10.0.13.101 CFLOW 162 total: 2 (v5) flows
└ Cisco_NETFLOW/IPFIX
  Version: 5
  Count: 2
  Sysuptime: 8700376
  └ Timestamp: Apr 25, 2012 18:28:46.691904526 Hora de verano romance
    CurrentSecs: 1335371326
    CurrentNSecs: 691904526
    FlowSequence: 268
    EngineType: RP (0)
    EngineId: 0
    00... .... .... = SamplingMode: No sampling mode configured (0)
    ..00 0000 0000 0000 = Samplerate: 0
  └ pdu 1/2
    SrcAddr: 10.0.13.101 (10.0.13.101)
    DstAddr: 192.168.0.2 (192.168.0.2)
    NextHop: 192.168.0.2 (192.168.0.2)
    InputInt: 3
    OutputInt: 2
    Packets: 187509
    Octets: 192759252
    └ [Duration: 32.256000000 seconds]
      StartTime: 8652.388000000 seconds
      Endtime: 8684.644000000 seconds
      SrcPort: 50984
      DstPort: 5001
      padding
      TCP Flags: 0x10
      Protocol: 17
      IP TOS: 0x00
      SrcAS: 0
      DstAS: 0
      SrcMask: 0 (prefix: 10.0.13.101/32)
      DstMask: 24 (prefix: 192.168.0.0/24)
      padding
    └ pdu 2/2
      SrcAddr: 10.0.13.101 (10.0.13.101)
      DstAddr: 10.0.13.100 (10.0.13.100)
      NextHop: 0.0.0.0 (0.0.0.0)
      InputInt: 3
      OutputInt: 0
      Packets: 26
      Octets: 1052
      └ [Duration: 16.096000000 seconds]
        StartTime: 8675.280000000 seconds
        Endtime: 8691.376000000 seconds
        SrcPort: 49831
        DstPort: 23

```

**Fig. 4.9 Captura datagrama Netflow con flujo 2 y telnet**

### 4.3. Pruebas de consumo de CPU

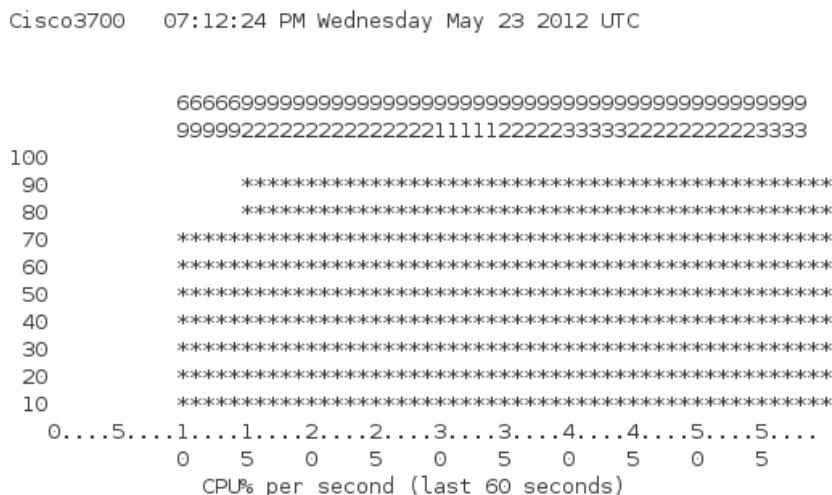
En la siguiente batería de pruebas se han realizado diferentes medidas para saber qué impacto tiene sobre el consumo de CPU en el router el tráfico generado con MGGEN con Netflow habilitado o deshabilitado.

Cisco nos proporciona grandes facilidades para la consulta de sus procesos. El comando para averiguar el consumo de CPU es el siguiente:

```
Cisco3700#sh processes cpu history
```

#### 4.3.1. Prueba de consumo de CPU sin Netflow

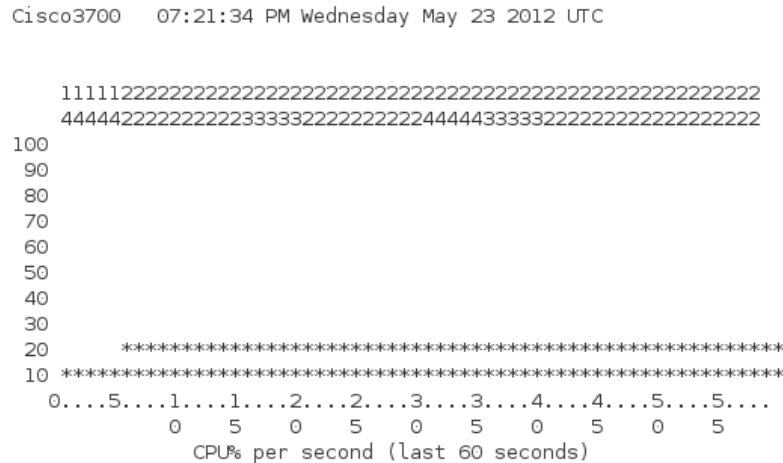
Con MGGEN generamos 50 Mbps con 300 bytes de datos útiles y 60 segundos de transmisión, el histórico del router ofrece lo siguiente:



**Fig. 4.10** Uso de CPU sin Netflow a 50 Mbps y 300 bytes

Como se puede observar en la imagen, el simple hecho de generar muchos paquetes por segundo (20833 paq/seg) ya casi satura la CPU con una media de 92% de utilización.

Por otro lado, si generamos 50 Mbps con 1400 bytes de datos útiles y 60 segundos de transmisión, el histórico del router ofrece lo siguiente:

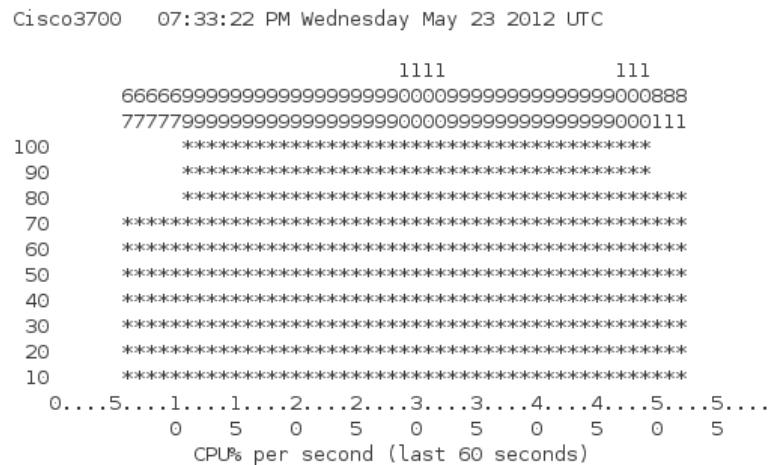


**Fig. 4.11** Uso de CPU sin Netflow a 50 Mbps y 1400 bytes

En este caso, al generar muchos menos paquetes por segundo incluso de mayor tamaño (4464 paq/seg), el router es mucho más estable con una utilización media de 22% de uso.

#### 4.3.2. Prueba de consumo de CPU con Full Netflow

Con MGGEN generamos 50 Mbps con 300 bytes de datos útiles y 60 segundos de transmisión, el histórico del router ofrece lo siguiente:

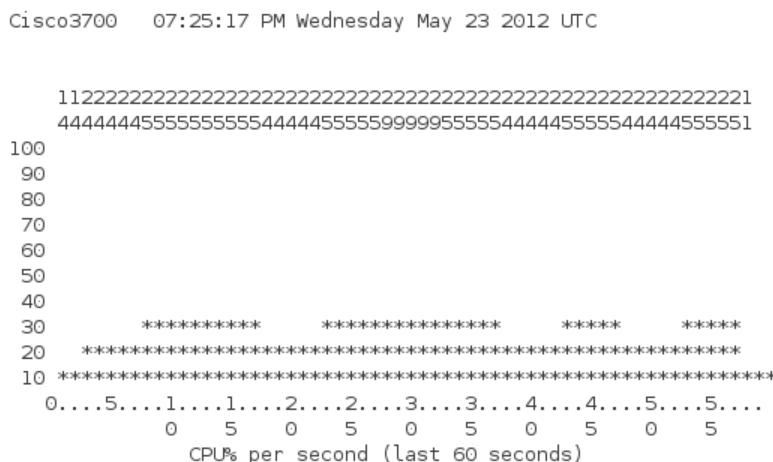


**Fig. 4.12** Uso de CPU con Full Netflow a 50 Mbps y 300 bytes

Como se observa, se puede apreciar una utilización total de la CPU con una media un 99% e incluso llegando al 100%, provocando inestabilidad en el sistema.

En comparación con el caso anterior, Netflow ha incrementado el uso de la CPU en un 7%, un incremento no muy notable contando que este sensor chequea cada uno de los paquetes entrantes, pero sí preocupante porque acaba por ocupar todo el uso de la CPU.

Por otro lado, si generamos 50 Mbps con 1400 bytes de datos útiles y 60 segundos de transmisión, el histórico del router ofrece lo siguiente:



**Fig. 4.13** Uso de CPU con Full Netflow a 50 Mbps y 1400 bytes

En esta imagen se ve más claramente como el incremento es bajo con respecto a la anterior con las mismas características y sin Netflow. La diferencia es aproximadamente un 4%. Nuevamente lo que más utiliza la CPU no es precisamente el continuo conteo de paquetes por parte del sensor, sino por la cantidad de paquetes por segundo que atraviesa el router.

Esto hace que todas las pruebas siguientes se realicen con un payload de 300 y 1400 bytes, de este modo no saturaremos la CPU del router.

## CAPÍTULO 5. GENERACIÓN CON MGEN, CAPTURA CON DAG Y MONITORIZACIÓN FULL/RANDOM NETFLOW

En este capítulo realizaremos cuatro pruebas diferentes. Observaremos cómo se comporta Full Netflow y Random Netflow frente a un flujo con diferentes tasas y tamaños de paquetes, y así tener una primera toma de contacto para posteriormente comprobar su efectividad con varios flujos, diferentes velocidades y repeticiones sobre el mismo experimento. De esta manera podremos obtener medias y desviaciones estándar para valorar su comportamiento.

### 5.1. Realización de pruebas con un flujo y Full Netflow

El objetivo de estas pruebas es comprobar si los paquetes totales enviados son coincidentes en las diferentes aplicaciones de captura, es decir, si coincide lo exportado por Netflow y posteriormente analizado por NfSen, y lo capturado por la DAG (en formato ERF) y el registro de llegada del propio MGEN (servidor).

Se espera que todos los paquetes entrantes al router sean muestreados y recibidos por la tarjeta DAG de alta precisión. Por otro lado representaremos si existen perdidas en el destino (RadPC).

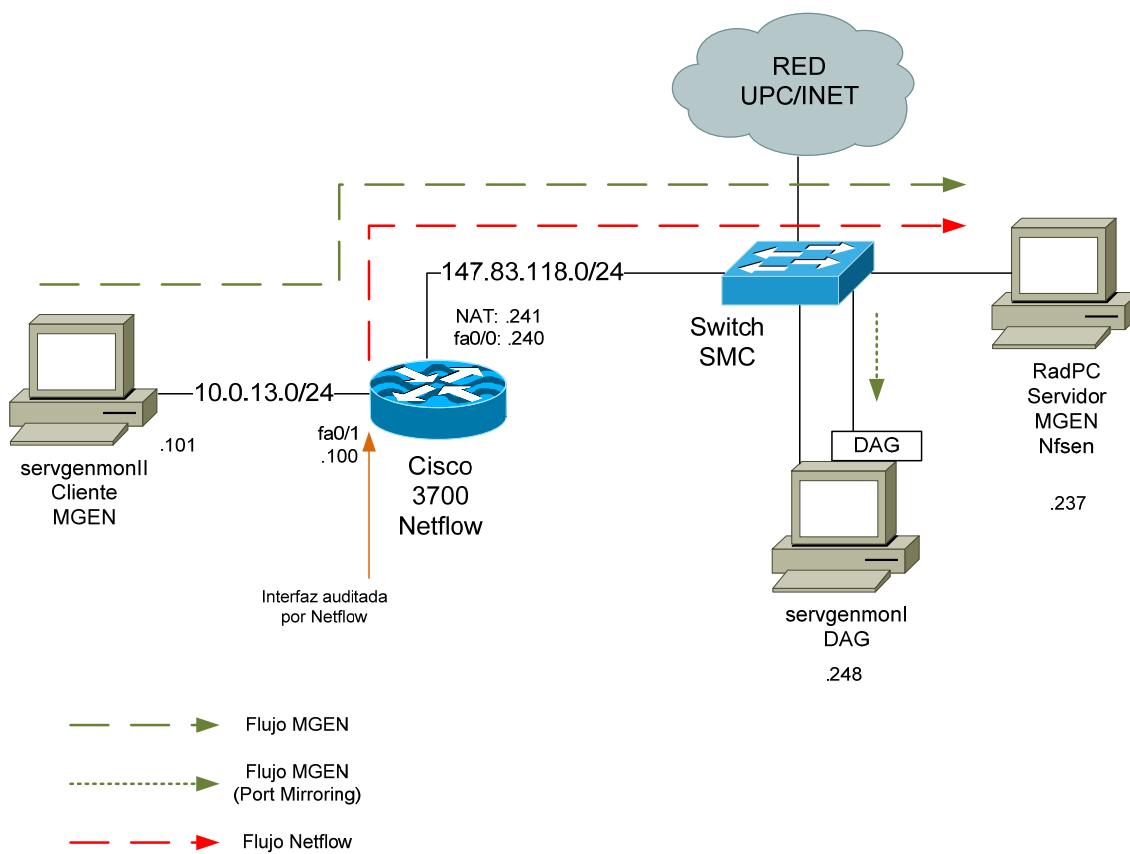
Emitiremos un flujo unicast de 60 segundos con dos tipos de payloads (300 y 1400 bytes). De esta manera comprobaremos cómo afectan las cabeceras y veremos la diferencia entre los paquetes enviados y recibidos, utilizando para ello dos tipos de velocidades (50 Mbps y 100Mbps):

1. Prueba de 1400 bytes a 50 y 100Mbps
2. Prueba a 300 bytes a 50 y 100Mbps

La Fig. 5.1 muestra el escenario de las pruebas. El PC servgenmonII (cliente) generará el tráfico MGEN de 60 segundos hacia RadPC (servidor). Para analizar ese mismo tráfico, hay que hacerlo llegar a la tarjeta DAG. El switch tiene una característica que permite configurarlo para que uno de sus puertos actúe como port mirroring, así todo el tráfico entrante al switch por parte del router será replicado por el puerto donde está conectado la DAG. Tanto los hosts (con radclock), la DAG (dagclock) y el router (con ntp) están sincronizados por NTP contra el servidor CESCA de l'Anella Científica ([ntp.cesca.cat](http://ntp.cesca.cat)) para tener una sincronización estable y estándar.

El método de análisis se evalúa mediante el histograma y el graficado de los intervalos de entre paquetes junto con la suma acumulativa de la cantidad de paquetes. En el [Anexo IV](#) podemos encontrar imágenes reales del laboratorio,

la configuración y el conexionado de los dispositivos de red, así como los resultados numéricos de las pruebas.



**Fig. 5.1** Escenario completo de captura con DAG

## 5.2. Resultados

Como se aprecia en la tabla 5.1, los resultados obtenidos son acordes con la teoría, en la que se deben muestrear todos los paquetes entrantes al router, obteniendo el 100% en todas las pruebas.

**Tabla 5.1.** Resultados con Full Netflow

| Full Netflow     |                 |                  |                |                  |              |                  |
|------------------|-----------------|------------------|----------------|------------------|--------------|------------------|
| Velocidad (Mbps) | Payload (bytes) | Nfsen (paquetes) | DAG (paquetes) | RadPC (paquetes) | % muestreado | % pérdidas RadPC |
| 50               | 1400            | 267858           | 267858         | 267858           | 100          | 0                |
|                  | 300             | 1249881          | 1249881        | 119176           | 100          | 4,69             |
| 100              | 1400            | 511707           | 511707         | 507286           | 100          | 0,86             |
|                  | 300             | 1322140          | 1322140        | 1307937          | 100          | 1,08             |

En el destino de la transmisión sí se aprecian pérdidas. En los casos con más pérdidas (300 bytes), es razonable encontrar alguna perdida ya que con este payload la cadencia de paquetes es muy elevada y recordando que RadPC es el equipo con menos prestaciones del escenario no es de extrañar obtener alguna pérdida. De todas maneras, estas pérdidas no afectan a los resultados que hacen referencia a Netflow, ya que como podemos observar que la tarjeta DAG recibe todos los paquetes, y por tanto el router se está comportando adecuadamente.

### 5.3. Realización de pruebas con un flujo y Random Netflow

Acabamos de comprobar que el router es capaz de capturar todos los paquetes en Full Netflow. En la siguiente batería de pruebas se realiza el mismo protocolo de actuación que en el anterior apartado pero con la diferencia de que se establece el muestreo aleatorio no determinístico por parte del sensor Netflow. La comprobación previa del rendimiento del router en Full Netflow nos hace confiar en que el resultado del muestreo no se verá afectado por pérdidas provocadas por sobrecarga de la CPU del router.

Random Netflow proporciona datos de un subconjunto de tráfico que atraviesa un router en función de sólo un paquete seleccionado al azar de  $n$  paquetes secuenciales, donde esta  $n$  es configurable por el usuario de 1 a 65535.

Este muestreo estadístico del tráfico reduce sustancialmente el consumo de recursos del router.

Para que Random Netflow funcione es imprescindible habilitar CEF (Cisco Express Forwarding) una tecnología de conmutación de capa 3 que optimiza el rendimiento de la red y permite un modo de conmutación más rápida con menos ciclos de CPU.

Tomando como base de referencia la tabla de enrutamiento IP, CEF crea su propia tabla de reenvío que se denomina Forwarding Information Base (FIB). La FIB es una tabla organizada de modo diferente que la tabla de enrutamiento, y es la que se utiliza para definir a qué interfaz se debe reenviar el paquete.

Se realizarán cuatro tipos de muestreos y se evaluarán cada uno de ellos:

- 1 de cada 10: Prueba de 1400 y 300 bytes a 50 y 100Mbps
- 1 de cada 100: Prueba de 1400 y 300 bytes a 50 y 100Mbps
- 1 de cada 1000: Prueba de 1400 y 300 bytes a 50 y 100Mbps
- 1 de cada 65535: Prueba de 1400 y 300 bytes a 50 y 100Mbps

El método de análisis se evalúa mediante el histograma y el graficado de los intervalos de tiempo entre paquetes. En el [Anexo V](#) podemos encontrar las diferentes pruebas y las gráficas realizadas.

## 5.4. Resultados

Los resultados obtenidos son acordes con la teoría, obteniendo resultados muy buenos para un solo flujo.

**Tabla 5.2.** Tabla comparativa de muestreo Random Netflow

|             |                 | Paquetes capturados por Random Netflow |          |           |            |
|-------------|-----------------|--|----------|-----------|------------|
| Tasa (Mbps) | Payload (bytes) | 1 de 10                                | 1 de 100 | 1 de 1000 | 1 de 65535 |
| 100         | 300             | 10,00%                                 | 0,99%    | 0,09%     | 0,00154%   |
|             | 1400            | 9,99%                                  | 1,00%    | 0,09%     | 0,00156%   |
| 50          | 300             | 10,00%                                 | 0,99%    | 0,09%     | 0,00151%   |
|             | 1400            | 10,00%                                 | 0,99%    | 0,09%     | 0,00111%   |
| Teórico     |                 | 10,00%                                 | 1,00%    | 0,10%     | 0,00152%   |

En la tabla 5.2 podemos ver como el muestreo de paquetes es muy acorde con el teórico, afirmando que Netflow se comporta adecuadamente en este caso.

**Tabla 5.3.** Tabla de pérdidas en destino

|             |                 | Pérdidas en destino (RadPC) |          |           |            |
|-------------|-----------------|-----------------------------|----------|-----------|------------|
| Tasa (Mbps) | Payload (bytes) | 1 de 10                     | 1 de 100 | 1 de 1000 | 1 de 65535 |
| 100         | 300             | 17,94%                      | 43,13%   | 38,52%    | 16,78%     |
|             | 1400            | 0,52%                       | 4,16%    | 0,10%     | 0,50%      |
| 50          | 300             | 10,37%                      | 32,75%   | 33,94%    | 11,96%     |
|             | 1400            | 0%                          | 0,13%    | 0%        | 0,01%      |

En la tabla 5.3 observamos, para las pérdidas en el PC de destino, resultados similares a los de la prueba anterior: para los flujos con alto payload, las pérdidas son mínimas por el bajo consumo de CPU y para bajos payloads lo contrario. De todas maneras, como en la prueba anterior, lo importante es que el router se comporta de la manera adecuada.

## 5.5. Comparativa final de muestreo

En la Tabla 5.4 podremos observar el tanto por ciento de muestreo que realiza Netflow con respecto a los paquetes recibidos por la DAG:

**Tabla 5.4.** Tabla comparativa de muestreo

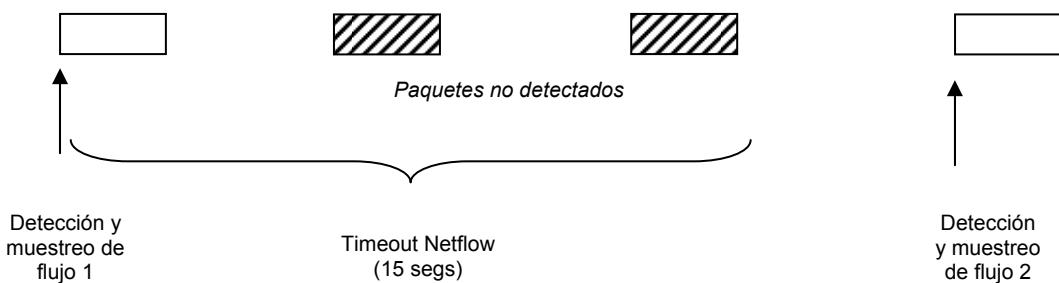
| Velocidad (Mbps) | Tamaño paquete (bytes) | Porcentaje de paquetes detectados según tipo de muestreo |         |          |           |
|------------------|------------------------|--|---------|----------|-----------|
|                  |                        | Random Netflow   |         |          |           |
|                  |                        | Full Netflow   | 1 de 10 | 1 de 100 | 1 de 1000 |
| 100              | 300                    | 100%   | 10,00%  | 0,99%    | 0,09%     |
|                  | 1400                   | 100%   | 9,99%   | 1,00%    | 0,09%     |
| 50               | 300                    | 100%   | 10,00%  | 0,99%    | 0,09%     |
|                  | 1400                   | 100%   | 9,99%   | 0,99%    | 0,09%     |
| Teórico          |                        | 100%   | 10,00%  | 1,00%    | 0,10%     |
|                  |                        |  |         |          | 0,00152%  |

En la tabla se puede apreciar como el conteo de paquetes por parte de Netflow es muy acertado independientemente del tamaño del paquete y la frecuencia de muestreo. Podemos afirmar entonces que Netflow se comporta adecuadamente con sólo un flujo activo.

## 5.6. Generación de múltiples flujos con MGGEN, captura con DAG y Full/Random Netflow

En esta prueba vamos a estresar el sensor Netflow emitiendo seis flujos unicast durante 60 segundos entre un cliente y un servidor con un payload de 1400 bytes. Gracias a la característica de port mirroring del switch SMC, replicaremos todo el tráfico a la tarjeta DAG de alta precisión. Obtendremos así una captura limpia antes de llegar el servidor y poder comparar si la recepción de paquetes en ambos es la misma.

El punto más importante de la prueba es saber si Netflow es capaz de detectar y evaluar correctamente varios flujos con diferentes velocidades, para diversas frecuencias de muestreo. El problema surge con los flujos de menor velocidad ya que el espacio entre paquetes es mayor, por lo que el muestreador puede interpretar un inicio y un fin de flujo erróneo o incluso no detectar el flujo, tal como se ilustra en la Fig. 5.2.

**Fig. 5.2** Representación errónea de flujos

En este ejemplo, Netflow detecta un flujo y muestrea un paquete: Supongamos que, el muestreo sea de 1 de cada 100 paquetes. Netflow debería escoger aleatoriamente uno de los 100 paquetes siguientes, pero no llega a detectarlos bien, por lo que el timeout va aumentando hasta los 15 segundos y es cuando genera un report con la información de ese flujo. Seguidamente, Netflow detecta erróneamente el flujo 2 asumiendo que el flujo 1 ha expirado. Esto conlleva realizar medidas menos precisas a causa tener un gran espaciado entre paquetes.

En la ecuación 5.1 podemos asociar la tasa mínima que debemos utilizar con un payload de 1400 bytes para que, con el mínimo tiempo configurable (15 segundos) de timeout, los flujos puedan ser detectados.

En la salida de MGEN aparecen 1442 bytes, los cuales 1400 son sólo data de MGEN, 8 de cabecera UDP, 20 de cabecera IP, 12 de direcciones MAC y 2 de campo L/T. A esto hay que añadirle el preámbulo y el CRC de nivel Ethernet, lo que nos hace un paquete de 1454 bytes a transmitir en 15 segundos. De aquí podemos obtener la velocidad mínima que podrá ser detectada por Netflow:

$$\text{Velocidad} = \frac{\text{tamaño}}{\text{tiempo}} = \frac{1454*8\text{bits}}{15 \text{ segs}} = 775,467 \text{ bps} \quad (5.1)$$

Si el flujo es inferior a 775,457 bits/s, Netflow es incapaz de detectar el flujo o lo hará incorrectamente. Es importante saber que (5.1) también depende de la frecuencia de muestreo configurada, es decir, en Full Netflow muestreará todos los paquetes, pero si estuviéramos en el caso de muestreo de 1 de cada 1000 y a esta velocidad hay mucha probabilidad de que ignore los paquetes.

En nuestro experimento inyectaremos al router seis flujos a la vez, con el mismo tamaño de paquete (1400 bytes) pero con diferentes velocidades:

- Flujo 1: 50 Mbps
- Flujo 2: 5 Mbps
- Flujo 3: 500 kbps
- Flujo 4: 50 kbps
- Flujo 5: 5 kbps
- Flujo 6: 500 bps

El script de generación de flujos con MGEN es el siguiente:

```
# Inicio script MGGEN

# Creación 6 flujos UDP con mismo destino, periódicos...

0.0 ON 1 UDP SRC 5501 DST 147.83.118.237/5501 PERIODIC [4464 1400]
0.0 ON 2 UDP SRC 5502 DST 147.83.118.237/5502 PERIODIC [446 1400]
0.0 ON 3 UDP SRC 5503 DST 147.83.118.237/5503 PERIODIC [44 1400]
0.0 ON 4 UDP SRC 5504 DST 147.83.118.237/5504 PERIODIC [4 1400]
0.0 ON 5 UDP SRC 5505 DST 147.83.118.237/5505 PERIODIC [0.44 1400]
0.0 ON 6 UDP SRC 5506 DST 147.83.118.237/5506 PERIODIC [0.044 1400]

# Parar los 6 flujos UDP con una duración de 60 segundos

60 OFF 1
60 OFF 2
60 OFF 3
60 OFF 4
60 OFF 5
60 OFF 6
```

En el [Anexo VI](#) podemos encontrar las diferentes pruebas y los resultados obtenidos.

### 5.6.1. Resultados pruebas iniciales

En estas pruebas realizaremos un pequeño estudio en el que compararemos el porcentaje de paquetes muestreados con lo capturado por la DAG en los cinco tipos de muestreo y para todos los flujos. De esta manera veremos cómo se comporta Netflow frente a diferentes flujos y diferentes velocidades.

**Tabla 5.5.** Tabla comparativa muestreo

| % paquetes muestreados |              |         |          |           |            |
|------------------------|--------------|---------|----------|-----------|------------|
| Tipo de flujo          | Full Netflow | 1 de 10 | 1 de 100 | 1 de 1000 | 1 de 65535 |
| Flujo 1 – 50Mbps       | 100%         | 10,03%  | 1,00%    | 0,10%     | 0,00186%   |
| Flujo 2 – 5Mbps        | 100%         | 9,68%   | 0,94%    | 0,09%     | 0%         |
| Flujo 3 – 500kbps      | 100%         | 9,35%   | 0,68%    | 0,03%     | 0%         |
| Flujo 4 – 50kbps       | 100%         | 9,95%   | 1,24%    | 0,41%     | 0%         |
| Flujo 5 – 5kbps        | 100%         | 29,62%  | 7,40%    | 3,70%     | 0%         |
| Flujo 6 – 500bps       | 100%         | 33,33%  | 0%       | 0%        | 0%         |
| Teórico                | 100%         | 10,00%  | 1,00%    | 0,10%     | 0,00152%   |

En estos resultados podemos comprobar cómo Netflow empieza a comportarse diferente respecto a los resultados del apartado [5.5](#).

Las discrepancias entre los valores teóricos y los obtenidos son a causa de las tasas y la cantidad de flujos utilizados en el experimento. Como pudimos comprobar en la comparativa del apartado [5.5](#), en que el muestreo con solo un flujo y con todos los tipos muestreos es muy bueno, en este experimento hemos encontrado desajustes en los resultados, obteniendo en los flujos con menos tasa un incremento en el muestreo respecto a lo teórico.

Cabe destacar que para el flujo más pequeño el tanto por ciento muestreado es muy irregular, siendo indetectable en la mayoría de muestreos.

## 5.7. Realización de 10 pruebas

Para obtener una correcta estimación fiable de las pruebas realizadas en el apartado [5.6.1](#) es repetirla diversas veces y calcular los intervalos de confianza asociados.

Estas pruebas las hemos extendido a una duración de 5 minutos para tener más precisión y más valores en los que basarse.

### 5.7.1. Resultados con Full Netflow

En estas pruebas realizaremos un estudio en el que compararemos el porcentaje muestreado con lo capturado por la DAG con Full Netflow, para todos los flujos y repetido 10 veces.

**Tabla 5.6.** Media y desviación con Full Netflow

| Número de Flujo | Full Netflow |          |          |          |          |          |          |          |          |           |            |
|-----------------|--------------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|------------|
|                 | 1/10 (%)     | 2/10 (%) | 3/10 (%) | 4/10 (%) | 5/10 (%) | 6/10 (%) | 7/10 (%) | 8/10 (%) | 9/10 (%) | 10/10 (%) | Media (%)  |
| 1-50M           | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 2-5M            | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 3-500k          | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 4-50k           | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 5-5k            | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 6-500b          | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |

Como se observa, con Full Netflow se muestrean todos y cada uno de los paquetes que atraviesan el router.

### 5.7.2. Resultados con Random Netflow

En estas pruebas realizaremos un estudio en el que compararemos el porcentaje muestreado con lo capturado por la DAG con todas las probabilidades de Random Netflow, para todos los flujos y repetido 10 veces.

A continuación veremos los intervalos de confianza de los diferentes flujos después de realizar 10 pruebas para cada uno de ellos y para cada tipo de muestreo:

**Tabla 5.7.** Resultados obtenidos con intervalos de confianza

| Flujo-Tasa            | Intervalos de confianza (95%) |                           |                          |                       |
|-----------------------|-------------------------------|---------------------------|--------------------------|-----------------------|
|                       | Tipo de muestreo              |                           |                          |                       |
| 1 de 10<br>(paquetes) | 1 de 100<br>(paquetes)        | 1 de 1000<br>(paquetes)   | 1 de 65535<br>(paquetes) |                       |
| 1-50M                 | 133972,80±204,55<br>(0,15%)   | 13407,30±19,66<br>(0,14%) | 1341,60±11,27<br>(0,84%) | 20,30±0,68<br>(3,35%) |
| 2-5M                  | 13333,40±196,73<br>(1,47%)    | 1321,60±18,52<br>(1,40%)  | 133,20±10,21<br>(7,66%)  | 2,00±0,83<br>(41,57%) |
| 3-500k                | 1326,3±12,53 (0,94%)          | 133,90±4,97<br>(3,71%)    | 12,30±1,71<br>(13,90%)   | 0,50±0,31<br>(61,97%) |
| 4-50k                 | 118,50±4,88 (4,12%)           | 12,00±2,51<br>(20,91%)    | 0,60±0,41<br>(68,52%)    | 0±0 (0%)              |
| 5-5k                  | 11,90±1,69 (14,25%)           | 1,40±0,50 (35,41%)        | 0,30±0,40<br>(132,28%)   | 0±0 (0%)              |
| 6-500b                | 1,26±0,78 (60,49%)            | 0,10±0,19 (185%)          | 0±0 (0%)                 | 0±0 (0%)              |

Podemos observar como para los flujos 4, 5 y 6, los valores de los intervalos de confianza son altos respecto a su media. Esto provoca tener más probabilidad de error.

Un posible criterio de valoración podría ser que, si el porcentaje del intervalo de confianza respecto a la media es superior al 10%, estos valores no son de plena confianza para el muestreo. El caso más destacado está en la probabilidad más pequeña (1 de cada 65535), en donde cinco de los seis flujos pueden ser despreciados por su alto error.

En la Tabla 5.8 se muestra la comparativa relacionada con la velocidad:

**Tabla 5.8.** Resultados obtenidos de paquetes y velocidad

| Flujo-Tasa | Intervalos de confianza (95%) |                      |                      |                        |
|------------|-------------------------------|----------------------|----------------------|------------------------|
|            | Tipo de muestreo              |                      |                      |                        |
|            | 1 de 10                       | 1 de 100             | 1 de 1000            | 1 de 65535             |
| 1-50M      | 50,01Mbps±76,36kbps           | 50,05Mbps±73,39kbps  | 50,08Mbps±420,87kbps | 49,66Mbps±1,66Mbps     |
| 2-5M       | 4,97Mbps±73,44kbps            | 4,93Mbps±69,14kbps   | 4,97Mbps±381,30kbps  | 4,89Mbps±2,03Mbps      |
| 3-500k     | 495,11kbps±4,678kbps          | 499,89kbps±18,55kbps | 459,20kbps±63,82kbps | 1223,32kbps±758,20kbps |
| 4-50k      | 44kbps±1,82kbps               | 44,8kbps±9,37kbps    | 22,40kbps±15,34kbps  | 0kbps±0kbps            |
| 5-5k       | 4kbps±0,63kbps                | 5,22kbps±1,85kbps    | 11,20kbps±14,81kbps  | 0kbps±0kbps            |
| 6-500b     | 485,33bps±293,60bps           | 373,33bps±694bps     | 0bps±0bps            | 0bps±0bps              |

Aquí podemos apreciar como el intervalo de confianza respecto a la velocidad va aumentando a medida que disminuye la velocidad del flujo. En el [Anexo VII](#) podemos encontrar las diferentes pruebas y los resultados obtenidos.

## CAPÍTULO 6. GENERACIÓN Y CAPTURA CON DAG

En este capítulo estudiaremos cómo se generan y se capturan trazas con el software de la tarjeta DAG. Aplicando estas ventajas con los datos de nuestro escenario, podremos realizar diferentes configuraciones de trazas y ver sus comportamientos, con el objetivo de definir una traza con gran aleatoriedad de longitud de paquetes y puertos.

### 6.1. Creación de trazas

Una de las características de la tarjeta DAG es que aparte de poder capturar tráfico también es posible generararlo. La tarjeta viene con una suite de programas, entre ellos: *dagflood*, *daggen* y *dagbits*.

*Dagflood* se encarga de transmitir a máxima velocidad un tráfico con paquetes guardados en un archivo ERF.

Este tráfico se genera a partir de un script basado en una sintaxis entendible por *daggen*. Este software nos permite crear en base a un script, un archivo ERF (traza) con el tráfico que debe ser injectado.

Una vez creado el archivo ERF, debemos asegurarnos que está alineado a 64 bits para que *dagflood* lo pueda transmitir, para ello utilizamos *dagbits*.

En este proyecto hemos trabajado con la implementación del TFC de Iván Pérez [6], el cual realizó un programa de generación basado en *dagflood* pero con tasa constante (*GenCBR*).

Para generar tráfico vamos a crear un script para que defina paquetes con diversas direcciones MAC e IP válidas para nuestro escenario y así poder evaluar como Netflow se enfrenta a muchos flujos y diferentes longitudes de paquete.

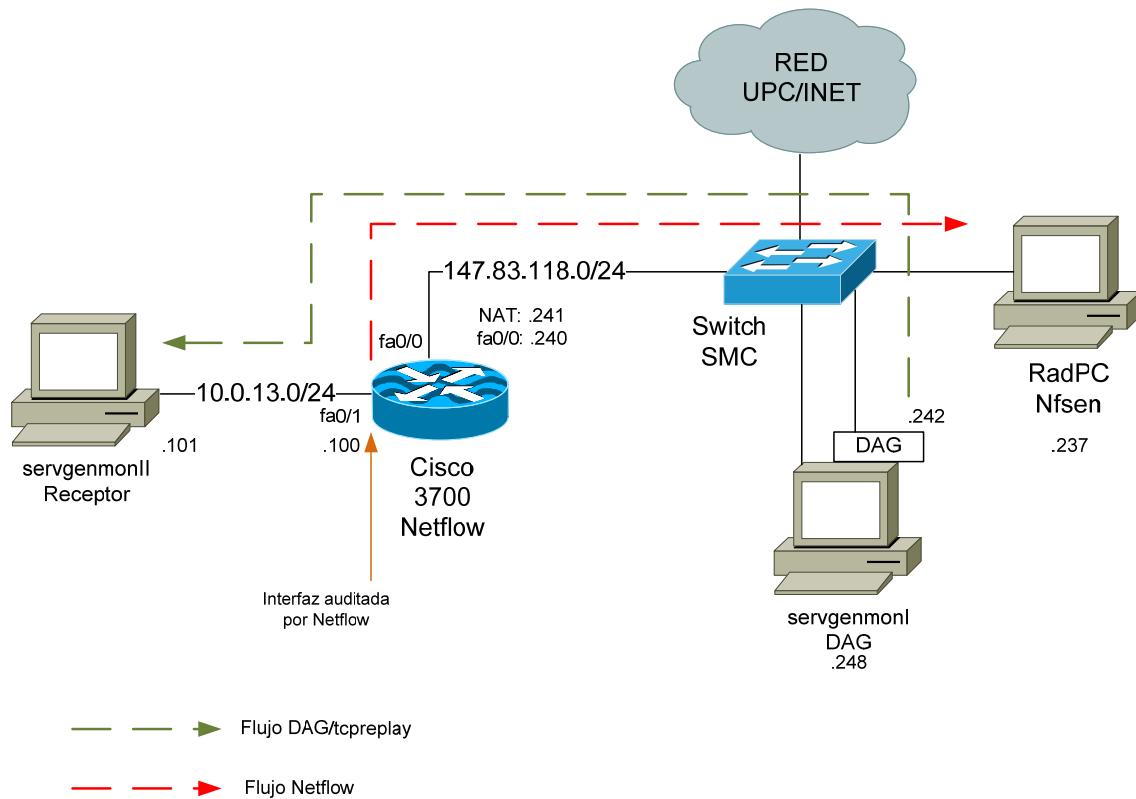
Hay que tener en cuenta que a la DAG no se le puede asignar una dirección IP, con lo que si queremos generar paquetes IP a partir del script, se tendrá que usar una MAC inventada y una IP válida del rango en el que operamos.

El script consiste en la construcción manual de un paquete IP y como hemos comentado anteriormente contendrá valores reales y ficticios.

*Daggen* nos permite flexibilidad a la hora de crear los paquetes y poner en todos los campos la información que nosotros queramos.

Si suponemos la situación en el que la DAG desea enviar paquetes al PC servgenmonII, éstos deberán llevar la IP de origen de la DAG y la IP de destino del PC servgenmonII como también la MAC origen la de la DAG y la MAC destino la de la interfaz de entrada del router (fa0/0). Así posteriormente el router cambia las MACs origen (fa0/1) y destino (PC servgenmonII).

En la Fig. 6.1 se muestra el escenario de pruebas.



**Fig. 6.1** Escenario completo de laboratorio

En cambio, si suponemos la situación en el que la tarjeta Ethernet del PC servgenmonI desea enviar paquetes a servgenmonII, éstos deberán llevar la IP de origen de la tarjeta Ethernet y la IP de destino del servgenmonII como también la MAC origen la de la tarjeta Ethernet y la MAC destino la de la interfaz de entrada del router (fa0/0). Así posteriormente el router cambia las MACs origen (fa0/1) y destino (servgenmonII).

A la hora de realizar las trazas hemos probado diferentes maneras de implementarlas. Inicialmente comprobamos la generación básica de una traza con paquetes de longitud fija y puertos fijos. Seguidamente desarrollamos otros scripts con paquetes de longitudes variables a partir de insertar distribuciones uniformes o normales, para tener variabilidad con la desviación estándar. Finalmente creamos un script válido para poder generar paquetes con longitudes cíclicas y puertos variables. Concretamente, se generarán 1362 tipos de tamaños, desde 38 hasta 1400 bytes de payload, con puertos origen y destino, comprendidos entre 5977:6021 y 6978:7021, respectivamente, con un total de 1892 posibles flujos origen-destino.

En el [Anexo VIII](#) podemos encontrar las diferentes generaciones con los scripts implementados y los resultados obtenidos.

## 6.2. Resultados

Los resultados obtenidos con la primera traza generada en la que la longitud y los puertos eran fijos fueron insatisfactorios, ya que no ofrecía diversidad ni de puertos ni de tamaño para un futuro uso.

En las pruebas realizadas con la segunda traza, en la que incluíamos longitudes variables a partir de distribuciones, hacían que, tanto el router como Netflow no sean capaces de detectar todos los flujos emitidos ni por la DAG ni por *tcp replay*, ya que la aleatoriedad en el tamaño del paquete conlleva que los campos de longitud de paquete IP (cabecera capa 3) y longitud de datagrama UDP (cabecera capa 4) no sean compatibles (por limitaciones en el proceso de generación con DAG) y el router descarte los paquetes con estas incoherencias.

Por ello se creó la traza con longitud cíclica<sup>3</sup> y puertos variables, en donde la única aleatoriedad está en los campos de puerto origen y destino de UDP. Estos dos campos no interfieren en el valor de los campos de longitud de paquete IP y longitud de datagrama UDP, por lo que el router no detectará anomalías en la formación de paquetes, y en cambio podemos asegurar que la traza tendrá flujos de velocidades variables.

Mediante la generación de scripts con *daggen* y la emisión con *GenCBR*, la DAG emite correctamente la traza a máxima velocidad (1Gbit/s) pero se producen pérdidas en el switch por la diferencia de velocidad con el router (100Mbits/s). Como la DAG no es una NIC normal y no se puede forzar su funcionamiento a 100 Mbit/s, no hay manera de solucionar este inconveniente.

Siendo esto un grave problema y como se verá en el siguiente capítulo, nos decantamos por utilizar *tcp replay* y la tarjeta Ethernet y así no producir colapso en los equipos.

---

<sup>3</sup> Es decir, el primer paquete tiene un tamaño de 38 bytes, el siguiente de 38+1, hasta el paquete 1400 con tamaño 1399+1 bytes, y a continuación empieza el ciclo de nuevo con el paquete de tamaño 38 bytes, el siguiente con 38+1, etc.

## CAPÍTULO 7. TCPREPLAY: EDICIÓN Y REPETICIÓN DE FICHEROS PCAP

En este capítulo presentaremos un software libre que nos permitirá editar y emitir archivos PCAP. El objetivo es replicar con *tcpreplay* una traza con suficiente volumen para poder evaluar el comportamiento de Netflow. Por otro lado, realizaremos pruebas de emisión con grandes trazas obtenidas por *daggen*.

### 7.1. Introducción a *tcpreplay*

*Tcpreplay* [10] es un potente software que ofrece múltiples posibilidades a la hora de editar archivos PCAP y replicar tráfico con extensión PCAP.

Se compone de varias herramientas, entre las más importantes: *tcpprep*, *tcprewrite* y *tcpreplay*.

- *Tcprep*: Es una utilidad preparatoria al *tcprewrite*. Crea un archivo *cache* (*input.cache*) que se utiliza para separar el tráfico entre cliente y servidor. Es una manera de dividir el tráfico entre dos nodos, aunque en la captura provengan de diferentes fuentes.
- *Tcprewrite*: Esta utilidad reescribe información y permite editar las cabeceras de los paquetes que están dentro de cualquier archivo PCAP. Es capaz de reescribir las cabeceras de capa 2, capa 3, capa 4....lo que conlleva poder cambiar direcciones MAC, direcciones IP, puertos, etc...
- *Tcpreplay*: Esta utilidad permite a partir de un archivo PCAP, replicar todo lo que este contenga por la interfaz que se desee.

El objetivo es reproducir en nuestro escenario el tráfico capturado previamente en una traza, concretamente, la proporcionada por CESCA [11], en las que tenemos un archivo ERF con todo el tráfico de todas las universidades de Catalunya durante un periodo de tiempo. Esto comporta tener múltiples fuentes y tamaños de paquete y, por tanto, flujos de tamaño variable. Las utilidades de *tcpreplay* nos permitirán retocar esta traza, consiguiendo cambiar las direcciones MAC e IP del archivo por las de nuestro escenario. Así podremos replicar el tráfico y muestrearlo para poderlo analizar posteriormente.

El proceso de tratado y comandos lo podemos ver en el [Anexo IX](#).

Los resultados obtenidos después de replicar la misma traza tanto con *tcpreplay* como con *GenCBR* fueron insatisfactorios ya que incomprensiblemente en la recepción sólo llegaban los paquetes con longitud 80 bytes. A pesar de haber analizado con detalle los procesos de preparación

de los archivos y las capturas resultantes no supimos razonarlo, incluso realizando muchas pruebas.

En los siguientes apartados realizaremos pruebas con las trazas generadas con *daggen*.

## 7.2. Creación, repetición y estudio del comportamiento de las trazas y Nfsen con $10^5$ y $10^6$ paquetes

Habiendo realizado las pruebas del apartado [7.1](#), en donde la traza de CESCA no se recibía correctamente, hemos optado por generar dos trazas con gran volumen de paquetes, longitud de tamaños cílicos y puertos aleatorios, usando *daggen*. De esta manera podremos evaluar el comportamiento de Netflow.

Las pruebas se han realizado con dos tipos trazas con diferente carga: una traza con 100000 paquetes cuya longitud sigue una ley cíclica<sup>4</sup> y una distribución normal de puertos con media 6000 y 7000 y desviación estándar 5, y otra traza con la misma configuración pero con un millón de paquetes. En los dos casos se monitoriza la transmisión con Full Netflow.

Al usar *tcpreplay*, la traza de 100000 paquetes tarda 76,33 segundos en transmitirse y la de un millón de paquetes tarda 766,98 segundos. En los dos casos se ha comprobado la recepción total de los paquetes, sin pérdida alguna.

En el [Anexo X](#) podemos encontrar el script de generación de trazas (*new\_script\_ports2\_eth0\_normal5.gen*), los comandos para generar el archivo PCAP (*new\_script\_ports2\_eth0\_normal5.pcap*) y los histogramas.

## 7.3. Resultados

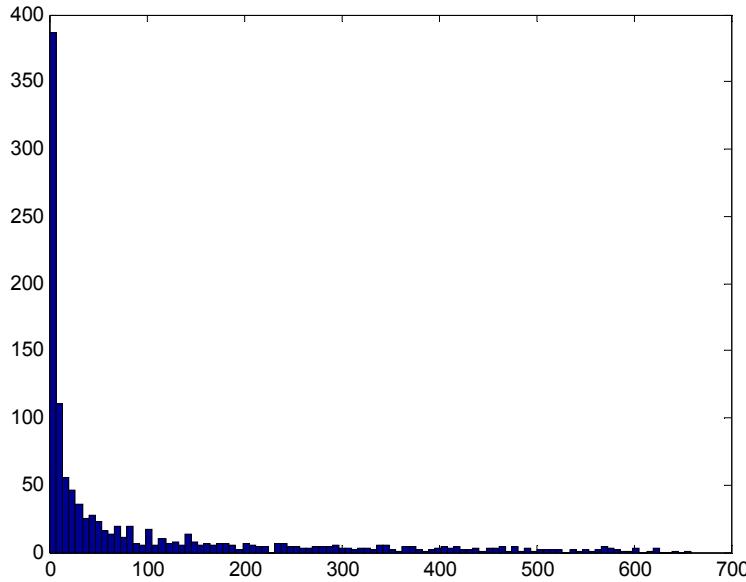
Las transmisiones utilizadas con *tcpreplay* generan que Netflow reporte correctamente todos los paquetes, en los dos casos, obteniendo los siguientes histogramas:

### 7.3.1. Traza $10^5$ paquetes

El histograma de paquetes por flujo queda de la siguiente manera:

---

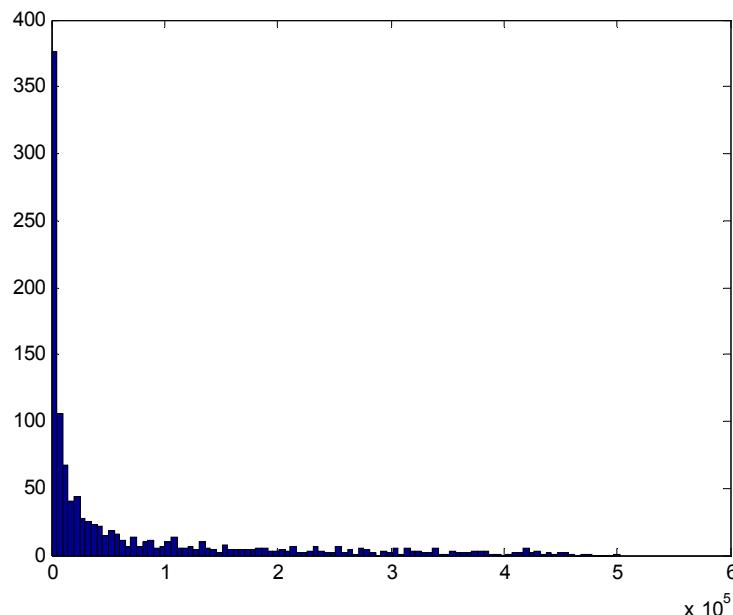
<sup>4</sup> Es decir, el primer paquete tiene un tamaño de 38 bytes, el siguiente de 38+1, hasta el paquete 1400 con tamaño 1399+1 bytes, y a continuación empieza el ciclo de nuevo con el paquete de tamaño 38 bytes, el siguiente con 38+1, etc.



**Fig. 7.1** Histograma de paquetes/flujo para la traza de 100000 paquetes

En la Fig. 7.1 podemos observar la cantidad de paquetes que contienen los flujos, que parece seguir una distribución exponencial negativa. Obtenemos un gran volumen de flujos con pocos paquetes por flujo, y muy pocos flujos con una gran cantidad de paquetes. La media es de 91,32 paquetes/flujo y la desviación estándar 144,23 paquetes/flujo. Los extremos son, un flujo máximo con 659 paquetes y un mínimo de varios flujos con un paquete.

El histograma de bytes por flujo queda de la siguiente manera:

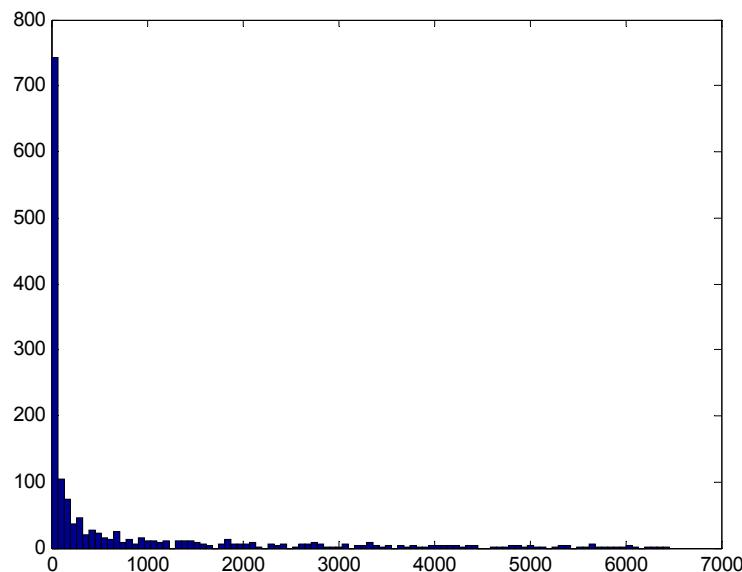


**Fig. 7.2** Histograma de bytes/flujo, para la traza de 100000 de paquetes

En la Fig. 7.2 se puede apreciar como en el caso de los bytes pasa algo muy parecido que con los paquetes, en donde hay una gran numero de flujos que contienen pocos bytes, en cambio la gráfica baja drásticamente en cuanto los bytes/flujo aumentan, con lo que hay pocos flujos que contengan muchos bytes. La media es de 68021,98 bytes/flujo y la desviación estándar es 107493,26 bytes/flujo. Los extremos son, un flujo máximo con 502505 bytes y un mínimo del orden de 380 flujos con 66 bytes.

### 7.3.2. Traza 1 millón

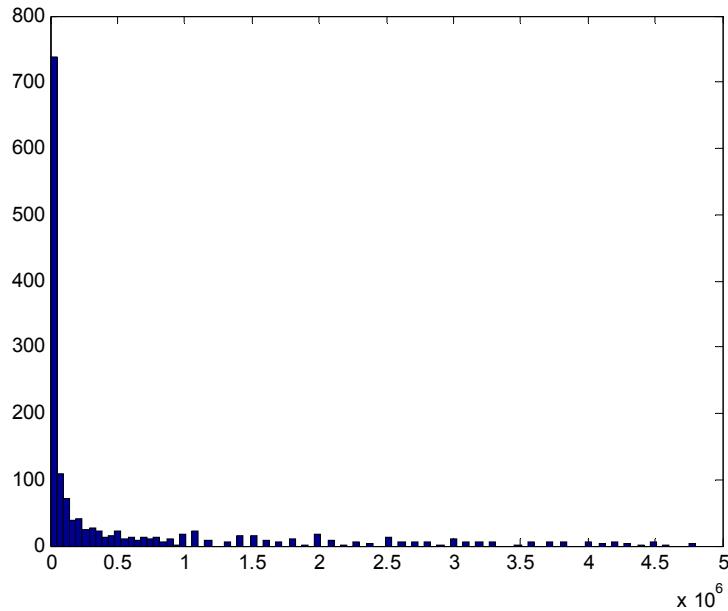
El histograma de paquetes por flujo queda de la siguiente manera:



**Fig. 7.3** Histograma de paquetes/flujo para la traza de un millón de paquetes

En la Fig. 7.3 podemos observar la cantidad de paquetes que contienen los flujos. Obteniendo un gran volumen de flujos con pocos paquetes por flujo. La media es de 683,06 paquetes/flujo y la desviación estándar 1305,04 paquetes/flujo. Los extremos son, un flujo máximo con 6465 paquetes y un mínimo de muchos flujos con un paquete.

El histograma de bytes por flujo queda de la siguiente manera:



**Fig. 7.4** Histograma de bytes/flujo, para la traza de un millón de paquetes

En la Fig. 7.4 se puede apreciar como en el caso de los bytes pasa algo muy parecido que con los paquetes, en donde hay una gran numero de flujos que contienen pocos bytes, en cambio la gráfica baja drásticamente en cuanto los bytes/flujo aumentan, con lo que hay pocos flujos que contengan muchos bytes. La media es de 510494,56 bytes/flujo y la desviación estándar es 975948,40 bytes/flujo. Los extremos son, varios flujos máximos con 4800000 bytes y un mínimo de un flujo con 85 bytes.

## CAPÍTULO 8. EVALUACIÓN DE SAMPLED NETFLOW

En este capítulo realizaremos una batería de pruebas en el que repetiremos cien veces la transmisión de la traza de  $10^5$  paquetes con cuatro diferentes probabilidades de detección. De este modo, veremos los errores que proporciona Netflow frente a múltiples experimentos. Por otro lado, analizaremos los métodos de obtención, extracción y tratado de datos para un posterior graficado.

### 8.1. Introducción

Se ha detectado que en capturas de tráfico real la distribución del tráfico suele tener una clara diferenciación entre flujos grandes (elefantes) y flujos pequeños (ratones) [15]. Es decir, un número pequeño de flujos elefante contribuyen a una gran cantidad de tráfico, mientras que el resto se consideran flujos ratón ya que suelen aparecer con mayor frecuencia pero con menor cantidad de tráfico. Esto supone un reto para los sistemas de monitorización basados en muestreo de paquetes, ya que los flujos pequeños pueden no ser detectados, y en caso de serlo, la estimación de su volumen es imprecisa. Cuanto más baja es la frecuencia de muestreo, peor fiabilidad tendrán los resultados obtenidos.

Una posible solución es monitorizar los flujos mediante el muestreo de bytes y no de paquetes, tal como se presenta en [12]. Esto conllevaría una modificación del funcionamiento de Netflow, ya que tal como está implementado no funciona así.

Un sistema de muestreo que ignore el tamaño de los paquetes puede afectar a la exactitud de las mediciones. Para ver esto, usamos la relación del error como una medida de precisión, que se define como la raíz cuadrada del error cuadrático medio (MSE) sobre el verdadero tamaño de un flujo. Definimos  $p$  ( $0,1, 0,01\dots$ ) como la probabilidad de muestreo,  $v$  el volumen por flujo,  $n$  la cantidad de paquetes por flujo y  $s$  como un tamaño fijo de paquete. En estas condiciones, se cumple que  $v=ns$ . Netflow muestrea cada paquete con probabilidad  $p$ , lo que lleva a que el número de paquetes muestreados  $n'$  sigue una distribución binomial y,  $n_{est}$  puede estimarse sin sesgo como  $n'/p$ . De la misma manera, y dado que asumimos un tamaño de paquete fijo,  $v_{est}$  puede estimarse sin sesgo como  $n_{est}s$ . En las condiciones anteriores, en [12] se demuestra que, el error relativo de la estimación del número de paquetes de los flujos y del volumen (en bytes) de los mismos siguen las expresiones:

$$\varepsilon = \sqrt{\frac{1-p}{pn}} = \sqrt{\frac{1-p}{pv}} \sqrt{s} \quad (8.1)$$

Como se puede observar, un tamaño grande de paquete (una  $s$  grande) provoca más error. Intuitivamente lo que sucede es que si fijamos el volumen (por ejemplo  $10^6$  bytes) y comparamos dos tamaños de paquetes diferentes (100 bytes y 1000 bytes), obtenemos una cantidad de paquetes diferente (10000 y 1000 paquetes respectivamente), y el error relativo del muestreo es mayor en el caso de tener pocos paquetes.

A partir de (8.1) y suponiendo que el tamaño de paquete es variable, el error relativo depende de la distribución de los tamaños de paquete de cada flujo. En [12] se demuestra que en este caso el error de cada flujo  $i$  sigue la expresión:

$$\varepsilon_i = \sqrt{\frac{1-p}{pn_i}} \sqrt{c_i^2 + 1} = \sqrt{\frac{1-p}{pn_i}} \sqrt{\frac{\sigma_i^2}{m_i^2} + 1} \quad (8.2)$$

donde  $p$  es la probabilidad de muestreo,  $n_i$  es la cantidad de paquetes por flujo,  $\sigma_i$  la desviación estándar del tamaño de los paquetes de cada flujo  $i$  y  $m_i$  la media de los tamaños de paquete de cada flujo  $i$ .

## 8.2. Descripción de las pruebas

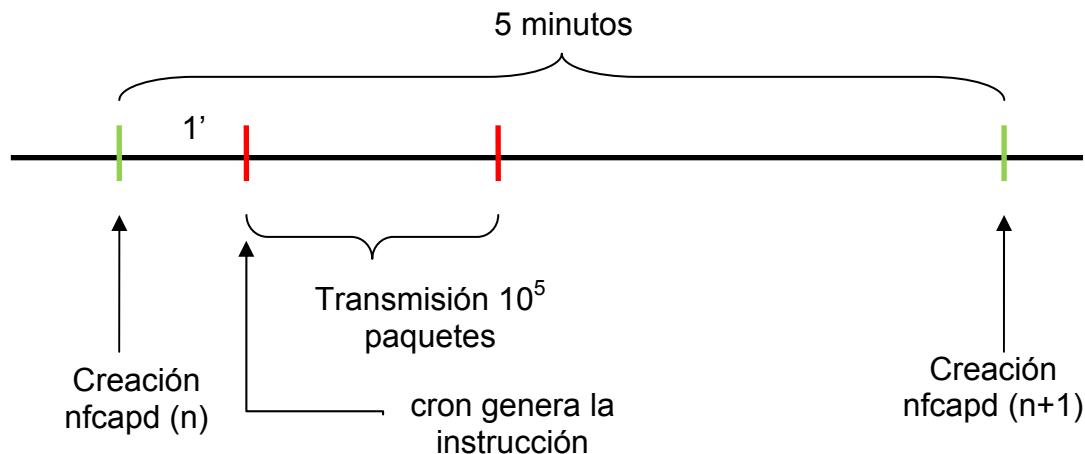
A continuación describiremos las pruebas y los procedimientos realizados para verificar el comportamiento de Netflow. El objetivo es repetir cien veces la transmisión de la traza de  $10^5$  paquetes para cada probabilidad de muestreo y obtener las estimaciones de número de paquetes por flujo y volumen por flujo, calcular los errores asociados comparando la cantidad de conversaciones (flujos en los que coinciden puerto origen, destino e IP) originales con las recibidas en las cien pruebas, y graficar los errores en diagramas de dispersión.

Como se comentó en el apartado [7.2](#), estas pruebas se han realizado en base a la traza de  $10^5$  paquetes cuya longitud sigue una ley cíclica y una distribución normal de puertos con desviación estándar 5. La transmisión de cada prueba dura aproximadamente unos 76,33 segundos y estará asociada a un archivo *nfcapd* por separado. A los ficheros *nfcapd* se le deberán extraer la máxima información posible mediante la herramienta *nfdump* y posteriormente compararlos con la traza original de  $10^5$  paquetes para obtener un cotejo entre flujos existentes y recibidos. Finalmente se realizarán cálculos para poder graficar los diagramas de dispersión.

## 8.3. Automatización con cron

Nfsen crea cada cinco minutos un archivo *nfcapd* con el registro de todos los flujos que ha detectado y enviado Netflow en los anteriores cinco minutos. Al tener una transmisión de 76,33 segundos, cada muestra de  $10^5$  paquetes estará comprendida dentro de cada archivo *nfcapd* generado. Con este

escenario, cron tiene un papel muy importante a la hora de enviar la instrucción, y la hará un minuto después de la creación del archivo *nfcapd* para que tenga suficiente margen, tanto para el envío del tráfico como para el timeout de NfSen. Veámoslo mejor en la Fig. 8.1:



**Fig. 8.1** Funcionamiento temporal del script

Al ser una prueba con muchas muestras, la automatización y la sincronización es clave. Por ello se ha utilizado la herramienta *cron* para que el PC ejecute la instrucción de emisión del tráfico (ejecución del script *100k.sh*) en el momento adecuado. Podemos ver los comandos, el script y los intervalos de emisión en el [Anexo XI](#). Así, obtendremos para cada tipo de muestreo 100 archivos *nfcapd* que trataremos en el siguiente apartado.

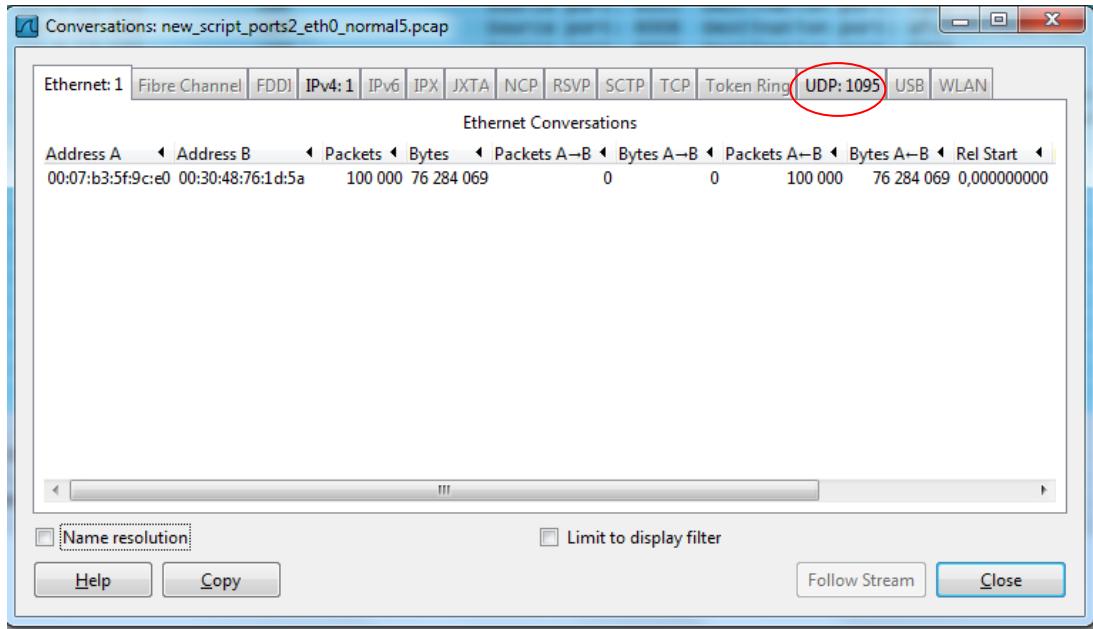
#### 8.4. Procesado de archivos *nfcapd*

Para cada tipo de muestreo se generarán 100 archivos *nfcapd* con los registros Netflow de cada una de las transmisiones de  $10^5$  paquetes. Al ser archivos binarios, se deben tratar con la herramienta *nfldump* para obtener resultados entendibles.

Nuestro objetivo es tener todas las conversations filtradas de las diferentes transmisiones realizadas, con lo que usaremos *nfldump* para filtrar el contenido. Se hará agregación de puertos origen y destino, en formato extended, es decir, con toda la información posible.

Para poder trabajar más a fondo y averiguar el comportamiento de los experimentos realizados se van a comparar los datos extraídos de las pruebas con los de la traza original.

Como se puede apreciar en la Fig. 8.2, la traza original (*new\_script\_ports2\_eth0\_normal15.pcap*) tiene 1095 conversaciones a nivel UDP; es decir, 1095 flujos. Recordemos que la manera de crear los flujos fue mediante la combinación aleatoria de puertos UDP.



**Fig. 8.2** Conversations en Wireshark

Finalmente sólo falta extraer con `nfdump` la información de los archivos `nfcapd`, por ello automatizaremos este trabajo con un script llamado `extended.sh` que generará 100 archivos txt con la información entendible y ordenada. Podemos ver el script en el [Anexo XII](#).

## 8.5. Cotejo de pruebas

Como se ha comentado en el apartado anterior, los flujos están filtrados y agrupados por puerto origen y destino e IP. El objetivo es buscar y comparar los flujos que tengan el mismo puerto origen y destino de la traza original con los de las diferentes 100 pruebas de cada tipo de muestreo. Por ello tendremos que realizar cuatro tipos de cotejos con 100 comprobaciones cada una:

- Traza original con muestreo 1/10
- Traza original con muestreo 1/100
- Traza original con muestreo 1/1000
- Traza original con muestreo 1/65535

Por ejemplo, en el caso de muestreo 1/10, si tenemos en la traza original un flujo con puerto origen 6006 y puerto destino 6996 debemos comprobar en cada una de las 100 pruebas si existe ese flujo. En caso afirmativo cogeremos tanto el valor de los bytes como de los paquetes y los copiaremos en dos archivos diferentes, en caso negativo escribiremos una X. Finalmente tendremos dos ficheros (`packets.txt` y `bytes.txt`) con el resultado de dos matrices con esta forma:

Número de experimentos (100 columnas) →

↓ Número de flujo (1095 filas)

|    |    |    |    |    |    |     |
|----|----|----|----|----|----|-----|
| 12 | 68 | 45 | 45 | 58 | 19 | 100 |
| 25 | 68 | 65 | 36 | 85 | 77 | 85  |
| 38 | 56 | X  | 20 | 48 | X  | 65  |
| 98 | 19 | 51 | 85 | 84 | 14 | 21  |
| 65 | X  | 21 | 74 | 49 | 62 | 84  |
| 47 | 65 | 10 | X  | 52 | 50 | 96  |
| 53 | X  | 28 | 88 | 57 | X  | 14  |
| .. | .. | .. | .. | .. | .. | ..  |

**Fig. 8.3 Ejemplo de archivo bytes.txt**

Para ello se ha desarrollado una aplicación en java llamada *inicio.java* para automatizar todo el proceso. De esta manera tendremos por separado en dos archivos los bytes y los paquetes de las 100 pruebas para cada tipo de muestreo, con un total de 8 archivos. Podemos ver el código fuente de la aplicación en el [Anexo XIII](#).

## 8.6. Tratado de matrices y graficado

A continuación realizaremos los cálculos necesarios para poder obtener los resultados de los experimentos. Estos cálculos se realizarán operando con las diferentes matrices obtenidas en el apartado [8.5](#) y la traza original.

En primer caso, para cada flujo encontrado en cada experimento, se calcula el error relativo corregido, es decir, para un flujo determinado se va calcular el error en función del valor original de la traza y la probabilidad de muestreo utilizada, para cada una de las veces que se ha detectado. En (8.3) podemos ver el ejemplo del error relativo del flujo uno y el experimento 1.

$$\varepsilon_{flujo}^{exp} = \varepsilon_1^1 = \left| \frac{\frac{V_1}{p} - V_1^1}{\frac{V_1}{p}} \right| \quad (8.3)$$

Donde  $V_1$  es el valor del flujo 1 de la traza original,  $p$  (10,100...) la probabilidad de muestreo usada y  $V_1^1$  el valor del flujo 1 del experimento 1. Este cálculo hay que realizarlo para todas las filas de la matriz cambiando el valor de los experimentos, es decir, para el mismo valor original ( $V_1$ ) hay que calcular el error relativo de todos los experimentos ( $V_1^1, V_1^2, V_1^3, V_1^4, V_1^5, \dots, V_1^{100}$ ):

Nos quedará una matriz como la de la Figura 8.4:

|                        |                        |                        |                        |                        |                        |                        |    |                            |
|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|----|----------------------------|
| $\varepsilon_1^1$      | $\varepsilon_1^2$      | $\varepsilon_1^3$      | $\varepsilon_1^4$      | $\varepsilon_1^5$      | $\varepsilon_1^6$      | $\varepsilon_1^7$      | .. | $\varepsilon_1^{100}$      |
| $\varepsilon_2^1$      | $\varepsilon_2^2$      | $\varepsilon_2^3$      | $\varepsilon_2^4$      | $\varepsilon_2^5$      | $\varepsilon_2^6$      | $\varepsilon_2^7$      | .. | $\varepsilon_2^{100}$      |
| $\varepsilon_3^1$      | $\varepsilon_3^2$      | $\varepsilon_3^3$      | $\varepsilon_3^4$      | $\varepsilon_3^5$      | $\varepsilon_3^6$      | $\varepsilon_3^7$      | .. | $\varepsilon_3^{100}$      |
| $\varepsilon_4^1$      | $\varepsilon_4^2$      | $\varepsilon_4^3$      | $\varepsilon_4^4$      | $\varepsilon_4^5$      | $\varepsilon_4^6$      | $\varepsilon_4^7$      | .. | $\varepsilon_4^{100}$      |
| $\varepsilon_5^1$      | $\varepsilon_5^2$      | $\varepsilon_5^3$      | $\varepsilon_5^4$      | $\varepsilon_5^5$      | $\varepsilon_5^6$      | $\varepsilon_5^7$      | .. | $\varepsilon_5^{100}$      |
| $\varepsilon_6^1$      | $\varepsilon_6^2$      | $\varepsilon_6^3$      | $\varepsilon_6^4$      | $\varepsilon_6^5$      | $\varepsilon_6^6$      | $\varepsilon_6^7$      | .. | $\varepsilon_6^{100}$      |
| ..                     | ..                     | ..                     | ..                     | ..                     | ..                     | ..                     | .. | ..                         |
| $\varepsilon_{1095}^1$ | $\varepsilon_{1095}^2$ | $\varepsilon_{1095}^3$ | $\varepsilon_{1095}^4$ | $\varepsilon_{1095}^5$ | $\varepsilon_{1095}^6$ | $\varepsilon_{1095}^7$ | .. | $\varepsilon_{1095}^{100}$ |

**Fig. 8.4** Ejemplo de la matriz resultante en Matlab

Posteriormente se calcula la media de todos los errores relativos de cada flujo respecto al número de veces que se ha detectado cada flujo.

$$\overline{\varepsilon_{\text{flujo}}} = \bar{\varepsilon}_1 = \frac{\sum_{i=1}^{100} \varepsilon_1^i}{\# \text{veces detectado}} \quad (8.4)$$

En (8.4) podemos ver el ejemplo para la media de errores del flujo uno. Para cada fila (flujo) operaremos de la misma manera. Así obtendremos una matriz con 1095 medias de errores, una para cada flujo.

Seguidamente, se debe calcular el coeficiente de variación de cada uno de los flujos ( $c^2_i$ ) de la traza original. Para ello, es necesario calcular el cuadrado de la longitud media de paquetes ( $m^2$ ) y la desviación estándar de cada flujo ( $\sigma^2$ ), tal y como se muestra en (8.5).

$$c_i^2 = \frac{\sigma_i^2}{m_i^2} \quad (8.5)$$

Para obtener estos datos ha sido necesaria la aplicación tcpstat [14] y un posterior script para extraer en un archivo txt dos columnas con todas las medias y deviaciones de todos los flujos. De este modo, podremos visualizar diagramas de dispersión donde se representará el error relativo de cada flujo en función de su volumen en bytes y de su número de paquetes, y compararlos con la expresión teórica (8.2).

Todos estos cálculos pueden hacerse y automatizarse con Matlab, en el [Anexo XIV](#) se detallan todos los comandos necesarios para crear las matrices de errores y el posterior graficado.

## 8.7. Diagramas de dispersión

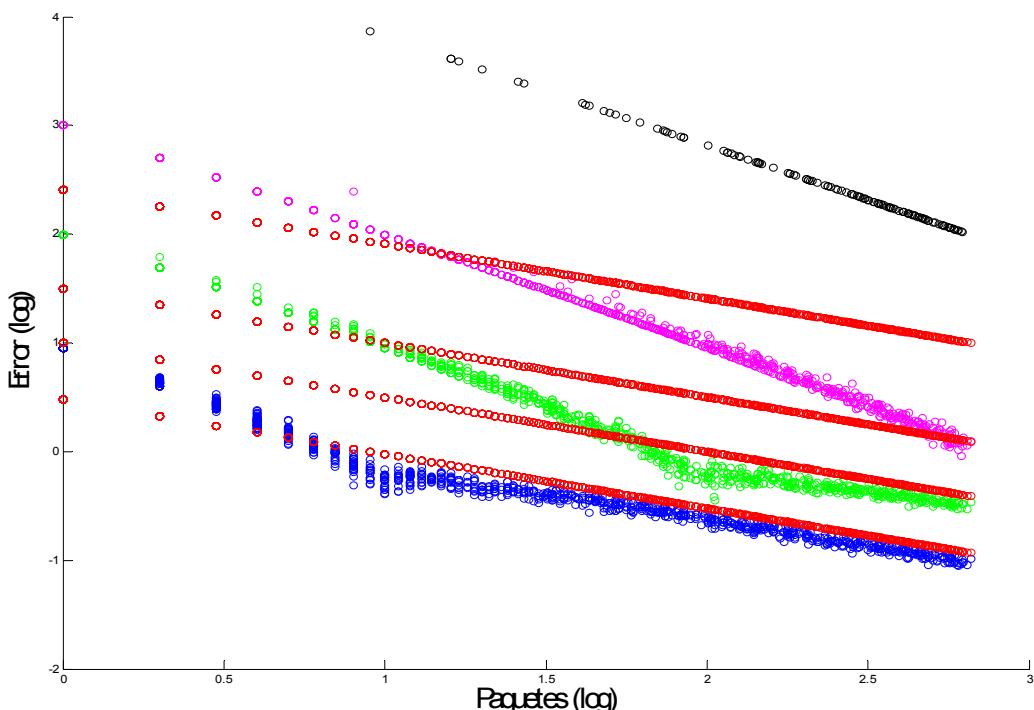
Los diagramas de dispersión representan el error relativo cometido en la estimación del volumen y del número de paquetes, para cada flujo. Conviene hacer una representación log-log, ya que aplicando logaritmo a las expresiones (8.1) y (8.2) obtenemos:

$$\log(\varepsilon) = \log \left[ \sqrt{\frac{1-p}{p}} \right] - \frac{1}{2} \log(n) \quad (8.6)$$

$$\log(\varepsilon_i) = \log \left[ \sqrt{\frac{1-p}{p}} \sqrt{c_i^2 + 1} \right] - \frac{1}{2} \log(n_i) \quad (8.7)$$

Y teniendo en cuenta que, para cada experimento, el valor de  $p$  es constante, aparecerá una recta de pendiente -0.5 cuando representemos el logaritmo del error en función del logaritmo del número de paquetes o del logaritmo del volumen.

En el siguiente diagrama podemos ver los cuatro muestreos realizados para los paquetes:



**Fig. 8.5** Diagrama de dispersión del error respecto al número de paquetes

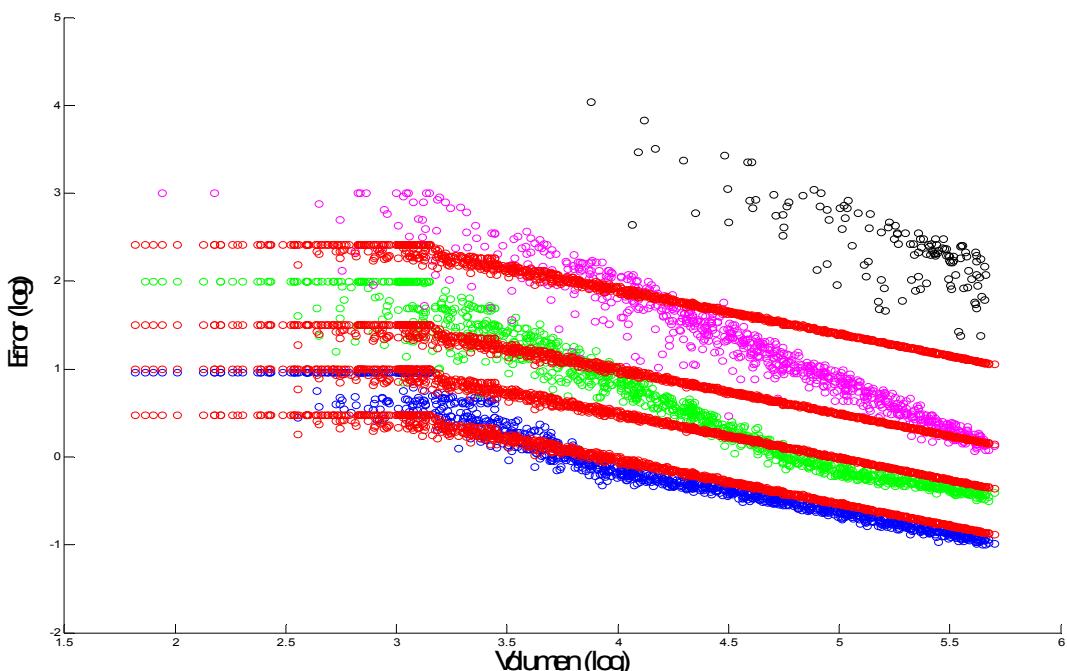
Los valores azules representan la probabilidad de muestro de 1 de cada 10, los verdes, 1 de cada 100, los rosas 1 de cada 1000 y los negros 1 de cada 65535. Los valores en rojo son los resultados teóricos obtenidos con (8.2).

En la Fig. 8.5 en que aparecen el 100% de los flujos detectados, podemos observar como a mayor concentración de paquetes por flujo, menor es el error relativo, como es lógico (mas paquetes implica mayor volumen, y por tanto menor error en la estimación basada en muestreo).

Las dos probabilidades más altas muestran un error obtenido inferior al teórico y su pendiente es acorde con la teoría, mientras que para las probabilidades más bajas el error teórico es más bajo que el obtenido y no presentan la pendiente esperada. Creemos que la explicación es que el “codo” que se observa para las probabilidades de 1/10 y 1/100 ha quedado fuera de la zona representada para los casos 1/1000 y 1/65535. Nuestras conclusiones son que la cantidad de muestras utilizadas en el experimento son insuficientes (en comparación con [12], en donde las trazas alcanzan los 10 GBytes) para analizar las dos probabilidades más bajas. De todas maneras, hay que destacar el correcto alineamiento para las probabilidades altas.

Por otra parte podemos ver poca dispersión, tal como esperábamos (recordemos que la ecuación (8.1) nos demuestra que la estimación del número de paquetes siempre tiene menos error relativo que la estimación de volúmenes).

En el siguiente diagrama podemos ver los cuatro muestreos realizados para el volumen en bytes:



**Fig. 8.6** Diagrama de dispersión del error respecto al volumen (en bytes)

En este caso podemos observar cómo se vuelve a verificar que a mayor cantidad de bytes (volumen) por flujo menor es el error. Contrariamente a la Fig. 8.5 se puede apreciar como la dispersión es mucho mayor en el caso de los bytes ya que el error relativo depende de la distribución de los tamaños de paquete de cada flujo.

Al igual que sucedía con los paquetes, los resultados vuelven a estar alineados con la teoría para las dos probabilidades de muestreo más altas, y no tanto en las más bajas, ya que volvemos a tener el fenómeno del codo. Por otro lado, cada una de ellas tiene una constante de intersección con el eje de las ordenadas diferente, lo que significa que a menor probabilidad, más error, como era de esperar (el volumen estimado es menor, y por tanto el error relativo aumenta).

Cabe destacar que en las dos probabilidades más altas, se observan muchos errores con el mismo valor. Eso se debe a que en el momento de muestreo se ha obtenido el mismo valor estimado para todas las veces que se ha detectado el flujo.

Los cálculos y los comandos necesarios para llevar a cabo los diagramas los podemos encontrar en el [Anexo XV](#).

## CAPÍTULO 9. CONCLUSIONES Y LÍNEAS FUTURAS

En este proyecto hemos podido configurar y montar un laboratorio de pruebas de altas prestaciones, con dispositivos de red de alta gama como un router Cisco 3700, un switch SMC Tigerswitch con conexiones a 1Gbps y fibra óptica, una tarjeta DAG de alta precisión y dos ordenadores con gran respuesta computacional.

Se ha presentado una comparativa de varios generadores de flujo, entre ellos Iperf, MGGEN y daggen junto con tcpreplay y GenCBR. Con Iperf se ha concluido que es un buen generador de tráfico con ciertas limitaciones e imperfecciones. En comparación con MGGEN, éste puede configurarse con diferentes patrones de tráfico e incluso la cantidad de paquetes por segundo que desee.

Daggen junto con tcpreplay nos ha ofrecido la posibilidad de generar trazas a nuestro gusto y con diferentes patrones pero con limitaciones muy estrictas en cuanto a programación. Gracias a él hemos podido estudiar una batería de pruebas con una traza de gran magnitud y ver los resultados que Netflow auditaba. Esto nos permite analizar el comportamiento de este software.

Por otro lado, nos hemos visto envueltos en varios problemas con la emisión con la DAG, pero no con la recepción, en la que su precisión es muy buena. Debido a su modo de funcionamiento, el marcado temporal de los paquetes se realiza de forma precisa, sin afectar a las características del tráfico que capturamos. Por ello, podemos llegar a la conclusión que para llevar a cabo una monitorización precisa es mejor utilizar esta solución.

La configuración de todo el escenario ha sido un objetivo primario y secundado con solvencia. Aunque el intento de replicación de la traza de CESCA no fue satisfactoria, otras trazas generadas por la DAG sí han podido ser estudiadas. Sería necesario estudiar por qué se producen estos errores localizados.

Se ha realizado una serie de experimentos para validar unas expresiones teóricas respecto al error relativo de la estimación del número de paquetes y del volumen de los flujos. Por lo que sabemos, es la primera vez que se realiza un estudio experimental de este tipo. Los resultados finales obtenidos para el volumen son válidos, ya que para las probabilidades más altas obtenemos unas medidas alineadas con la teoría. Para probabilidades más bajas los resultados no son satisfactorios. Estas discordancias son debidas a que en las pruebas realizadas se basan en una traza de  $10^5$  paquetes, una cantidad correcta pero poco comparable con las grandes trazas utilizadas en [12]. En el caso de los resultados de los paquetes, hemos visto como la precisión es mucho mayor, coincidiendo con el muestreo realizado por Netflow basado en paquetes y no bytes.

Podemos llegar a la conclusión de que Netflow es un buen muestreador de paquetes, pero hay que llegar a un compromiso con la calidad de los resultados que se requiera ya que dependiendo de la probabilidad es más preciso o no.

Como líneas futuras, sería interesante poder adaptar grandes trazas como las de CESCA, NZX-II y otros operadores, al laboratorio montado, así poder inspeccionar y tratar los datos frente a cualquier modo de configuración de Netflow.

Los resultados obtenidos en este TFC no presentan en sí mismos una amenaza para el medioambiente. Por otra parte las máquinas necesarias para llevar a cabo una correcta monitorización de tráfico suponen un consumo energético, teniendo en cuenta que en el proceso pueden intervenir múltiples dispositivos (PC, switches, routers, pantallas...) que, además, suelen estar ubicados en salas que necesitan sistemas de refrigeración este consumo puede ser muy elevado. Por lo tanto habría que tomar medidas de obtención de energía limpias como puede ser la solar para alimentar los dispositivos menos potentes y sistemas de ahorro de energía para los que más consumen. Además habría que regular las emisiones de los sistemas de refrigeración mediante catalizadores u otros dispositivos reguladores.

## GLOSARIO

|              |  |
|--------------|--|
| <b>ARP</b>   | <b>Address Resolution Protocol</b>                       |
| <b>ATM</b>   | <b>Asynchronous Transfer Mode</b>                        |
| <b>BDD</b>   | <b>Base de Datos</b>                                     |
| <b>CEF</b>   | <b>Cisco Express Forwarding</b>                          |
| <b>CPU</b>   | <b>Central Processing Unit</b>                           |
| <b>CRC</b>   | <b>Cyclic Redundancy Check</b>                           |
| <b>CSV</b>   | <b>Comma-Separated Values</b>                            |
| <b>DAG</b>   | <b>Data Acquisition and Generation</b>                   |
| <b>DOS</b>   | <b>Denial Of Service</b>                                 |
| <b>ERF</b>   | <b>Extensible Record Format</b>                          |
| <b>FCS</b>   | <b>Frame Check Sequence</b>                              |
| <b>FIFO</b>  | <b>First In First Out</b>                                |
| <b>IEEE</b>  | <b>Institute of Electrical and Electronics Engineers</b> |
| <b>IOS</b>   | <b>Internetwork Operating System</b>                     |
| <b>IP</b>    | <b>Internet Protocol</b>                                 |
| <b>ISP</b>   | <b>Internet Service Provider</b>                         |
| <b>L/T</b>   | <b>Length/Type</b>                                       |
| <b>MAC</b>   | <b>Media Access Control</b>                              |
| <b>MSE</b>   | <b>Mean Square Error</b>                                 |
| <b>MTU</b>   | <b>Maximum Transmission Unit</b>                         |
| <b>NAT</b>   | <b>Network Address Translation</b>                       |
| <b>NIC</b>   | <b>Network Interface Controller</b>                      |
| <b>NTP</b>   | <b>Network Time Protocol</b>                             |
| <b>NVRAM</b> | <b>Non-Volatile Random Access Memory</b>                 |
| <b>PCI-X</b> | <b>Peripheral Component Interconnect Express</b>         |
| <b>SDRAM</b> | <b>Synchronous Dynamic Random Access Memory</b>          |
| <b>SFP</b>   | <b>Small Form-factor Pluggable</b>                       |
| <b>TAR</b>   | <b>Tape ARchiver</b>                                     |
| <b>TCP</b>   | <b>Transmission Control Protocol</b>                     |
| <b>TOS</b>   | <b>Type Of Service</b>                                   |
| <b>UDP</b>   | <b>User Datagram Protocol</b>                            |
| <b>WIC</b>   | <b>Wan Interface Card</b>                                |

## BIBLIOGRAFÍA

- [1] Cisco. *Netflow Services Solutions Guide*. , 22-01-2007. Available from: <[http://www.cisco.com/en/US/docs/ios/solutions\\_docs/netflow/nfwhite.html](http://www.cisco.com/en/US/docs/ios/solutions_docs/netflow/nfwhite.html)>.
- [2] Networks. *Configuring J-Flow Statistics*. Available from: <<http://www.juniper.net/techpubs/software/erx/junose82/swconfig-ip-services/html/ip-iflow-stats-config2.html>>.
- [3] Gray. *GNU Cflow Manual*. , 11-10-2011. Available from: <<http://www.gnu.org/software/cflow/manual/index.html>>.
- [4] H3C. *NetStream Configuration*. Available from: <[http://www.h3c.com/portal/Technical\\_Support/Documents/Technical\\_Documents/Security\\_Products/H3C\\_SecBlade\\_NetStream\\_Card/Configuration/User\\_Manual/H3C\\_SecBlade\\_NetStream\\_Card\\_Overview-6PW103/02-NetStream/201006/678512\\_1285\\_0.htm](http://www.h3c.com/portal/Technical_Support/Documents/Technical_Documents/Security_Products/H3C_SecBlade_NetStream_Card/Configuration/User_Manual/H3C_SecBlade_NetStream_Card_Overview-6PW103/02-NetStream/201006/678512_1285_0.htm)>.
- [5] Endace. *Endace Web Page*. Available from: <<http://www.endace.com>>.
- [6] PÉREZ RODRÍGUEZ, Iván. Analizador/generador Gigabit Ethernet De Altas Prestaciones. , 07-07-2006. Available from: <<http://upcommons.upc.edu/pfc/handle/2099.1/3759>>.
- [7] SOLER CAMÍ, Jordi. *Experimentación Con Relojes De Alta Precisión Para La Monitorización De Tráfico En Redes De Ordenadores*. , 07-05-2012. Available from: <<http://upcommons.upc.edu/pfc/handle/2099.1/15221>>.
- [8] Wikipedia. *Iperf*. Available from: <<http://en.wikipedia.org/wiki/Iperf>>.
- [9] Naval Research Laboratory. *MGEN*. Available from: <<http://pf.itd.nrl.navy.mil/mgen/mgen.html>>.
- [10] Trac. *TCPReplay*. Available from: <<http://tcpreplay.synfin.net/wiki/>>.
- [11] Cesca. *CESCA*. Available from: <<http://www.cesca.cat>>.
- [12] RASPALL CHAURE, Frederic. Efficient Packet Sampling for Accurate Traffic Measurements. , 19-04-2012. Available from: <[http://ac.els-cdn.com/S1389128611004142/1-s2.0-S1389128611004142-main.pdf?\\_tid=c9f55490-07fc-11e2-85d7-0000aacb362&acdnat=1348679487\\_ab40dddb66c60ed58e25e4dd437bc1b8](http://ac.els-cdn.com/S1389128611004142/1-s2.0-S1389128611004142-main.pdf?_tid=c9f55490-07fc-11e2-85d7-0000aacb362&acdnat=1348679487_ab40dddb66c60ed58e25e4dd437bc1b8)>
- [13] Caida. Netflow Export Headers. , 30-09-1999. Available from: <<http://www.caida.org/tools/measurement/cflowd/configuration/configuration-9.html>>.
- [14] Herman, Paul. Tcpstat, 16-06-2009. Available from: <<http://www.frenchfries.net/paul/tcpstat/>>.

[15] Yobot. Elephant flow, 5-12-2012. Available from: <[http://en.wikipedia.org/wiki/Elephant\\_flow](http://en.wikipedia.org/wiki/Elephant_flow)>.

## Bibliografía de interés

<http://linux.byexamples.com/archives/283/simple-usage-of-tcpdump/>

<http://www.thegeekstuff.com/2010/08/tcpdump-command-examples/>

[http://www.cisco.com/en/US/docs/ios/netflow/configuration/guide/12\\_4/nf\\_12\\_4\\_book.html](http://www.cisco.com/en/US/docs/ios/netflow/configuration/guide/12_4/nf_12_4_book.html)

<http://www.matrixlab-examples.com/hist.html>

[http://www.cisco.com/en/US/docs/ios/12\\_0s/feature/guide/nfstatsa.html](http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/nfstatsa.html)

<http://etutorials.org/Networking/network+management/Part+II+Implementations+on+the+Cisco+Devices/Chapter+7.+NetFlow/Supported+Devices+and+IOS+Versions/>

[http://www.cisco.com/en/US/docs/ios/12\\_0s/feature/guide/12s\\_sanf.html](http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html)

<http://librosnetworking.blogspot.com.es/2007/01/introduccin-cef.html>

[http://www.cisco.com/en/US/docs/ios/12\\_2/switch/configuration/guide/xcfnfov.html](http://www.cisco.com/en/US/docs/ios/12_2/switch/configuration/guide/xcfnfov.html)

[http://netflow.caligare.com/netflow\\_v5.htm](http://netflow.caligare.com/netflow_v5.htm)

<http://escuela.med.puc.cl/recursos/recepidem/EPIANAL9.HTM>

[http://es.wikipedia.org/wiki/Intervalo\\_de\\_confianza](http://es.wikipedia.org/wiki/Intervalo_de_confianza)

<http://www.mathworks.es/help/techdoc/ref/hist.html>

<http://www.eyeintheskygroup.com/Azar-Ciencia/Analisis-Estadistico-Juegos-de-Azar/Desviacion-Estandar-Intervalo-de-Confianza.htm>

<http://www.vadenumeros.es/sociales/estimacion-muestras.htm>

<http://www.ual.es/~aposadas/TeoriaErrores.pdf>

<http://www.pucpr.edu/facultad/ejaviles/ED%20800%20PDF%20Files/ED%20800%20Estimados%20y%20Tama%F1os%20de%20Muestras.pdf>

<https://wiki.man.poznan.pl/Personar-mdm/index.php/Dag.sh>

[http://www.bow.co.kr/filedata/Endace/dag-3.1.0\\_Release\\_Notes.pdf](http://www.bow.co.kr/filedata/Endace/dag-3.1.0_Release_Notes.pdf)

<http://www.bow.co.kr/filedata/Endace/>

<http://informaticaesp.forosweb.net/t194-como-hacer-un-script-consola-de-linux>

- <http://www.ubuntu-es.org/node/14511#.UBAj3mE0Pkg>
- [http://www.linuxtotal.com.mx/index.php?cont=info\\_admon\\_006](http://www.linuxtotal.com.mx/index.php?cont=info_admon_006)
- [http://es.wikipedia.org/wiki/Cron\\_\(Unix\)](http://es.wikipedia.org/wiki/Cron_(Unix))
- [http://www.nebrija.es/~abustind/tutorial\\_matlab.htm](http://www.nebrija.es/~abustind/tutorial_matlab.htm)
- <http://es.scribd.com/doc/8974427/Matlab-Intro>
- <http://hp.vector.co.jp/authors/VA014830/english/FlexRena/>
- [http://www.zyxel.com/products\\_services/gs1100\\_series.shtml?t=p](http://www.zyxel.com/products_services/gs1100_series.shtml?t=p)
- [http://www.smc.com/files/AP/MN\\_8624T\\_V2\\_INS.pdf](http://www.smc.com/files/AP/MN_8624T_V2_INS.pdf)
- <http://upcommons.upc.edu/pfc/handle/2099.1/11288> (Oscar Rodríguez)
- <http://upcommons.upc.edu/pfc/handle/2099.1/12986> (Rubén Hinojosa)
- <http://upcommons.upc.edu/pfc/handle/2099.1/10799> (Iván Minguillón)

## ANEXOS

### I. Instalación de software

Para la instalación de Nfsen contamos con una maquina Suse Linux 2.6.32 con tarjeta a 1000Mbps. La ejecución del front-end de Nfsen requiere módulos de Apache2, PHP5 y Perl. A continuación se enumeran los diferentes complementos necesarios para este software.

#### Instalación de preliminares y dependencias

Suse Linux incorpora un potente gestor de aplicaciones y dependencias para una sencilla administración del software, este es YaST2.

Dentro del gestor de aplicaciones YaST2, buscamos los siguientes paquetes y lo marcamos para que se instalen automáticamente al darle a aceptar:

- ✓ apache2
- ✓ php5
- ✓ php5-sockets
- ✓ perl
- ✓ perl-mailtools
- ✓ perl-socket6
- ✓ perl-regexp-common
- ✓ regexp
- ✓ bison
- ✓ byacc

Este software es básico para la instalación de futuros paquetes que forman las aplicaciones principales del proyecto:

- ✓ RRDtools: Es un conjunto de herramientas que nos permitirá trabajar con las bases de datos RRD.
- ✓ Nfdump: Las herramientas nfdump forman el conjunto de herramientas de back-end que emplea Nfsen. Se encargan de recoger y procesar los datos de Netflow.
- ✓ Nfsen: Front-end en formato web para la interpretación de los datos recogidos por las dos anteriores aplicaciones. Grafica e interpreta datos.

## Instalación RRDtools

La instalación de RRDtools es muy sencilla si se utiliza YaST2, simplemente hay que buscarla con el gestor e instalarla, aunque es sumamente recomendable realizarla a mano.

Para ello debemos descargar la última versión desde la página oficial:

<http://oss.oetiker.ch/rrdtool/pub/?M=D> (Descargar archivo rrdtool-X.X.X.tar.gz)

Descomprimir el archivo con: # `tar -xvf rrdtool-X.X.X.tar.gz`

Entrar en el directorio y realizar el configure:

# `./configure --prefix=/usr/local/rrdtool --disable-tcl`

Tras finalizar el configure, realizar los make's:

# `make`

# `make install`

## Instalación Nfdump

El conjunto de herramientas nfdump no es posible descargarla desde YaST2, así que habrá que realizarse manualmente:

Descargamos la última versión desde la página oficial:

<http://sourceforge.net/projects/nfdump/> (Descargar archivo nfdump-X.X.X.tar.gz)

Descomprimir el archivo con: # `tar -xvf nfdump-X.X.X.tar.gz`

Al configurar nfdump con la opción nfprofile, necesaria para instalar NfSEN, el configure espera encontrar rrdtools en /usr, con `rrd.h` en /usr/include y `librrd` en /usr/lib. El directorio donde está rrdtools se puede especificar con la opción `-enable-nfprofile--with-rrdpath=...`, pero tanto la cabecera `rrd.h` como las librerías hay que copiarlas a los directorios correspondientes, ya que si no da error:

```
# cp /usr/local/rrdtool/include/rrd.h /usr/include/  
# cp /usr/local/rrdtool/lib/librrd* /usr/lib/
```

Seguidamente ya se puede realizar el configure con todas las opciones:

```
# ./configure --enable-nfprofile --with-  
rrdpath=/usr/local/rrdtool/lib
```

Entrar en el directorio y realizar los make's:

```
# make
# make install
```

## Instalación Nfsen

Una vez logramos instalar los prerequisitos en nuestro sistema, podemos proceder con la instalación de Nfsen en el mismo.

Antes de comenzar con el proceso, tenemos que crear nuevo usuario Nfsen miembro del grupo www con directorio /usr/local/nfsen. Podemos hacerlo desde YaST2 o bien a través del siguiente comando:

```
# useradd -d /usr/local/nfsen -g www nfsen
```

Para saber si se ha creado correctamente realizamos:

```
# groups nfsen
```

El software Nfsen no es posible descargarlo desde YaST2, así que habrá que realizarse manualmente:

Descargamos la última versión desde la página oficial:

<http://sourceforge.net/projects/nfsen/> (Descargar archivo nfsen-X.X.X.tar.gz)

Descomprimir el archivo con: # tar -xvf nfsen-X.X.X.tar.gz

Al descomprimir se genera un archivo de configuración en la ubicación etc/nfsen-dist.conf. Este lleva una plantilla de configuración para modificar el valor de las variables. El archivo de configuración debe tener el nombre nfsen.conf, así que se provoca una copia del archivo actual con este nombre para configurarlo posteriormente y a la vez realizamos una copia de seguridad de la plantilla base.

Entramos en el directorio y realizamos una copia del archivo de configuración:

```
# cd nfsen-X.X.X
#cp etc/nfsen-dist.conf etc/nfsen.conf
```

Una vez tenemos la copia correcta ya se pueden modificar las variables necesarias para la correcta configuración:

```
# nano etc/nfsen.conf
```

Los valores a modificar son los siguientes:

```
$BASEDIR = "/usr/local/nfsen";
$HTMLDIR      = "/usr/local/nfsen/www/htdocs/nfsen";
```

```
$USER      = "nfsen";
$WWWUSER   = "nfsen";
$ZIPcollected      = 0;
$ZIPprofiles       = 0;
%sources = (
    'prueba1'     => { 'port' => '9910', 'col' => '#0000ff',
    'type' => 'netflow' }, );
);
```

Una vez configurado debemos asignar permisos para poder acceder a la web de Nfsen con lo que debemos agregar permisos a la configuración de Apache en `/etc/apache2/default-server.conf` con la siguiente información:

```
Alias /nfsen "/usr/local/nfsen/www/htdocs/nfsen"

<Directory "/usr/local/nfsen/www/htdocs/nfsen">
    Options None
    AllowOverride None
    Order allow,deny
    Allow from all
    AddType application/x-httpd-php .php
</Directory>
```

Una guardado el archivo, nos dirigimos a la carpeta descomprimida de nfsen y realizamos la instalación:

```
# ./install.pl etc/nfsen.conf
```

### Problemas encontrados

Si a la hora de cargar la página no se carga y pide la descarga del propio archivo `nfsen.php` hay que confirmar que los módulos `php5` y `perl` se estén cargados correctamente.

Para comprobarlo vamos a la carpeta `/etc/apache2/sysconfig.d` y editamos con `nano` el archivo `loadmodule.conf`.

Dentro de él deben estar las siguientes líneas, sino hay que ponerlas:

```
LoadModule php5_module      /usr/lib/apache2/mod_php5.so
LoadModuleperl_module      /usr/lib/apache2/mod_perl.so
```

## Actualización del software de la DAG 4.3 GE

La suite de software que lleva la DAG permite múltiples soluciones, emisores de flujo, conversores de formato, programación de trazas, etc... Estas utilidades requieren actualizaciones para ampliar sus operaciones o reparar bugs.

En nuestro caso nos ha hecho falta realizar conversiones de formato ERF a PCAP y viceversa. La versión ya instalada sólo permitía realizar conversiones de PCAP a ERF y teniendo la necesidad de realizar la operación contraria ha sido necesario actualizar el software.

La nueva actualización incorpora las siguientes mejoras:

- Actualización de firmware: Soporte a VLANs, corrección de errores, etc..
- Asignación de dirección MAC a los puertos
- Soporte hasta la versión 2.6.21 de los kernels de Linux
- Actualización y corrección de errores en las utilidades

Al instalar la nueva versión del software se generaron varios errores de compilación. A continuación se explica la instalación, sus errores generados y su solución:

Descargar archivo de: <http://www.bow.co.kr/filedata/Endace/> (sitio no oficial)

Descomprimir el archivo con: # ***tar -xvf dag-3.1.0.tar.gz***

Entrar en el directorio y realizar el configure:

***# ./configure***

Tras finalizar el configure, realizar los make's:

***# make***

***# make install***

Seguidamente al realizar un *dagload* para cargar la DAG nos generaba un error, por lo que probamos la aplicación con:

*servgenmonI:~# modprobe dag*

```
/lib/modules/2.4.27-speakup/endace/dagmem: The module you
are trying to load (/lib/modules/2.4.27-
speakup/endace/dagmem) is compiled with a gcc version 2
compiler, while the kernel you are running is compiled with
a gcc version 3 compiler. This is known to not work.
```

```
/lib/modules/2.4.27-speakup/endace/dagmem: insmod
/lib/modules/2.4.27-speakup/endace/dagmem failed
```

```
/lib/modules/2.4.27-speakup/endace/dagmem: insmod dag  
failed
```

Esto advierte de que la aplicación ha sido compilada con una versión diferente a la que está compilado el kernel.

Gracias al manual (software installation guide) que está incluido en el TAR, pudimos observar que para deshabilitar la detección de *gcc* es necesario realizar la instalación de esta manera:

Entrar en el directorio y realizar el configure:

```
# ./configure --disable-gcc-detect
```

Tras finalizar el configure, realizar los make's:

```
# make
```

```
# make install
```

De este modo los drivers de la tarjeta se instalan correctamente.

## II. Pruebas Iperf

### Emisión y recepción

Como se ha comentado anteriormente se configurarán dos equipos, uno como cliente y otro como servidor, ambos con Linux e iperf arrancado. La captura de datos se realizará mediante la herramienta tcpdump, un software libre que permite capturar todos los paquetes de la interfaz (gracias a la librería pcap) y que no exige tantos recursos de CPU como Wireshark.

Tcpdump hará el papel de recolector de paquetes en el PC servidor. La sintaxis estándar que utilizaremos para capturar en todas las pruebas será la siguiente:

```
tcpdump -i eth0 -w 64k_100MB.pcap -n -tttt UDP and port 5001
```

Donde *-i* es la interfaz que usaremos, *-w* nos indica que la salida será a un archivo pcap que nombraremos nosotros, *-n* indica la captura con direcciones IP y no hosts, *-tttt* indica que capturará los timestamps y *UDP and port* nos indica que queremos sólo capturar paquetes UDP que cumplan el puerto que elijamos nosotros.

Para emitir el tráfico utilizaremos los comandos de consola de iperf. En el siguiente ejemplo se generará un flujo de 100 Mbps y 64 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
iperf -c -u -b 100m -i 1 -l 64
```

Donde: *-c* simboliza el modo cliente, *-u* formato de flujo UDP, *-b* el ancho de banda a generar, *-i* el intervalo en segundos para escribir en pantalla los resultados y *-l* la cantidad de bytes útiles del paquete.

El servidor se configura con el siguiente comando:

```
iperf -s -u -i 1
```

Donde: `-s` simboliza el modo servidor, `-u` formato de flujo UDP y `-i` el intervalo en segundos para escribir en pantalla los resultados.

Para cada una de las pruebas habrá que cambiar los valores de `100m` y `64` dependiendo del tipo de prueba que se desee realizar. De este modo obtendremos un registro de Wireshark para cada prueba.

## Graficado con Matlab

El tratado de datos se basa en obtener el Delta Time (intervalo entre paquetes), pero no todos los datos, sino la mayoría de ellos, ya que, la primera fase de establecimiento de conexión (unos 10 paquetes).

Después de capturar los datos con tcpdump, se importan con Wireshark a formato CSV para tratarlos posteriormente con Excel y Matlab. Una vez se selecciona la columna de interés (Delta time) con Excel, se exportará y generaremos un histograma para ver el comportamiento del intervalo entre paquetes con la herramienta Matlab.

Los histogramas con Matlab se generan con el siguiente comando:

```
hist(delta(find(delta<0.00005)), 100)
```

Donde: `find(delta<0.00005)` genera un array con el número del paquete que cumple la sentencia en que delta es menor a 0.00005; `delta(find(delta<0.00005))` genera otro array con el valor del número del paquete que cumple la sentencia en que delta es menor a 0.00005 y `hist`, representa el histograma en base a los datos importados del array delta y troceado en 100 intervalos.

Para cada tipo de prueba y dependiendo de la precisión que queramos graficar, deberemos cambiar el valor umbral `0.00005` y el número de intervalos (`100`).

Por otro lado, hay que tener en cuenta que el comando anterior provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,00005 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.00005))
size(find(delta>0.00005))
```

Así podremos saber cuántos valores escogemos para graficar el histograma y cuantos estamos despreciando.

Finalmente, calcularemos la media (`mean`) y la desviación estándar (`std`) de todos los valores con los siguientes comandos:

```
mean (delta)
std (delta)
```

## Prueba a 100 Mbps y 64 bytes

Iperf permite configurar muchos parámetros para la generación de flujos. En el siguiente ejemplo se generará un flujo de 100 Mbps y 64 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
iperf -c -u -b 100m -i 1 -l 64
```

donde: `-c` simboliza el modo cliente, `-u` formato de flujo UDP, `-b` el ancho de banda a generar, `-i` el intervalo en segundos para escribir en pantalla los resultados y `-l` la cantidad de bytes útiles del paquete.

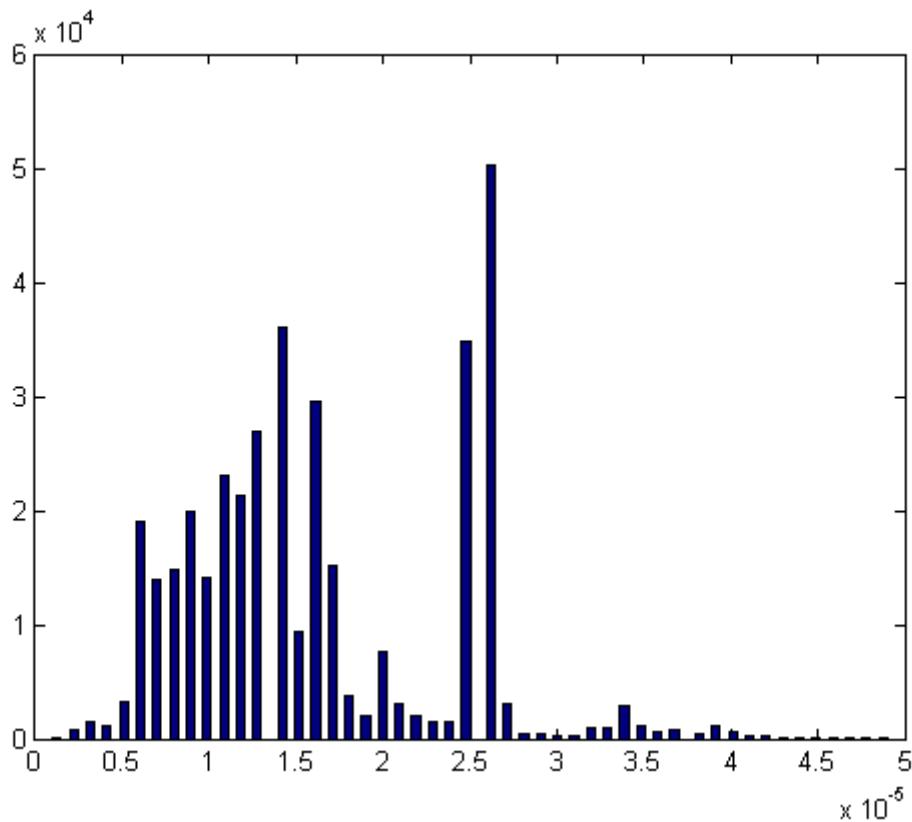
Bajo la premisa de la expresión (3.1), teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 117924,52, y el Delta time la inversa: 8,48  $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.00005)), 100)
```

donde: **find(delta<0.00005)** genera un array con el número del paquete que cumple la sentencia en que delta es menor a 0.00005; **delta(find(delta<0.00005))** genera otro array con el valor del número del paquete que cumple la sentencia en que delta es menor a 0.00005 y **hist**, representa el histograma en base a los datos importados del array delta y troceado en **100** intervalos.

El histograma para esta prueba se muestra en la figura 3.2.



**Fig. A2.1** Histograma a 100 Mbps y 64 bytes

En este gráfico se puede observar como la mayor cantidad de paquetes se concentran en 14μs, 245μs y 26μs, aunque hay dos zonas claramente diferenciadas.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,00005 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.00005))
size(find(delta>0.00005))
```

Con  $\text{delta}<0.00005$  se obtienen 372203 valores (los graficados), los descartados con  $\text{delta}>0.00005$ , y se obtienen 2465 valores.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 64 bytes a 100Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 193,96μs y la media es de 26,68μs.

En este caso, este resultado sí proporciona el mismo resultado que el report del propio programa iperf, ya que 26,68 $\mu$ s son unos 31,78 Mbps y es parecido a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 38.0 MBytes 31.9 Mbits/sec 0.001 ms 0/622226 (0%)
```

**Fig. A2.2** Resultado servidor Iperf de la prueba a 100 Mbps y 64 bytes

### Prueba a 100 Mbps y 300 bytes

En esta prueba se generará un flujo de 100 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
iperf -c -u -b 100m -i 1 -l 300
```

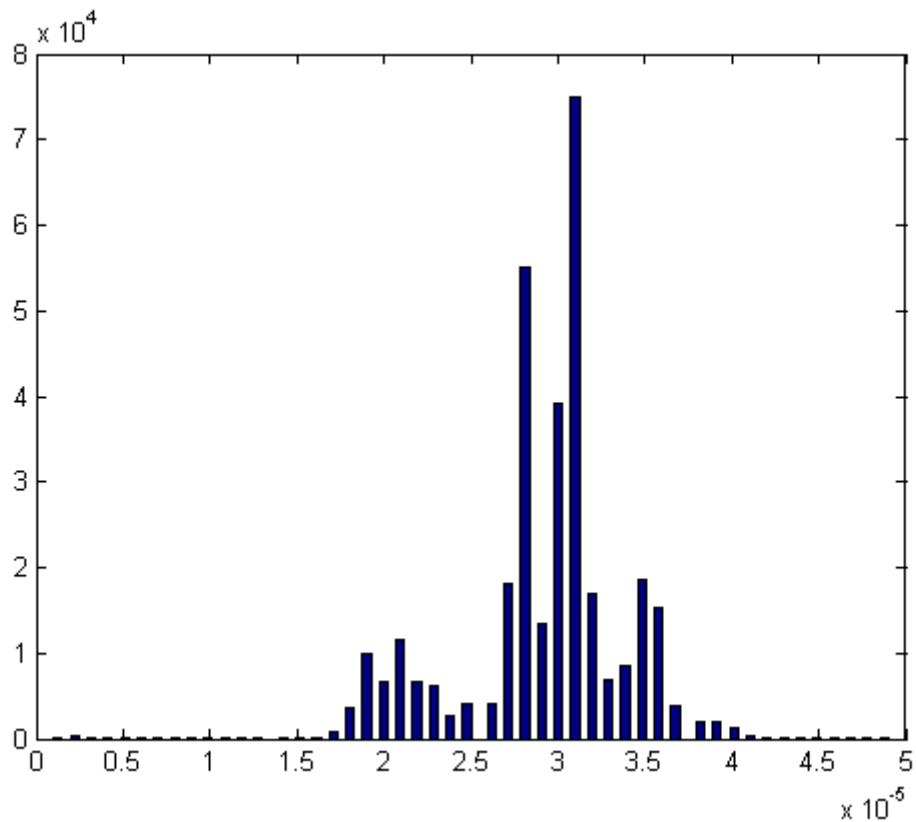
Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 36549.70, y el Delta time la inversa: 27,36  $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.00005)), 100)
```

Se utiliza 5e-5 ya que así abarcamos gran parte de los datos, tanto los superiores como los inferiores a 27,36 $\mu$ s.

El histograma es el siguiente:



**Fig. A2.3** Histograma a 100 Mbps y 300 bytes

En este gráfico se puede observar como la mayor cantidad de paquetes se concentran entre 28μs y 31μs, aunque la zona desde 15μs hasta 40μs es donde mayor número de paquetes se generan.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,00005 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.00005))
size(find(delta>0.00005))
```

Con  $\text{delta}<0.00005$  se obtienen 334829 valores (los graficados), los descartados con  $\text{delta}>0.00005$ , y se obtienen 432 valores.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 300 bytes a 100Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 42,853μs y la media es de 29,83μs.

En este caso, este resultado no proporciona el mismo resultado que el report del propio programa iperf, ya que 29,83 $\mu$ s son unos 91,71 Mbps y no es parecido a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 97.8 MBytes 82.0 Mbits/sec 0.026 ms 0/341713 (0%)
```

**Fig. A2.4** Resultado servidor Iperf de la prueba a 100 Mbps y 300 bytes

### Prueba a 100 Mbps y 1000 bytes

En esta prueba se generará un flujo de 100 Mbps y 1000 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
iperf -c -u -b 100m -i 1 -l 1000
```

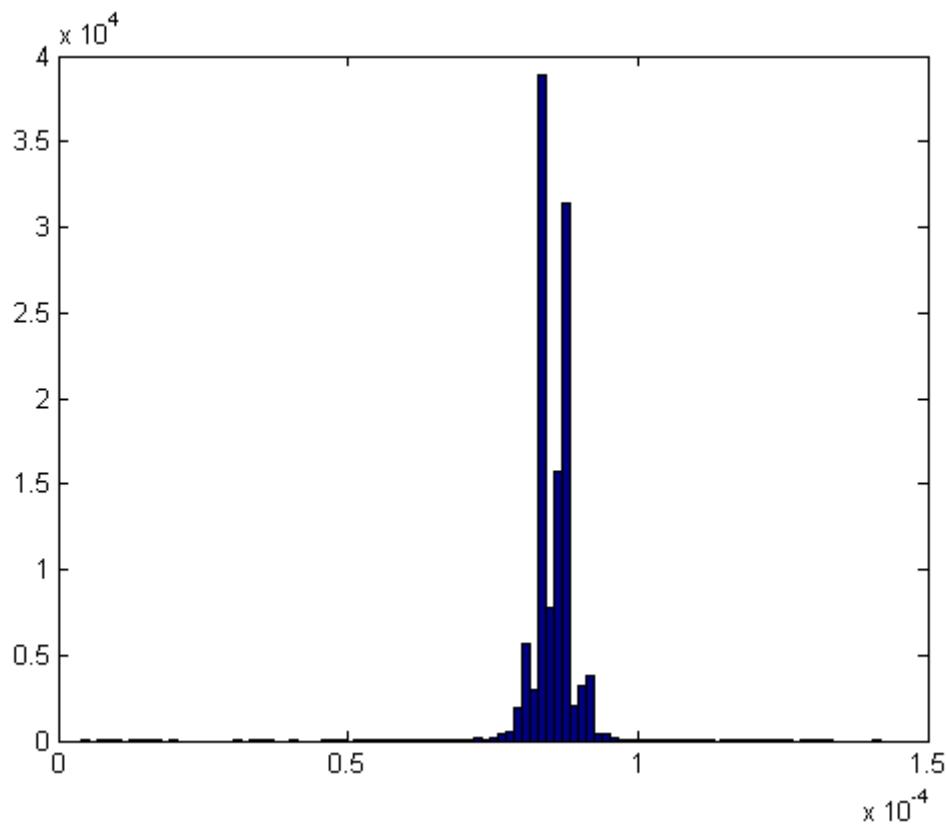
Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 11996,16, y el Delta time la inversa: 83,36 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

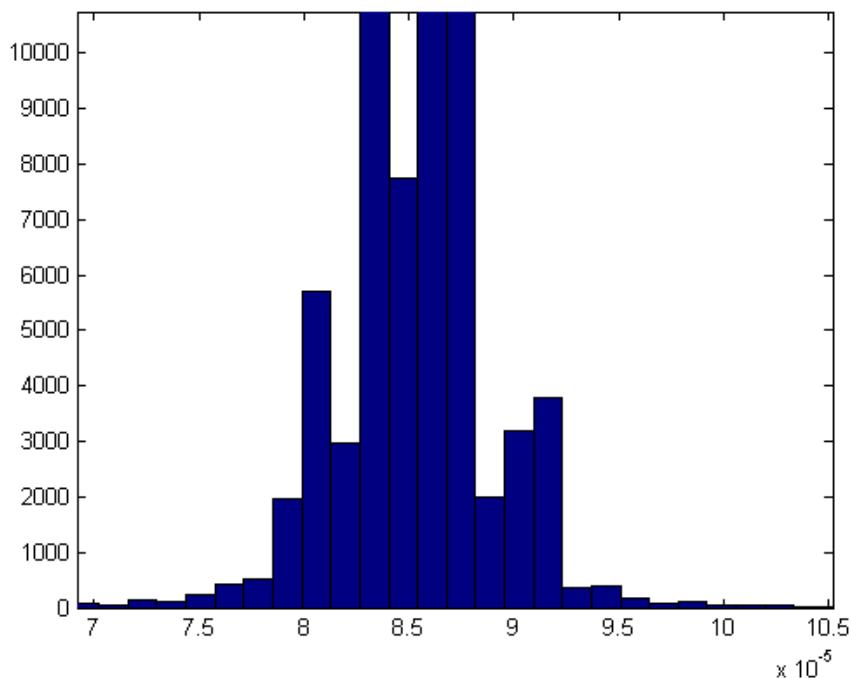
```
hist(delta(find(delta<0.00015)), 100)
```

Se utiliza 15e-5 ya que así abarcamos los datos inferiores y superiores a 83,36  $\mu$ s.

El histograma es el siguiente:



Si se amplía se pueden observar estos valores:



**Fig. A2.6** Histograma a 100 Mbps y 1000 bytes ampliado

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,00015 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.00015))
size(find(delta>0.00015))
```

Con  $\delta < 0.00015$  se obtienen 116506 valores (los graficados), los descartados con  $\delta > 0.00015$ , y se obtienen 58 valores.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1000 bytes a 100Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 37, 02 $\mu$ s y la media es de 85,82 $\mu$ s.

En este caso, este resultado no proporciona el mismo resultado que el report del propio programa iperf, ya que 85,82 $\mu$ s son unos 97,13 Mbps y no es parecido a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:
[ 3] 0.0-10.0 sec 112 MBytes 93.8 Mbits/sec 0.322 ms 0/117351 (0%)
```

**Fig. A2.7** Resultado servidor Iperf de la prueba a 100 Mbps y 1000 bytes

### Prueba a 100 Mbps y 1400 bytes

En esta prueba se generará un flujo de 100 Mbps y 1400 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
iperf -c -u -b 100m -i 1 -l 1400
```

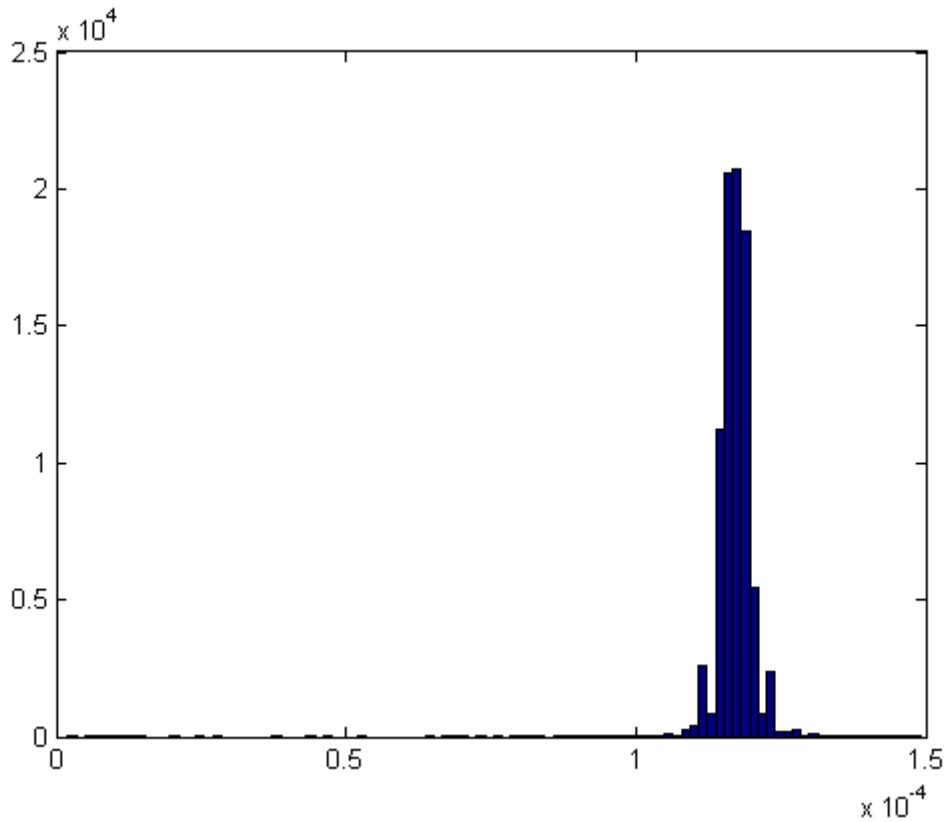
Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 8668,51, y el Delta time la inversa: 115,36 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.00015)), 100)
```

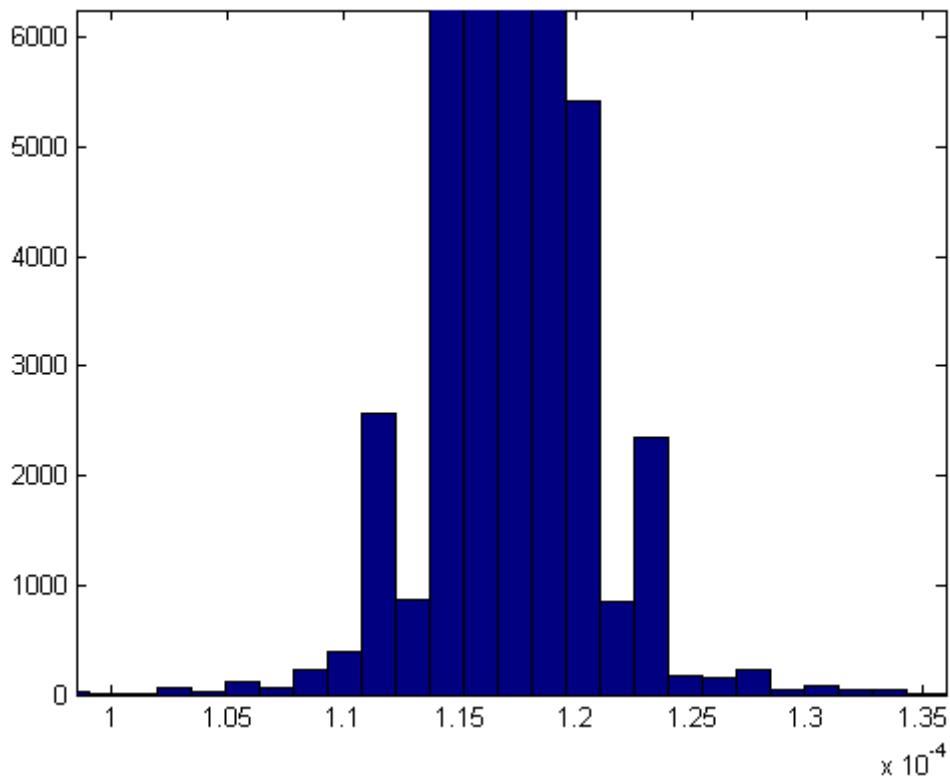
Se utiliza 15e-5 segundos ya que así abarcamos los datos inferiores a ese valor.

El histograma es el siguiente:



**Fig. A2.8** Histograma a 100 Mbps y 1400 bytes

En este gráfico se puede observar como la mayor cantidad de paquetes se concentran entre 100 $\mu$ s y 150 $\mu$ s. Si se realiza un zoom se puede observar como la zona de mayor densidad está entre 110 $\mu$ s y 125 $\mu$ s. Si se amplía se pueden observar estos valores:



**Fig. A2.9** Histograma a 100 Mbps y 1400 bytes ampliado

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,00015 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.00015))
size(find(delta>0.00015))
```

Con  $\delta < 0.00015$  se obtienen 85000 valores (los graficados), los descartados con  $\delta > 0.00015$ , y se obtienen 64 valores.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1400 bytes a 100Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 26,84 $\mu$ s y la media es de 117,62 $\mu$ s.

En este caso, este resultado sí proporciona el mismo resultado que el report del propio programa iperf, ya que 117,62 $\mu$ s son unos 98,07 Mbps y es parecido a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 114 MBytes 95.5 Mbits/sec 0.378 ms 0/85354 (0%)
```

**Fig. A2.10** Resultado servidor Iperf de la prueba a 100 Mbps y 1400 bytes

### Prueba a 300 Mbps y 64 bytes

En esta prueba se generará un flujo de 300 Mbps y 64 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
iperf -c -u -b 300m -i 1 -l 64
```

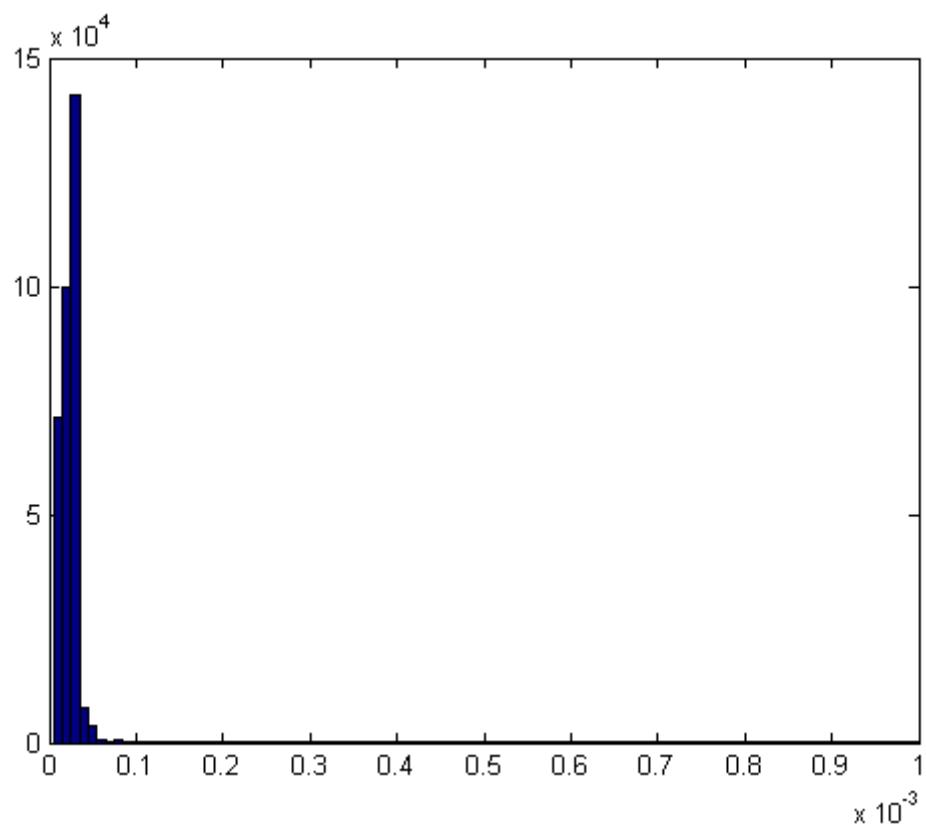
Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 353773,58, y el Delta time la inversa: 2,82 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.001)), 100)
```

Se utiliza 1e-3 segundos ya que así abarcamos los datos inferiores a ese valor.

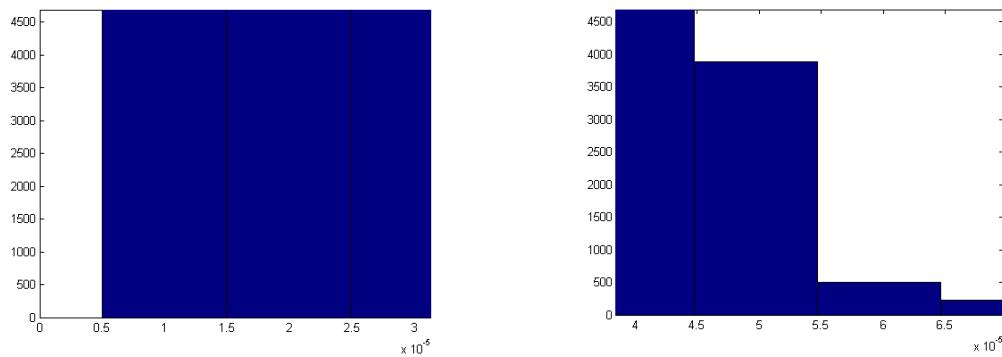
El histograma es el siguiente:



**Fig. A2.11** Histograma a 300 Mbps y 64 bytes

En este gráfico se puede observar como la mayor cantidad de paquetes se concentran entre 0 y 0,1ms. Si se realiza un zoom se puede observar como la zona de mayor densidad está entre 50 $\mu$ s y 65 $\mu$ s.

Si se amplía se pueden observar estos valores:



**Fig. A2.12** Histograma a 300 Mbps y 64 bytes ampliado

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0018 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0018))
size(find(delta>0.0018))
```

Con  $\delta < 0.0018$  se obtienen 328515 valores (los graficados), los descartados con  $\delta > 0.0018$ , y se obtienen 327 valores.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 64 bytes a 300Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 139,43 $\mu$ s y la media es de 30,40 $\mu$ s.

En este caso, este resultado proporciona un resultado algo inferior al generado por el report del propio programa iperf, ya que 30,40 $\mu$ s son unos 27,89 Mbps y como se aprecia en la imagen, no es el mismo:



```
[ 3] Server Report:
[ 3] 0.0-10.0 sec 34.7 MBytes 29.1 Mbits/sec 0.019 ms 0/569075 (0%)
```

**Fig. A2.13** Resultado servidor Iperf de la prueba a 300 Mbps y 64 bytes

### Prueba a 300 Mbps y 300 bytes

En esta prueba se generará un flujo de 300 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
iperf -c -u -b 300m -i 1 -l 300
```

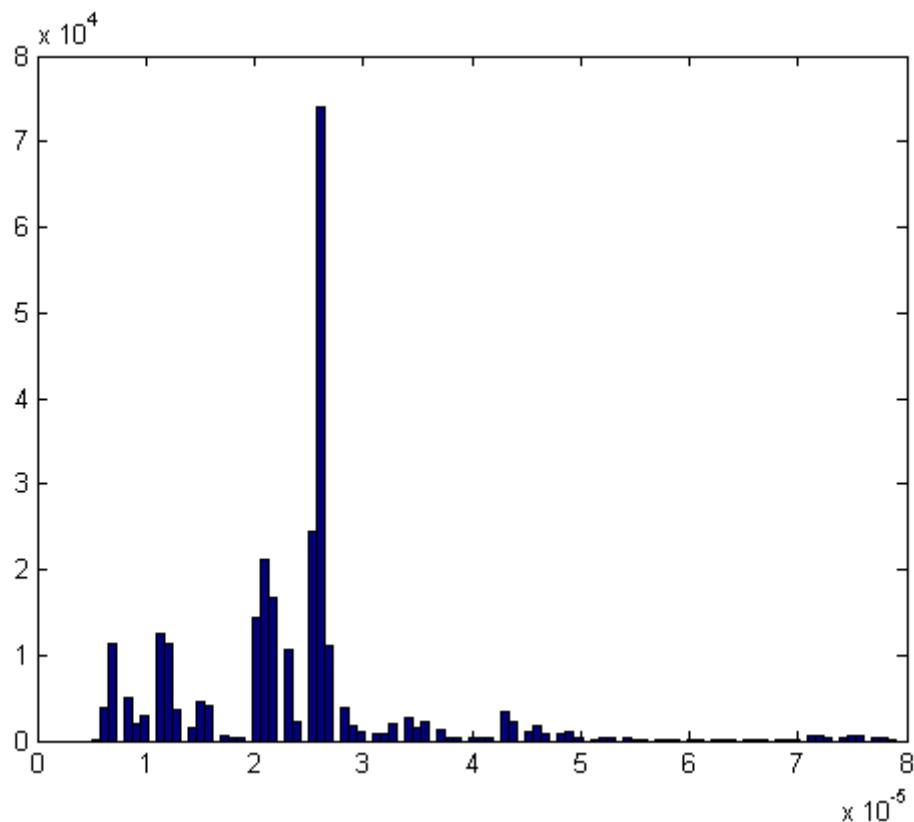
Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 109649,12, y el Delta time la inversa: 9,12 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.00008)), 100)
```

Se utiliza 8e-5 segundos ya que así abarcamos los datos inferiores a ese valor.

El histograma es el siguiente:



**Fig. A2.14** Histograma a 300 Mbps y 300 bytes

En este gráfico se puede observar como la mayor cantidad de paquetes se concentran entre  $20\mu s$  y  $30\mu s$ .

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a  $0,00008$  segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.00008))
size(find(delta>0.00008))
```

Con  $\text{delta}<0.00008$  se obtienen 275198 valores (los graficados), los descartados con  $\text{delta}>0.00008$ , y se obtienen 3611 valores.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 300 bytes a 300Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 271,06 $\mu$ s y la media es de 35,839 $\mu$ s.

En este caso, este resultado proporciona un resultado muy diferente al generado por el report del propio programa iperf, ya que 35,839 $\mu$ s son unos 76,34 Mbps no comparable con el report:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 159 MBytes 134 Mbits/sec 0.002 ms 469/557675 (0.084%)
```

**Fig. A2.15** Resultado servidor Iperf de la prueba a 300 Mbps y 300 bytes

### Prueba a 300 Mbps y 1000 bytes

En esta prueba se generará un flujo de 300 Mbps y 1000 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
iperf -c -u -b 300m -i 1 -l 1000
```

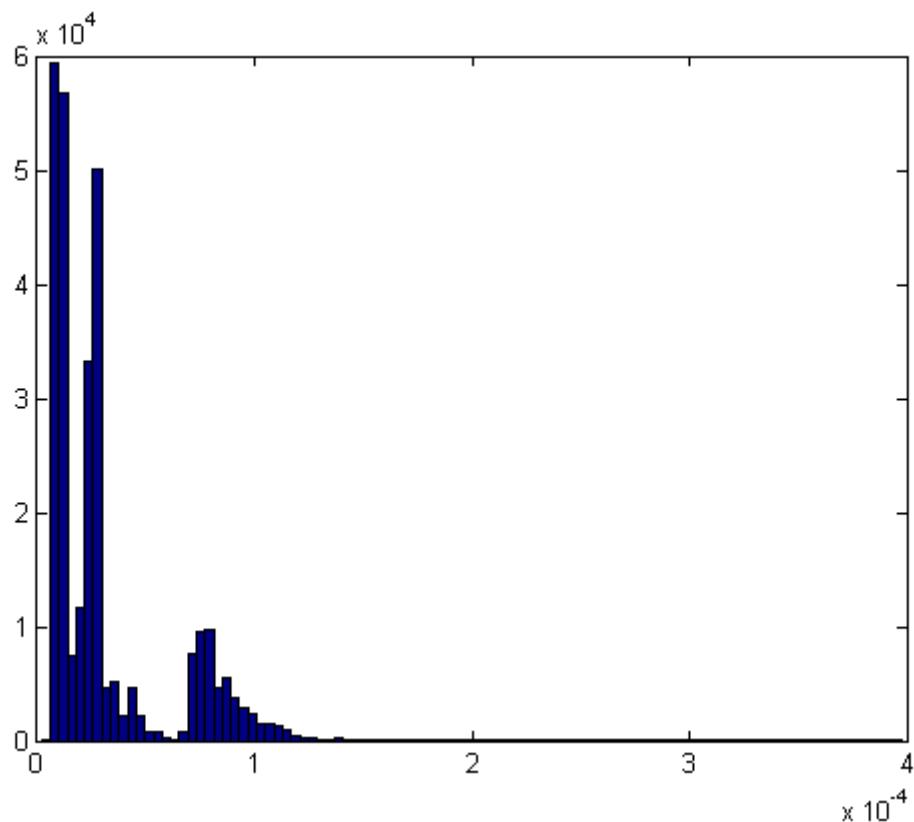
Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 35988,48, y el Delta time la inversa: 27,78 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0004)), 100)
```

Se utiliza 4e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

El histograma es el siguiente:



**Fig. A2.16**Histograma a 300 Mbps y 1000 bytes

En este gráfico se puede observar como los paquetes se concentran en dos ráfagas entre 0 y 140 $\mu s$ .

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0004 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0004))
size(find(delta>0.0004))
```

Con  $\delta < 0.0004$  se obtienen 294334 valores (los graficados), los descartados con  $\delta > 0.0004$ , y se obtienen 615 valores.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1000 bytes a 300Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 134,65 $\mu s$  y la media es de 33,90 $\mu s$ .

En este caso, este resultado sí proporciona un resultado parecido al report del propio programa iperf, ya que 33,90 $\mu$ s son unos 245,89 Mbps y es similar a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 293 MBytes 246 Mbits/sec 0.013 ms 20/306921 (0.0065%)
```

**Fig. A2.17 Resultado servidor Iperf de la prueba a 300 Mbps y 1000 bytes**

### Prueba a 300 Mbps y 1400 bytes

En esta prueba se generará un flujo de 300 Mbps y 1000 bytes de datos útiles. El siguiente comando generará dicho flujo:

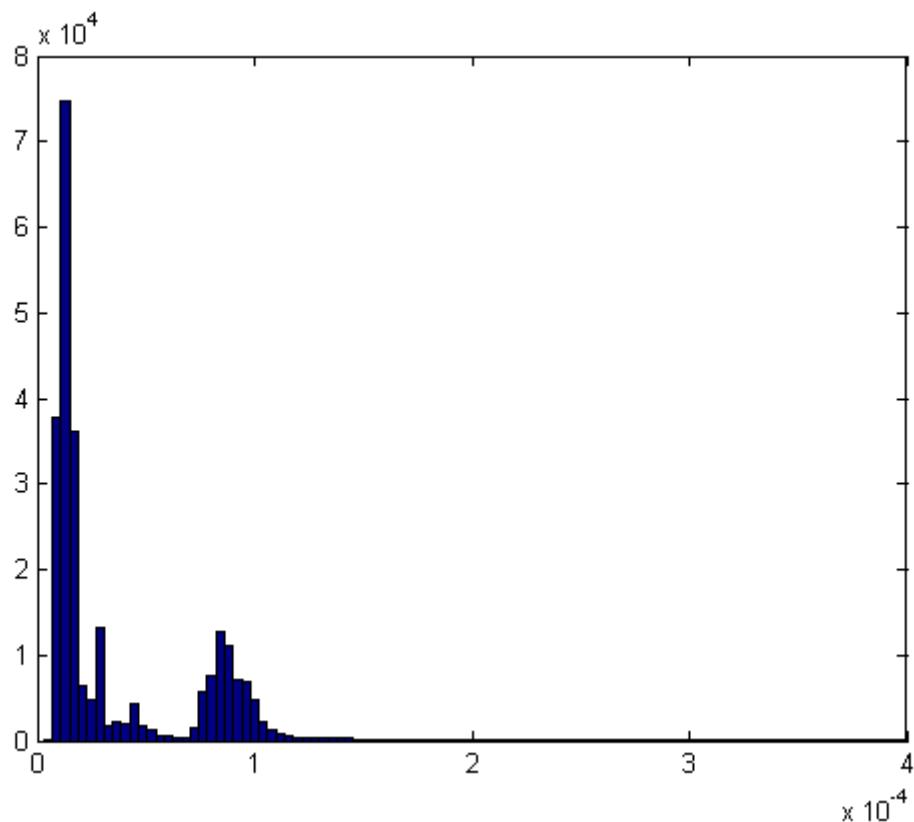
```
iperf -c -u -b 300m -i 1 -l 1400
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 26005,54, y el Delta time la inversa: 38,45 $\mu$ s segundos.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0004)), 100)
```

El histograma es el siguiente:



**Fig. A2.18** Histograma a 300 Mbps y 1400 bytes

En este gráfico se puede observar como los paquetes se concentran en dos ráfagas entre 0 y 140 $\mu s$ , muy parecido al resultado anterior.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0004 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0004))
size(find(delta>0.0004))
```

Con  $\delta < 0.0004$  se obtienen 254417 valores (los graficados), los descartados con  $\delta > 0.0004$ , y se obtienen 483 valores.

Se utiliza 4e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1400 bytes a 300Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 166,69 $\mu s$  y la media es de 39,22 $\mu s$ .

En este caso, este resultado sí proporciona un resultado parecido al report del propio programa iperf, ya que 39,22 $\mu$ s son unos 294,13 Mbps y es similar a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 350 MBytes 293 Mbits/sec 0.020 ms 0/261869 (0%)
```

**Fig. A2.19** Resultado servidor Iperf de la prueba a 300 Mbps y 1400 bytes

### Prueba a 500 Mbps y 64 bytes

En esta prueba se generará un flujo de 500 Mbps y 64 bytes de datos útiles. El siguiente comando generará dicho flujo:

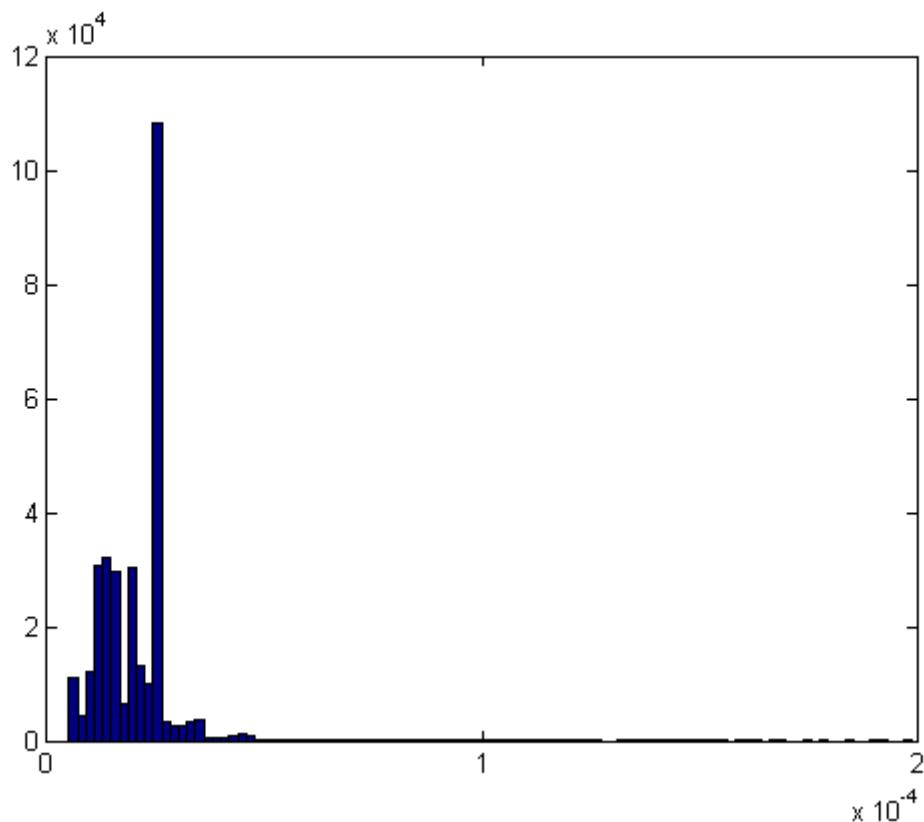
```
iperf -c -u -b 500m -i 1 -l 64
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 589622,64, y el Delta time la inversa: 1,69 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.002)), 100)
```

El histograma es el siguiente:



**Fig. A2.20** Histograma a 500 Mbps y 64 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y 50 $\mu$ s con un valor muy denotado que es 250 $\mu$ s con gran cantidad de valores.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,002 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.002))
size(find(delta>0.002))
```

Con  $\delta < 0.002$  se obtienen 312599 valores (los graficados), los descartados con  $\delta > 0.002$ , y se obtienen 772 valores.

Se utiliza 2e-3 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 64 bytes a 500Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 195,64 $\mu$ s y la media es de 31,91 $\mu$ s.

En este caso, este resultado no proporciona un resultado parecido al report del propio programa iperf, ya que 31,91 $\mu$ s son unos 26,57 Mbps y no es similar a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 39.0 MBytes 32.7 Mbits/sec 0.001 ms 16/639630 (0.0025%)
```

**Fig. A2.21** Resultado servidor Iperf de la prueba a 500 Mbps y 64 bytes

### Prueba a 500 Mbps y 300 bytes

En esta prueba se generará un flujo de 500 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

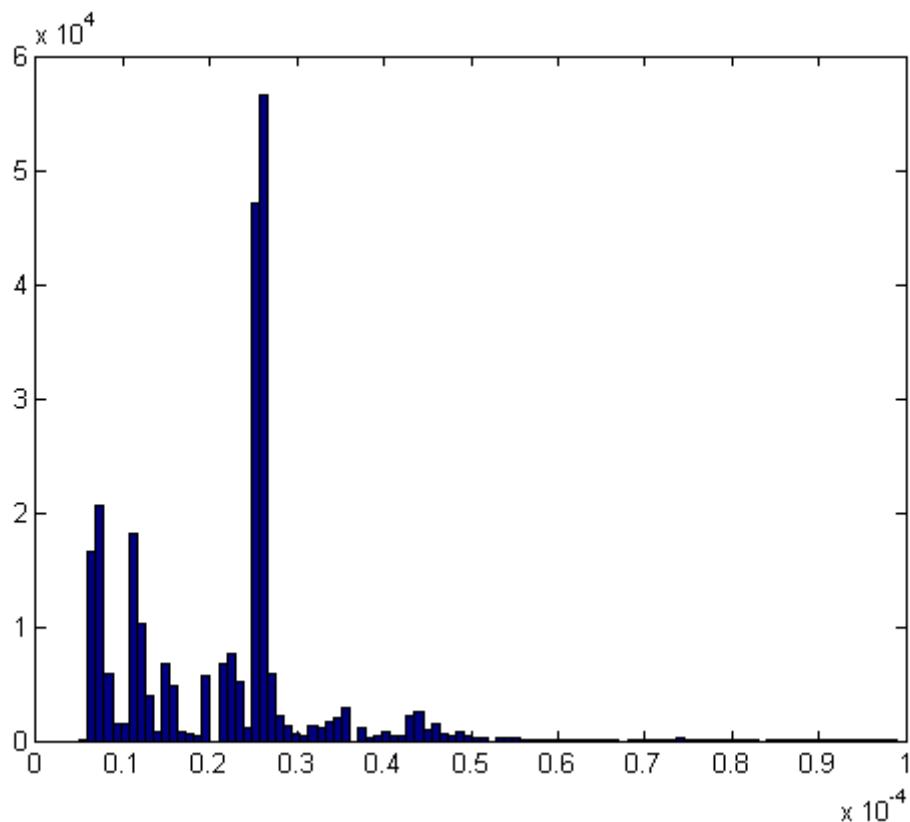
```
iperf -c -u -b 500m -i 1 -l 300
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 182748,53, y el Delta time la inversa: 5,47 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 100)
```

El histograma es el siguiente:



**Fig. A2.22** Histograma a 500 Mbps y 300 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y 50 $\mu s$  con un valor muy denotado que es 25 $\mu s$  con gran cantidad de valores.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,002 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 258513 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 2511 valores.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 300 bytes a 500Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 237,74 $\mu s$  y la media es de 38,31 $\mu s$ .

En este caso, este resultado no proporciona un resultado parecido al report del propio programa iperf, ya que 38,31 $\mu$ s son unos 71,41 Mbps y no es similar a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 173 MBytes 145 Mbits/sec 0.022 ms 3841/609983 (0.63%)
```

**Fig. A2.23 Resultado servidor Iperf de la prueba a 500 Mbps y 300 bytes**

### Prueba a 500 Mbps y 1000 bytes

En esta prueba se generará un flujo de 500 Mbps y 1000 bytes de datos útiles. El siguiente comando generará dicho flujo:

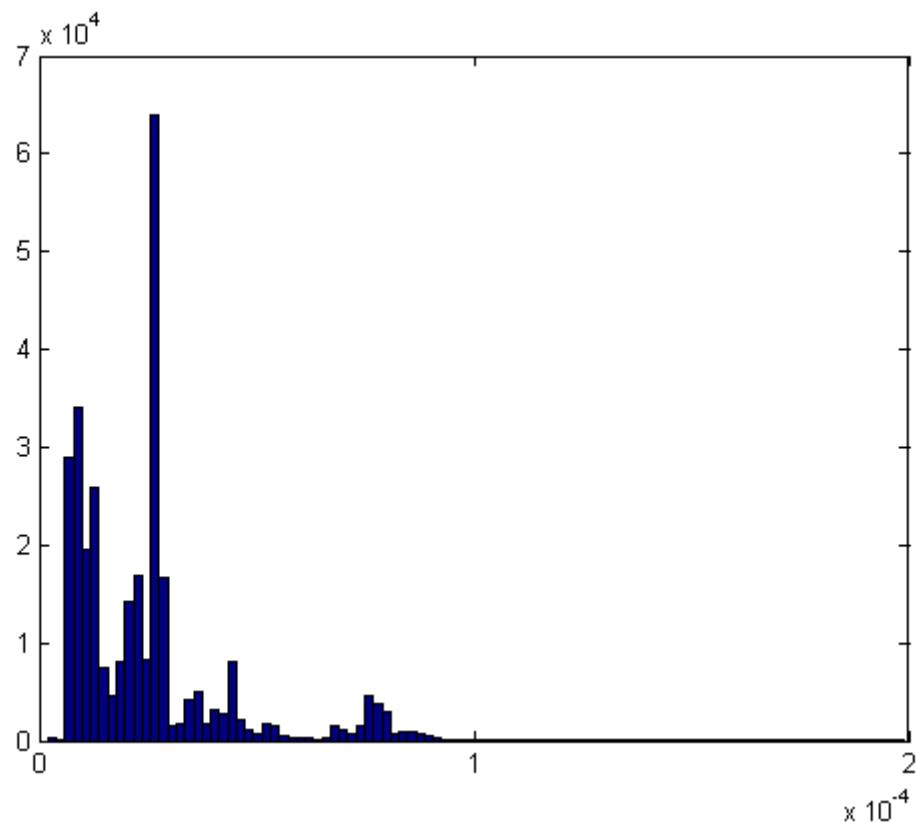
```
iperf -c -u -b 500m -i 1 -l 1000
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 59980,80, y el Delta time la inversa: 16,67 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0002)), 100)
```

El histograma es el siguiente:



**Fig. A2.24** Histograma a 500 Mbps y 1000 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y  $100\mu s$  con un valor muy denotado que es  $26\mu s$  con gran cantidad de valores y seguido de otro valor con gran cantidad también:  $9\mu s$ .

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0002 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0002))
size(find(delta>0.0002))
```

Con  $\delta < 0.0002$  se obtienen 309512 valores (los graficados), los descartados con  $\delta > 0.0002$ , y se obtienen 3186 valores.

Se utiliza  $2e-4$  segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1000 bytes a 500Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de  $481,15\mu s$  y la media es de  $31,97\mu s$ .

En este caso, este resultado no proporciona un resultado parecido al report del propio programa iperf, ya que 31,97 $\mu$ s son unos 260,74 Mbps y no es similar a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 375 MBytes 315 Mbits/sec 0.003 ms 175/393696 (0.044%)
```

**Fig. A2.25** Resultado servidor Iperf de la prueba a 500 Mbps y 1000 bytes

### Prueba a 500 Mbps y 1400 bytes

En esta prueba se generará un flujo de 500 Mbps y 1400 bytes de datos útiles. El siguiente comando generará dicho flujo:

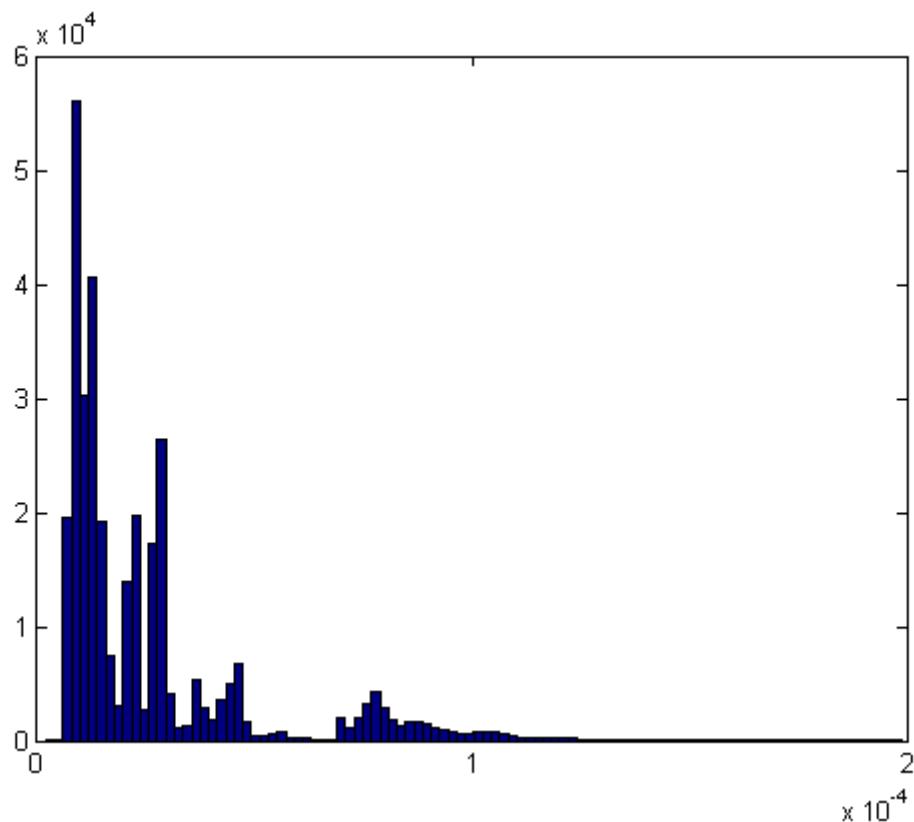
```
iperf -c -u -b 500m -i 1 -l 1400
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 43342,57, y el Delta time la inversa: 23,07 $\mu$ s segundos.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0002)), 100)
```

El histograma es el siguiente:



**Fig. A2.26** Histograma a 500 Mbps y 1400 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y 130 $\mu$ s con varios valores denotados: 90 $\mu$ s, 100 $\mu$ s.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0002 segundos. Hay un comando que cambiando el operador nos permite saber cuantos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0002))
size(find(delta>0.0002))
```

Con  $\delta < 0.0002$  se obtienen 328435 valores (los graficados), los descartados con  $\delta > 0.0002$ , y se obtienen 2878 valores.

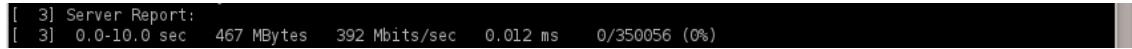
Se utiliza 2e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1000 bytes a 500Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 246,13 $\mu$ s y la media es de 30,17 $\mu$ s.

En este caso, este resultado sí proporciona un resultado parecido al report del propio programa iperf, ya que 30,17 $\mu$ s son unos 382,36 Mbps y es similar a lo mostrado en la siguiente imagen:



```
[ 3] Server Report:
[ 3] 0.0-10.0 sec 467 MBytes 392 Mbits/sec 0.012 ms 0/350056 (0%)
```

A terminal window showing the output of an iperf server report. The output includes the number of streams (3), the duration (0.0-10.0 sec), the total bytes transferred (467 MBytes), the throughput (392 Mbits/sec), the round trip time (0.012 ms), and the number of packets lost (0/350056 or 0%).

**Fig. A2.27** Resultado servidor Iperf de la prueba a 500 Mbps y 1400 bytes

### Prueba a 1000 Mbps y 64 bytes

En esta prueba se generará un flujo de 1000 Mbps y 64 bytes de datos útiles. El siguiente comando generará dicho flujo:

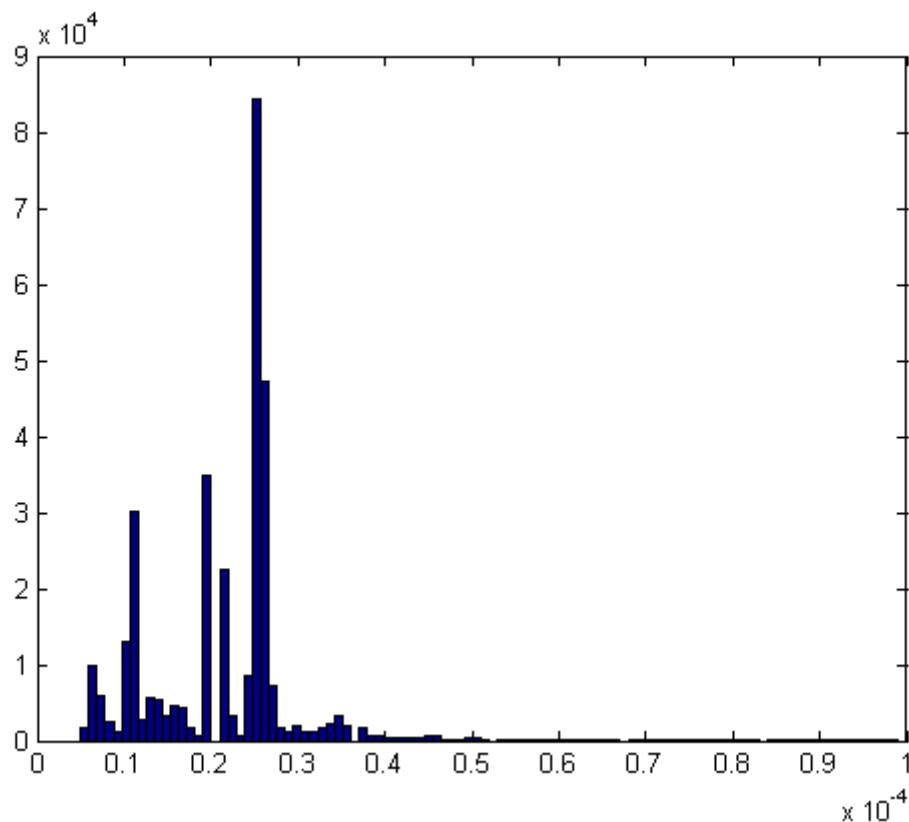
```
iperf -c -u -b 1000m -i 1 -l 64
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 1179245,28, y el Delta time la inversa: 0,84 $\mu$ s segundos.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0002)), 100)
```

El histograma es el siguiente:



**Fig. A2.28** Histograma a 1000 Mbps y 64 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y 50 $\mu$ s con varios valores denotados: 25 $\mu$ s y 26 $\mu$ s.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta t < 0.0001$  se obtienen 327689 valores (los graficados), los descartados con  $\delta t > 0.0001$ , y se obtienen 2249 valores.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 64 bytes a 1000Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 141,27 $\mu$ s y la media es de 30,30 $\mu$ s.

En este caso, este resultado sí proporciona un resultado parecido al report del propio programa iperf, ya que 30,30 $\mu$ s son unos 37,98 Mbps y es similar a lo mostrado en la siguiente imagen:



```
[ 3] Server Report:
[ 3] 0.0-10.0 sec 35.6 MBytes 29.8 Mbits/sec 0.022 ms 240/582963 (0.041%)
```

**Fig. A2.29** Resultado servidor Iperf de la prueba a 1000 Mbps y 64 bytes

### Prueba a 1000 Mbps y 300 bytes

En esta prueba se generará un flujo de 1000 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

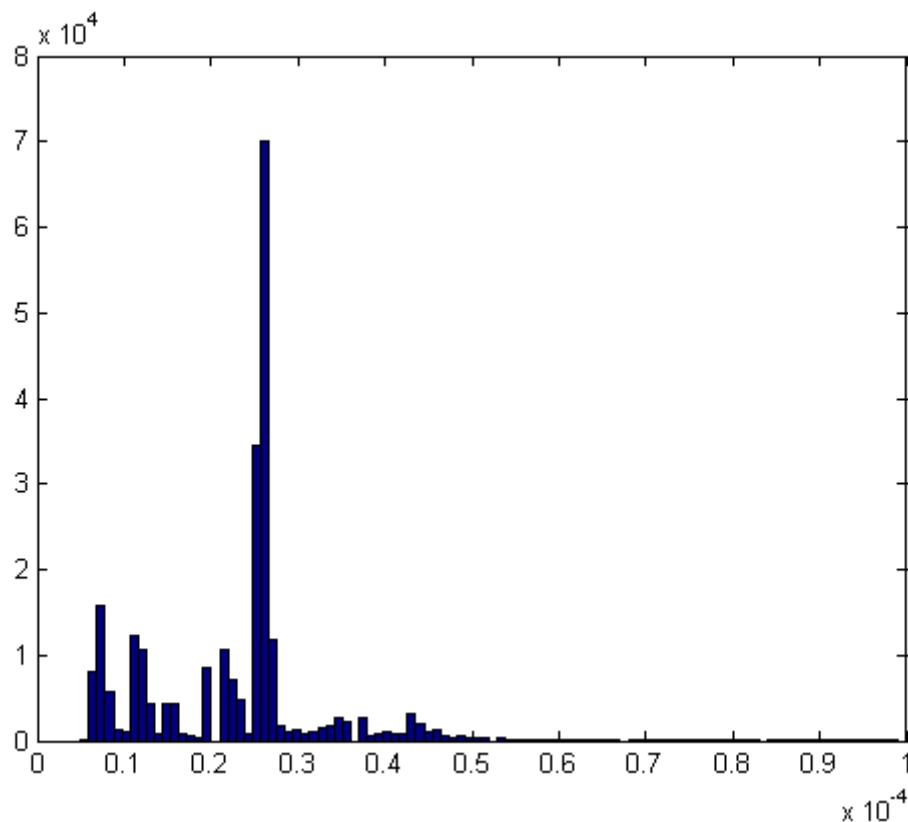
```
iperf -c -u -b 1000m -i 1 -l 300
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 365497,07, y el Delta time la inversa: 2,73 $\mu$ s.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 100)
```

El histograma es el siguiente:



**Fig. A2.30** Histograma a 1000 Mbps y 300 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y 50 $\mu$ s con varios valores denotados: 25 $\mu$ s y 26 $\mu$ s.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 252185 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 2217 valores.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 300 bytes a 1000Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 235,24 $\mu$ s y la media es de 39,30 $\mu$ s.

En este caso, este resultado no proporciona un resultado parecido al report del propio programa iperf, ya que 39,30 $\mu$ s son unos 69,61 Mbps y no es similar a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 168 MBytes 141 Mbits/sec 0.002 ms 3008/591283 (0.51%)
```

**Fig. A2.31** Resultado servidor Iperf de la prueba a 1000 Mbps y 300 bytes

### Prueba a 1000 Mbps y 1000 bytes

En esta prueba se generará un flujo de 1000 Mbps y 1000 bytes de datos útiles.

El siguiente comando generará dicho flujo:

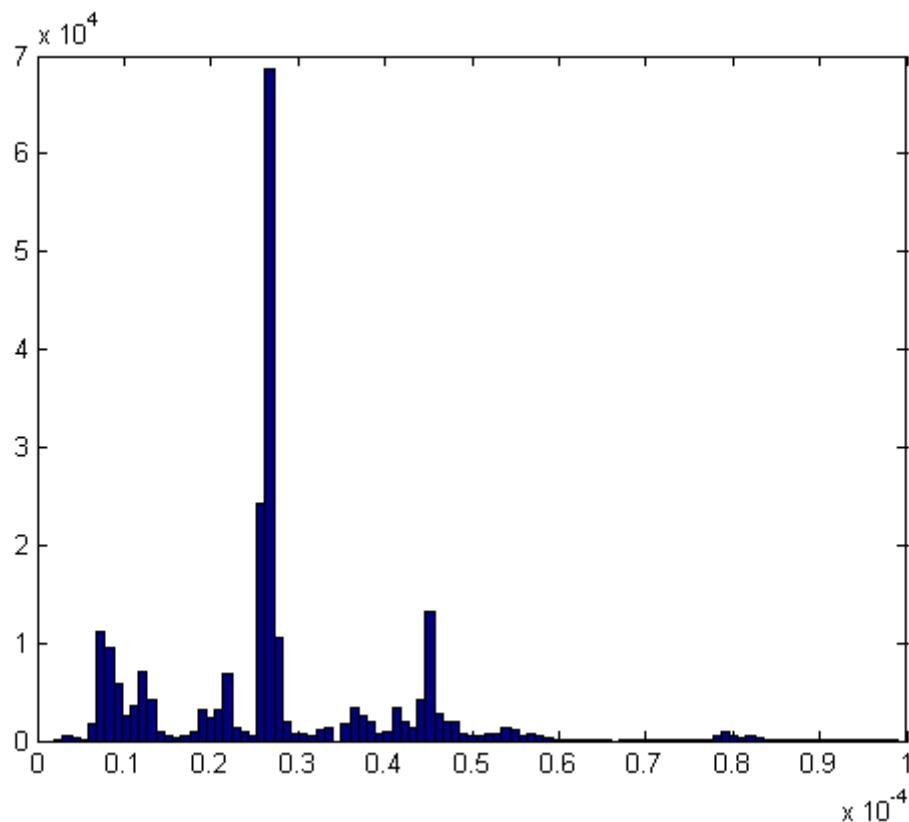
```
iperf -c -u -b 1000m -i 1 -l 1000
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 119961,61, y el Delta time la inversa: 8,3 $\mu$ s segundos.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 100)
```

El histograma es el siguiente:



**Fig. A2.32** Histograma a 1000 Mbps y 1000 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y 60 $\mu$ s con varios valores denotados: 25,5 $\mu$ s y 26,6 $\mu$ s.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 232928 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 3222 valores.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1000 bytes a 1000Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 198,37 $\mu$ s y la media es de 42,335 $\mu$ s.

En este caso, este resultado no proporciona un resultado parecido al report del propio programa iperf, ya que 42,33 $\mu$ s son unos 196,90 Mbps y no es similar a lo mostrado en la siguiente imagen:

```
[ 3] Server Report:  
[ 3] 0.0-10.0 sec 466 MBytes 391 Mbits/sec 0.016 ms 900/489308 (0.18%)
```

**Fig. A2.33** Resultado servidor Iperf de la prueba a 1000 Mbps y 100 bytes

### Prueba a 1000 Mbps y 1400 bytes

En esta prueba se generará un flujo de 1000 Mbps y 1400 bytes de datos útiles.

El siguiente comando generará dicho flujo:

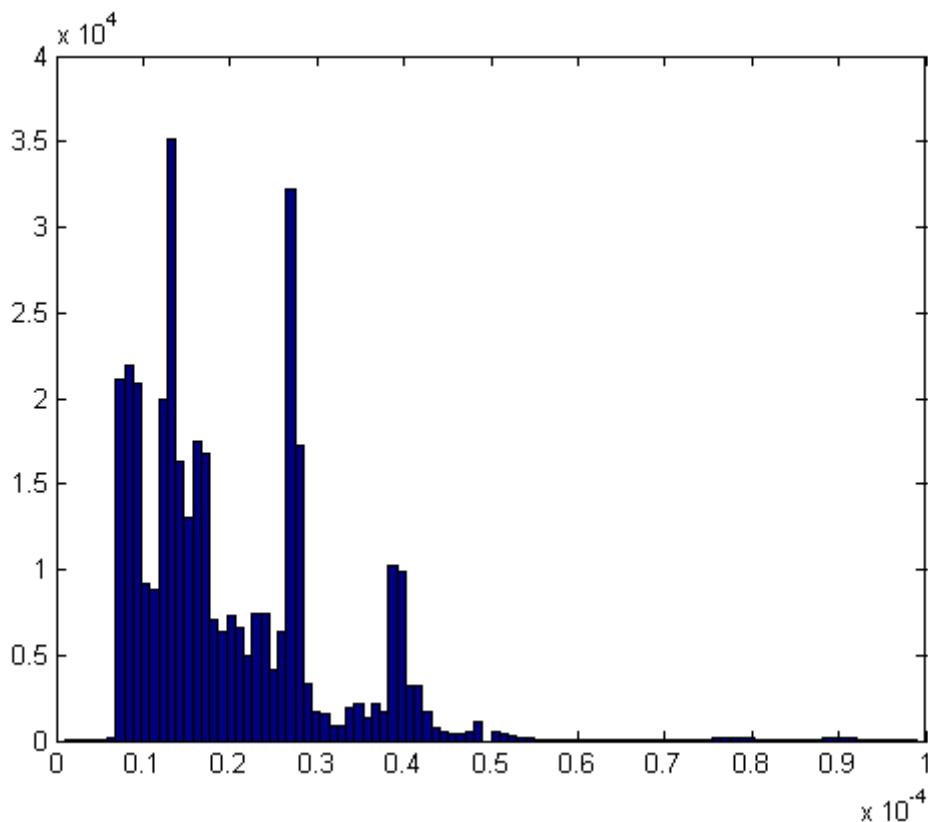
```
iperf -c -u -b 1000m -i 1 -l 1400
```

Teóricamente, en base a este tamaño de paquete con cabeceras y a esta velocidad, los paquetes por segundo generados es de 86685,15, y el Delta time la inversa: 11,5 $\mu$ s segundos.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 100)
```

El histograma es el siguiente:



**Fig. A2.34** Histograma a 1000 Mbps y 1400 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y 50 $\mu\text{s}$  con varios valores denotados: 13 $\mu\text{s}$  y 28 $\mu\text{s}$ .

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 362851 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 2487 valores.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1400 bytes a 1000Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 1,5ms y la media es de 27,36 $\mu\text{s}$ .

En este caso, este resultado no proporciona un resultado parecido al report del propio programa iperf, ya que 27,36μs son unos 421,63 Mbps y no es similar a lo mostrado en la siguiente imagen:



```
[ 3] Server Report:
[ 3] 0.0-10.0 sec 700 MBytes 587 Mbits/sec 0.019 ms 289/524769 (0.055%)
```

**Fig. A2.35** Resultado servidor Iperf de la prueba a 1000 Mbps y 1400 bytes

### III. Pruebas MGEN

#### Emisión y recepción

Como se ha comentado anteriormente se configurarán dos equipos, uno como cliente y otro como servidor, ambos con Linux y con los scripts arrancados. La captura de datos se realizará mediante la herramienta tcpdump, un software libre que permite capturar todos los paquetes de la interfaz (gracias a la librería pcap) y que no exige tantos recursos de CPU como Wireshark.

El método de captura es el mismo que se utiliza en el [Anexo II](#).

#### Generación de scripts

Esta herramienta requiere el uso de scripts para poner en marcha los ordenadores de cualquier simulación. Tiene una sintaxis fácil y muchos parámetros con los que interactuar. En el siguiente ejemplo se explicarán los parámetros utilizados en las pruebas realizadas:

```
# Inicio script cliente MGEN (1_flujo_64B_100M.mgn)
# Creación de 1 flujo UDP con dirección ip destino, patrón...
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [195312 64]

# Parar el flujo UDP con una duración de 5 segundos
5 OFF 1
```

Sintaxis:

<segundo inicio><on/off><id\_flujo><protocolo>SRC<puerto\_origen>

DST<ip\_destino>/<puerto><patron\_de\_datos> [paquetes/seg  
número\_bytes\_útiles]

Para cada prueba realizada se ha tenido que cambiar tanto los paquetes por segundo como en número de bytes útiles, así obtener las mismas pruebas que las realizadas con Iperf.

Una vez creado el script sólo hay que arrancarlo con el siguiente comando:

```
mgen input 1_flujo_64B_100M.mgn
```

El servidor tiene una sintaxis más sencilla y se ejecuta de la misma manera que el servidor pero cambiando el nombre del archivo:

```
# Inicio script servidor MGGEN (server.mgn)
# Creación de 1 flujo UDP con dirección ip destino, patrón...
0.0 LISTEN UDP 5500
```

Para pararlo sólo hay que presionar Control+C.

## Graficando con Matlab

Los histogramas se realizan de la misma manera que en las pruebas de Iperf, se desarrolla en la sección [Anexo II](#).

## Prueba a 100 Mbps y 64 bytes

En esta prueba se generará un flujo de 100 Mbps y 64 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_64B_100M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGGEN
# Creación de 1 flujo UDP con dirección ip destino, periodicidad...
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [195312
64]

# Parar el flujo UDP con una duración de 5 segundos
5 OFF 1
```

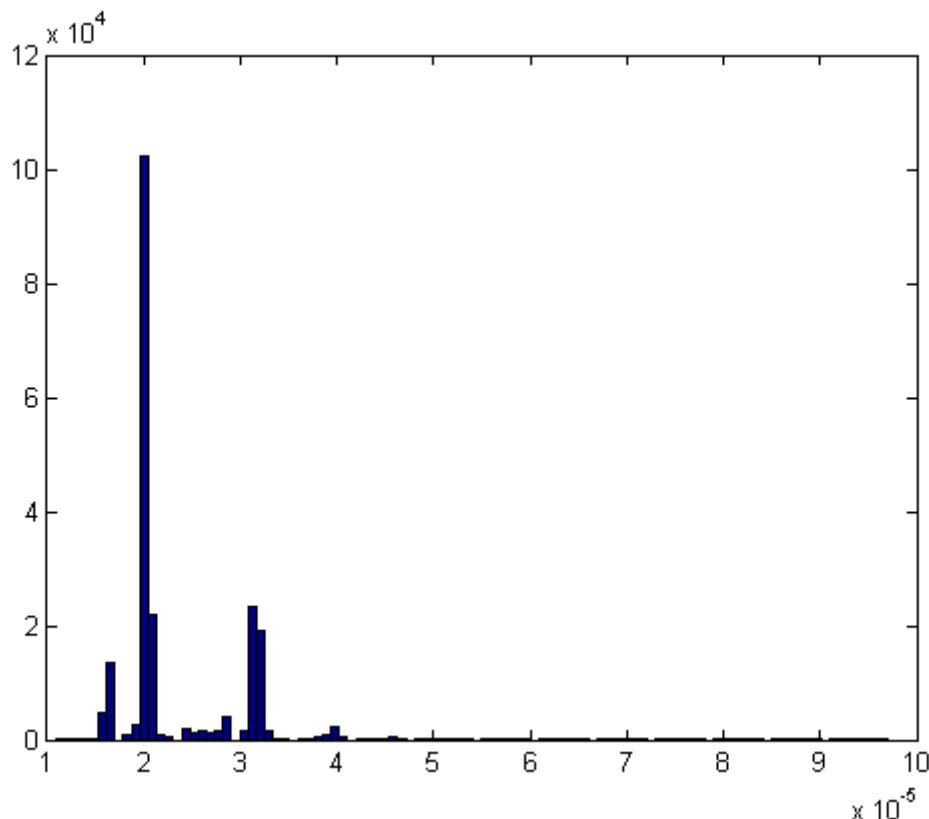
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de  $5,12\mu\text{s}$ , veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 100)
```

El histograma es el siguiente:



**Fig. A3.1** Histograma a 100 Mbps y 64 bytes

En este gráfico se puede observar como los paquetes se concentran entre 0 y  $40\mu\text{s}$  con un valor denotado en  $20\mu\text{s}$ . En este histograma se puede observar como la dispersión de los paquetes es mucho menor que con iperf, es decir, la cantidad de paquete se concentran más en pocas zonas y no muchas.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))  
size(find(delta>0.0001))
```

Con delta<0.0001 se obtienen 212449 valores (los graficados), los descartados con delta>0.0001, y se obtienen 999 valores, es decir, un 0,46802%.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 64 bytes a 100Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)  
mean (delta)
```

La desviación estándar es de 137,08μs y la media es de 29,26μs.

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a 5μs que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 28,64 Mbps

## Prueba a 100 Mbps y 300 bytes

En esta prueba se generará un flujo de 100 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgm input 1_flujo_300B_100M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN  
  
# Creación de 1 flujo UDP con dirección ip destino, periodicidad...  
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [41666  
300]  
  
# Parar el flujo UDP con una duración de 5 segundos  
5 OFF 1
```

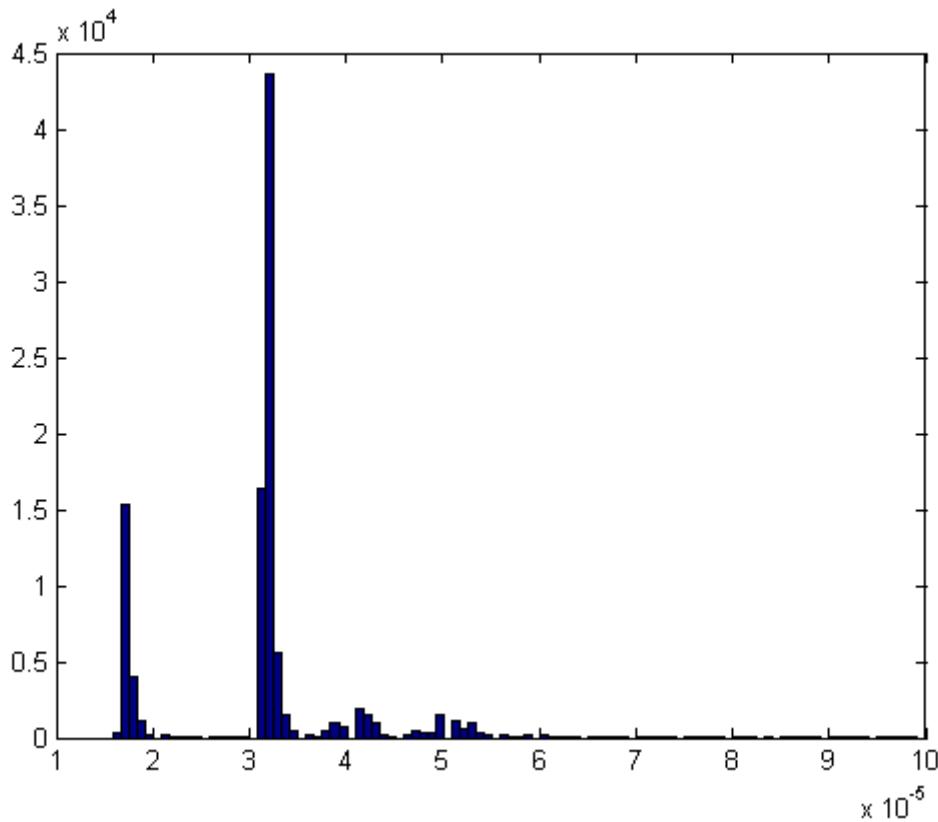
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 24μs, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 100)
```

El histograma es el siguiente:



**Fig. A3.2** Histograma a 100 Mbps y 300 bytes

En este gráfico se puede observar como los paquetes se concentran entre 1,6 y 60μs con un valor denotado en 32μs. Como ya se ha comentado anteriormente la dispersión es pequeña.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con delta<0.0001 se obtienen 104250 valores (los graficados), los descartados con delta>0.0001, y se obtienen 746 valores, es decir, un 0,7105%.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 300 bytes a 100Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 243 $\mu$ s y la media es de 48,33 $\mu$ s.

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a 24 $\mu$ s que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 56,61 Mbps

## Prueba a 100 Mbps y 1000 bytes

En esta prueba se generará un flujo de 100 Mbps y 1000 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_1000B_100M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación de 1 flujo UDP con dirección ip destino, periodicidad...
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [12500
1000]

# Parar el flujo UDP con una duración de 5 segundos
5 OFF 1
```

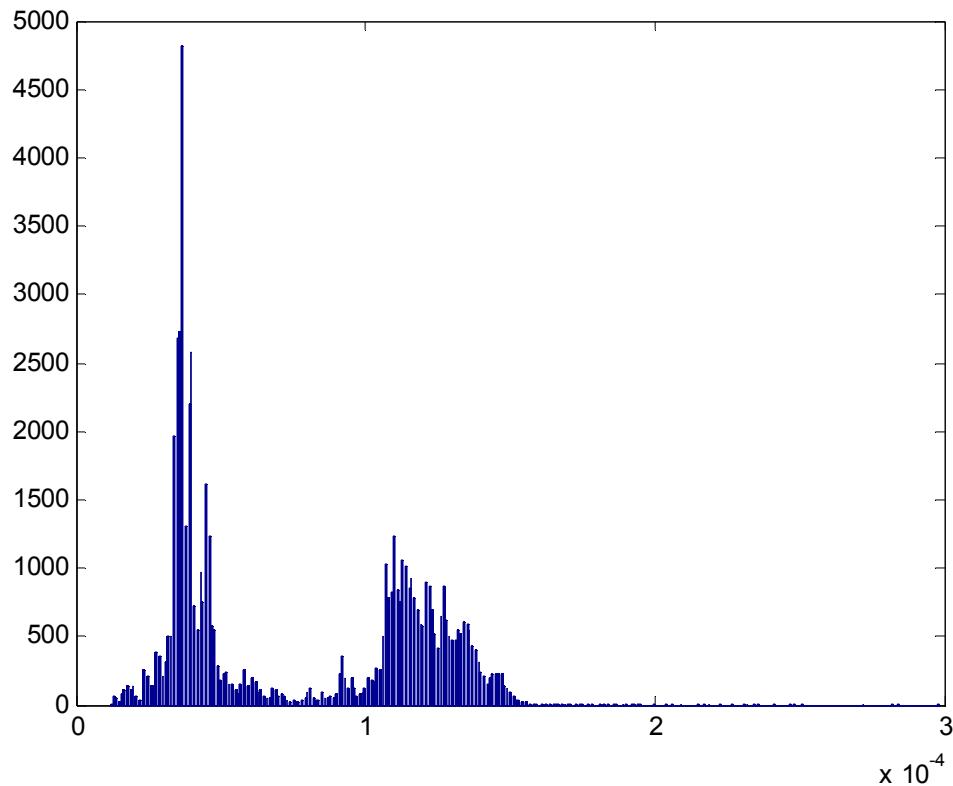
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 80 $\mu$ s, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0003)), 500)
```

El histograma es el siguiente:



**Fig. A3.3** Histograma a 100 Mbps y 1000 bytes

En este gráfico se puede observar como los paquetes se concentran entre  $10\mu\text{s}$  y  $160\mu\text{s}$  con un valor denotado en  $37\mu\text{s}$ . Como ya se ha comentado anteriormente la dispersión es pequeña.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a  $0,0003$  segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0003))
size(find(delta>0.0003))
```

Con  $\text{delta}<0.0003$  se obtienen 61957 valores (los graficados), los descartados con  $\text{delta}>0.0003$ , y se obtienen 146 valores, es decir, un 0,2350%.

Se utiliza  $3e-4$  segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1000 bytes a 100Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 122,39 $\mu$ s y la media es de 80,49 $\mu$ s.

En este caso, este resultado sí proporciona un resultado correcto, ya que la media debería ser parecida a 80 $\mu$ s que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 103,56 Mbps.

## Prueba a 100 Mbps y 1400 bytes

En esta prueba se generará un flujo de 100 Mbps y 1400 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_1400B_100M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGGEN
# Creación de 1 flujo UDP con dirección ip destino, periodicidad...
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [8928
1400]

# Parar el flujo UDP con una duración de 5 segundos
5 OFF 1
```

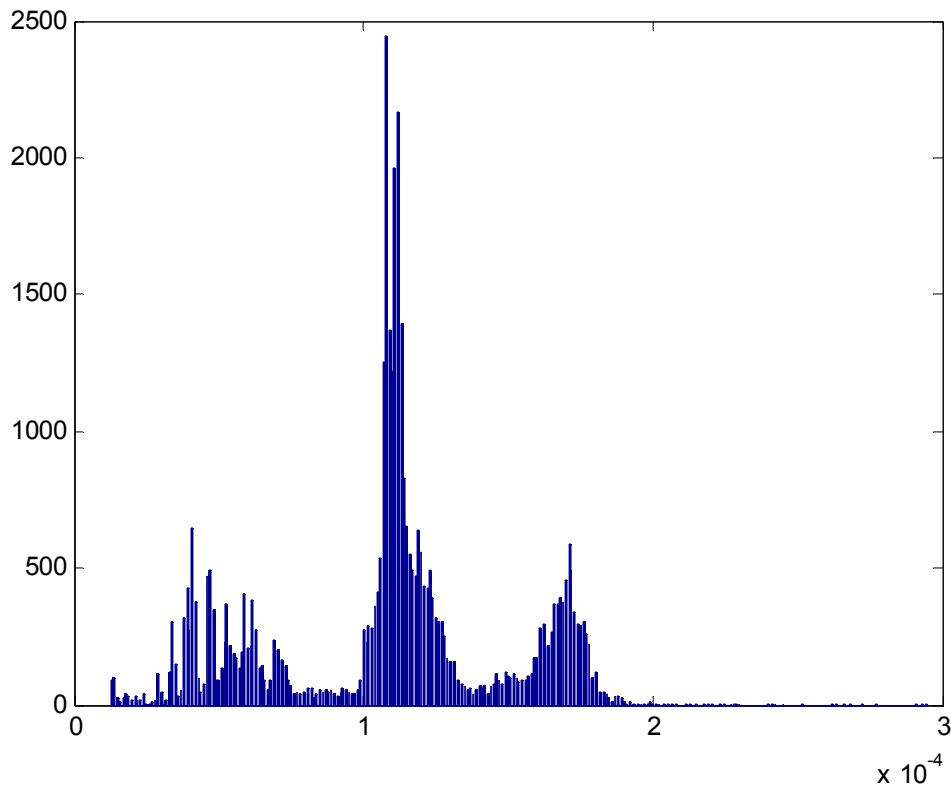
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 112 $\mu$ s, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0003)), 500)
```

El histograma es el siguiente:



**Fig. A3.4** Histograma a 100 Mbps y 1400 bytes

En este gráfico se puede observar como los paquetes se concentran entre  $10\mu\text{s}$  y  $200\mu\text{s}$  con un valor denotado en  $108\mu\text{s}$ . Como ya se ha comentado anteriormente la dispersión es pequeña.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0003 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0003))
size(find(delta>0.0003))
```

Con  $\text{delta}<0.0001$  se obtienen 42021 valores (los graficados), los descartados con  $\text{delta}>0.0001$ , y se obtienen 134 valores, es decir, un 0,3178%.

Se utiliza  $3e-4$  segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1400 bytes a 100Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 1,3ms y la media es de 118,58μs.

En este caso, este resultado sí proporciona un resultado correcto, ya que la media debería ser parecida a 112μs que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 97,28 Mbps.

### Prueba a 300 Mbps y 64 bytes

En esta prueba se generará un flujo de 300 Mbps y 64 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_64B_300M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación 5 flujos UDP con mismo destino, periódicos,
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [585937
64]

# Parar los 5 flujos UDP con una duración de 5 segundos
5 OFF 1
```

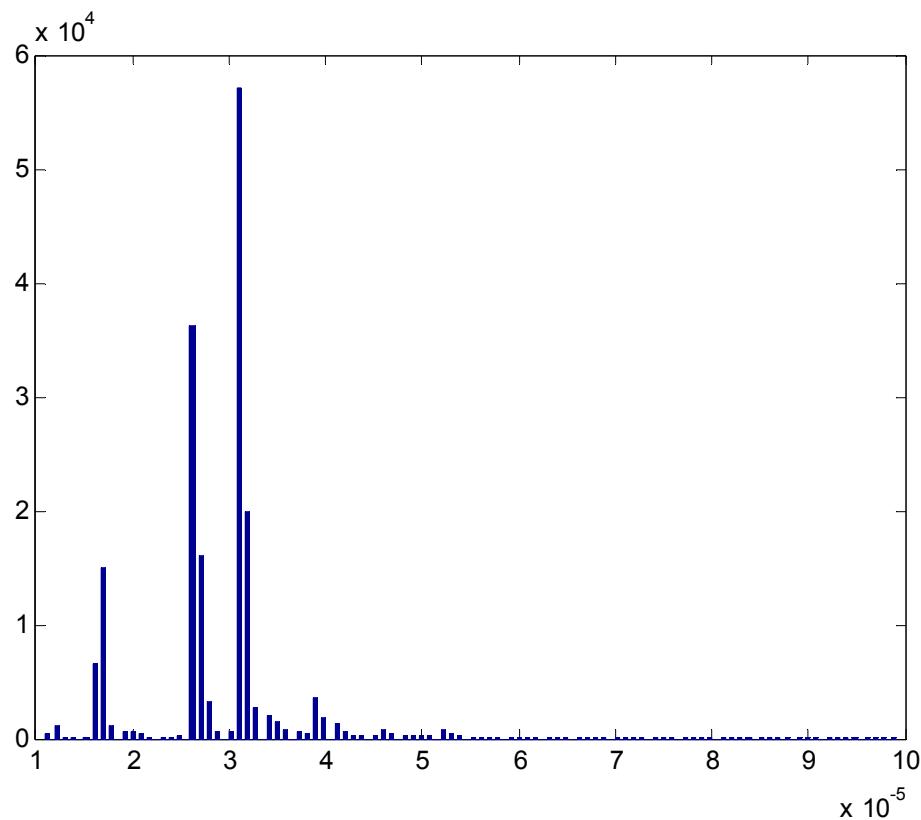
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 1,7μs, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 200)
```

El histograma es el siguiente:



**Fig. A3.5** Histograma a 300 Mbps y 64 bytes

En este gráfico se puede observar como los paquetes se concentran entre 1 y  $20\mu\text{s}$  con un valor denotado en  $31\mu\text{s}$ . Se aprecia dispersión.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\text{delta}<0.0001$  se obtienen 181800 valores (los graficados), los descartados con  $\text{delta}>0.0001$ , y se obtienen 1144 valores, es decir, un 0,6253%.

Se utiliza  $1\text{e}-4$  segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 64 bytes a 300Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 171,97 $\mu$ s y la media es de 38,90 $\mu$ s.

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a 17 $\mu$ s que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 21,79 Mbps.

### Prueba a 300 Mbps y 300 bytes

En esta prueba se generará un flujo de 300 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_300B_300M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación 5 flujos UDP con mismo destino, periódicos,
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [125000
300]

# Parar los 5 flujos UDP con una duración de 5 segundos
5 OFF 1
```

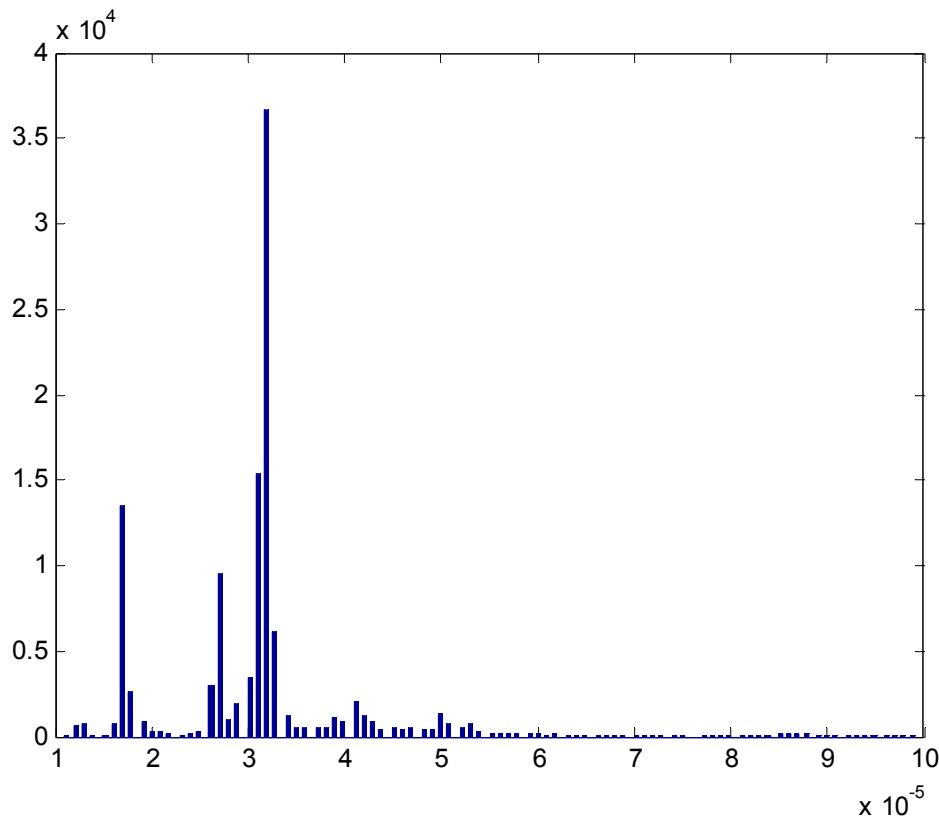
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 8 $\mu$ s, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 200)
```

El histograma es el siguiente:



**Fig. A3.6** Histograma a 300 Mbps y 300 bytes

En este gráfico se puede observar como los paquetes se concentran entre 10 $\mu$ s y 30 $\mu$ s con un valor denotado en 31 $\mu$ s. Se aprecia dispersión.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 116095 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 743 valores, es decir, un 0,6359%.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 300 bytes a 300Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de  $214,87\mu s$  y la media es de  $45,32\mu s$ .

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a  $8\mu s$  que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de  $60,37$  Mbps.

### Prueba a 300 Mbps y 1000 bytes

En esta prueba se generará un flujo de 300 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_1000B_300M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación 5 flujos UDP con mismo destino, periódicos,
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [37500
1000]

# Parar los 5 flujos UDP con una duración de 5 segundos
5 OFF 1
```

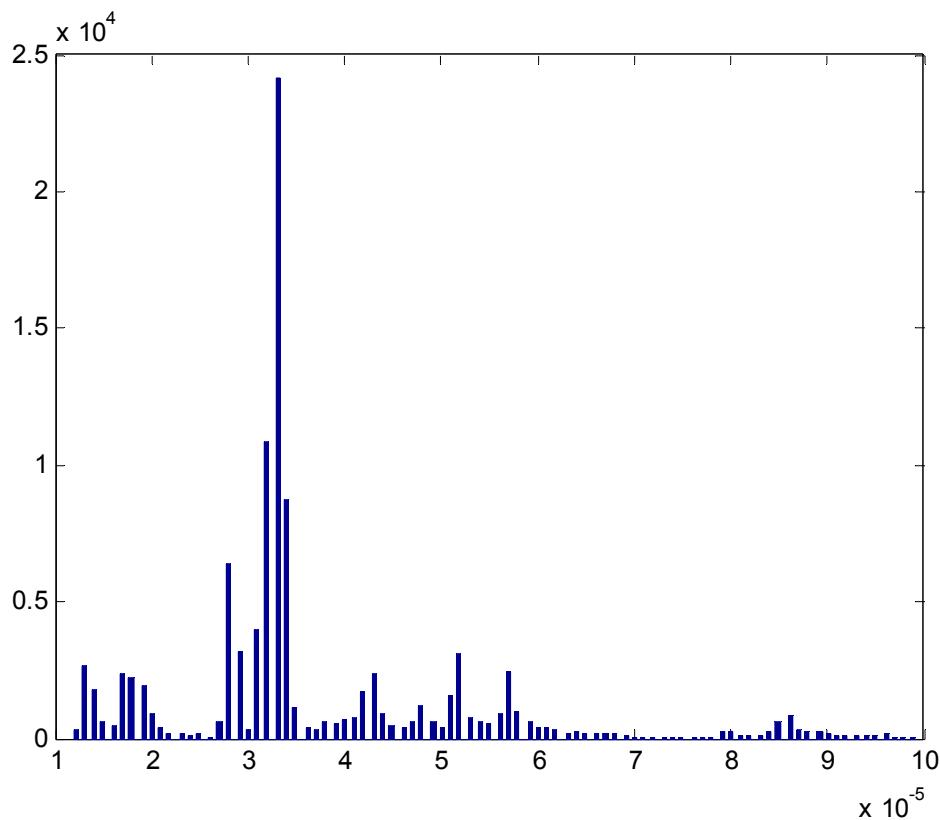
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de  $26,67\mu s$ , veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 200)
```

El histograma es el siguiente:



**Fig. A3.7** Histograma a 300 Mbps y 1000 bytes

En este gráfico se puede observar como los paquetes se concentran entre 10 $\mu$ s y 70 $\mu$ s con un valor denotado en 31 $\mu$ s. Se aprecia dispersión.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 104489 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 699 valores, es decir, un 0,6645%.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1000 bytes a 300Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 177,73 $\mu$ s y la media es de 47,54 $\mu$ s.

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a 26,67 $\mu$ s que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 175,34 Mbps.

### Prueba a 300 Mbps y 1400 bytes

En esta prueba se generará un flujo de 300 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_1400B_300M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación 5 flujos UDP con mismo destino, periódicos,
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [26785
1400]

# Parar los 5 flujos UDP con una duración de 5 segundos
5 OFF 1
```

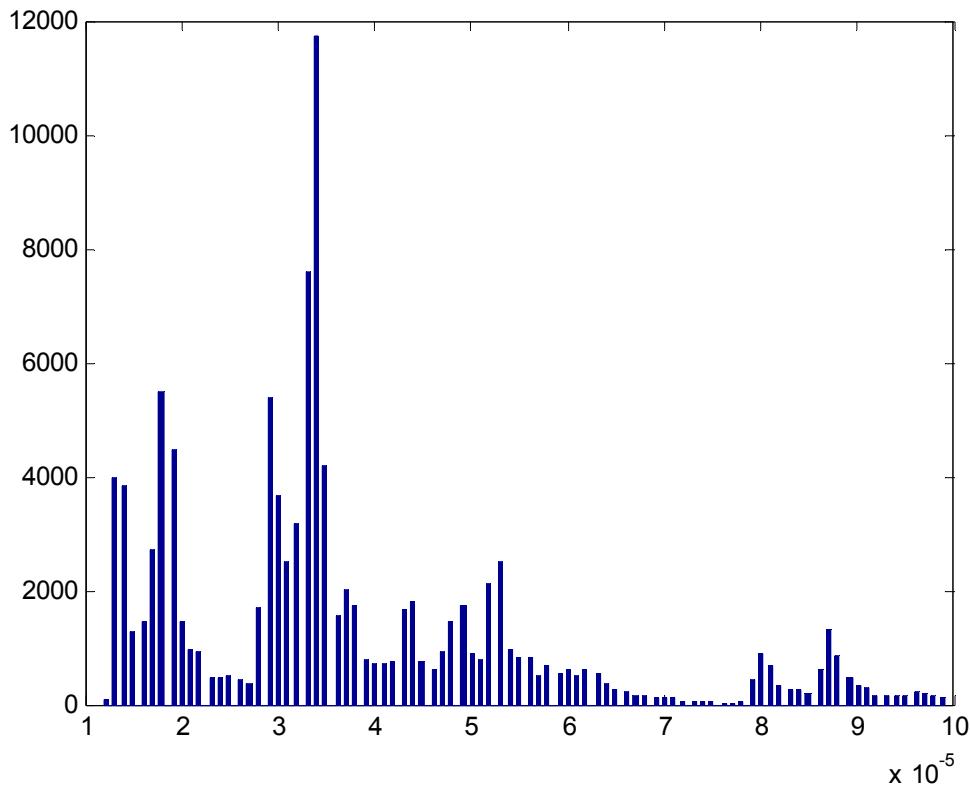
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 37,3 $\mu$ s, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 200)
```

El histograma es el siguiente:



**Fig. A3.8** Histograma a 300 Mbps y 1400 bytes

En este gráfico se puede observar como los paquetes se concentran entre  $10\mu s$  y  $100\mu s$  con un valor denotado en  $31\mu s$ . Se aprecia dispersión.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\text{delta}<0.0001$  se obtienen 109299 valores (los graficados), los descartados con  $\text{delta}>0.0001$ , y se obtienen 2227 valores, es decir, un 1,9968%.

Se utiliza  $1e-4$  segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1400 bytes a 300Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de  $116,54\mu s$  y la media es  $44,78\mu s$ .

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a  $37,3\mu s$  que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 257,61 Mbps.

### Prueba a 500 Mbps y 64 bytes

En esta prueba se generará un flujo de 300 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_64B_500M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación 5 flujos UDP con mismo destino, periódicos,
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [976562
64]

# Parar los 5 flujos UDP con una duración de 5 segundos
5 OFF 1
```

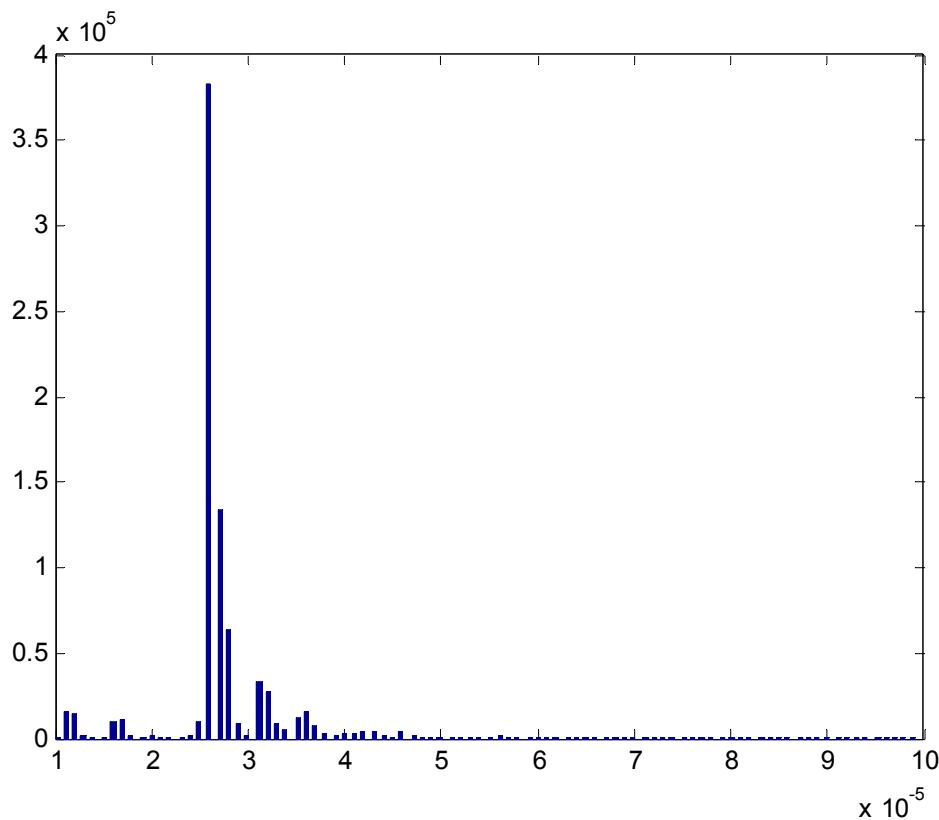
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de  $1,06\mu s$ , veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 200)
```

El histograma es el siguiente:



**Fig. A3.9** Histograma a 500 Mbps y 64 bytes

En este gráfico se puede observar como los paquetes se concentran entre  $10\mu s$  y  $50\mu s$  con un valor denotado en  $27\mu s$ . Se aprecia dispersión.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\text{delta}<0.0001$  se obtienen 811626 valores (los graficados), los descartados con  $\text{delta}>0.0001$ , y se obtienen 1784 valores, es decir, un 0,2193%.

Se utiliza  $1e-4$  segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 64 bytes a 500Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 119,52 $\mu$ s y la media es de 31,90 $\mu$ s.

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a 1,06 $\mu$ s que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 26,58 Mbps.

## Prueba a 500 Mbps y 300 bytes

En esta prueba se generará un flujo de 300 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_300B_500M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación 5 flujos UDP con mismo destino, periódicos,
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [208333
300]

# Parar los 5 flujos UDP con una duración de 5 segundos
5 OFF 1
```

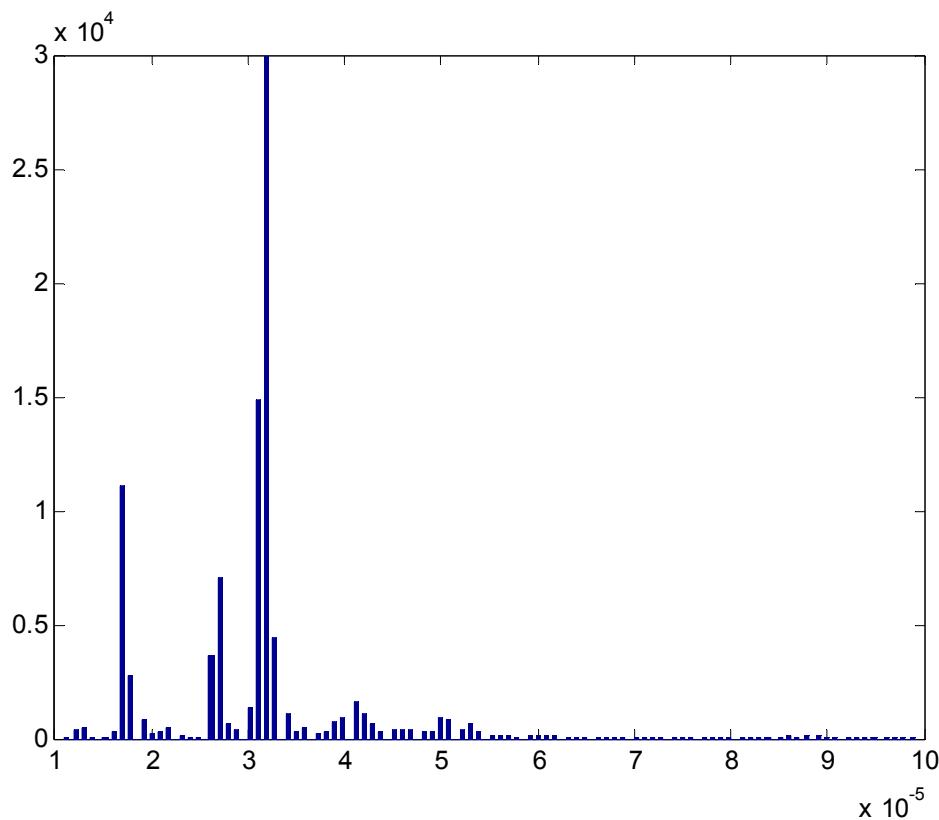
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 4,80 $\mu$ s, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 200)
```

El histograma es el siguiente:



**Fig. A3.10** Histograma a 500 Mbps y 300 bytes

En este gráfico se puede observar como los paquetes se concentran entre  $10\mu\text{s}$  y  $60\mu\text{s}$  con un valor denotado en  $31\mu\text{s}$ . Se aprecia dispersión.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\text{delta}<0.0001$  se obtienen 94928 valores (los graficados), los descartados con  $\text{delta}>0.0001$ , y se obtienen 677 valores, es decir, un 0,7081%.

Se utiliza  $1\text{e}-4$  segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 300 bytes a 500Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 213,53 $\mu$ s y la media es de 45,26 $\mu$ s.

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a 4,80 $\mu$ s que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 60,45 Mbps.

### Prueba a 500 Mbps y 1000 bytes

En esta prueba se generará un flujo de 300 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_1000B_500M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación 5 flujos UDP con mismo destino, periódicos,
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [62500
1000]

# Parar los 5 flujos UDP con una duración de 5 segundos
5 OFF 1
```

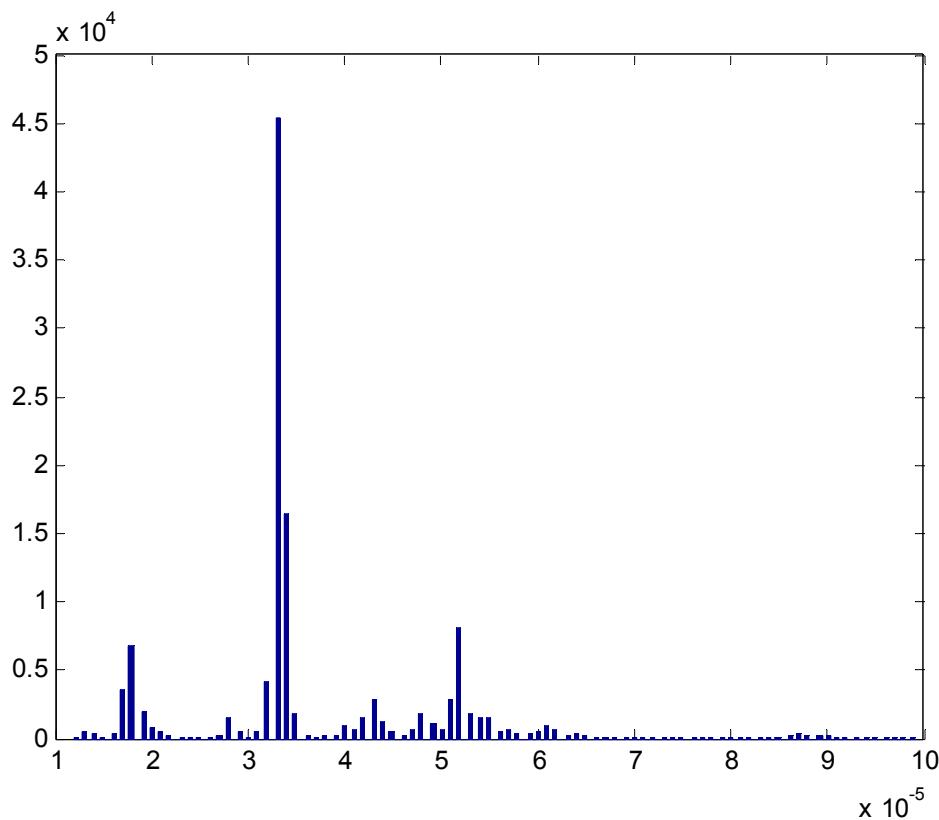
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 16 $\mu$ s, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 200)
```

El histograma es el siguiente:



**Fig. A3.11** Histograma a 500 Mbps y 1000 bytes

En este gráfico se puede observar como los paquetes se concentran entre  $10\mu\text{s}$  y  $60\mu\text{s}$  con un valor denotado en  $33\mu\text{s}$ . Se aprecia dispersión.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\text{delta}<0.0001$  se obtienen 122264 valores (los graficados), los descartados con  $\text{delta}>0.0001$ , y se obtienen 745 valores, es decir, un 0,6056%.

Se utiliza  $1\text{e}-4$  segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1000 bytes a 500Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 1,6ms y la media es de 51,35μs.

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a 16μs que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 162,33 Mbps.

## Prueba a 500 Mbps y 1400 bytes

En esta prueba se generará un flujo de 300 Mbps y 300 bytes de datos útiles. El siguiente comando generará dicho flujo:

```
mgen input 1_flujo_1400B_500M.mgn
```

Y el script generador es el siguiente:

```
# Inicio script MGEN
# Creación 5 flujos UDP con mismo destino, periódicos,
0.0 ON 1 UDP SRC 5501 DST 147.83.118.250/5500 PERIODIC [44642
1400]

# Parar los 5 flujos UDP con una duración de 5 segundos
5 OFF 1
```

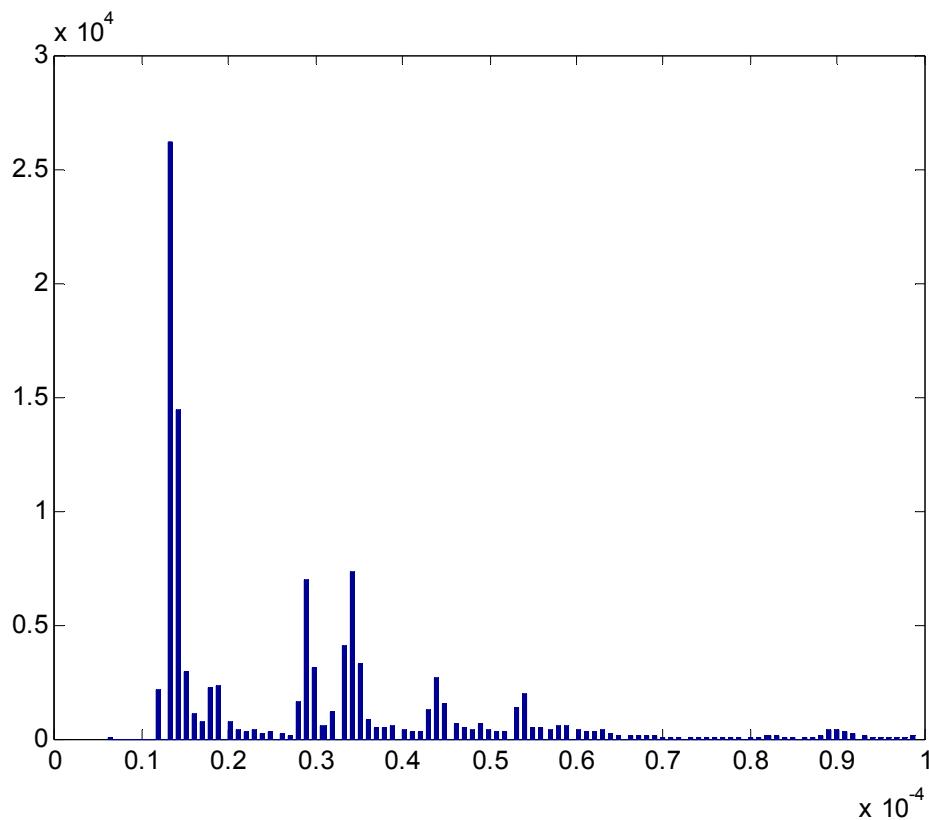
En negrita a parecen los parámetros que se deben ir cambiando para obtener la velocidad deseada (paquetes/seg) y el tamaño del paquete de la prueba.

Teóricamente se debería obtener un delta time de 22,4μs, veamos con Matlab que histograma aparece y que media se obtiene.

El histograma con Matlab se genera con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 200)
```

El histograma es el siguiente:



**Fig. A3.12** Histograma a 500 Mbps y 1400 bytes

En este gráfico se puede observar como los paquetes se concentran entre 10 $\mu$ s y 100 $\mu$ s con un valor denotado en 14 $\mu$ s. Se aprecia dispersión.

Por otro lado, hay que tener en cuenta que la anterior grafica provoca que muchos valores no se ponderen para el resultado final ya que filtramos los datos inferiores a 0,0001 segundos. Hay un comando que cambiando el operador nos permite saber cuántos valores están por encima y cuantos por debajo:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 106076 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 2533 valores, es decir, un 2,3322%.

Se utiliza 1e-4 segundos ya que así abarcamos los datos inferiores a ese valor.

Para obtener una estadística de todas las pruebas, se realiza un cálculo de la desviación estándar (std) y media (mean) de cada una. El caso de 1400 bytes a 500Mbps estos valores se obtienen con los siguientes comandos:

```
std (delta)
mean (delta)
```

La desviación estándar es de 1,3ms y la media es de 46,82 $\mu$ s.

En este caso, este resultado no proporciona un resultado correcto, ya que la media debería ser parecida a 22,4 $\mu$ s que es la inversa de los paquetes por segundo generados por MGEN. Esta media proporciona una velocidad de 246,39 Mbps.

## IV. Configuración dispositivos

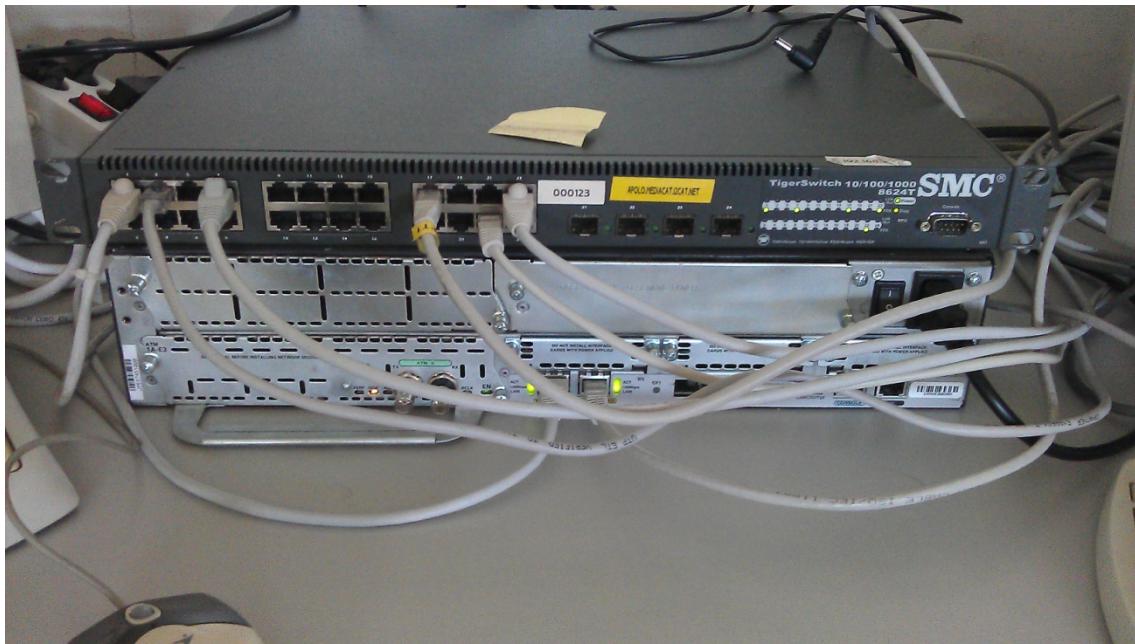
### Escenario

El PC servgenmonll (cliente) generará el tráfico MGEN de 60 segundos hacia RadPC (servidor). Para analizar ese mismo tráfico, hay que hacerlo llegar a la tarjeta DAG. El switch tiene una característica que permite configurarlo para que uno de sus puertos actúe como port mirroring, así todo el tráfico entrante al switch por parte del router será replicado por el puerto donde está conectado la DAG.

Tanto los hosts (con radclock), la DAG (dagclock) y el router (con ntp) están sincronizados por NTP contra el servidor CESCA de l'Anella Científica ([ntp.cesca.cat](http://ntp.cesca.cat)) para tener una sincronización estable y estándar.



**Fig. A4.1** El escenario completo



**Fig. A4.2** Conexiones del router cisco 3700 y el switch SMC

### Configuración Cisco: Full Netflow y NAT

Es necesario aclarar que, al tener diferente direccionamiento y estando conectado a la red de la escuela, es necesario la traducción de direcciones (NAT) y una ruta por defecto (0.0.0.0) tanto para salir al exterior como para la conectividad entre las máquinas del escenario.

A continuación se muestra la configuración de Full Netflow y NAT:

1. Configurar Full Netflow
2. Habilitar Full Netflow en la interfaz a auditar
3. Configurar ruta por defecto
4. Configurar NAT
5. Habilitar NAT en las interfaces
6. Configurar NTP
7. Guardar la configuración

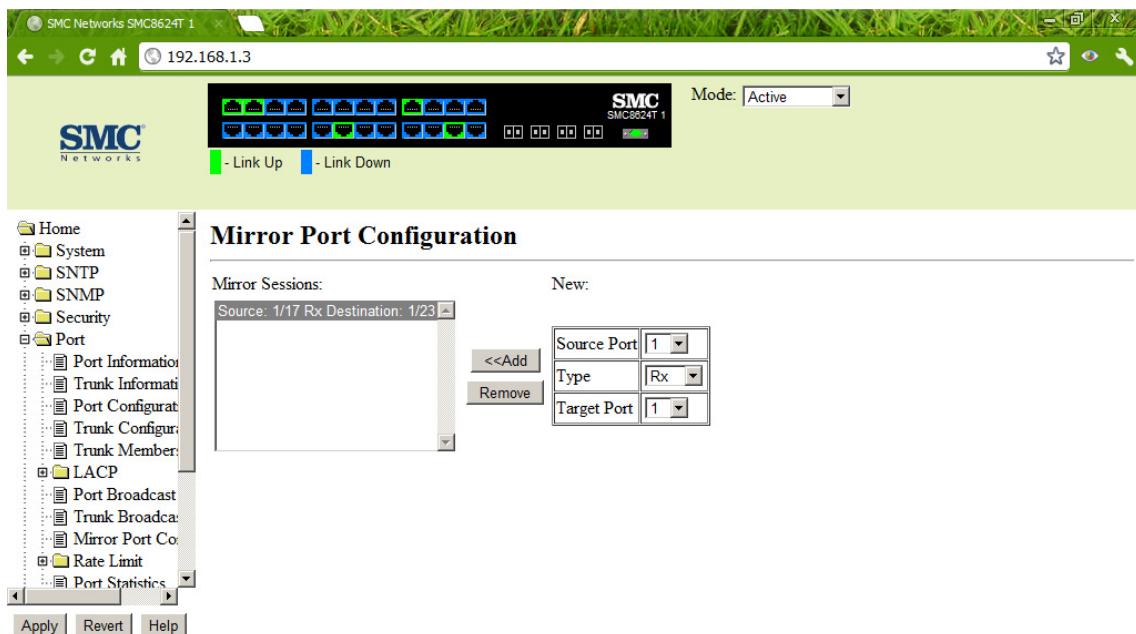
**Configuración Full Netflow****Configuración de la interfaz origen de NetFlow***ip flow-export source <interfaz>***Configuración de la versión de NetFlow***ip flow-export version <version>***Configuración de la IP/puerto destino del colector NetFlow***ip flow-export destination <dirección IP> <puerto>*Cisco3700(config)#*ip flow-export source FastEthernet0/1*Cisco3700(config)#*ip flow-export version 5*Cisco3700(config)#*ip flow-export destination 147.83.118.237 9910***Habilitar Full NetFlow en la interfaz a auditar***ip route cache flow*Cisco3700(config)#*int fa0/1*Cisco3700(config-if)#*ip route-cache flow***Configurar ruta por defecto al gateway de la escuela***ip route <red origen> <red destino> <ip gateway>*Cisco3700(config)#*ip route 0.0.0.0 0.0.0.0 147.83.118.1***Configurar NAT***ip nat inside source static <ip a traducir> <ip traducida>*Cisco3700(config)#*ip nat inside source static 10.0.13.101  
147.83.118.241***Habilitar NAT en las interfaces***ip nat <sentido de nat>*Cisco3700(config)#*int fa0/1*Cisco3700(config)#*ip nat inside*Cisco3700(config)#*int fa0/0*Cisco3700(config)#*ip nat outside***Configurar NTP***ip server <ip server> prefer*Cisco3700(config)#*ntp server ntp.cesca.cat prefer***Guardar la configuración***wr*Cisco3700#*wr*

## Configuración Switch SMC: Port mirroring

Una vez realizada la configuración del router, hay que habilitar el port mirroring en el switch. Éste soporta tanto la configuración vía telnet como por web, se explicará vía web por ser el método más intuitivo.

Al tener un direccionamiento diferente al del escenario, ha sido necesario conectar un portátil para poder gestionarlo. Los pasos a seguir son los siguientes:

1. Entrar al switch vía web
2. Ir a Home->Port->Mirror Port Configuration
3. Configurar puertos
4. Aplicar configuración



**Fig. A4.3** Configuración port mirroring switch SMC

Una vez entramos con la ip 192.168.1.3 (admin/admin), se va a la sección Mirror Port Configuration y se especifica el puerto origen de los datos, el tipo de tráfico a replicar y finalmente a que puerto destino se quiere copiar. En nuestro caso hemos replicado lo que se recibe por el puerto 17 (lo que recibe el switch del router) y se ha replicado por el 23 (destino a la DAG).

## Configuración tarjeta DAG 4.3GE

La tarjeta de alta precisión DAG, también debe configurarse adecuadamente para que pueda operar de la manera que necesitamos. El proceso es el siguiente:

1. Activar tarjeta DAG
2. Revisar configuración
3. Configurar recepción de paquetes y transmisión de paquetes
4. Cambiar el firmware de la tarjeta para que permita recibir y transmitir
5. Sincronizar con NTP
6. Capturar y exportar a .erf

**Activar tarjeta DAG**

```
servgenmonI:/home#dagload
```

**Revisar configuración**

```
servgenmonI:/home#dagfour default
```

```
linkA nonic noeql norxpkts notxpkts crclong=1518 enablea
linkB nonic noeql norxpkts notxpkts crclong=1518 enableb
packet varlenslen=48 noalign64
packetA drop=0
packetB drop=0
pcix    133MHz 64-bit nodrop routessource=stream0 buf=512MiB
rxstreams=1 txstreams=1 mem=496:16
```

**Configurar recepción de paquetes y transmisión de paquetes**

```
servgenmonI:/home#dagfour -d /dev/dag0 rxpkts txpkts
```

```
linkA nonic noeql rxpkts xpkts crclong=1518 enablea
linkB nonic noeql rxpkts txpkts crclong=1518 enableb
packet varlenslen=48 noalign64
packetA drop=0
packetB drop=0
pcix    133MHz 64-bit drop routessource=stream0 buf=512MiB
rxstreams=1 txstreams=1 mem=496:16
```

**Cambiar el firmware de la tarjeta para que permita recibir y transmitir**

```
servgenmonI:/home#dagrom -p (cambia el * de stable a current)
servgenmonI:/home#dagrom -x (visualiza roms)
```

```
current: edag43epci_terf_v2_5 2v1000ff896 2004/11/26 14:55:56 *
```

```
stable: edag43epci_erf_v2_9 2v1000ff896 2004/04/27 10:26:32
```

```
Card Serial: 5335
```

**Sincronizar con NTP**

```
servgenmonI:/home#dagclock -d /dev/dag0 none overin
```

```
muxin   over
muxout  none
status   Synchronised Threshold 11921ns Failures 5 Resyncs 3
error    Freq 1000ppb Phase 1000ns Worst Freq 2509281ppb Worst
Phase 385575000ns
crystal Actual 99998812Hz Synthesized 67108864Hz
input    Total 931579 Bad 3 Singles Missed 0 Longest Sequence
Missed 502006
```

```
start   Thu Jun  7 18:34:54 2012
host    Tue Jul  3 11:25:53 2012
dag     Tue Jul  3 11:25:53 2012
TSC    6662641321958302
```

**Capturar y exportar a erf**

```
servgenmonI:/home#dagsnap -d /dev/dag0 -o ejemplo.erf -v
```

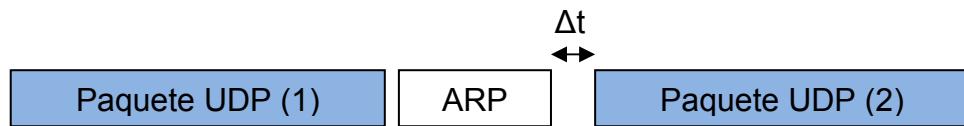
De este modo se configura la tarjeta para operar sólo en recepción. La captura exige que le describas que dispositivo se utilizará ( $-d$ ; device), que nombre de archivo se usará ( $-o$ ; output) y que imprima por pantalla los datos más significativos al momento ( $-v$ ; verbose).

### Términos a tener en cuenta

Al realizar la prueba nos hemos dado cuenta que al crear las gráficas e histogramas a partir del delta time de Wireshark se generaban errores, ya que la diferencia temporal entre paquetes no sólo eran de la transferencia entre cliente y servidor sino también de otros paquetes que pasaban por la interfaz auditada, con lo que nos era más preciso escoger la diferencia del tiempo absoluto de los paquetes de la conversión en cuestión. Esto se realiza filtrando la captura por el flujo deseado y exportando a una nueva captura, de este modo, los paquetes que se entrelazaban con los necesarios se eliminan.

Los siguientes esquemas demuestran gráficamente como cambia la diferencia entre las dos capturas:

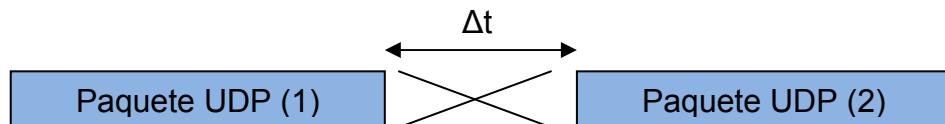
Captura original:



**Fig. A4.4** Diferencia temporal sin filtrar

En este caso, aunque se filtre por UDP y puerto, si se exporta a CSV, el paquete ARP desaparecerá pero el delta time del paquete UDP 2 seguirá siendo respecto al paquete ARP con lo que es una medida errónea.

Captura guardada nuevamente (Save As):



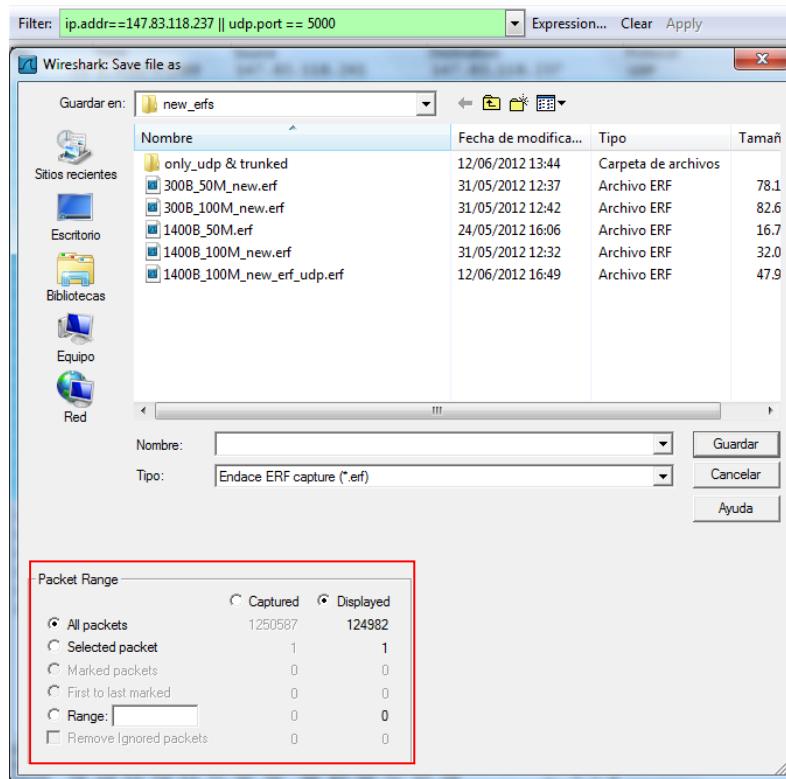
**Fig. A4.5** Diferencia temporal filtrada

Si se filtra por UDP/puerto y se guarda en una nueva captura, el tiempo absoluto se corrige, por lo que el delta time será correcto, es decir, al exportarlo a una nueva captura, el paquete ARP desaparece y el delta time del paquete UDP 2 será respecto al paquete UDP 1.

Por otro lado, al tratarse de captura con un gran volumen de paquetes (alrededor de 1,2 millones) existe un problema a la hora de exportarlo a CSV, ya que éste sólo soporta como máximo un millón por archivo, así que, a la hora de exportar se debe dividir en tantas partes como millones se tengan, en este caso dos.

El ciclo de los archivos deseados es el siguiente:

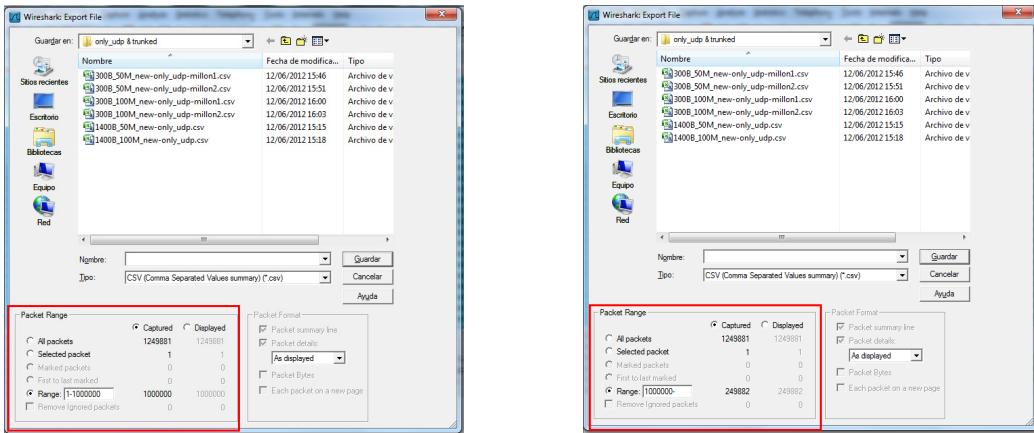
DAG → ejemplo\_capturado.erf → Abrir con Wireshark: filtro UDP y puerto → Save As nanosecond pcap (displayed) → ejemplo\_filtrado.pcap



**Fig. A4.6 Filtro displayed Wireshark**

Abrir ejemplo\_filtrado pcap con Wireshark → Export .csv → Packet Range → Range: 1-1000000.

De nuevo, volver a exportar con la diferencia entre 1000000 y el total de paquetes con: Export .csv → Packet Range → Range: 1000000-



**Fig. A4.7 Filtros de rango Wireshark**

Posteriormente se deben exportar a Matlab para que puedan ser tratados. Al tener que trabajar con dos archivos, Matlab tiene la posibilidad de concatenar vectores con el siguiente comando:

Si el millón 1 se llama delta1 y lo restante delta2:

```
delta=[delta1;delta2]
```

Una vez tengamos los dos vectores en uno, ya es posible graficar y trabajar con todo el grueso de paquetes.

El tiempo de transmisión mínimo para los dos casos que se van a estudiar son los siguientes:

Para 300 bytes de datos, hay que sumarle: 8 de cabecera UDP, 20 de cabecera IP, 14 cabecera Ethernet (SA+DA+L/T)= 342 bytes. Pero el tiempo de transmisión también debe incluir el FCS, el preámbulo y el IFG, lo que nos hace agregar 4+8+12 y esto hace un total de 342+24 = 366 bytes.

Por tanto = Tamaño total del paquete/Velocidad de la línea= $366 \times 8 / 100e6 = 29.28\mu s$

Pero en 100Mbps, en recepción el IFG puede ser 0, con lo que sólo se contaría el sólo el FCS+Preamble y quedaría  $(342+4+8)8 / 100e6 = 28.32\mu s$

Para 1400 bytes de datos, hay que sumarle: 8 de cabecera UDP, 20 de cabecera IP, 14 cabecera Ethernet (SA+DA+L/T)= 1442 bytes. Pero el tiempo de transmisión también debe incluir el FCS, el preámbulo y el IFG, lo que nos hace agregar 4+8+12 y esto hace un total de 1442+24 = 1466 bytes.

Por tanto = Tamaño total del paquete/Velocidad de la línea= $1466 \times 8 / 100e6 = 117.28\mu s$

Pero en 100Mbps, en recepción el IFG puede ser 0, con lo que sólo se contaría el sólo el FCS+Preamble y quedaría  $(1442+4+8)8 / 100e6 = 116.32\mu s$

Estos valores son los umbrales mínimos que no deben sobrepasar los delta time de los paquetes.

## Pruebas Full Netflow con MGEN, DAG y Nfsen

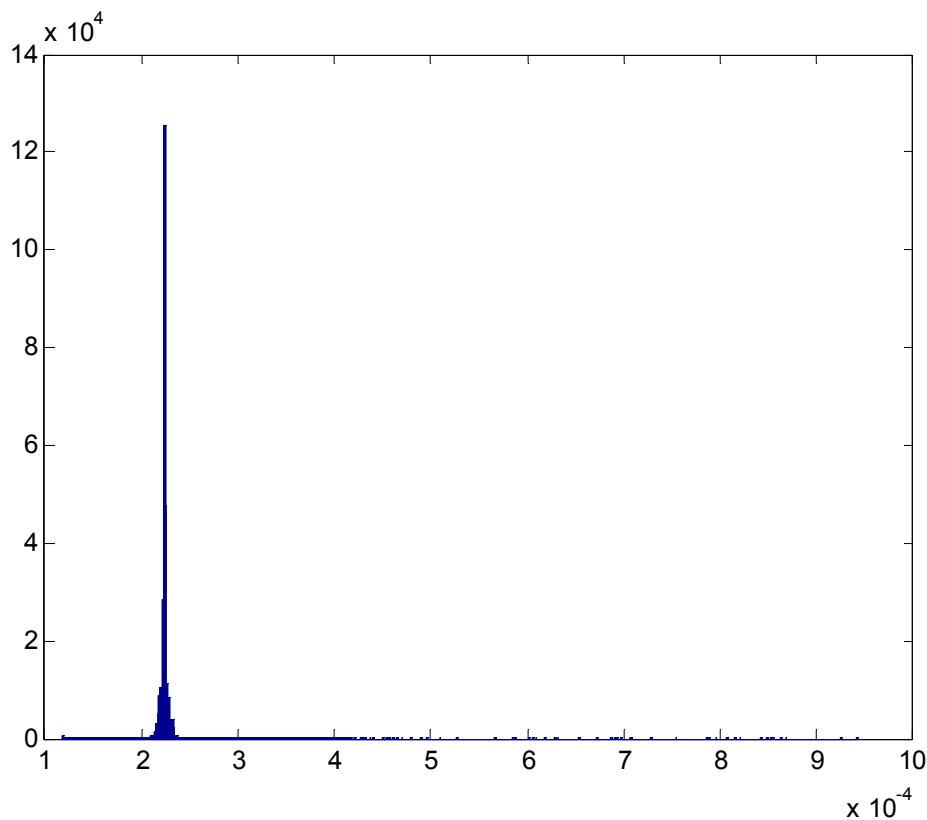
### Prueba de 1400 bytes a 50 Mbps

En esta prueba se ha cotejado que tanto Nfsen, la captura de la DAG y el registro de salida de MGEN han obtenido el mismo resultado de paquetes totales recibidos, siendo estos: 267858 paquetes con una velocidad obtenida de 50Mbps. Una medida muy precisa y exacta.

En este caso Netflow muestrea un 100% de los paquetes totales recibidos por la DAG, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta,500)
```



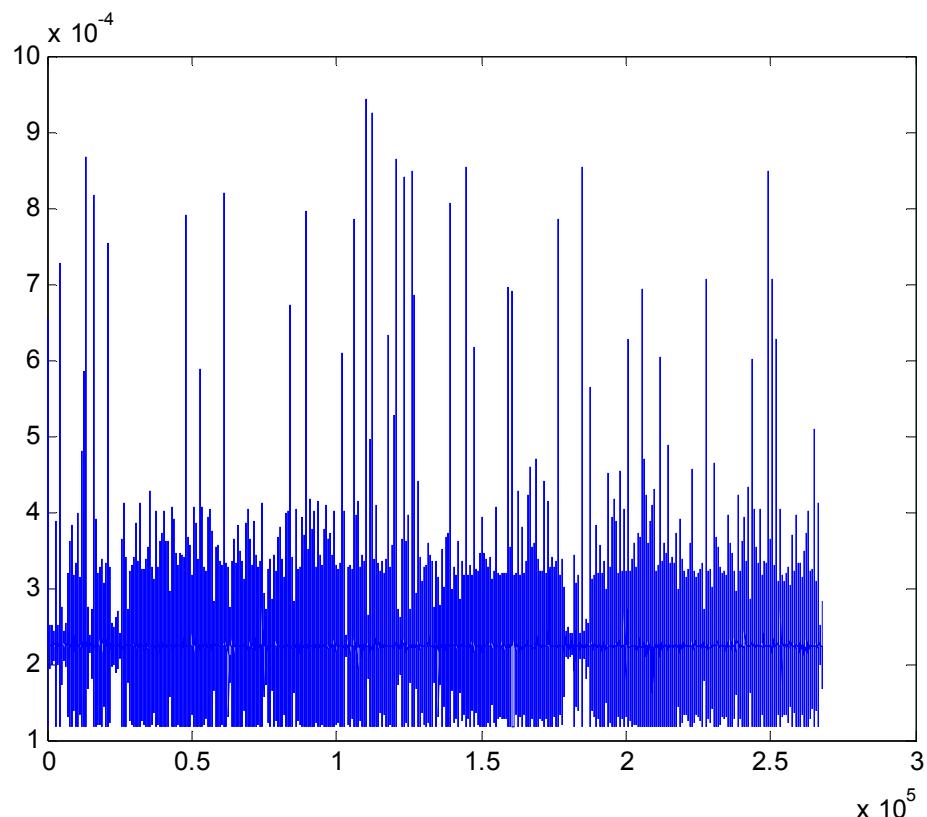
**Fig. A5.1** Histograma a 50 Mbps y 1400 bytes

Se puede observar como hay algo de dispersión, concentrándose la mayoría de paquetes en  $223,98\mu s$ , siendo la media y la desviación estándar:  $223,98 \mu s$  y  $11,39\mu s$  respectivamente.

Primeramente realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

*plot (delta)*

Con este comando obtenemos la siguiente gráfica:

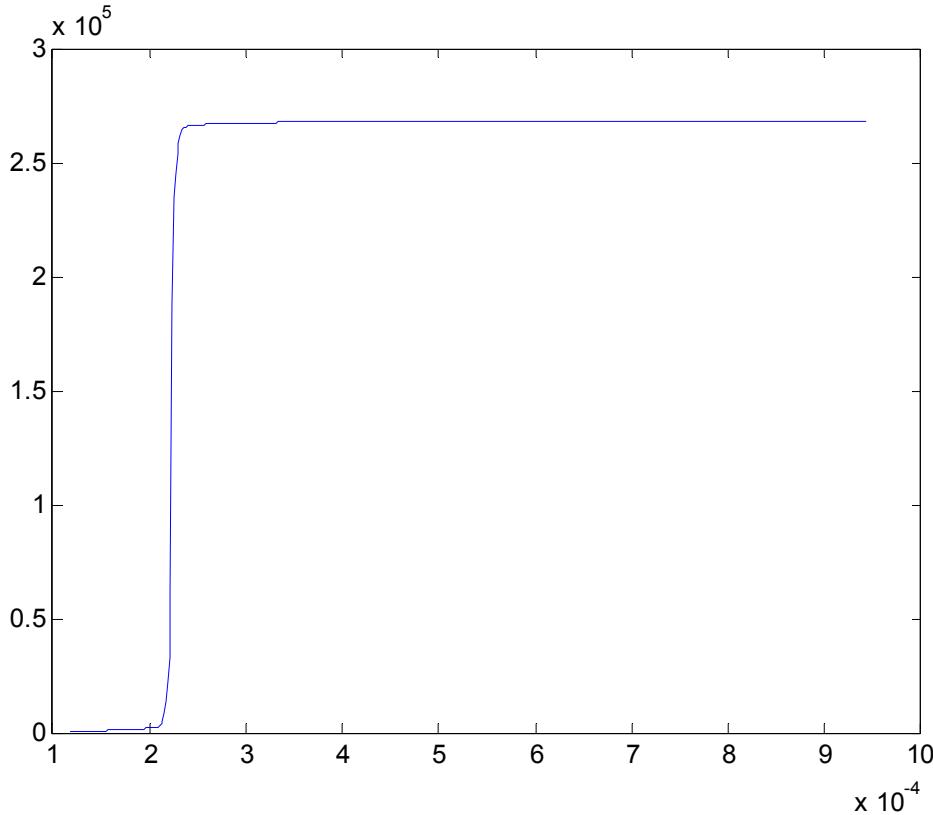


**Fig. A5.2** Histograma a 50 Mbps y 1400 bytes

Observamos que el valor máximo no supera los  $943,58\mu s$  ni están por debajo de  $117,21\mu s$ .

Seguidamente graficaremos los valores de la suma acumulativa en función de los valores de la prueba:

```
[N,X]=hist(delta,500)
plot(X,cumsum(N))
```



**Fig. A5.3** Suma acumulativa a 50 Mbps y 1400 bytes

En este gráfico se puede destacar que la suma acumulativa hasta el valor medio es pequeña y a partir del este, la suma crece hasta llegar casi a la totalidad de paquetes recibidos.

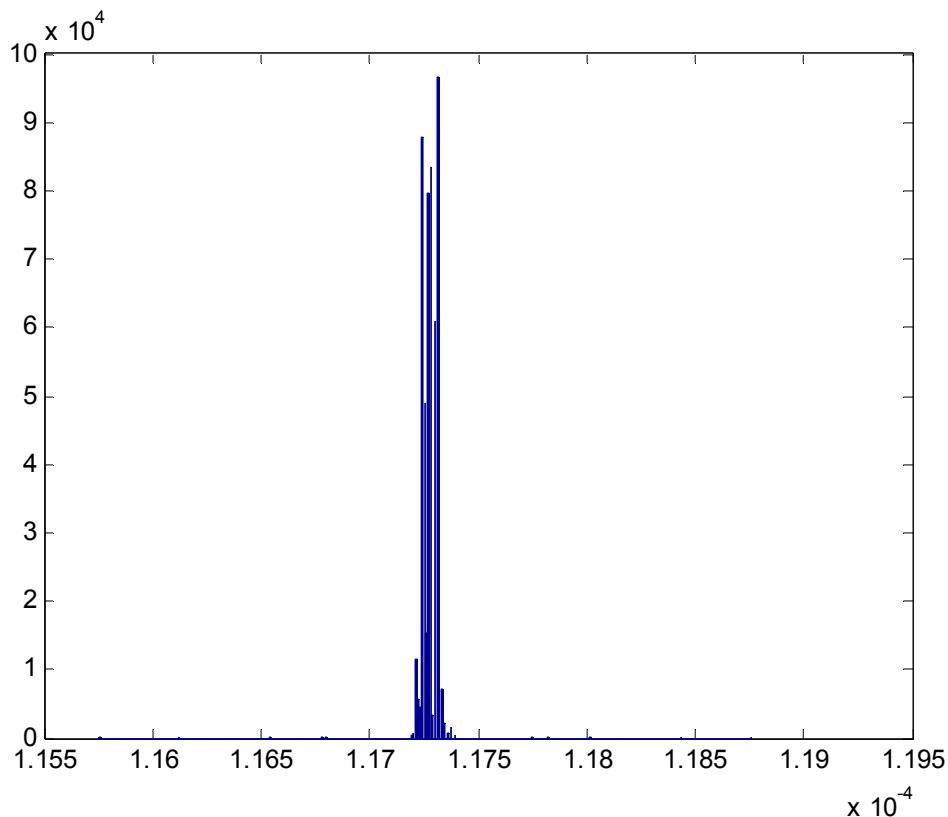
### Prueba de 1400 bytes a 100 Mbps

En esta prueba se ha cotejado que tanto Nfsen como la captura de la DAG han obtenido el mismo resultado de paquetes totales recibidos, siendo estos: 511707 paquetes con una velocidad obtenida de 97 Mbps. En cambio, el registro de salida de MGEN nos señala 507286 paquetes recibidos, algo inferior a lo capturado por la DAG, lo que nos lleva tener un 0,86% de pérdidas.

En este caso Netflow muestrea un 100% de los paquetes totales recibidos por la DAG, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00012)),500)
```



**Fig. A5.4** Plot a 100 Mbps y 1400 bytes

Para obtener una gráfica más precisa, se debe afinar más la resolución y despreciar las muestras superiores a 0.00012 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

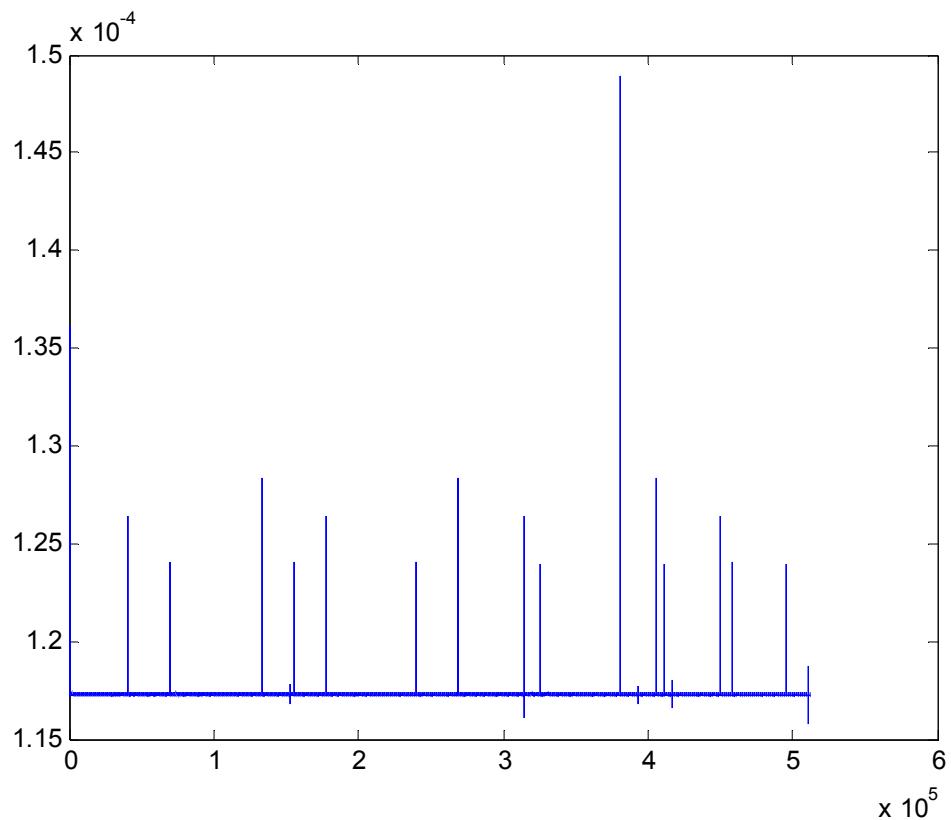
```
size(find(delta<0.00012)))
size(find(delta>0.00012))
```

Con  $\text{delta} < 0.00012$  se obtienen 511676 valores (los graficados), los descartados con  $\text{delta} > 0.00012$ , y se obtienen 29 valores, es decir un 0,005667%.

Se puede apreciar como hay muy poca dispersión de paquetes y la mayoría se concentran en 117,27μs, siendo la media y la desviación estándar: 117,27μs y 81,92ns respectivamente.

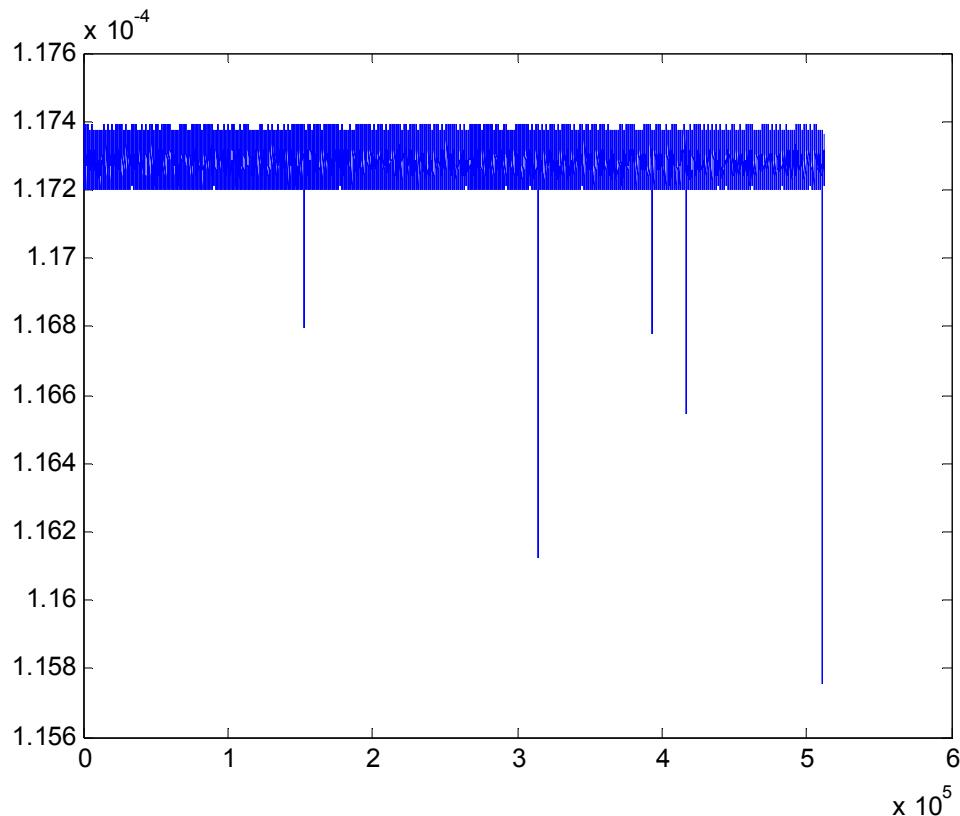
Seguidamente se realiza una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo.

```
plot(delta)
```



**Fig. A5.5** Plot a 100 Mbps y 1400 bytes

Si ampliamos podemos observar mejor la gráfica y ver si los valores mínimos cumplen con el tiempo de transmisión:



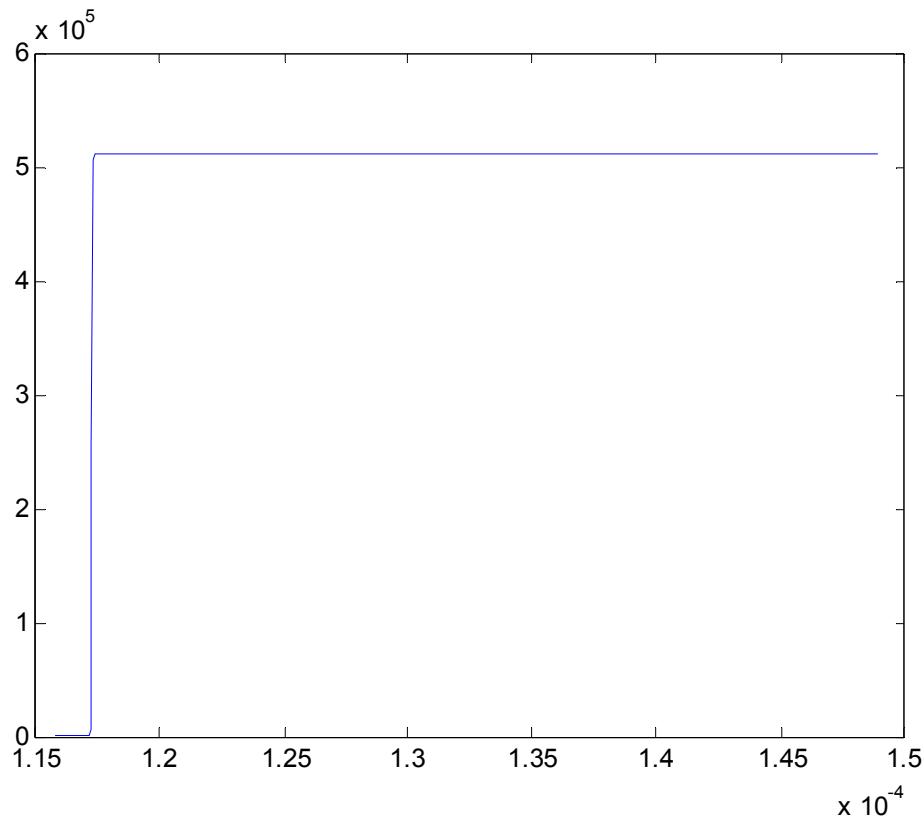
**Fig. A5.6** Plot ampliado

Observamos que el valor máximo no supera los  $148,92\mu s$  segundos ni están por debajo de  $115,75\mu s$ .

En valores más bajos que la media: Si miramos la captura de wireshark podemos comprobar cómo la diferencia del paquete anterior tiene entre medio otro paquete, que filtramos, pero que el tiempo absoluto no corrige y hace que se produzcan intervalos inferiores. Por otro lado, la precisión del reloj de emisión también interviene en la medida.

A continuación graficaremos los valores de la suma acumulativa en función de los valores de la prueba:

```
[N,X]=hist(delta,500)  
plot(X,cumsum(N))
```



**Fig. A5.7** Suma acumulativa a 100 Mbps y 1400 bytes

Como vemos, existen muy pocos valores desde cero hasta  $117\mu s$ , por eso el acumulativo es mínimo. A partir de ese valor, la cantidad máxima de muestras forman un pico muy abrupto y poco suavizado debido a la alta frecuencia de valores en esa zona.

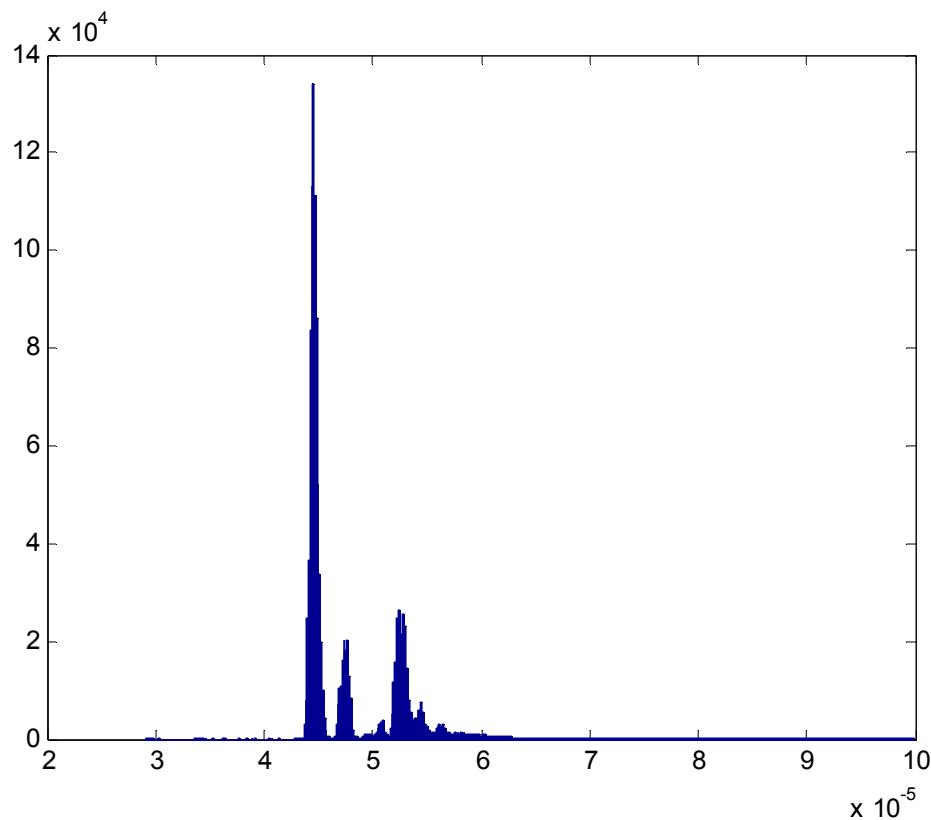
### Prueba de 300 bytes a 50 Mbps

En esta prueba se ha cotejado que tanto Nfsen como la captura de la DAG han obtenido el mismo resultado de paquetes totales recibidos, siendo estos: 1249881 paquetes con una velocidad obtenida de 51 Mbps. En cambio, el registro de salida de MGEN nos señala 1191176 paquetes recibidos, algo inferior a lo capturado por la DAG, lo que nos lleva tener un 4,69% de pérdidas.

En este caso Netflow muestrea un 100% de los paquetes totales recibidos por la DAG, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico:

```
hist(delta(find(delta<0.0001)), 500)
```



**Fig. A5.8** Histograma a 50 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.0001 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

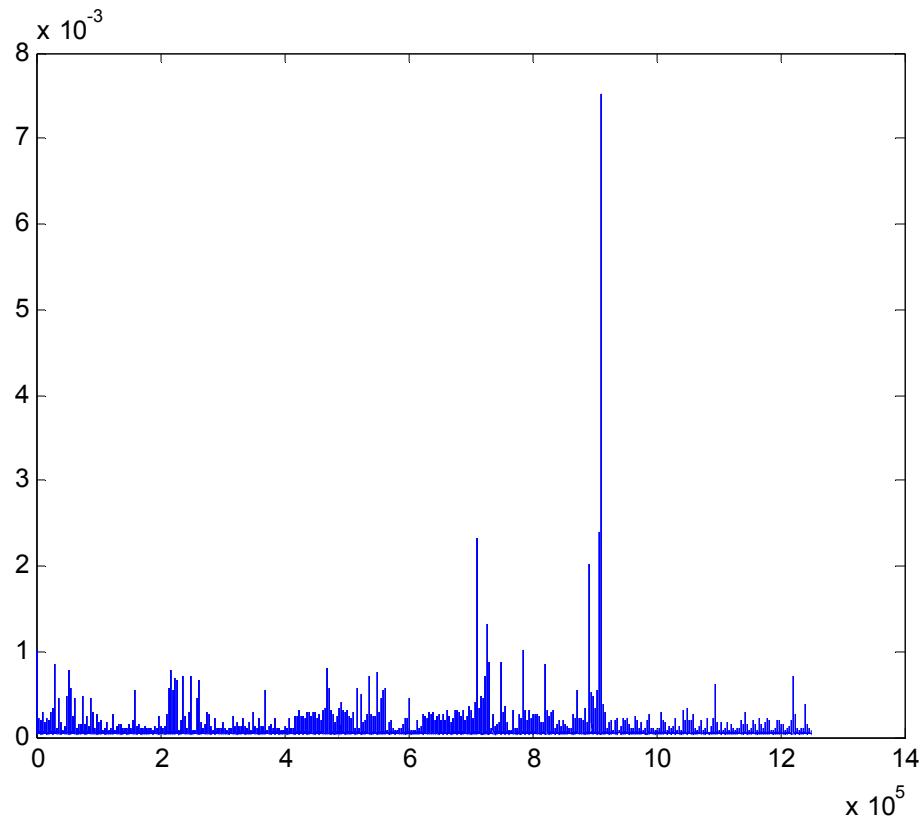
```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 1248692 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 1188 valores, es decir un 0,095049%.

Como se puede ver en la gráfica ampliada, existe dispersión de paquetes y la mayoría se concentran en  $44.6 \mu\text{s}$ , siendo la media y la desviación estándar:  $48.01 \mu\text{s}$  y  $11.22 \mu\text{s}$  respectivamente.

Seguidamente se realiza una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo.

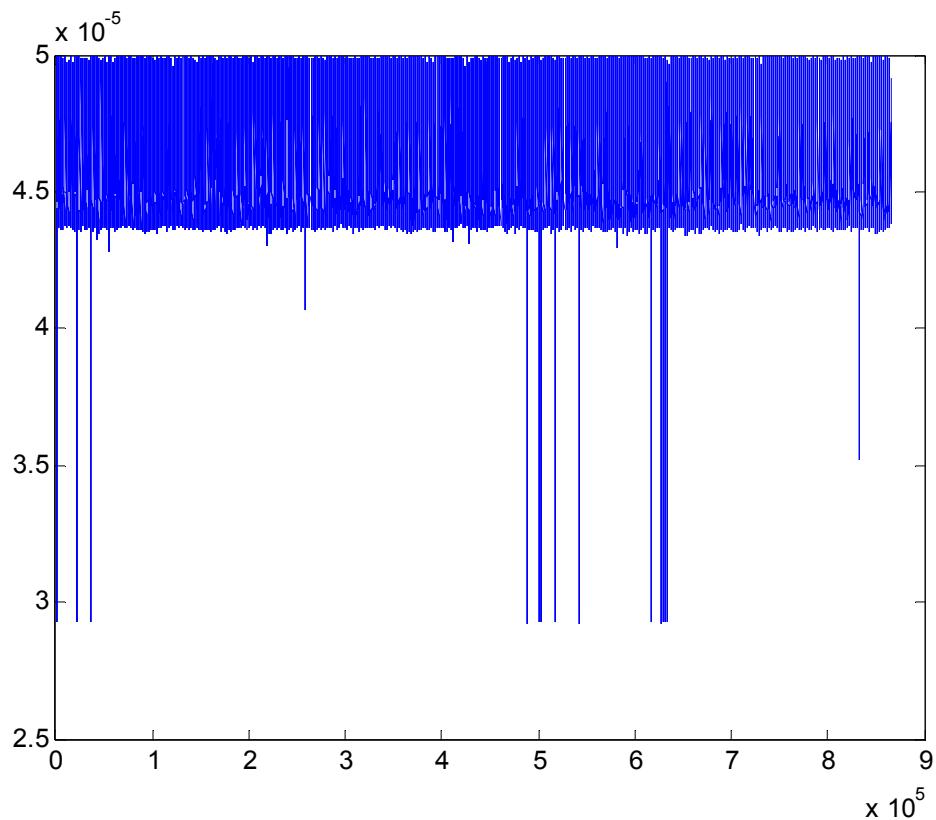
```
plot(delta)
```



**Fig. A5.9** Plot a 50 Mbps y 300 bytes

Observamos que el valor máximo no supera los 7,510126ms y el mínimo 29,20μs. Lo que nos lleva a no obtener anomalías. En la siguiente gráfica afinamos más sobre estos datos con valores que van más rápidos que la velocidad media pero que están dentro del tiempo de transmisión de línea:

Con `plot(Z(find(Z<0.00005)))`

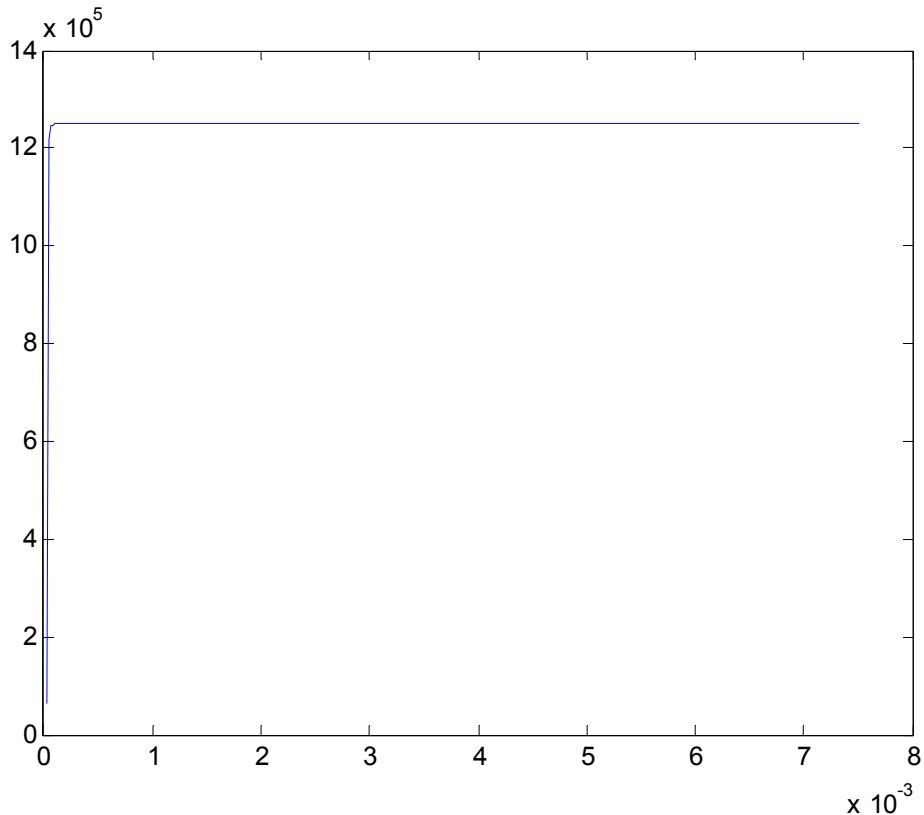


**Fig. A5.10** Plot ampliado

En este caso, al analizar detenidamente esta anomalía a partir de la captura de Wireshark, nos damos cuenta que no hay ningún paquete entrelazado con los demás, con lo que se puede suponer que aparte de tener el router al 92% de CPU podemos achacar el problema a imprecisiones del reloj o del propio software generador MGEN.

A continuación graficaremos los valores de la suma acumulativa en función de los valores de la prueba:

```
[N,X]=hist(delta,500)  
plot(X,cumsum(N))
```



**Fig. A5.11** Suma acumulativa 50 Mbps y 300 bytes

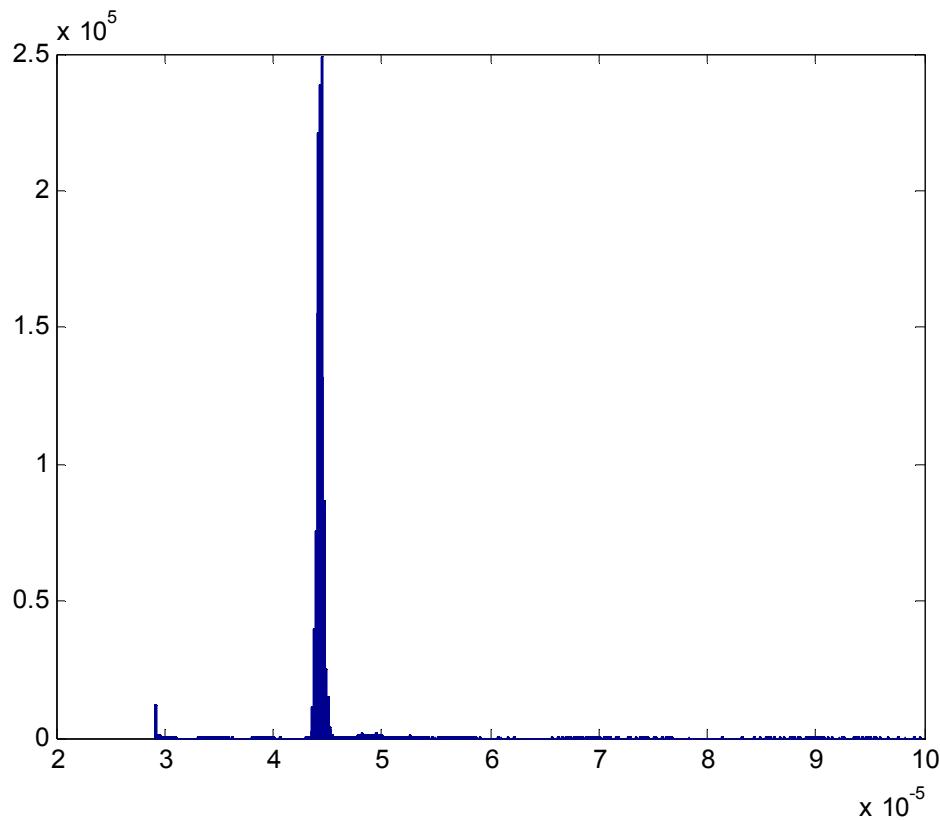
Como vemos, la cantidad máxima de valores forma un pico muy abrupto y poco suavizado debido a la alta frecuencia de valores en esa zona (a partir de 43,8 $\mu$ s) y a los despreciables valores poco acumulativos posteriores a este.

## Prueba de 300 bytes a 100 Mbps

En esta prueba se ha cotejado que tanto Nfsen como la captura de la DAG han obtenido el mismo resultado de paquetes totales recibidos, siendo estos: 1322140 paquetes con una velocidad obtenida de 57,8 Mbps. En cambio, el registro de salida de MGEN nos señala 1307937 paquetes recibidos, algo inferior a lo capturado por la DAG, lo que nos lleva tener un 1,07 % de pérdidas.

A continuación veremos el histograma del tráfico:

```
hist(delta(find(delta<0.0001)),500)
```



**Fig. A5.12** Plot a 100 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.0001 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

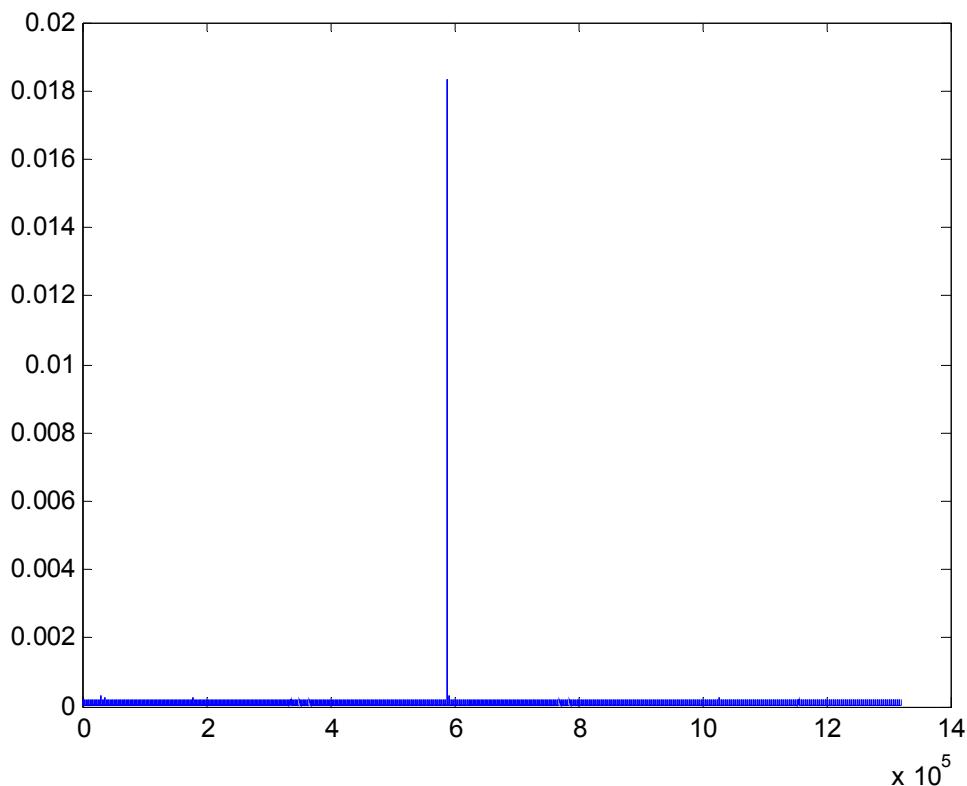
```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

Con  $\delta < 0.0001$  se obtienen 1311711 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 10428 valores, es decir un 0,788721%.

Como se puede ver en la gráfica ampliada, apenas existe dispersión de paquetes y la mayoría se concentran en 44,5 $\mu$ s, siendo la media y la desviación media: 45,38 $\mu$ s y 20,31 $\mu$ s respectivamente.

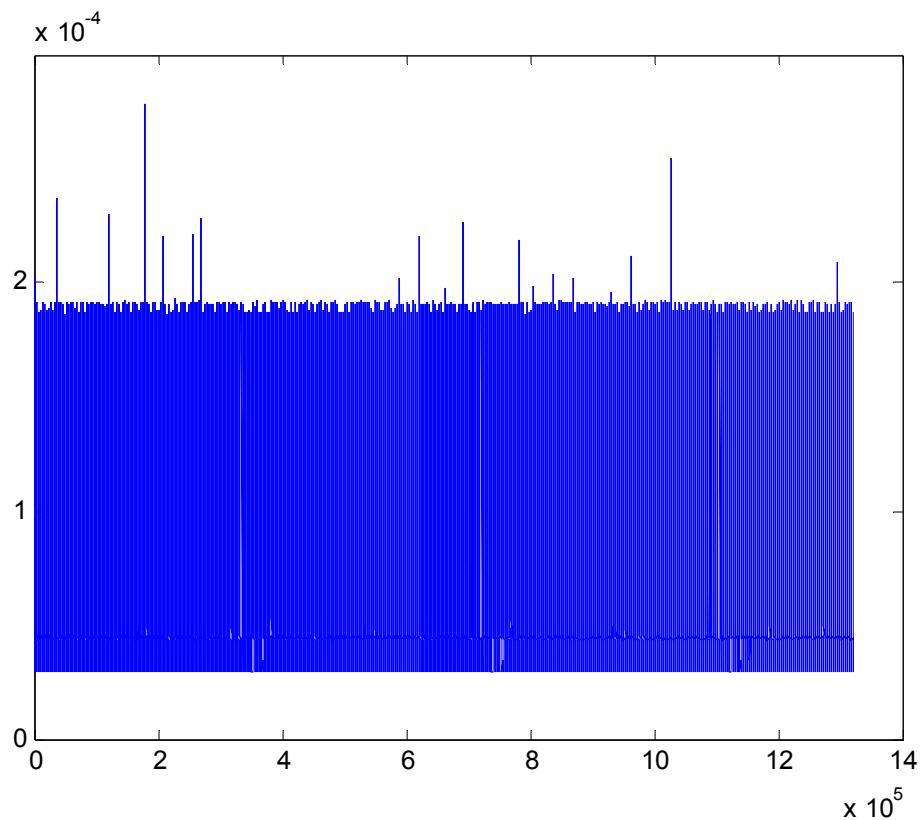
Seguidamente se realiza una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

*plot(delta)*



**Fig. A5.13** Plot a 100 Mbps y 300 bytes

Como se ha comentado anteriormente, se observa dispersión mínima a excepción de una muestra con valor 18,30ms y que filtraremos en la siguiente gráfica:

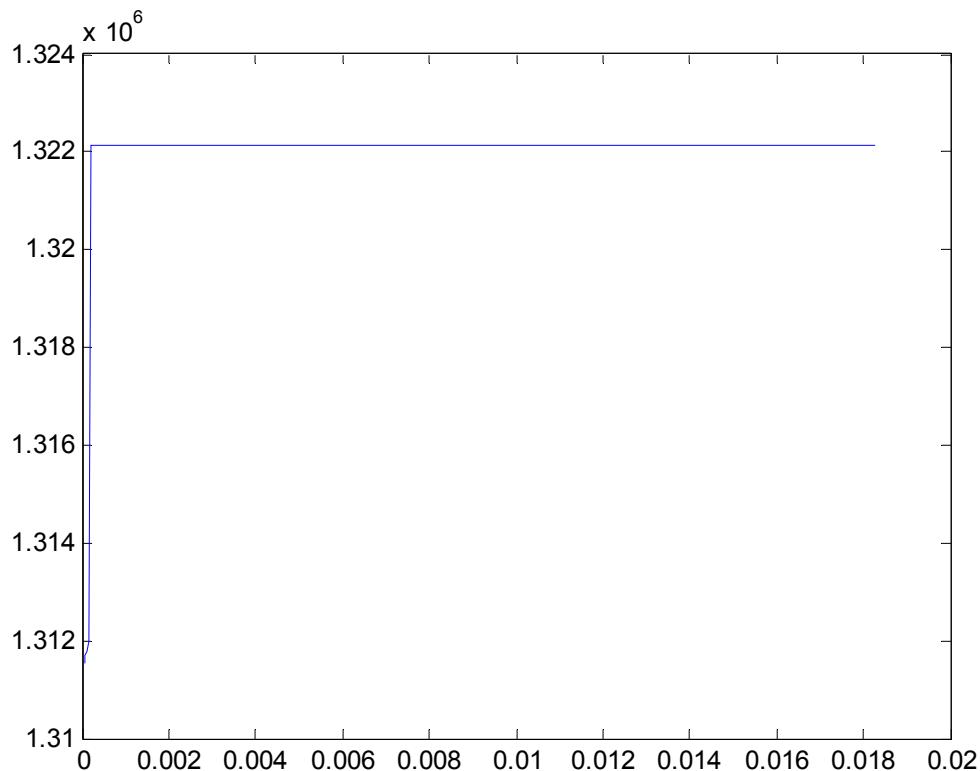


**Fig. A5.14** Plot ampliado

La transmisión es bastante estable. El valor mínimo es  $29,17\mu s$  segundos, o como se puede suponer con la gráfica,  $29,17\mu s$  lo que nos hace estar dentro del umbral correcto.

En este caso el único valor que desvía exageradamente de la media es una diferencia entre dos paquetes en los que se entrelazan entre ellos 5 paquetes de flujos distintos (HSRP, CDP y Loop)

A continuación graficaremos los valores de la suma acumulativa en función de los valores de la prueba:



**Fig. A5.15** Suma acumulativa a 50 Mbps y 1400 bytes

Como vemos, la cantidad máxima de valores forma un pico muy abrupto y poco suavizado debido a la alta frecuencia de valores en esa zona ( $44,5\mu s$ ) y a los despreciables valores poco acumulativos posteriores a este.

## V. Pruebas Random Netflow con MGEN, DAG y Nfsen

### Muestreo 1 de cada 10 paquetes

La configuración es la siguiente para uno de cada 10:

```
Crear mapa de muestreo, modo y frecuencia
flow-sampler-map nombre_mapa
mode random one-out-of <frecuencia_muestreo>
Cisco3700(config)#flow-sampler-map TEST1-10
Cisco3700(config-sampler)#mode random one-out-of 10

Habilitar mapa de muestreo en interfaz
flow-sampler <nombre_mapa>
Deshabilitar Full Netflow
no ip route-cache flow

Cisco3700(config-if)#flow-sampler TEST1-10
Cisco3700(config-if)#no ip route-cache flow
```

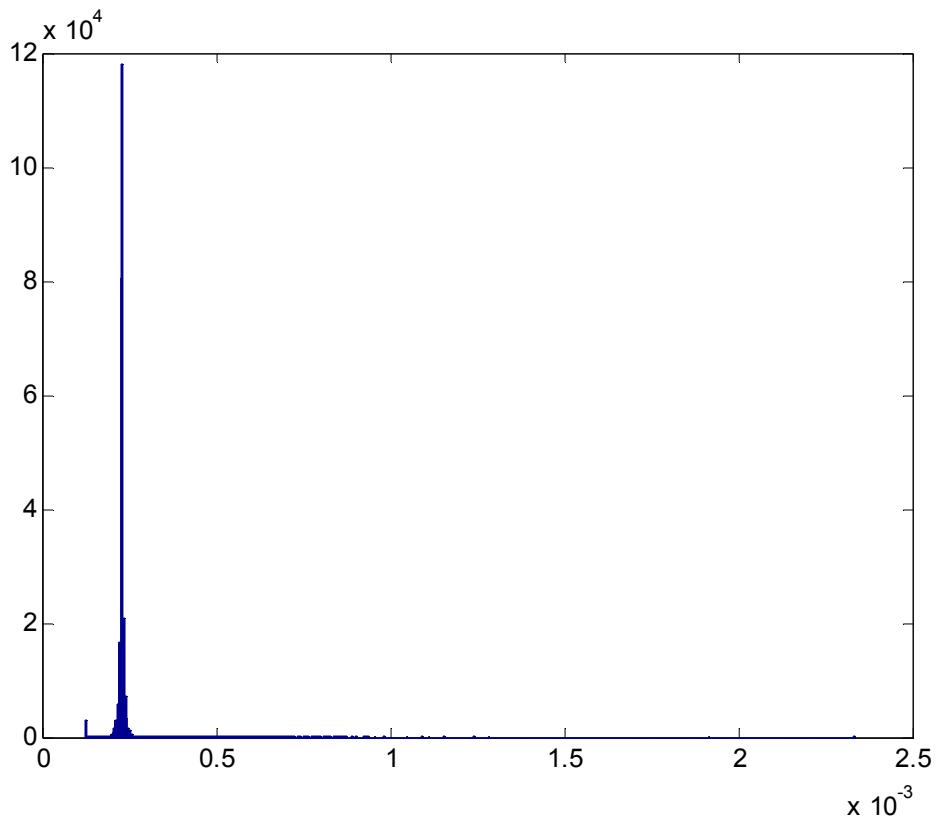
### Prueba de 1400 bytes a 50 Mbps

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogido por la DAG, Nfsen y el registro de salida de MGEN.

Por parte del colector, se interpretan 26785 paquetes recibidos, mientras que por la DAG se reciben 267858 paquetes y finalmente para el registro de MGEN se reciben 267619 paquetes con una velocidad obtenida de 5,1 Mbps. En este caso Netflow muestrea un 10,00% de los paquetes totales recibidos por la DAG, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta,500)
```



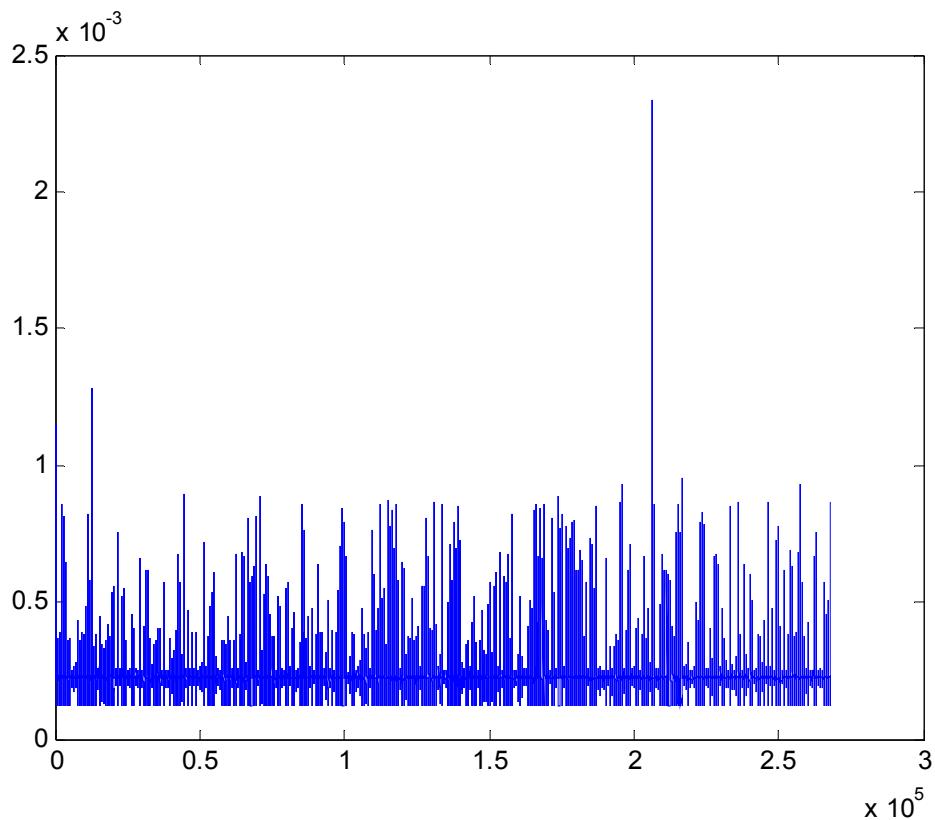
**Fig. A5.1** Histograma a 50 Mbps y 1400 bytes

Se puede observar como existe algo de dispersión. Concentrándose la mayoría de paquetes en  $2.26\text{e-}04$  segundos, siendo la media y la desviación estándar:  $223,99\mu\text{s}$  y  $26,18\mu\text{s}$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:



**Fig. A5.2 Plot a 50 Mbps y 1400 bytes**

Observamos que los valores máximos no superan el 1ms segundo ni están por debajo de 117,19 $\mu$ s segundos. El valor denotado llega a ser 2,33ms, una muestra muy elevada que no tiene explicación alguna, ya que visualizando con wireshark no se encuentran anomalías ni paquetes intercalados en esa posición.

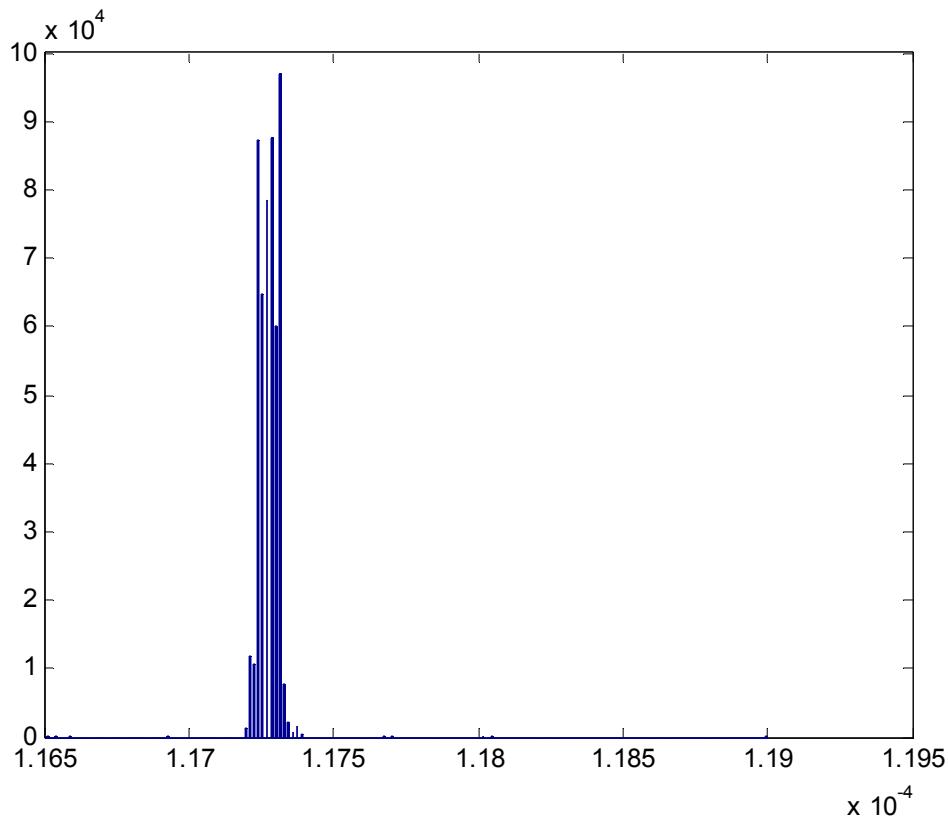
### Prueba de 1400 bytes a 100 Mbps

En este caso, por parte del colector, se interpretan 51169 paquetes recibidos, mientras que por la DAG se reciben 511695 paquetes y finalmente para el registro de MGGEN se reciben 509028 paquetes con una velocidad obtenida de 9,2Mbps.

En este caso Netflow muestrea un 9,99% de los paquetes totales recibidos por la DAG, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00012)), 500)
```



**Fig. A5.3** Histograma a 100 Mbps y 1400 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.00012 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.00012))
size(find(delta>0.00012))
```

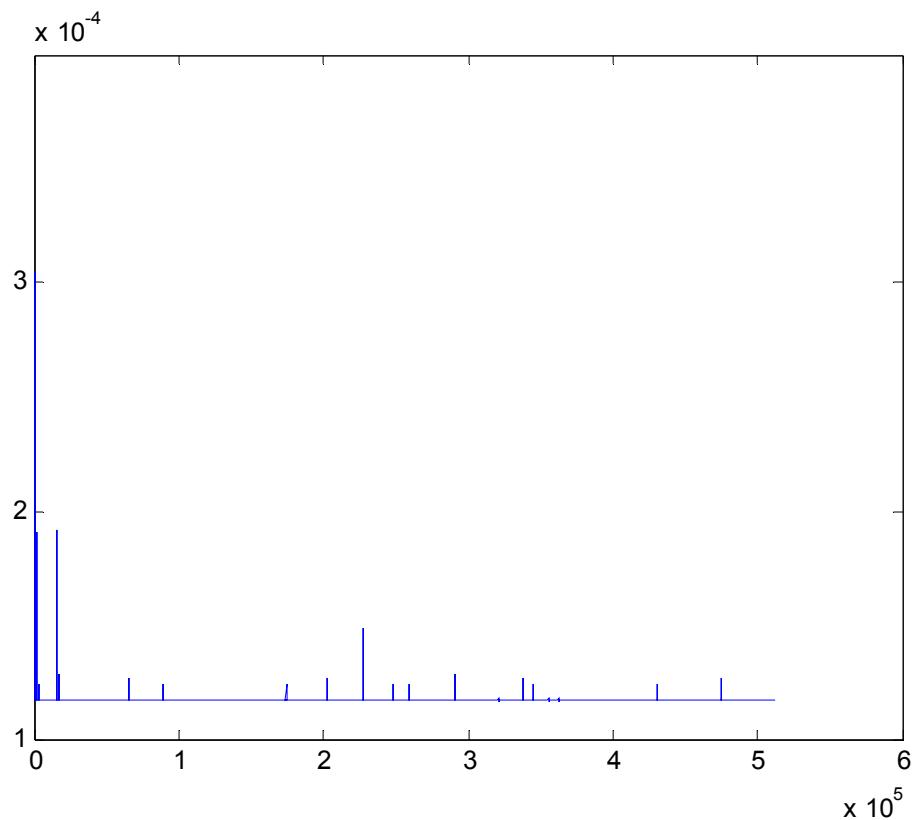
Con  $\text{delta}<0.00012$  se obtienen 511675 valores (los graficados), los descartados con  $\text{delta}>0.00012$ , y se obtienen 19 valores, es decir un 0,0037131%.

Se puede observar como existe muy poca dispersión. Concentrándose la mayoría de paquetes en 117 $\mu\text{s}$  segundos, siendo la media y la desviación estándar: 117,27 $\mu\text{s}$  y 0,30 $\mu\text{s}$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

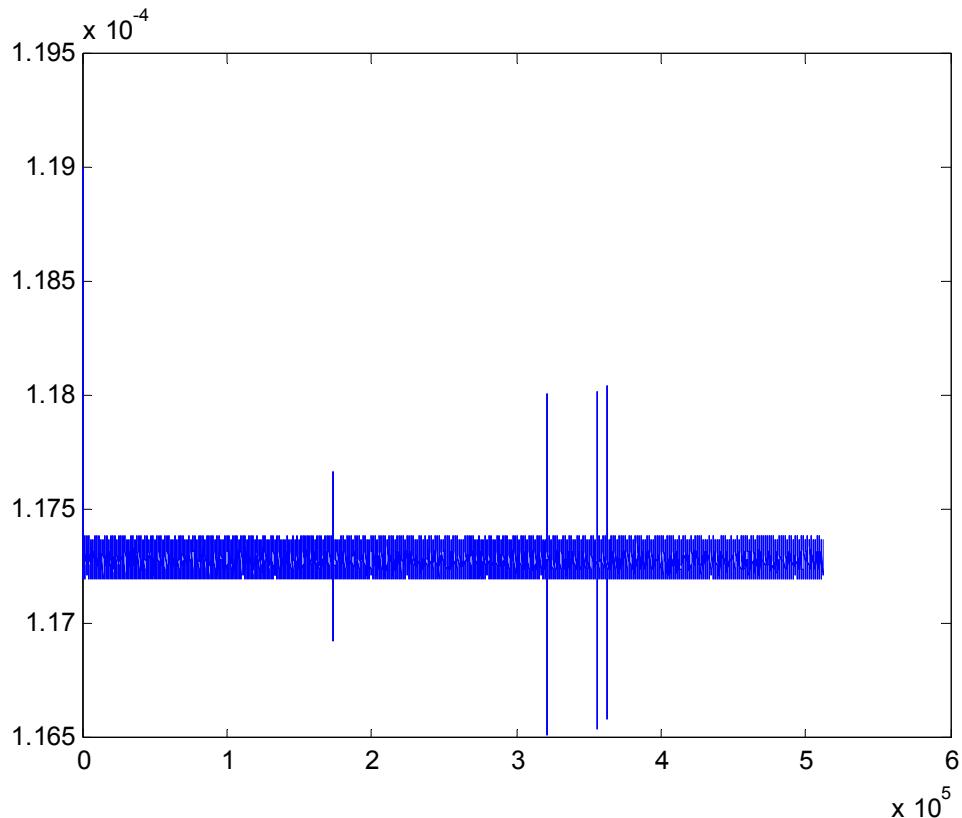
Con este comando obtenemos la siguiente gráfica:



**Fig. A5.4** Plot a 100 Mbps y 1400 bytes

En este caso los superiores a la media son paquetes que tienen entre ellos paquetes NBNS. La cantidad de muestras se consigue filtrando por valores superiores a 0.000175 con `size(find(delta4>0.0001175))` y resultan 25 valores.

Si afinamos más el gráfico podemos observar mejor la poca dispersión que existe:



**Fig. A5.5** Plot ampliado

Vemos como hay cuatro valores por debajo de la media, los cuales tienen paquetes entre medio de broadcast e ipv6. La gran cantidad de muestras están por encima de 117 $\mu$ s y las muestras más bajas cumplen con el umbral.

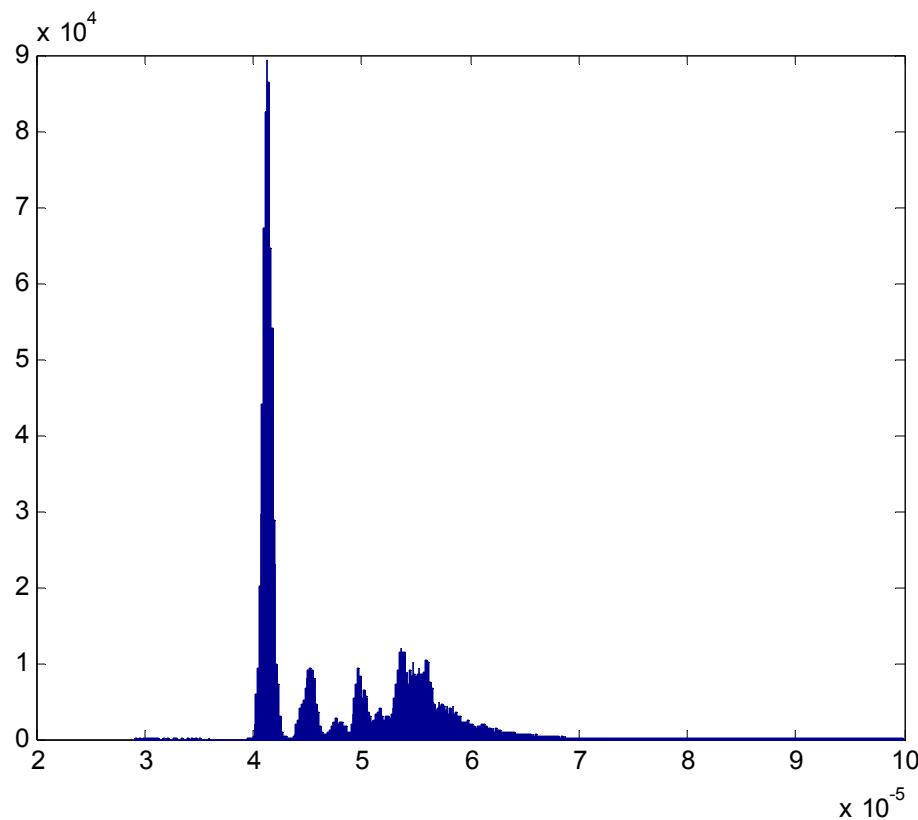
### Prueba de 300 bytes a 50 Mbps

En este caso, por parte del colector, se interpretan 124949 paquetes recibidos, mientras que por la DAG se reciben 1249485 paquetes y finalmente para el registro de MGEN se reciben 1119822 paquetes con una velocidad obtenida de 5,5 Mbps.

En este caso Netflow muestrea un 10,00% de los paquetes totales recibidos por la DAG, es decir un resultado algo superior a lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.0001)),500)
```



**Fig. A5.6** Histograma a 50 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.0001 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

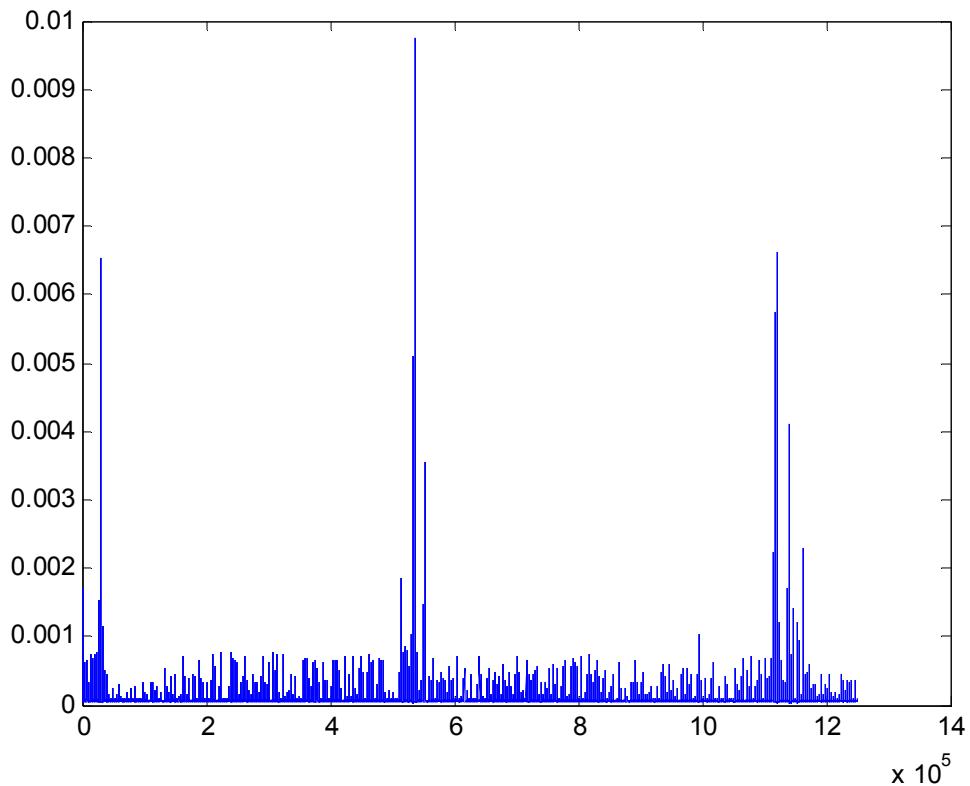
Con  $\delta < 0.0001$  se obtienen 1247143 valores (los graficados), los descartados con  $\delta > 0.0001$ , y se obtienen 2341 valores, es decir un 0,187357%.

Como se puede ver en la gráfica ampliada, existe poca dispersión de paquetes y la mayoría se concentran en  $41.4\mu\text{s}$ , siendo la media y la desviación estándar:  $48.01\mu\text{s}$  y  $24.75\mu\text{s}$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

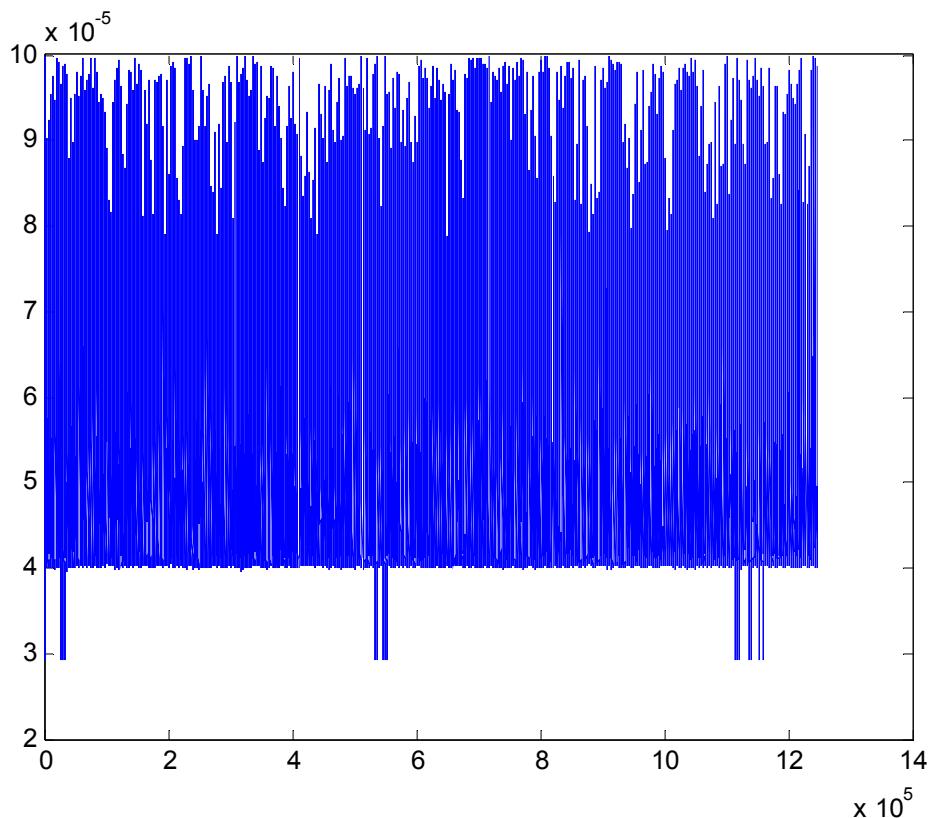
Con este comando obtenemos la siguiente gráfica:



**Fig. A5.7 Plot a 50 Mbps y 300 bytes**

Si nos fijamos observamos ciertas anomalías al inicio de la transmisión, algo que podría ser lógico por el propio programa MGEN al generar tanta cantidad de paquetes por segundo. Para los valores superiores no se encuentra razón, ya que no existen paquetes intercalados entre esas muestras. Podría ser CPU de la maquina generadora.

Si afinamos más el gráfico podemos observar mejor la poca dispersión que existe:



**Fig. A5.8** Plot ampliado

Si afinamos la gráfica para obtener valores inferiores a 1ms obtenemos algo más estable dentro de la media. Aunque se continúan viendo muestras inferiores a 40 $\mu$ s, éstas están dentro del tiempo de transmisión mínimo.

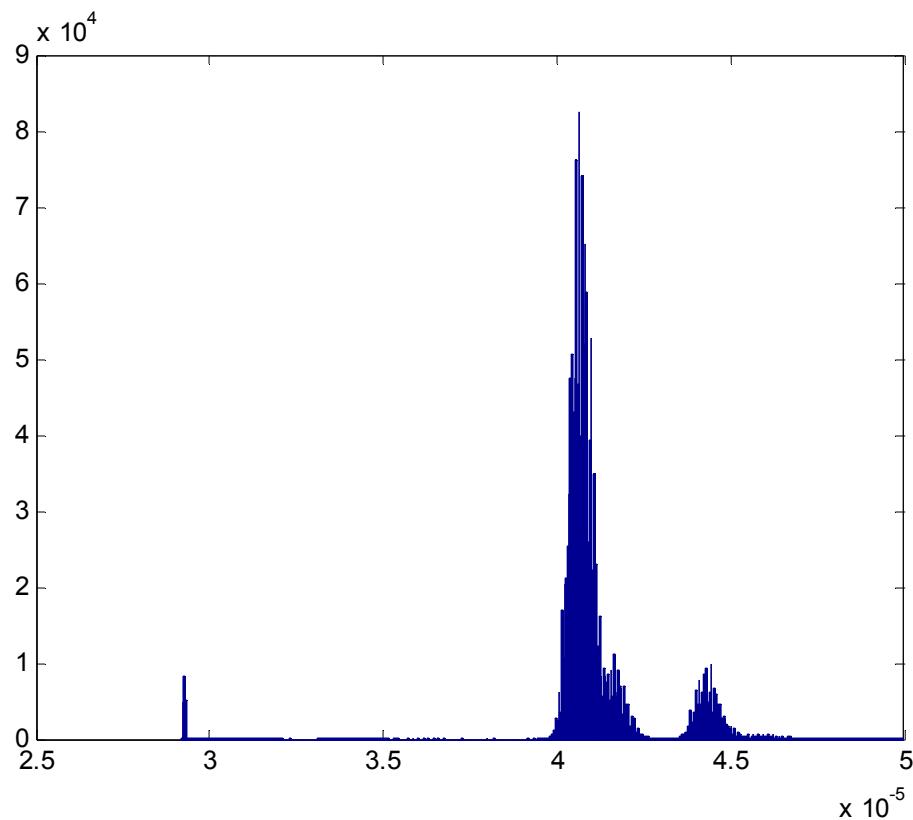
### Prueba de 300 bytes a 100 Mbps

En este caso, por parte del colector, se interpretan 142210 paquetes recibidos, mientras que por la DAG se reciben 1422086 paquetes y finalmente para el registro de MGEN se reciben 1166897 paquetes con una velocidad obtenida de 6,2 Mbps.

En este caso Netflow muestrea un 10,00% de los paquetes totales recibidos por la DAG, es decir un resultado algo superior a lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00005)),500)
```



**Fig. A5.9** Histograma a 100 Mbps y 300bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.00005 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.00005))
size(find(delta>0.00005))
```

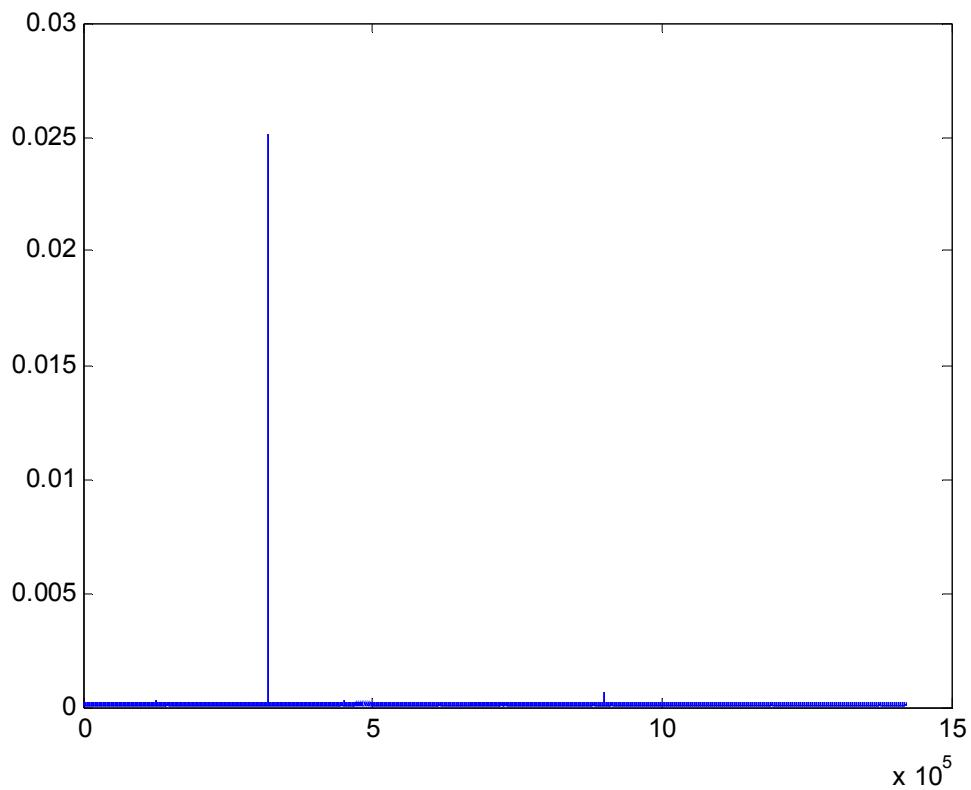
Con  $\text{delta} < 0.00005$  se obtienen 1409754 valores (los graficados), los descartados con  $\text{delta} > 0.00005$ , y se obtienen 12331valores, es decir un 0,867107%.

Como se puede ver en la gráfica ampliada, existe poca dispersión de paquetes y la mayoría se concentran en 4.06e-5, siendo la media y la desviación estándar: 42,19 $\mu$ s y 24,42 $\mu$ s respectivamente.

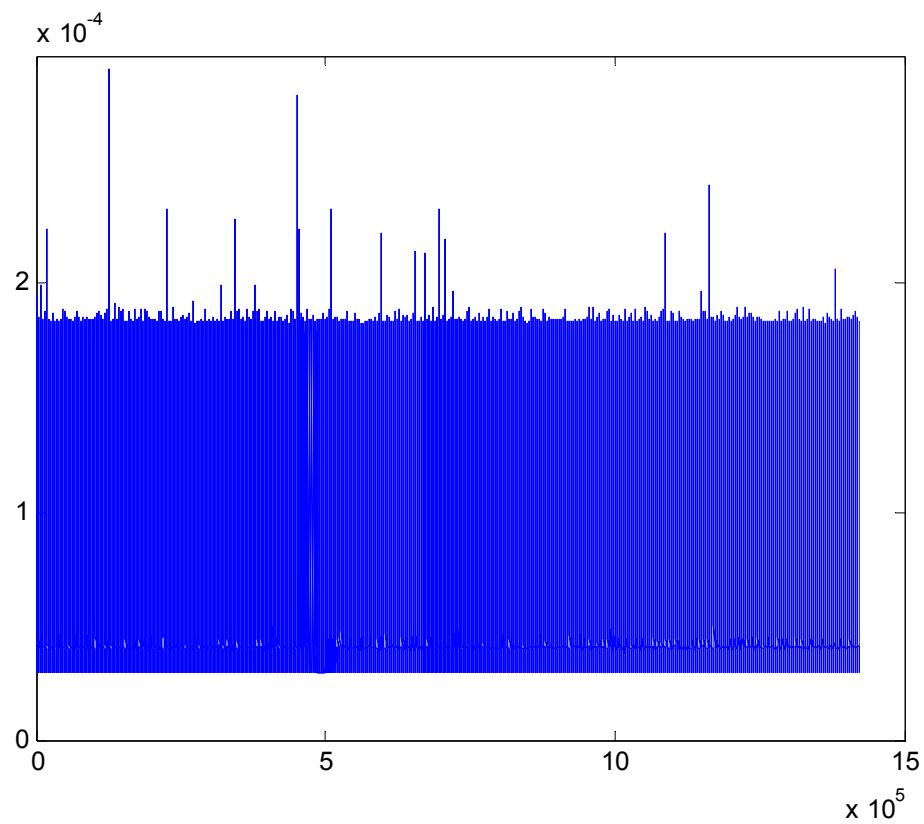
A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:



**Fig. A5.10** Plot a 100 Mbps y 300bytes



### Fig. A5.11 Plot ampliado

Observamos que el valor máximo no supera los 25ms, siendo este una cantidad mínima. Como se puede apreciar los valores mínimos no bajan de 29,17 $\mu$ s con lo que están por encima del umbral mínimo.

En este caso el único valor que desvía exageradamente de la media es una diferencia entre dos paquetes en los que se entrelaza con un Loop.

## Muestreo 1 de cada 100 paquetes

La configuración es la siguiente para uno de cada 100:

```
Crear mapa de muestreo, modo y frecuencia
flow-sampler-map nombre_mapa
mode random one-out-of <frecuencia_muestreo>

Cisco3700(config)#flow-sampler-map TEST1-100
Cisco3700(config-sampler)#mode random one-out-of 100

Habilitar mapa de muestreo en interfaz
flow-sampler <nombre_mapa>
Deshabilitar Full Netflow
no ip route-cache flow

Cisco3700(config-if)#flow-sampler TEST1-100
Cisco3700(config-if)#no ip route-cache flow
```

## Prueba de 1400 bytes a 50 Mbps

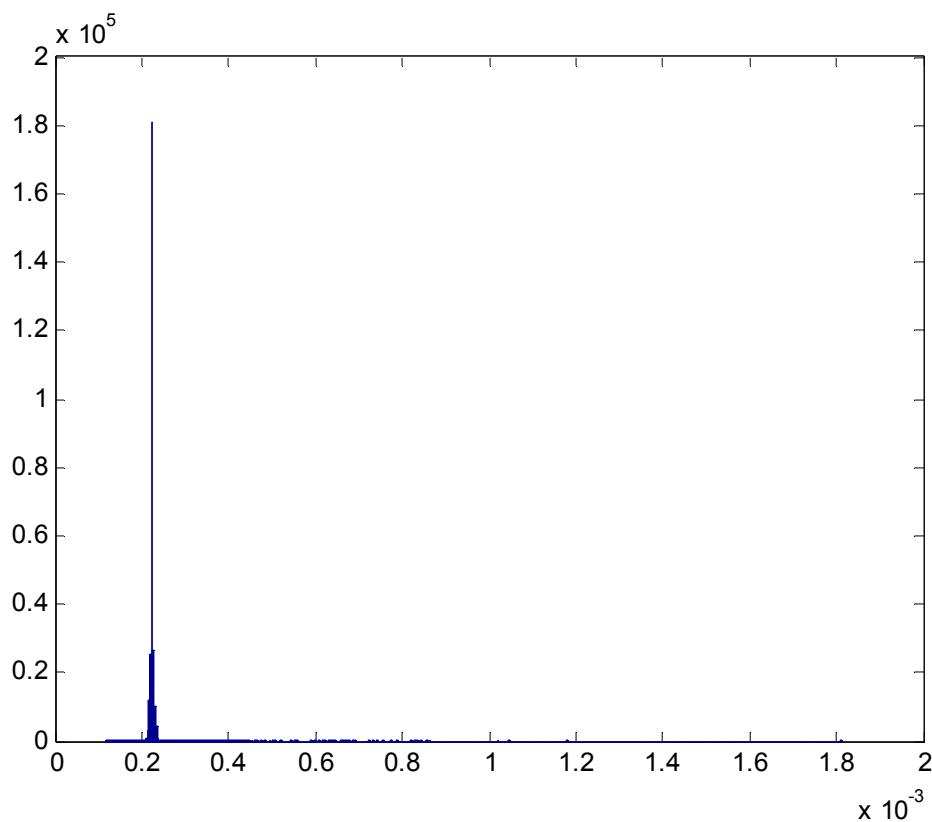
En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogido por la DAG, Nfsen y el registro de salida de MGEN.

Por parte del colector, se interpretan 2678 paquetes recibidos, mientras que por la DAG se reciben 267858 paquetes y finalmente para el registro de MGEN se reciben 267493 paquetes con una velocidad obtenida de 510,027 kbps.

En este caso Netflow muestrea un 0,99% de los paquetes totales recibidos, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta,500)
```



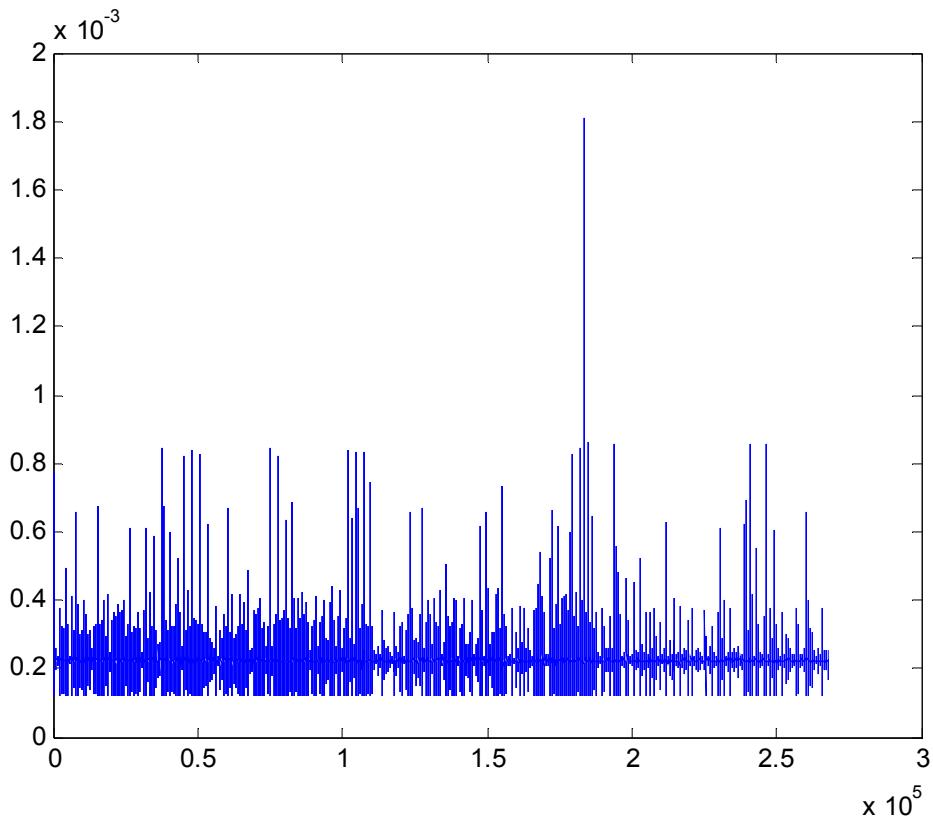
**Fig. A5.12** Histograma a 50 Mbps y 1400 bytes

Se puede observar como existe algo de dispersión. Concentrándose la mayoría de paquetes en  $224 \mu s$ , siendo la media y la desviación estándar:  $223,98 \mu s$  y  $12,01 \mu s$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:



**Fig. A5.13 Plot a 50 Mbps y 1400 bytes**

Observamos que el valor máximo no supera los 2ms segundos ni están por debajo del valor mínimo de transmisión, estando por encima de 117.1 $\mu$ s. El valor denotado llega a ser 1,81ms una muestra muy elevada que no tiene explicación alguna, ya que visualizando con wireshark no se encuentran anomalías ni paquetes intercalados en esa posición.

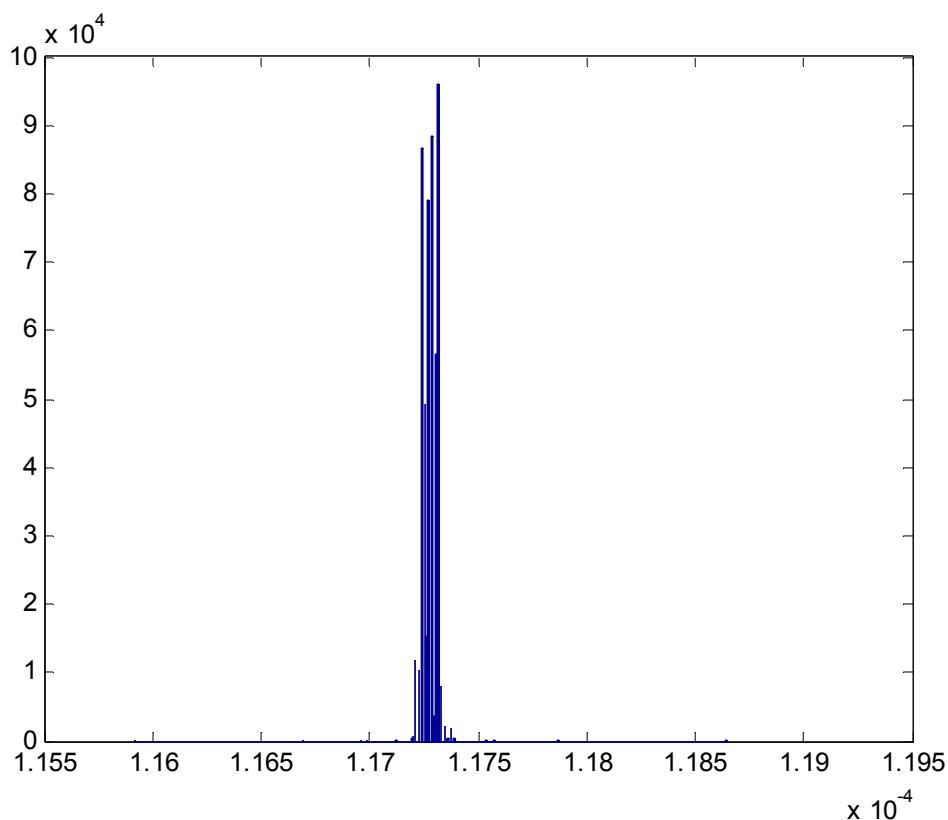
### Prueba de 1400 bytes a 100 Mbps

En este caso, por parte del colector, se interpretan 5117 paquetes recibidos, mientras que por la DAG se reciben 511676 paquetes y finalmente para el registro de MGGEN se reciben 490348 paquetes con una velocidad obtenida de 974,211 kbps.

En este caso Netflow muestrea un 1,00% de los paquetes totales recibidos, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00012)),500)
```



**Fig. A5.14** Histograma a 100 Mbps y 1400 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.00012 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.00012))
size(find(delta>0.00012))
```

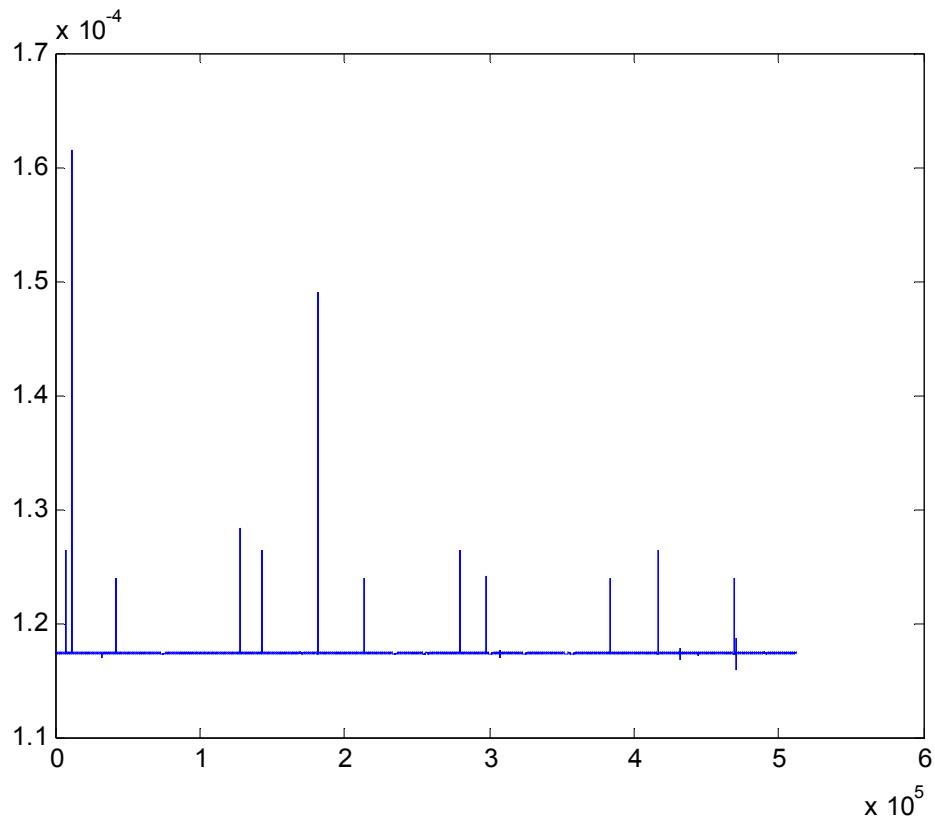
Con  $\delta < 0.00012$  se obtienen 511662 valores (los graficados), los descartados con  $\delta > 0.00012$ , y se obtienen 14 valores, es decir un 0,00254067%.

Se puede observar como existe muy poca dispersión. Concentrándose la mayoría de paquetes en  $117\mu s$  segundos, siendo la media y la dispersión estándar:  $117,27\mu s$  y  $0,91\mu s$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

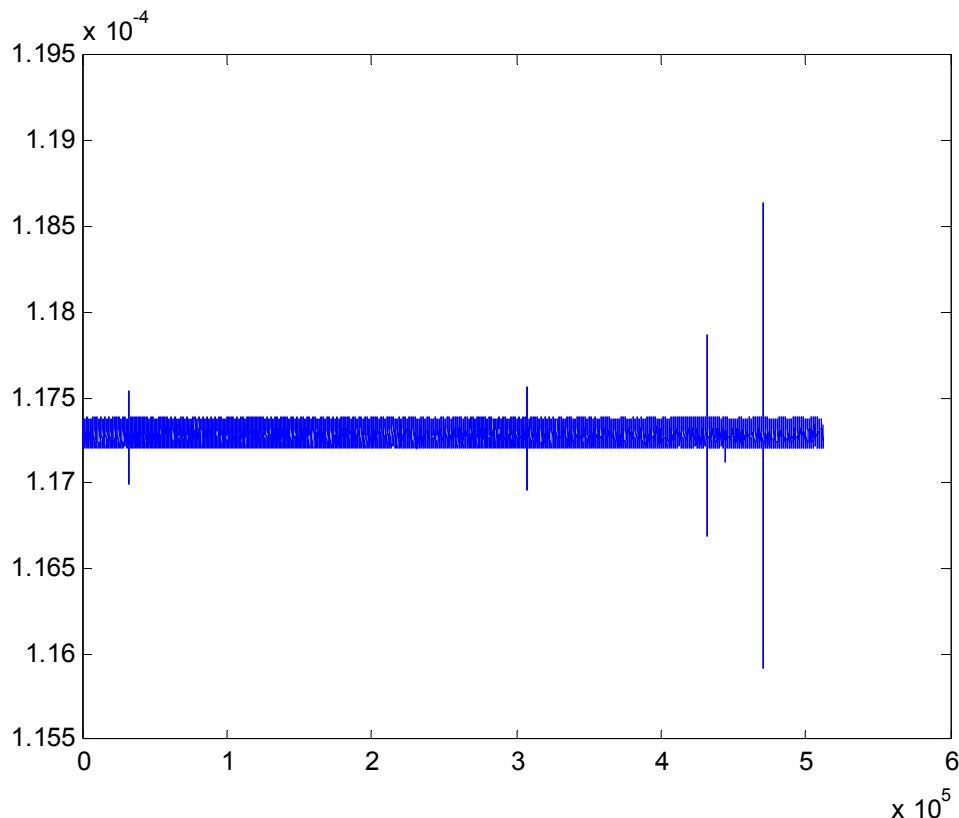
Con este comando obtenemos la siguiente gráfica:



**Fig. A5.15** Plot a 100 Mbps y 1400 bytes

En este caso los superiores a la media son paquetes que tienen entre ellos paquetes NBNS. La cantidad de muestras se consigue filtrando por valores superiores a 0.000175 con `size(find(delta4>0.0001175))` y resultan 17 valores.

Si afinamos más el gráfico podemos observar mejor la dispersión:



**Fig. A5.16** Plot ampliado

Vemos como hay cuatro valores por debajo de la media, los cuales tienen paquetes entre medio de broadcast e ipv6.

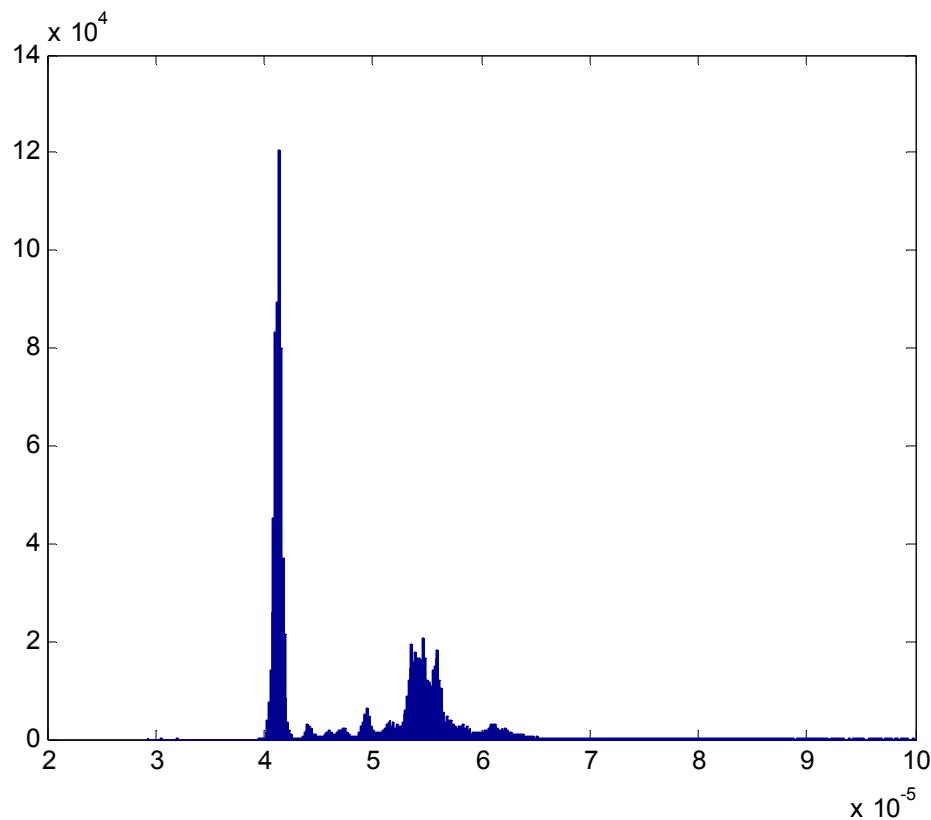
### Prueba de 300 bytes a 50 Mbps

En este caso, por parte del colector, se interpretan 12500 paquetes recibidos, mientras que por la DAG se reciben 1250002 paquetes y finalmente para el registro de MGEN se reciben 840626 paquetes con una velocidad obtenida de 510,285 kbps.

En este caso Netflow muestrea un 0,99% de los paquetes totales recibidos, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 500)
```



**Fig. A5.17** Histograma a 50 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.0001 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

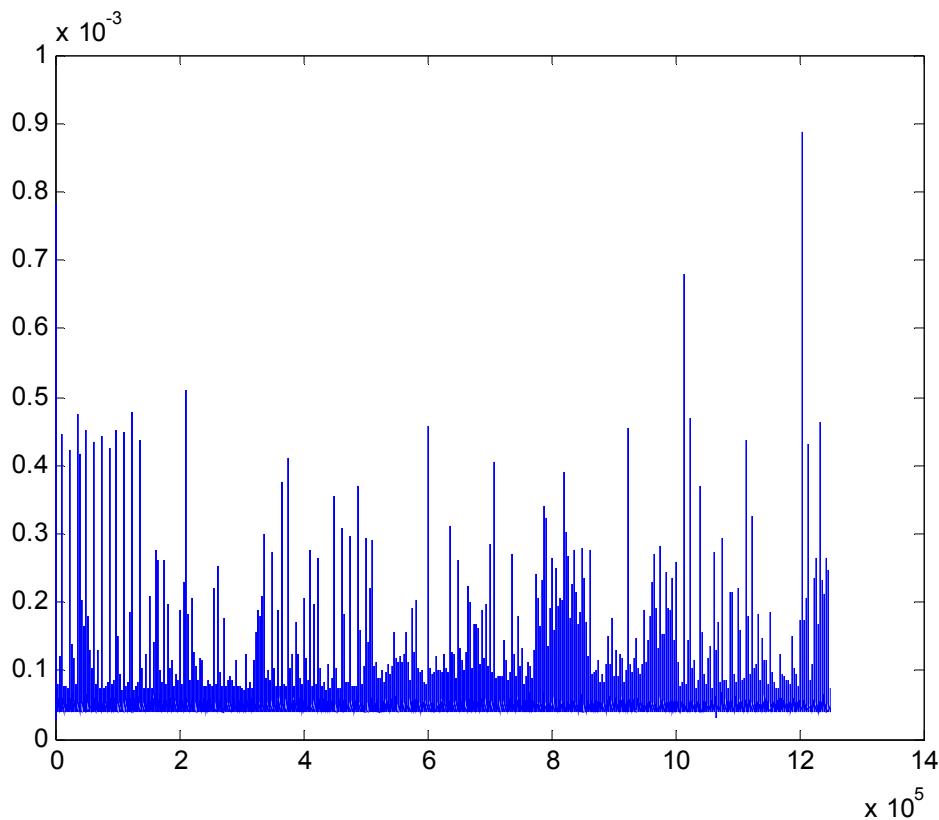
Con  $\text{delta}<0.0001$  se obtienen 1249506 valores (los graficados), los descartados con  $\text{delta}>0.0001$ , y se obtienen 494 valores, es decir un 0,03952%.

Como se puede ver en la gráfica ampliada, existe dispersión de paquetes y la mayoría se concentran en 41,3μs, siendo la media y la desviación estándar: 48,00μs y 8,08μs respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

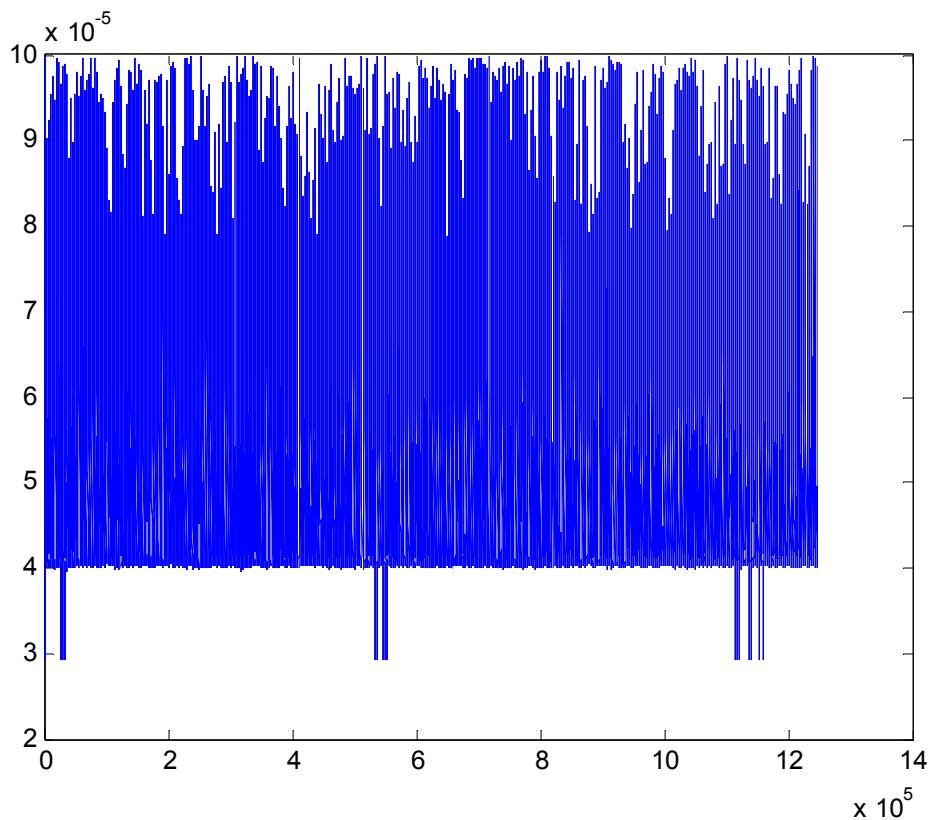
Con este comando obtenemos la siguiente gráfica:



**Fig. A5.18 Plot a 50 Mbps y 300 bytes**

Si nos fijamos observamos ciertas anomalías al inicio de la transmisión, algo que podría ser lógico por el propio programa MGEN al generar tanta cantidad de paquetes por segundo. Para los valores superiores no se encuentra razón, ya que no existen paquetes intercalados entre esas muestras. Podría ser CPU de la maquina generadora. No existen paquetes entrelazados.

Si afinamos más el gráfico podemos observar mejor la poca dispersión que existe:



**Fig. A5.19** Plot ampliado

Si afinamos la gráfica para obtener valores inferiores a 1ms obtenemos algo más estable dentro de la media. . Aunque se continúan viendo muestras inferiores a 40 $\mu$ s, éstas están dentro del tiempo de transmisión mínimo.

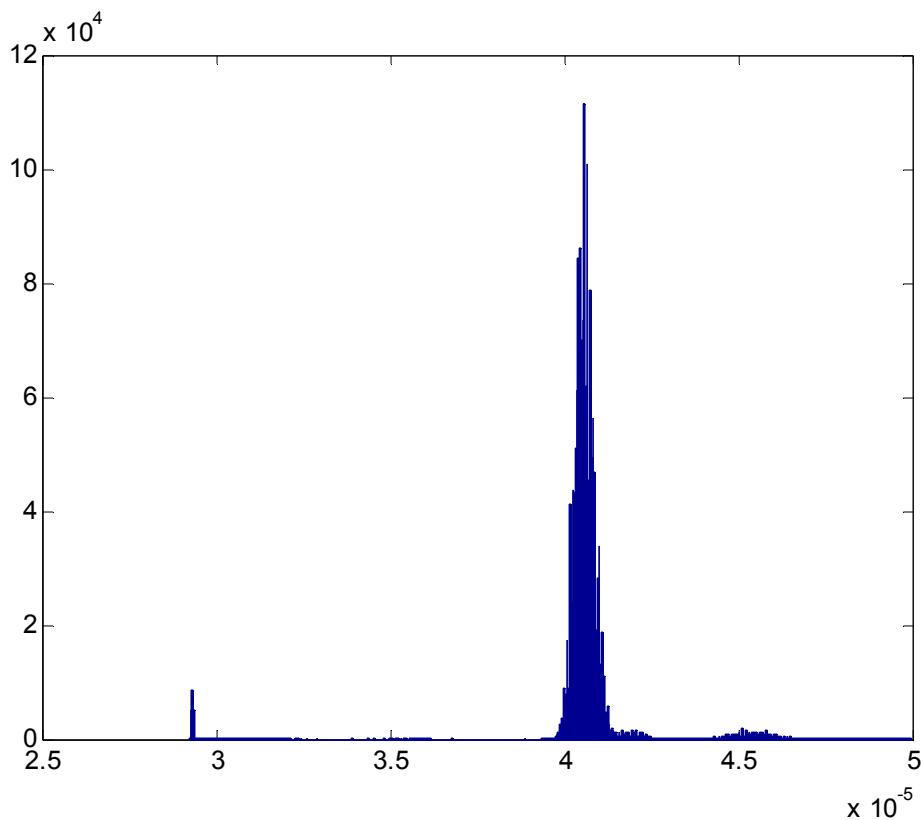
### Prueba de 300 bytes a 100 Mbps

En este caso, por parte del colector, se interpretan 14395 paquetes recibidos, mientras que por la DAG se reciben 1439522 paquetes y finalmente para el registro de MGEN se reciben 818565 paquetes con una velocidad obtenida de 629,499 kbps.

En este caso Netflow muestrea un 0,99% de los paquetes totales recibidos, es decir un resultado algo superior a lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00005)),500)
```



**Fig. A5.20** Plot a 100 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.00005 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.00005))
size(find(delta>0.00005))
```

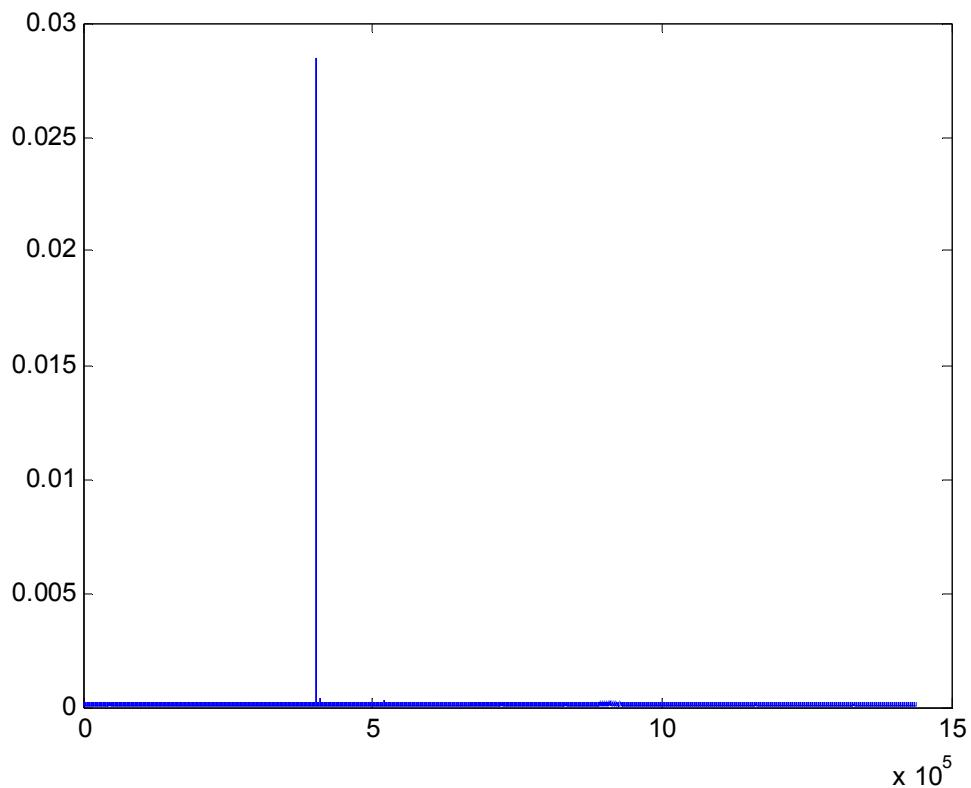
Con  $\delta t < 0.00005$  se obtienen 1427535 valores (los graficados), los descartados con  $\delta t > 0.00005$ , y se obtienen 11986 valores, es decir un 0,832638%.

Como se puede ver en la gráfica ampliada, existe dispersión de paquetes y la mayoría se concentran en 40,6 $\mu$ s, siendo la media y la desviación estándar: 41,68 $\mu$ s y 26,69vs respectivamente.

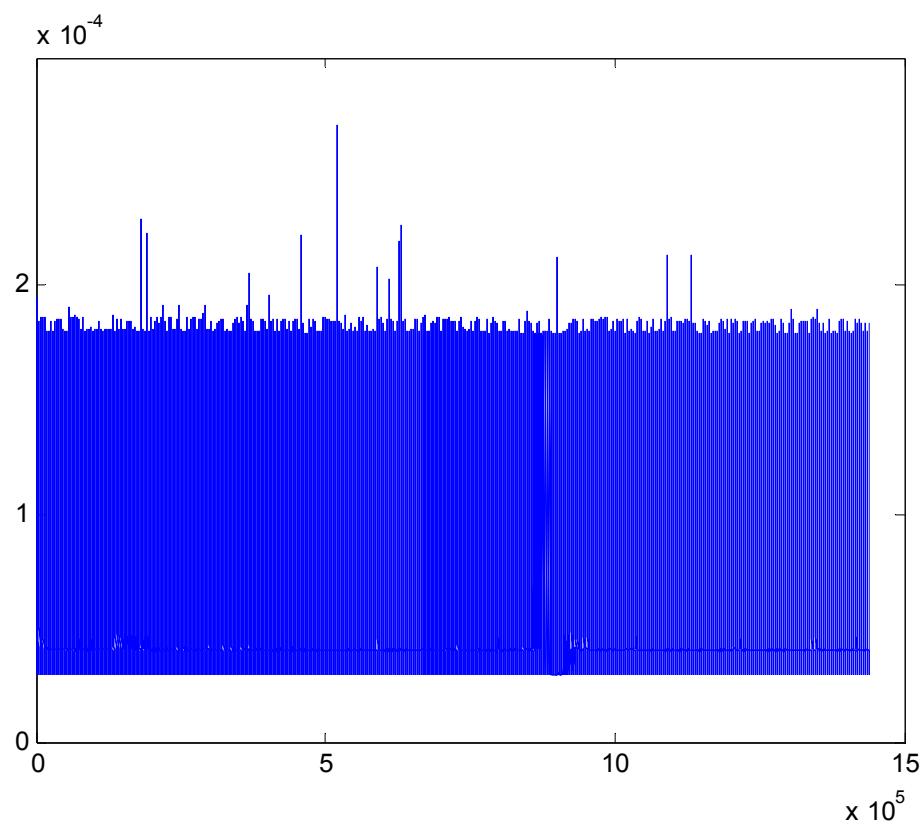
A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:



**Fig. A5.21** Plot a 100 Mbps y 300bytes



### Fig. A5.22 Plot ampliado

Observamos que el valor máximo no supera los 0.03 segundos, siendo este una cantidad mínima. Como se puede apreciar los valores mínimos no bajan de 29,17 $\mu$ s con lo que están por encima del umbral mínimo.

En este caso el único valor que desvía exageradamente de la media es una diferencia entre dos paquetes en los que se entrelazan tres paquetes, Loop, CDP y ARP.

## Muestreo 1 de cada 1000 paquetes

La configuración es la siguiente para uno de cada 1000:

```
Crear mapa de muestreo, modo y frecuencia
flow-sampler-map nombre_mapa
mode random one-out-of <frecuencia_muestreo>

Cisco3700(config)#flow-sampler-map TEST1-1000
Cisco3700(config-sampler)#mode random one-out-of 1000

Habilitar mapa de muestreo en interfaz
flow-sampler <nombre_mapa>
Deshabilitar Full Netflow
no ip route-cache flow

Cisco3700(config-if)#flow-sampler TEST1-1000
Cisco3700(config-if)#no ip route-cache flow
```

## Prueba de 1400 bytes a 50 Mbps

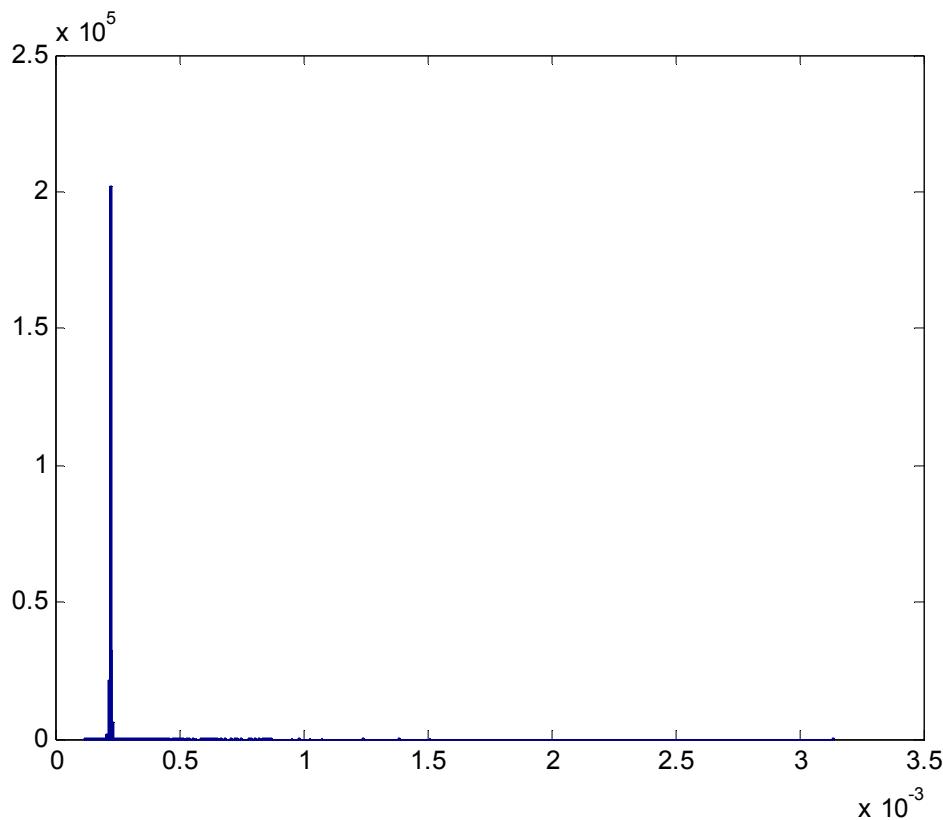
En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogido por la DAG, Nfsen y el registro de salida de MGEN.

Por parte del colector, se interpretan 267 paquetes recibidos, mientras que por la DAG se reciben 267858 paquetes y finalmente para el registro de MGEN se reciben 267858 paquetes con una velocidad obtenida de 51,047 kbps.

En este caso Netflow muestrea un 0,09% de los paquetes totales recibidos, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta,500)
```



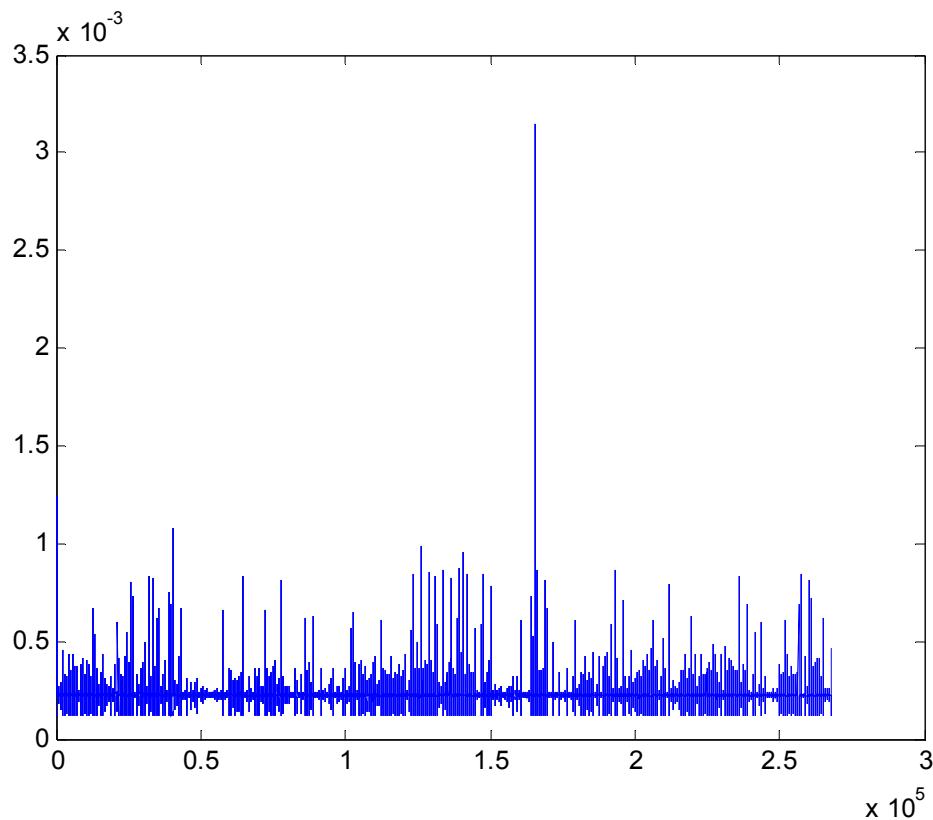
**Fig. A5.23** Histograma a 50 Mbps y 1400 bytes

Se puede observar como existe algo de dispersión. Concentrándose la mayoría de paquetes en 230 $\mu s$ , siendo la media y la desviación estándar: 224,01 $\mu s$  y 13,91 $\mu s$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:



**Fig. A5.24 Plot a 50 Mbps y 1400 bytes**

Observamos que el valor máximo no supera los 3,5ms segundos ni están por debajo del valor mínimo de transmisión. Las imprecisiones del reloj interno hacen que exista algo de dispersión.

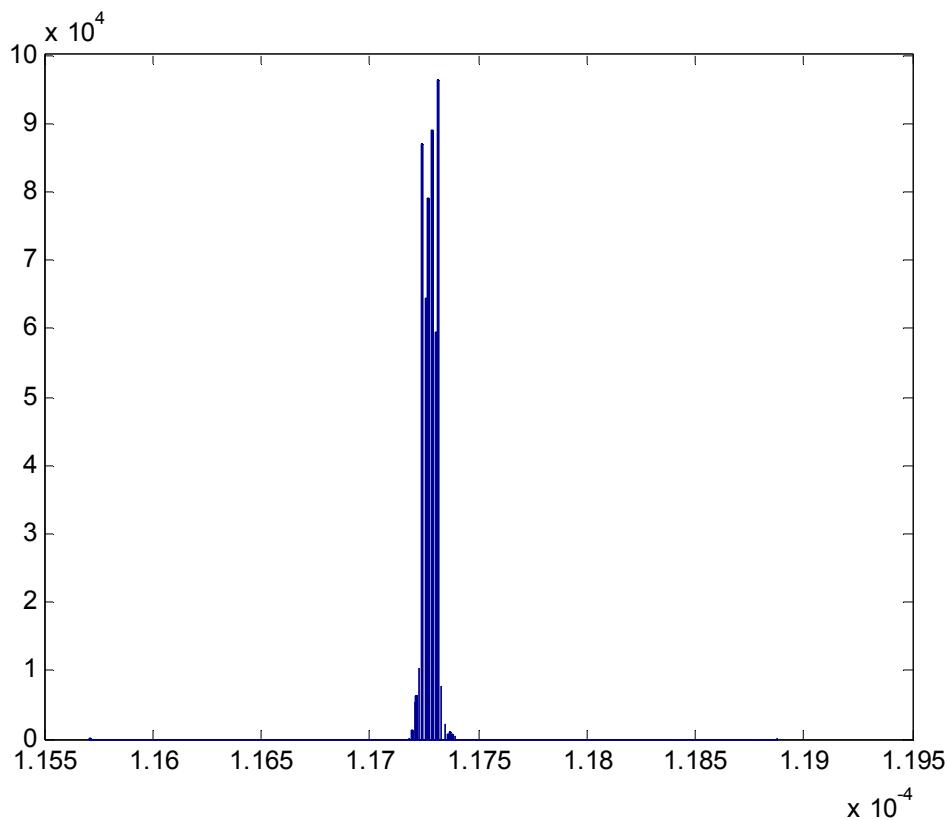
### Prueba de 1400 bytes a 100 Mbps

En este caso, por parte del colector, se interpretan 511 paquetes recibidos, mientras que por la DAG se reciben 511668 paquetes y finalmente para el registro de MGEN se reciben 511150 paquetes con una velocidad obtenida de 97,737 kbps.

En este caso Netflow muestrea un 0,09% de los paquetes totales recibidos, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00012)),500)
```



**Fig. A5.25** Histograma a 50 Mbps y 1400 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.00012 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.00012))
size(find(delta>0.00012))
```

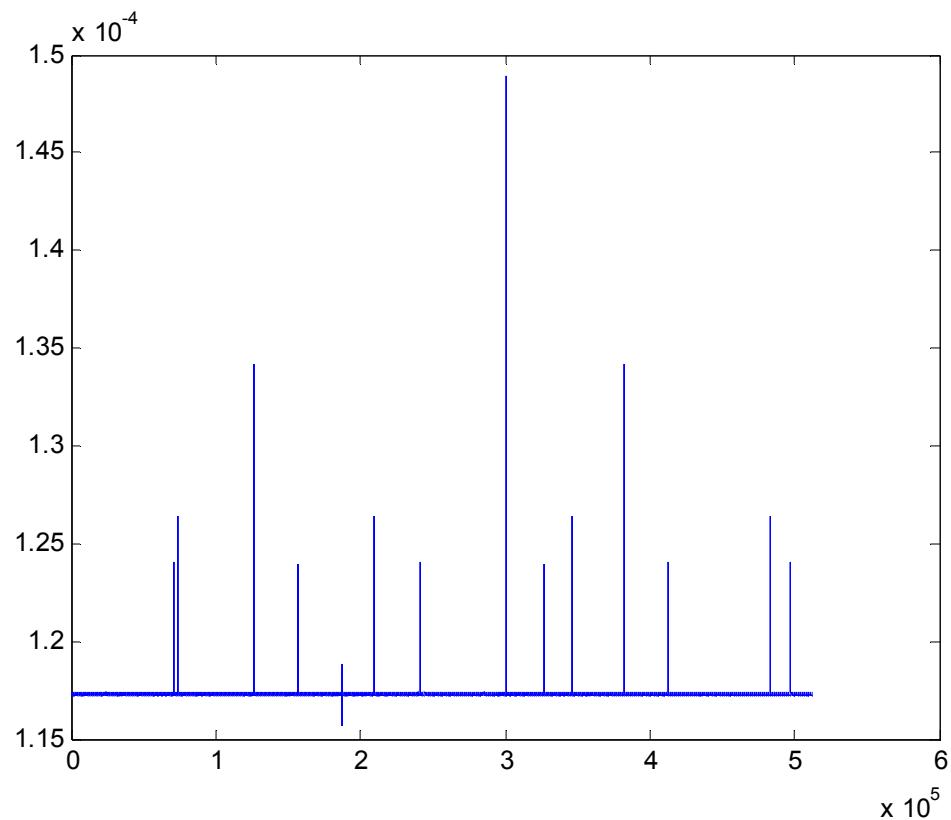
Con  $\text{delta}<0.00012$  se obtienen 511654 valores (los graficados), los descartados con  $\text{delta}>0.00012$ , y se obtienen 13 valores, es decir un 0,00254071%.

Se puede observar como existe muy poca dispersión. Concentrándose la mayoría de paquetes en  $117\mu\text{s}$ , siendo la media y la dispersión estándar:  $117,27\mu\text{s}$  y  $72,08\text{ns}$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

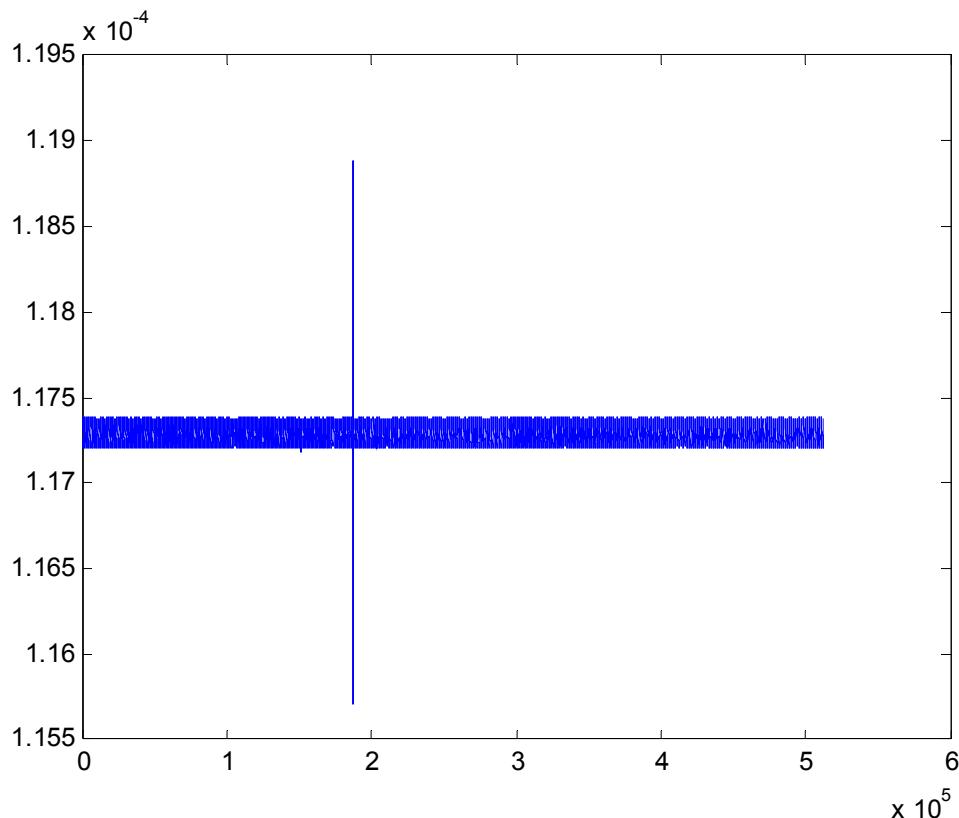
Con este comando obtenemos la siguiente gráfica:



**Fig. A5.26** Plot a 100 Mbps y 1400 bytes

En este caso los superiores a la media son paquetes que tienen entre ellos paquetes NBNS. La cantidad de muestras se consigue filtrando por valores superiores a 0.000175 con `size(find(delta4>0.0001175))` y resultan 14 valores.

Si afinamos más el gráfico podemos observar mejor la dispersión:



**Fig. A5.27** Plot ampliado

Vemos como hay un valor por debajo de la media, el cual la diferencia con el anterior paquete es muy pequeña, de nuevo imprecisiones de reloj. La gran cantidad de muestras están por encima de  $117\mu s$  por lo que cumplen con el umbral mínimo.

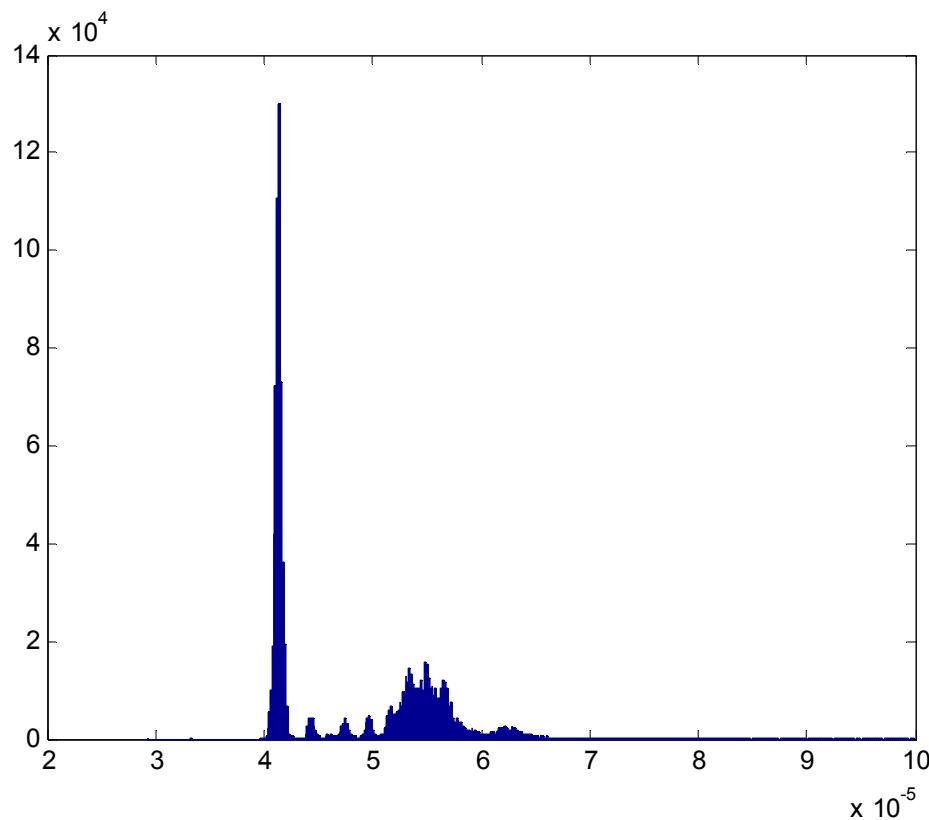
### Prueba de 300 bytes a 50 Mbps

En este caso, por parte del colector, se interpretan 1250 paquetes recibidos, mientras que por la DAG se reciben 1250001 paquetes y finalmente para el registro de MGEN se reciben 825684 paquetes con una velocidad obtenida de 54,699 kbps.

En este caso Netflow muestrea un 0,09% de los paquetes totales recibidos, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 500)
```



**Fig. A5.28** Histograma a 50 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.0001 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

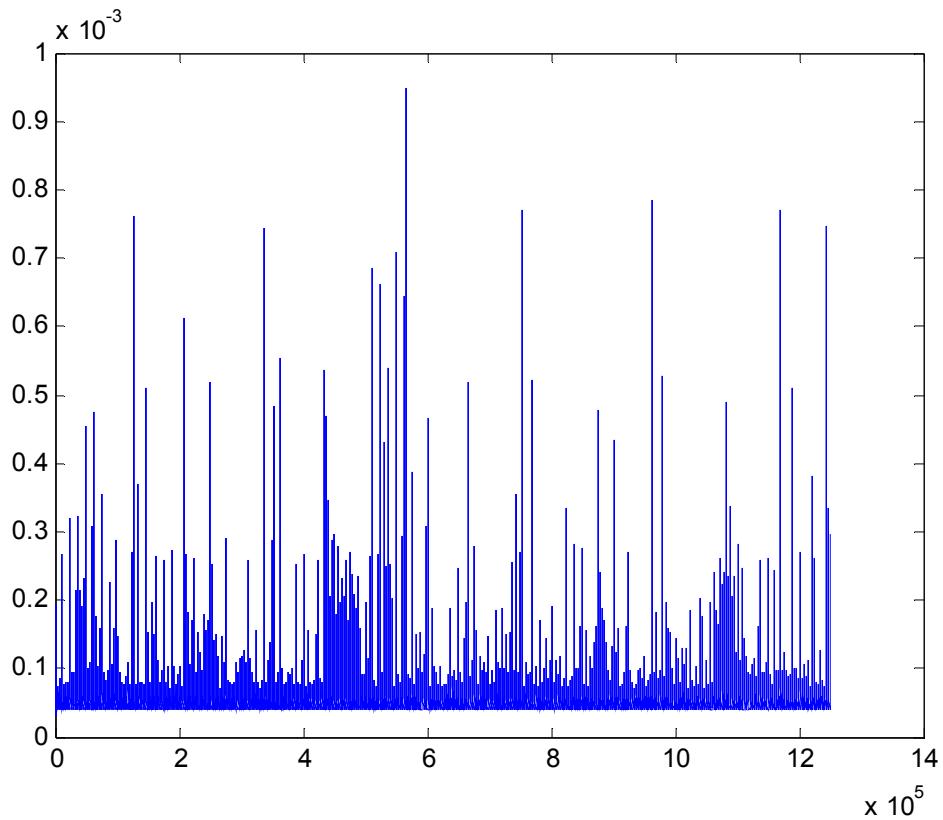
Con  $\text{delta}<0.0001$  se obtienen 1249558 valores (los graficados), los descartados con  $\text{delta}>0.0001$ , y se obtienen 441 valores, es decir un 0,035288%.

Como se puede ver en la gráfica ampliada, existe dispersión de paquetes y la mayoría se concentran en  $41,5\mu s$ , siendo la media y la desviación estándar:  $47,99\mu s$  y  $8,36\mu s$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

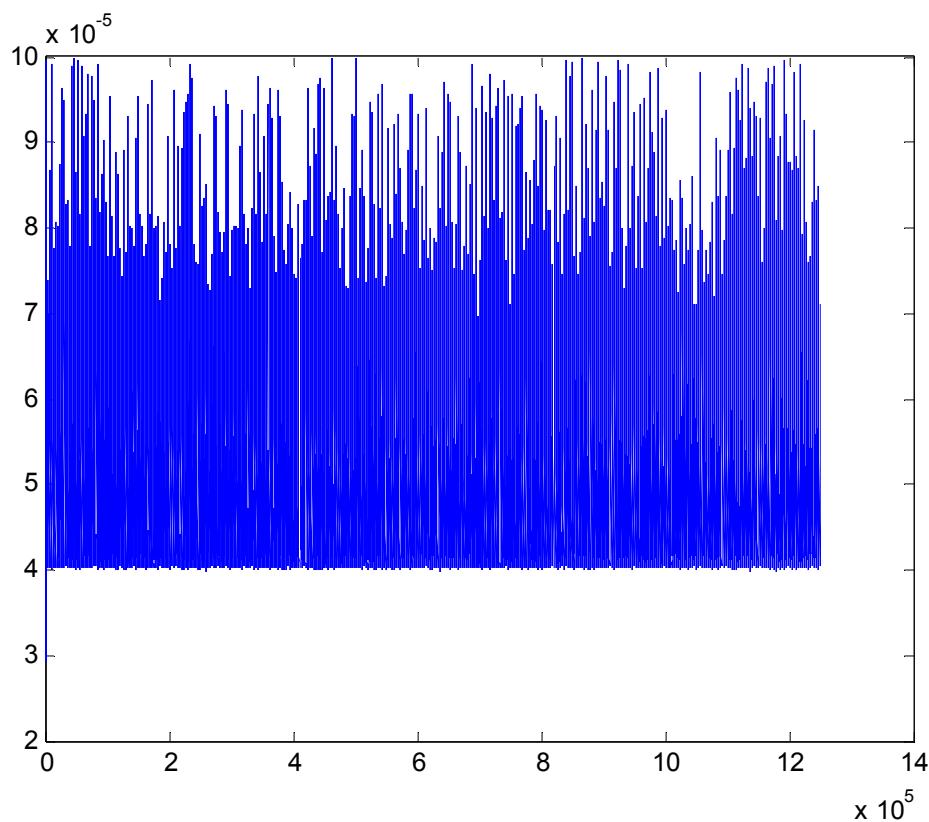
Con este comando obtenemos la siguiente gráfica:



**Fig. A5.29** Plot a 50 Mbps y 300 bytes

Para los valores superiores no se encuentra razón, ya que no existen paquetes intercalados entre esas muestras. Lo más probable es que sea culpa de la generación por parte de MGEN.

Si afinamos más el gráfico podemos observar mejor la poca dispersión que existe:



**Fig. A5.30** Plot ampliado

Si afinamos la gráfica para obtener valores inferiores a 1ms obtenemos algo más estable dentro de la media, y no se ven muestras inferiores a esta. Todos los valores están dentro del tiempo de transmisión mínimo.

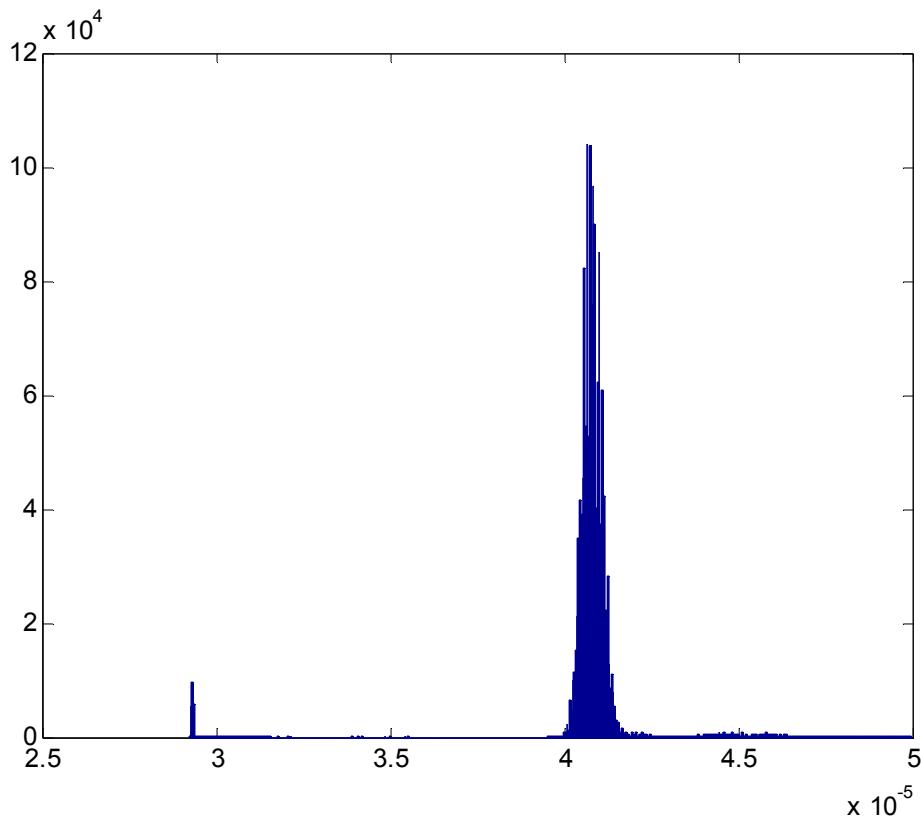
### Prueba de 300 bytes a 100 Mbps

En este caso, por parte del colector, se interpretan 1434 paquetes recibidos, mientras que por la DAG se reciben 1434236 paquetes y finalmente para el registro de MGEN se reciben 881717 paquetes con una velocidad obtenida de 629,499 kbps.

En este caso Netflow muestrea un 0,09% de los paquetes totales recibidos, es decir un resultado muy acorde con lo esperado.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00005)),500)
```



**Fig. A5.31** Plot a 100 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.00005 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.00005))
size(find(delta>0.00005))
```

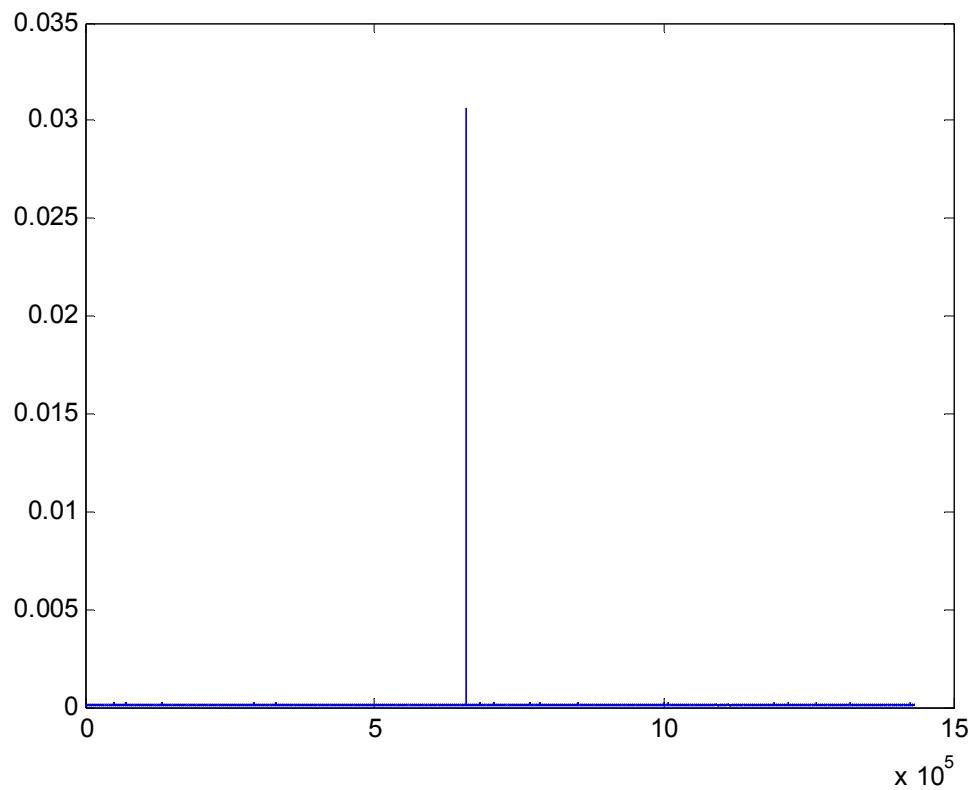
Con  $\delta t < 0.00005$  se obtienen 1422424 valores (los graficados), los descartados con  $\delta t > 0.00005$ , y se obtienen 11811valores, es decir un 0,823505%.

Como se puede ver en la gráfica ampliada, existe muy poca dispersión de paquetes y la mayoría se concentran en 40,7 $\mu$ s, siendo la media y la desviación estándar: 41,84 $\mu$ s y 28,32 $\mu$ s respectivamente.

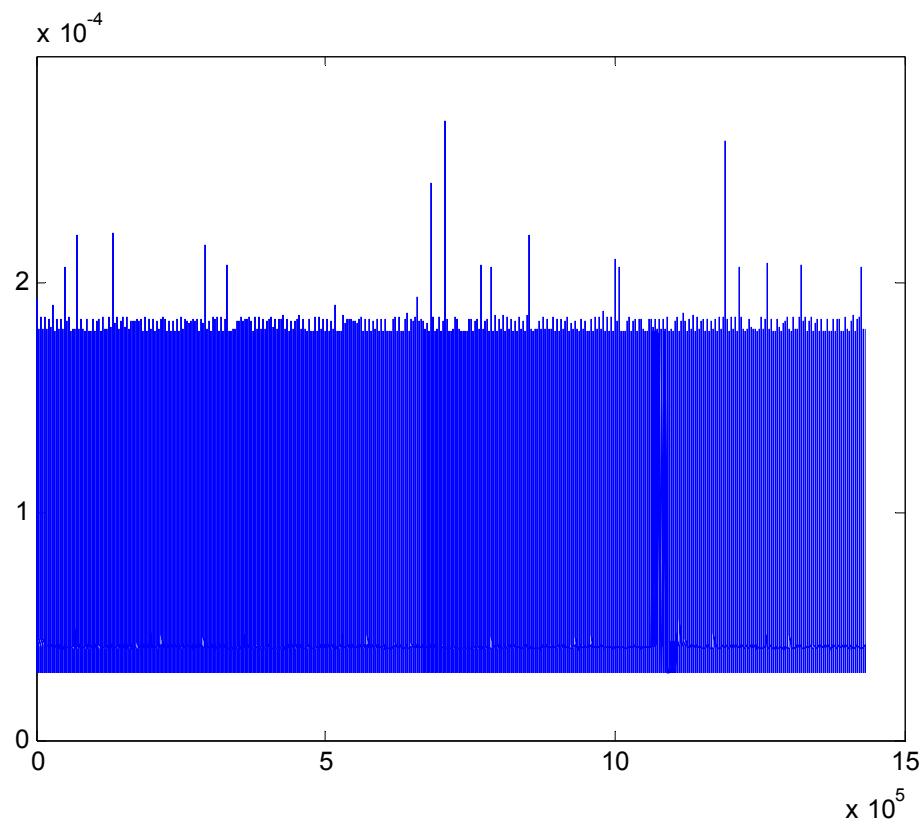
A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:



**Fig. A5.32** Plot a 100 Mbps y 300 bytes



**Fig. A5.33 Plot ampliado**

Observamos que el valor máximo no supera los 0.03 segundos, siendo este una cantidad mínima. Como se puede apreciar los valores mínimos no bajan de 29,17 $\mu$ s con lo que están por encima del umbral mínimo.

En este caso el único valor que desvía exageradamente de la media es una diferencia entre dos paquetes en los que se entrelaza con un Loop y un paquete CDP.

## Muestreo 1 de cada 65535 paquetes

La configuración es la siguiente para uno de cada 1000:

```
Crear mapa de muestreo, modo y frecuencia
flow-sampler-map nombre_mapa
mode random one-out-of <frecuencia_muestreo>

Cisco3700(config)#flow-sampler-map TEST1-65535
Cisco3700(config-sampler)#mode random one-out-of 65535

Habilitar mapa de muestreo en interfaz
flow-sampler <nombre_mapa>
Deshabilitar Full Netflow
no ip route-cache flow

Cisco3700(config-if)#flow-sampler TEST1-65535
Cisco3700(config-if)#no ip route-cache flow
```

## Prueba de 1400 bytes a 50 Mbps

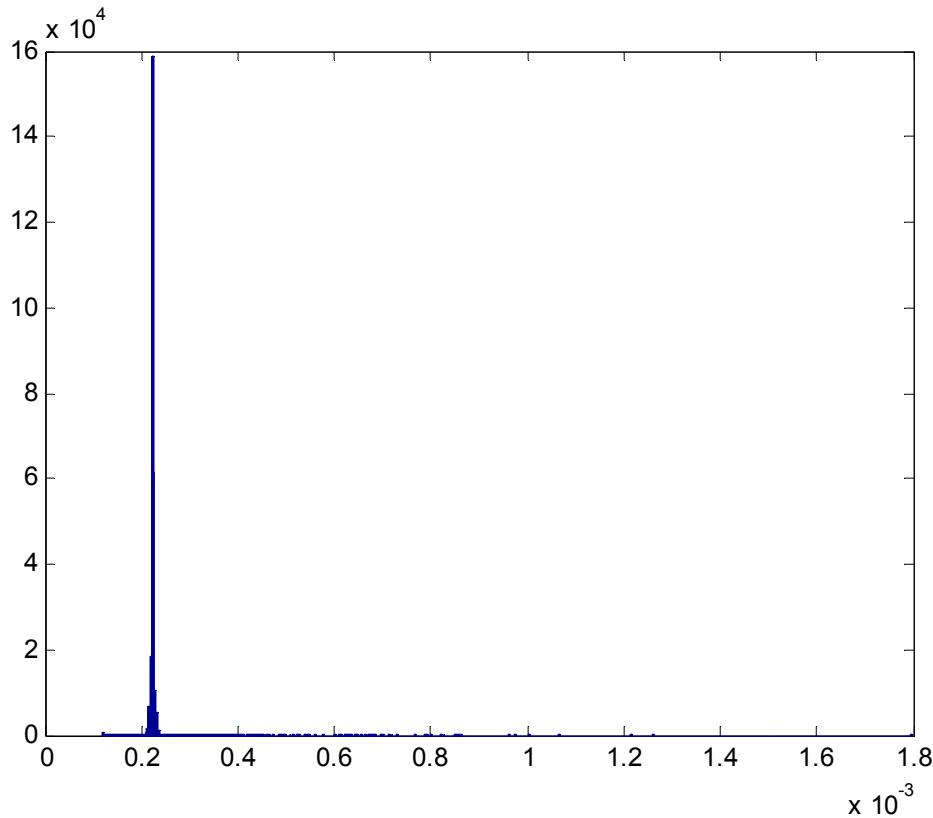
En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogido por la DAG, Nfsen y el registro de salida de MGEN.

Por parte del colector, se interpretan 3 paquetes recibidos en un flujo y 1 en otro, mientras que por la DAG se reciben 267858 paquetes y finalmente para el registro de MGEN se reciben 267818 paquetes con una velocidad obtenida de 1,232 kbps.

En este caso Netflow muestrea un 0,0011% de los paquetes totales recibidos, siendo un muestreo de 1 de cada 65535 se deberían obtener 4 paquetes respecto a la DAG, con lo que afirmamos que Nfsen tiene un error de un 0%.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta,500)
```



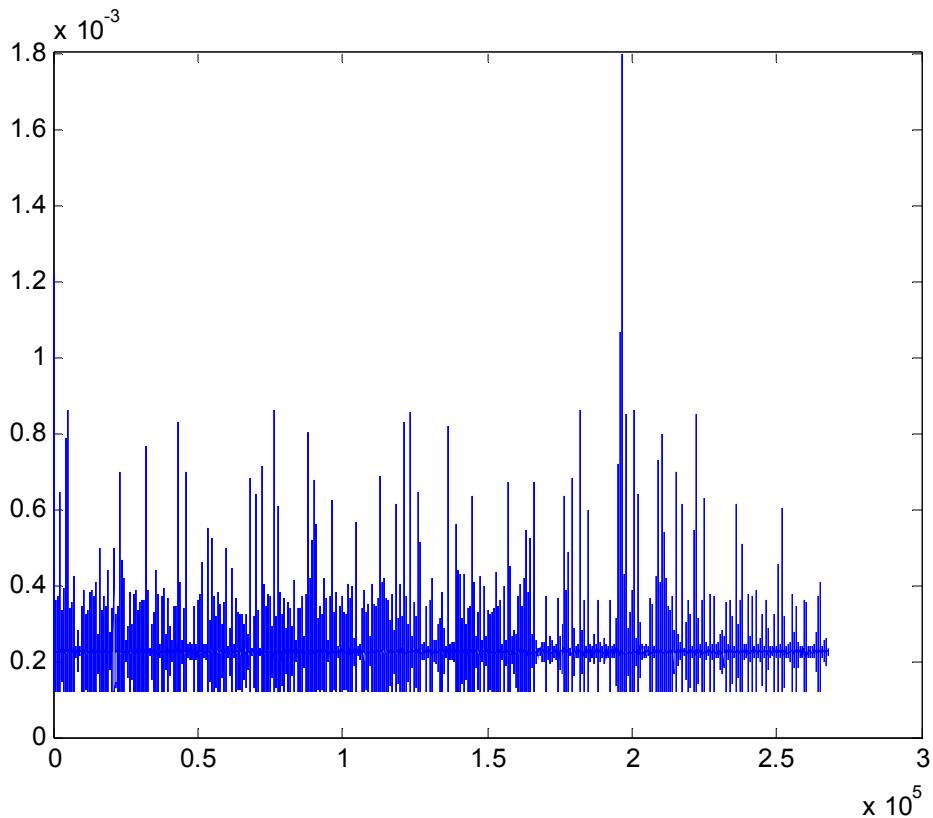
**Fig. A5.34** Histograma a 50 Mbps y 1400 bytes

Se puede observar como existe algo de dispersión. Concentrándose la mayoría de paquetes en 223 $\mu$ s segundos, siendo la media y la desviación estándar: 223,98 $\mu$ s y 11,96 $\mu$ s respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:



**Fig. A5.35 Plot a 50 Mbps y 1400 bytes**

Observamos que el valor máximo no supera los 1.8ms segundos ni están por debajo del valor mínimo de transmisión. El valor denotado llega a ser 1,81ms, una muestra muy elevada que no tiene explicación alguna, ya que visualizando con wireshark no se encuentran anomalías ni paquetes intercalados en esa posición.

### Prueba de 1400 bytes a 100 Mbps

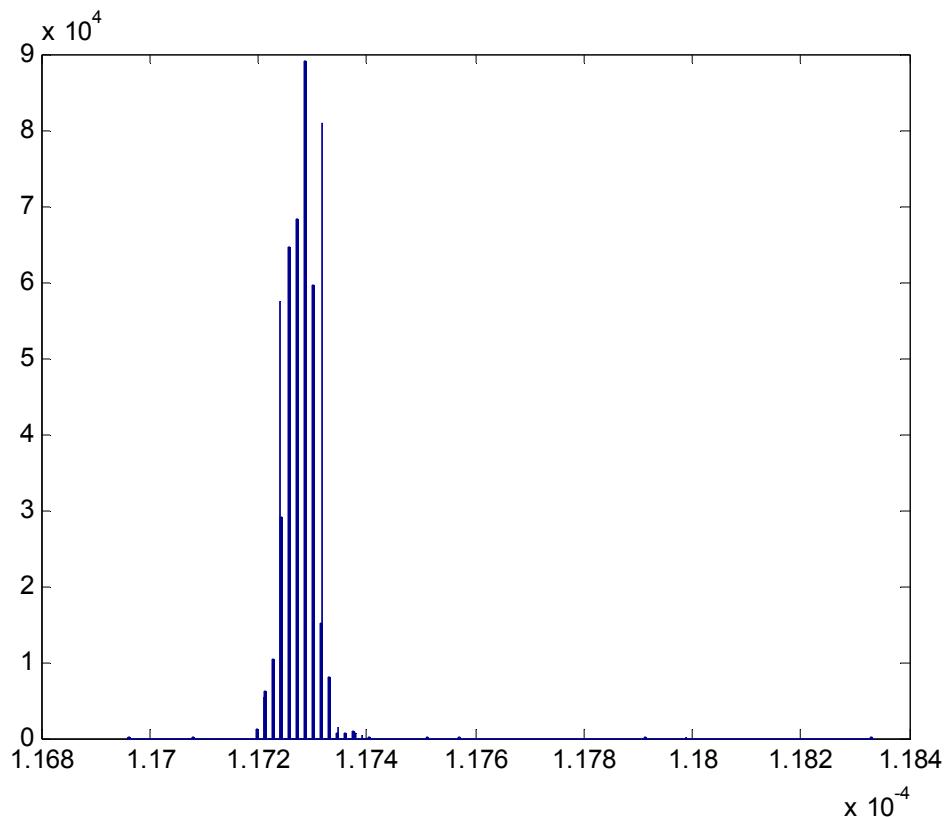
En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogido por la DAG, Nfsen y el registro de salida de MGEN.

Por parte del colector, se interpretan 8 paquetes recibidos, mientras que por la DAG se reciben 511667 paquetes y finalmente para el registro de MGEN se reciben 509066 paquetes con una velocidad obtenida de 1,707 kbps.

En este caso Netflow muestrea un 0,0015% de los paquetes totales recibidos, siendo un muestreo de 1 de cada 65535 se deberían obtener 7 paquetes respecto a la DAG, con lo que afirmamos que Nfsen a veces redondea hacia arriba o hacia abajo.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00012)),500)
```



**Fig. A5.36** Histograma a 100 Mbps y 1400 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.00012 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.00012))
size(find(delta>0.00012))
```

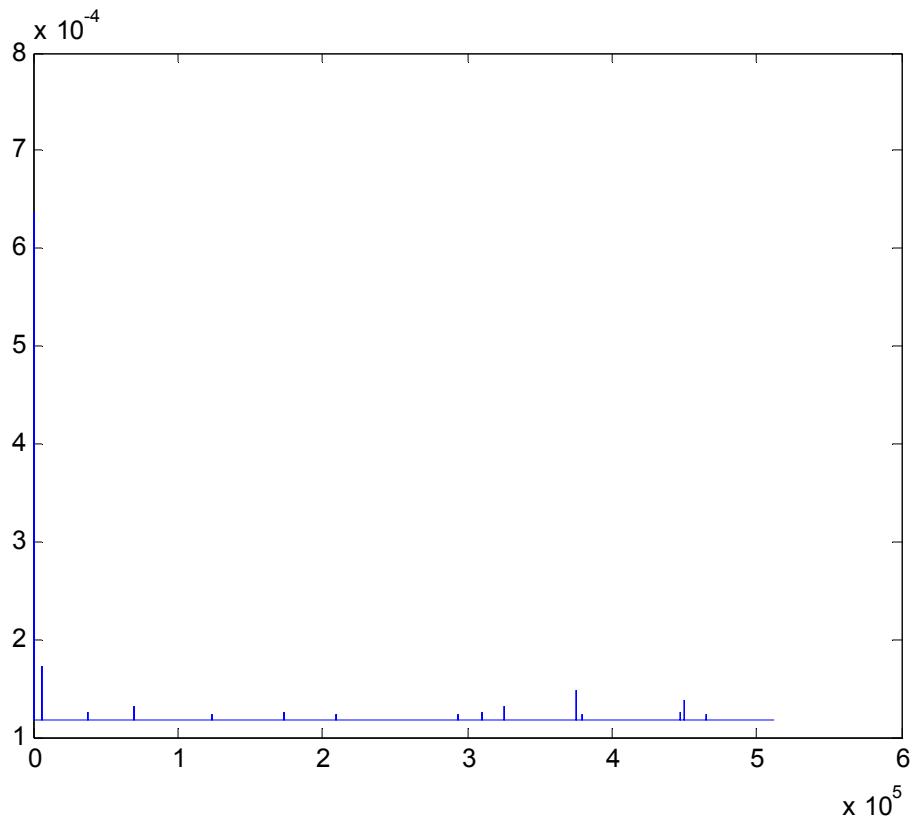
Con  $\text{delta} < 0.00012$  se obtienen 511647 valores (los graficados), los descartados con  $\text{delta} > 0.00012$ , y se obtienen 19 valores, es decir un 0,003713%.

Se puede observar como existe muy poca dispersión. Concentrándose la mayoría de paquetes en 117,3 $\mu$ s, siendo la media y la dispersión estándar: 117,28 $\mu$ s y 0,73 $\mu$ s respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:

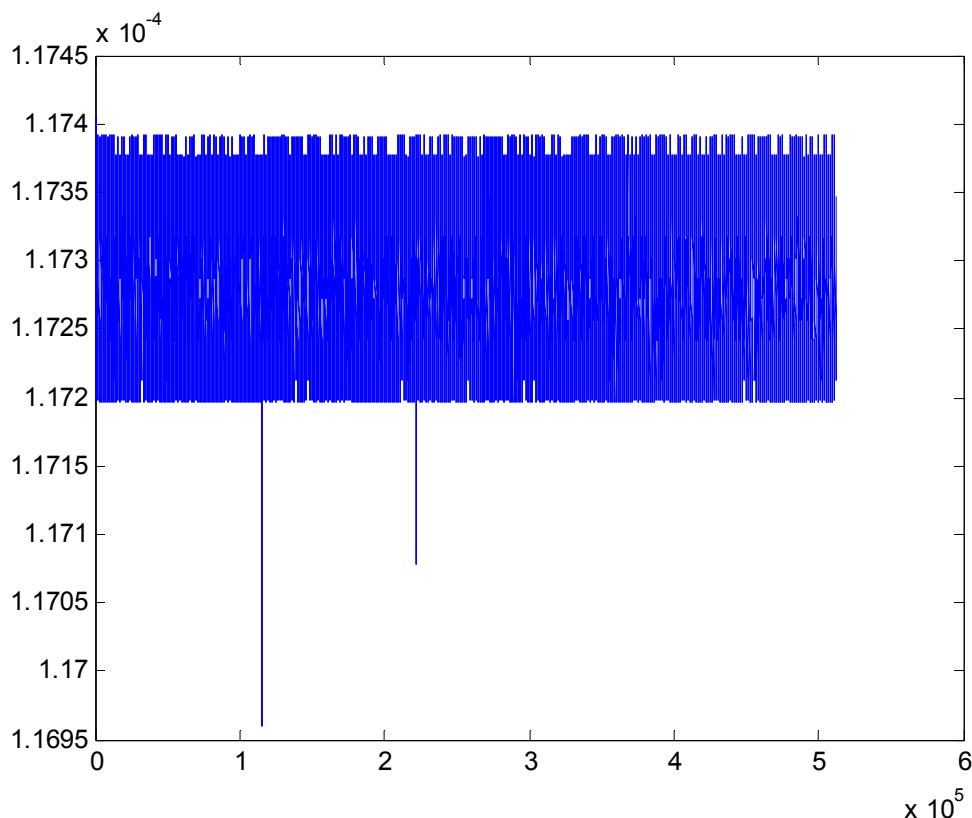


**Fig. A5.37** Plot a 100 Mbps y 1400 bytes

En este caso los superiores a la media son paquetes que tienen entre ellos paquetes NBNS. La cantidad de muestras se consigue filtrando por valores superiores a 0.000175 con `size(find(delta4>0.0001175))` y resultan 24 valores.

El valor más alto no tiene explicación ya que no tiene paquetes intercalados, suponemos que es error de generación de MGEN.

Si afinamos más el gráfico podemos observar mejor la dispersión:



**Fig. A5.38** Plot ampliado

Vemos como hay dos valores por debajo de la media, los cuales tienen paquetes entre medio de broadcast e ipv6. La gran cantidad de muestras están por encima de 117,2 $\mu$ s por lo que estamos en el mínimo.

### Prueba de 300 bytes a 50 Mbps

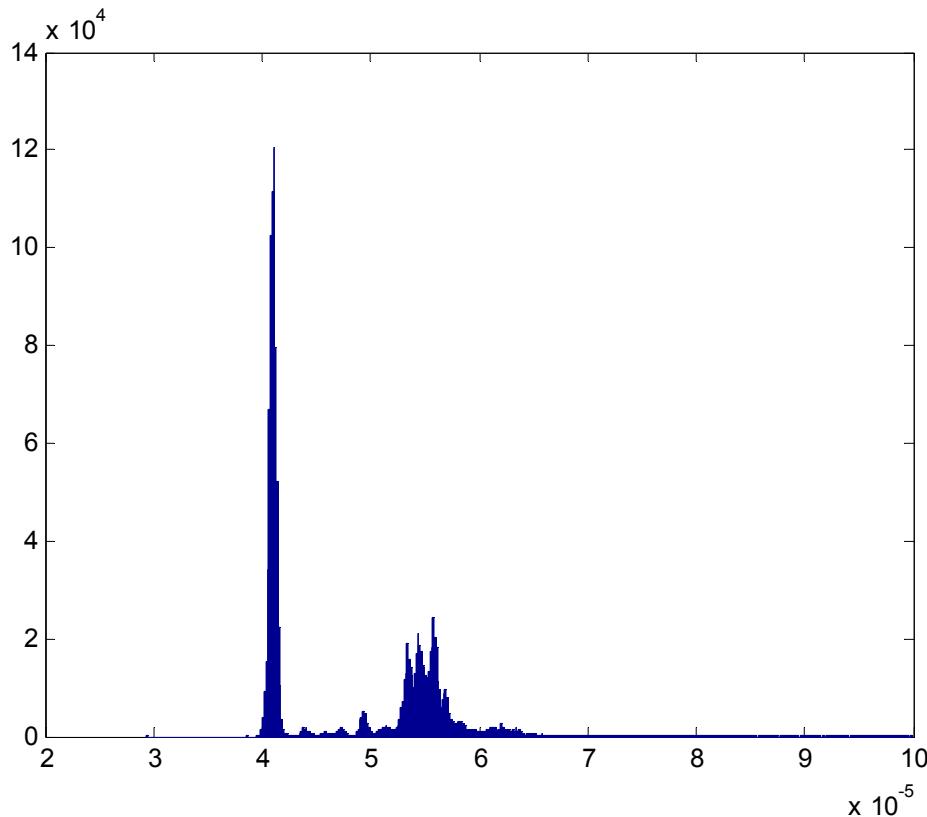
En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogido por la DAG, Nfsen y el registro de salida de MGEN.

Por parte del colector, se interpretan 19 paquetes recibidos, mientras que por la DAG se reciben 1250001 paquetes y finalmente para el registro de MGEN se reciben 1100399 paquetes con una velocidad obtenida de 879bps.

En este caso Netflow muestrea un 0,0015% de los paquetes totales recibidos, siendo un muestreo de 1 de cada 65535 se deberían obtener 19 paquetes respecto a la DAG, con lo que afirmamos que Nfsen tiene un error de un 0%.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.0001)), 500)
```



**Fig. A5.39** Histograma a 50 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.0001 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.0001))
size(find(delta>0.0001))
```

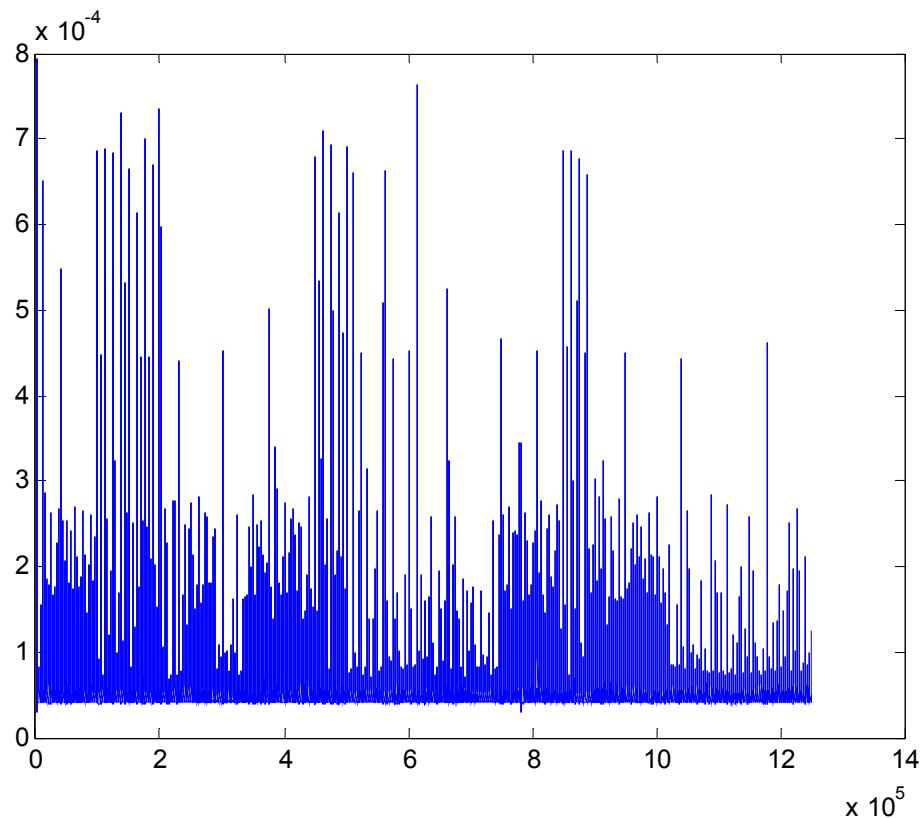
Con  $\text{delta} < 0.0001$  se obtienen 1249174 valores (los graficados), los descartados con  $\text{delta} > 0.0001$ , y se obtienen 826 valores, es decir un 0,06608%.

Como se puede ver en la gráfica ampliada, existe dispersión de paquetes y la mayoría se concentran en 41,1  $\mu s$ , siendo la media y la desviación estándar: 48,01  $\mu s$  y 8,96  $\mu s$  respectivamente.

A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

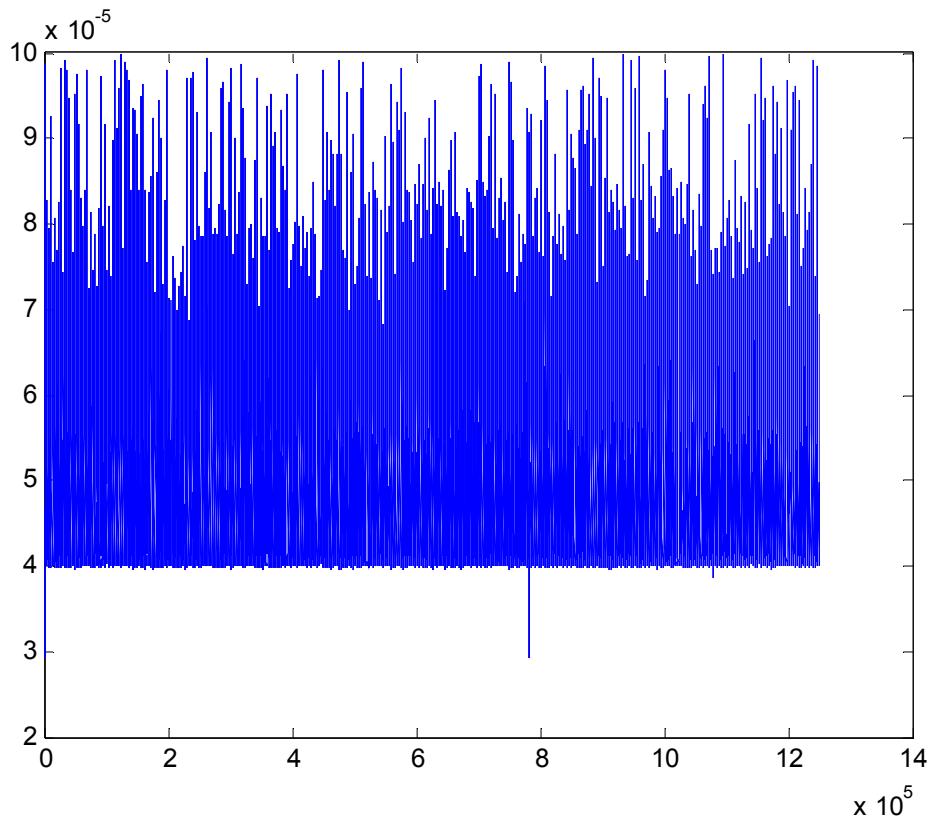
Con este comando obtenemos la siguiente gráfica:



**Fig. A5.40** Plot a 50 Mbps y 300 bytes

Para los valores superiores no se encuentra razón, ya que no existen paquetes intercalados entre esas muestras. La generación de tantos paquetes por segundo puede provocar imprecisiones al software MGEN.

Si afinamos más el gráfico podemos observar mejor la poca dispersión que existe:



**Fig. A5.41** Plot ampliado

Si afinamos la gráfica para obtener valores inferiores a 0,1ms obtenemos algo más estable dentro de la media, se ven muy pocas muestras inferiores a esta. Todos los valores están dentro del tiempo de transmisión mínimo.

### Prueba de 300 bytes a 100 Mbps

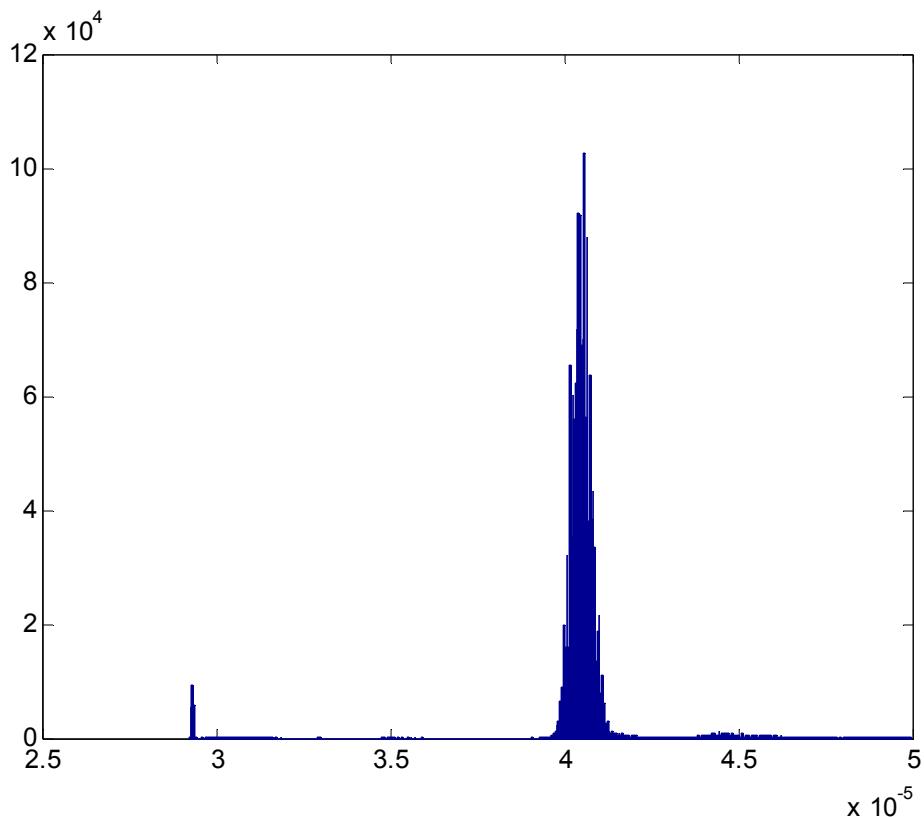
En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogido por la DAG, Nfsen y el registro de salida de MGGEN.

Por parte del colector, se interpretan 22 paquetes recibidos, mientras que por la DAG se reciben 1421460 paquetes y finalmente para el registro de MGGEN se reciben 1182802 paquetes con una velocidad obtenida de 991bps.

En este caso Netflow muestrea un 0,0015% de los paquetes totales recibidos, siendo un muestreo de 1 de cada 65535 se deberían obtener 21 paquetes respecto a la DAG, con lo que afirmamos que Nfsen tiene un error de un 4,54%.

A continuación veremos el histograma del tráfico con el siguiente comando:

```
hist(delta(find(delta<0.00005)),500)
```



**Fig. A5.42** Histograma a 100 Mbps y 300 bytes

Al ser una prueba con alta cantidad de paquetes, se debe afinar más la resolución de la gráfica y despreciar las muestras superiores a 0.00005 segundos para poder visualizar el comportamiento del histograma. Los valores utilizados y los despreciados se pueden ver con estos comandos:

```
size(find(delta<0.00005))
size(find(delta>0.00005))
```

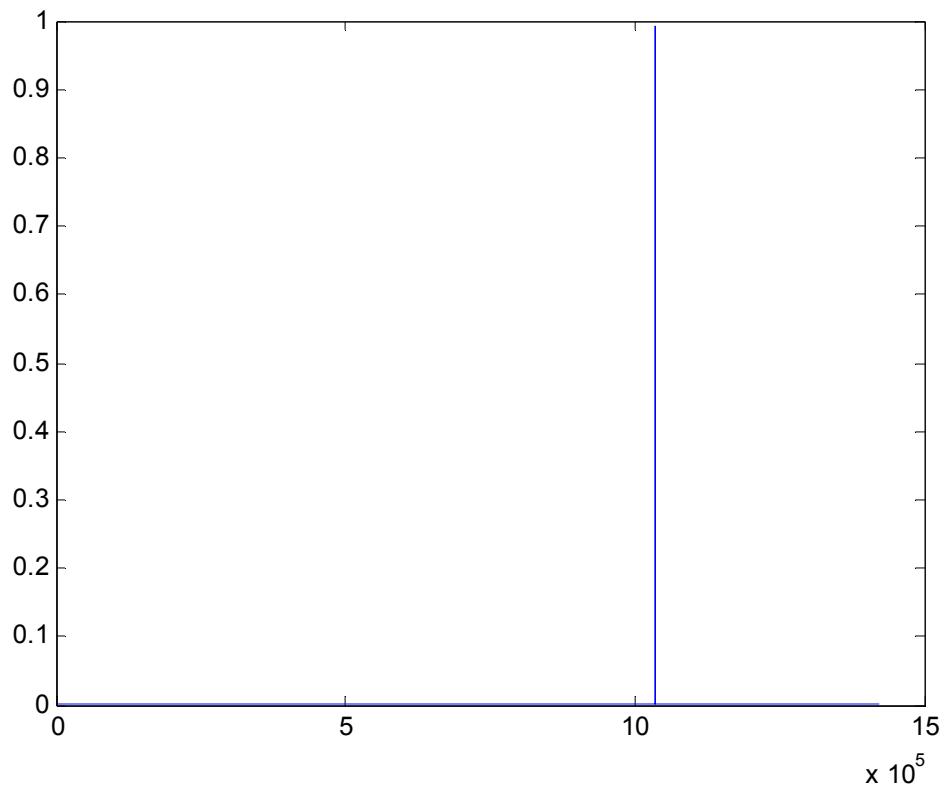
Con  $\text{delta}<0.00005$  se obtienen 1409903 valores (los graficados), los descartados con  $\text{delta}>0.00005$ , y se obtienen 11556 valores, es decir un 0,812967%.

Como se puede ver en la gráfica ampliada, existe muy poca dispersión de paquetes y la mayoría se concentran en 40,6 $\mu$ s, siendo la media y la desviación estándar: 42,21 $\mu$ s y 832,34 $\mu$ s respectivamente.

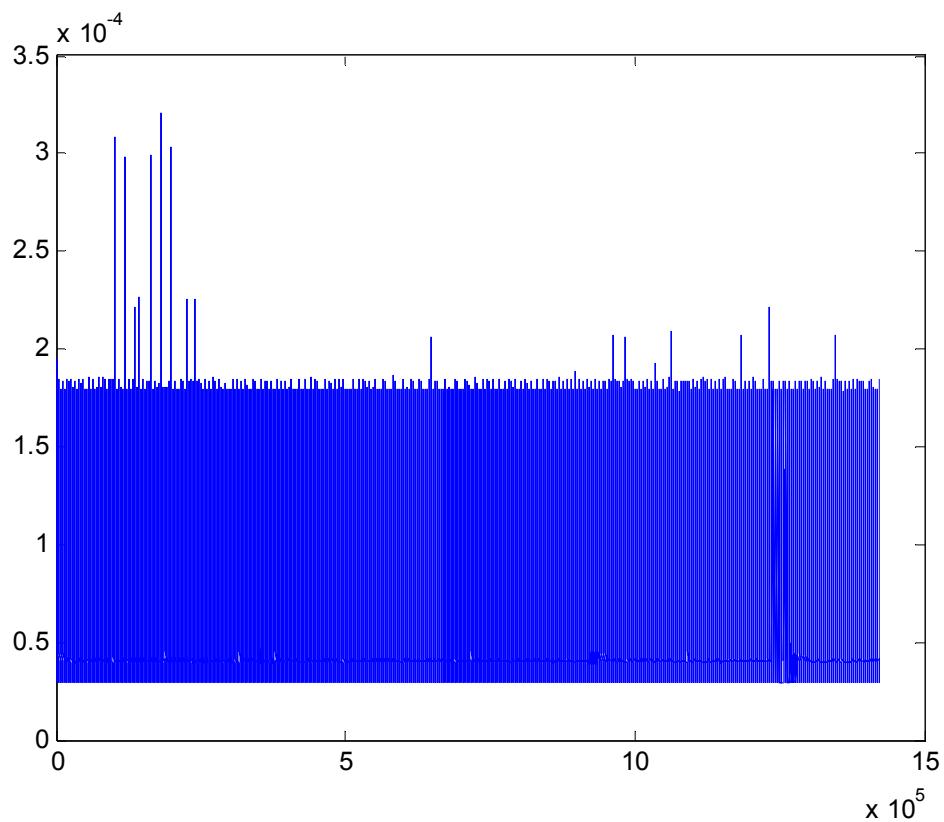
A continuación realizaremos una gráfica estándar para comprobar el estado de los valores y verificar su valor máximo y mínimo.

```
plot(delta)
```

Con este comando obtenemos la siguiente gráfica:



**Fig. A5.43** Plot a 100 Mbps y 300 bytes



**Fig. A5.44 Plot ampliado**

Observamos que el valor máximo no supera los 35ms segundos, siendo este una cantidad mínima. Como se puede apreciar los valores mínimos no bajan de 29,17 $\mu$ s con lo que están por encima del tiempo mínimo de transmisión.

En este caso el único valor que desvía exageradamente de la media es una diferencia entre dos paquetes en los que se entrelazan 44 paquetes, lo que hace obtener una diferencia muy alta.

## VI. Pruebas iniciales de 6 flujos

### Prueba de 6 flujos con Full Netflow

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

Nfsen nos proporciona la siguiente información:

```
** nfdump -M /usr/local/nfsen/profiles-data/live/upstream1 -T -r 2012/06/19/nfcapd.201206191215 -o extended -m -c 20
nfdump filter:
proto UDP and dst ip 147.83.118.237
Date flow start Duration Proto Src IP Addr:Port Dst IP Addr:Port Flags Tos Packets Bytes pps bps Bpp Flows
2012-06-19 12:16:29.199 60.000 UDP 10.0.13.101:5501 -> 147.83.118.237:5501 .A.... 0 267858 382.5 M 4464 51.0 M 1428 1
2012-06-19 12:16:29.201 0.000 UDP 10.0.13.101:5506 -> 147.83.118.237:5506 .A.... 0 1 1428 0 0 1428 1
2012-06-19 12:16:29.203 59.996 UDP 10.0.13.101:5502 -> 147.83.118.237:5502 .A.... 0 26762 38.2 M 446 5.1 M 1428 1
2012-06-19 12:16:29.203 59.088 UDP 10.0.13.101:5505 -> 147.83.118.237:5505 .A.... 0 27 38556 0 5220 1428 1
2012-06-19 12:16:29.203 59.996 UDP 10.0.13.101:5503 -> 147.83.118.237:5503 .A.... 0 2641 3.8 M 44 502879 1428 1
2012-06-19 12:16:29.203 59.996 UDP 10.0.13.101:5504 -> 147.83.118.237:5504 .A.... 0 241 344148 4 45889 1428 1
2012-06-19 12:16:51.929 0.000 UDP 10.0.13.101:5506 -> 147.83.118.237:5506 .A.... 0 1 1428 0 0 1428 1
2012-06-19 12:17:14.654 0.000 UDP 10.0.13.101:5506 -> 147.83.118.237:5506 .A.... 0 1 1428 0 0 1428 1
Summary: total flows: 8, total bytes: 424.9 M, total packets: 297532, avg bps: 56.7 M, avg pps: 4958, avg bpp: 1428
Time window: 2012-04-30 19:11:58 - 2012-06-19 12:19:13
Total flows processed: 179, Blocks skipped: 0, Bytes read: 9336
Sys: 0.004s flows/second: 44750.0 Wall: 0.000s flows/second: 983516.5
```

**Fig. A6.1 Reporte de los 6 flujos en Nfsen con Full Netflow**

A continuación analizamos el porcentaje muestreado por Netflow respecto a lo capturado por la DAG:

**Tabla A6.1. Muestreo Full Netflow vs DAG**

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 267858                    | 267858                         | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 26762                     | 26762                          | 100%                         |
| Flujo3 – 500kbps | 1                  | 2641                      | 2641                           | 100%                         |
| Flujo 4 – 50kbps | 1                  | 241                       | 241                            | 100%                         |

|                  |   |    |    |      |
|------------------|---|----|----|------|
| Flujo 5 – 5kbps  | 1 | 27 | 27 | 100% |
| Flujo 6 – 500bps | 3 | 3  | 3  | 100% |

Al analizar esta tabla podemos decir que Netflow no muestra ningún error en el muestreo de paquetes, incluso partiendo el último flujo en tres.

## Prueba de 6 flujos con Random Netflow

### Muestreo 1 de cada 10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

Nfsen nos proporciona la siguiente información:

```
** nfdump -M /usr/local/nfse[n] profiles-data/live/upstream1 -T -r 2012/06/20/nfcapd.201206201140 -o extended -m -c 20
nfdump filter:
proto UDP and dst ip 147.83.118.237
Date flow start Duration Proto Src IP Addr:Port Dst IP Addr:Port Flags Tos Packets Bytes pps bps Bpp Flows
2012-06-20 11:42:20.057 0.000 UDP 10.0.13.101:5506 -> 147.83.118.237:5506 .A.... 0 1 1428 0 0 1428 1
2012-06-20 11:42:20.059 59.992 UDP 10.0.13.101:5501 -> 147.83.118.237:5501 .A.... 0 26883 38.4 M 448 5.1 M 1428 1
2012-06-20 11:42:20.103 59.700 UDP 10.0.13.101:5503 -> 147.83.118.237:5503 .A.... 0 247 352716 4 47265 1428 1
2012-06-20 11:42:20.115 59.924 UDP 10.0.13.101:5502 -> 147.83.118.237:5502 .A.... 0 2592 3.7 M 43 494142 1428 1
2012-06-20 11:42:22.330 18.176 UDP 10.0.13.101:5505 -> 147.83.118.237:5505 .A.... 0 4 5712 0 2514 1428 1
2012-06-20 11:42:22.807 56.496 UDP 10.0.13.101:5504 -> 147.83.118.237:5504 .A.... 0 24 34272 0 4853 1428 1
2012-06-20 11:42:56.419 13.632 UDP 10.0.13.101:5505 -> 147.83.118.237:5505 .A.... 0 4 5712 0 3352 1428 1
Summary: total flows: 7, total bytes: 42.5 M, total packets: 29755, avg bps: 5.7 M, avg pps: 495, avg bpp: 1428
Time window: 2012-06-20 11:40:01 - 2012-06-20 11:43:47
Total flows processed: 14, Blocks skipped: 0, Bytes read: 756
Sys: 0.004s flows/second: 3500.0 Wall: 0.000s flows/second: 91503.3
```

**Fig. A6.2 Reporte de los 6 flujos en Nfsen con Random Netflow 1/10**

A continuación analizamos el porcentaje muestreado por Netflow respecto a lo capturado por la DAG:

**Tabla A6.2. Muestreo Random Netflow 1/10 vs DAG**

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 267858                    | 26883                          | 10,03628788%                 |
| Flujo 2 – 5Mbps  | 1                  | 26762                     | 2592                           | 9,685374785%                 |
| Flujo3 – 500kbps | 1                  | 2641                      | 247                            | 9,352517986%                 |
| Flujo 4 – 50kbps | 1                  | 241                       | 24                             | 9,958506224%                 |
| Flujo 5 – 5kbps  | 2                  | 27                        | 8                              | 29,62962963%                 |
| Flujo 6 – 500bps | 1                  | 3                         | 1                              | 33,33333333%                 |

Al analizar esta tabla podemos decir que Netflow no muestra ningún error en el muestreo de paquetes, incluso partiendo el flujo 5 en dos, está muestreando más. Para el flujo 6, al muestrear un paquete de los tres transmitidos el porcentaje es muy alto.

### Muestreo 1 de cada 100

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

Nfsen nos proporciona la siguiente información:

```
** nfdump -M /usr/local/nfsen/profiles-data/live/upstream1 -T -r 2012/06/20/nfcapd.201206201245 -o extended -m -c 20
nfdump filter:
proto UDP and dst ip 147.83.118.237
Date flow start Duration Proto Src IP Addr:Port Dst IP Addr:Port Flags Tos Packets Bytes pps bps Bpp Flows
2012-06-20 12:47:00.279 59.860 UDP 10.0.13.101:5502 -> 147.83.118.237:5502 .A.... 0 252 359856 4 48093 1428 1
2012-06-20 12:47:00.307 59.964 UDP 10.0.13.101:5501 -> 147.83.118.237:5501 .A.... 0 2701 3.9 M 45 514579 1428 1
2012-06-20 12:47:02.819 51.800 UDP 10.0.13.101:5503 -> 147.83.118.237:5503 .A.... 0 18 25704 0 3969 1428 1
2012-06-20 12:47:30.274 0.252 UDP 10.0.13.101:5504 -> 147.83.118.237:5504 .A.... 0 2 2856 7 90666 1428 1
2012-06-20 12:47:36.643 13.636 UDP 10.0.13.101:5505 -> 147.83.118.237:5505 .A.... 0 2 2856 0 1675 1428 1
2012-06-20 12:47:57.527 0.000 UDP 10.0.13.101:5504 -> 147.83.118.237:5504 .A.... 0 1 1428 0 0 1428 1
Summary: total flows: 6, total bytes: 4.2 M, total packets: 2976, avg bps: 566705, avg pps: 49, avg bpp: 1428
Time window: 2012-06-20 12:47:00 - 2012-06-20 12:48:00
Total flows processed: 6, Blocks skipped: 0, Bytes read: 340
Sys: 0.004s flows/second: 1500.0 Wall: 0.000s flows/second: 47619.0
```

**Fig. A6.3 Reporte de los 6 flujos en Nfsen con Random Netflow 1/100**

A continuación analizamos el porcentaje muestreado por Netflow respecto a lo capturado por la DAG:

**Tabla A6.3. Muestreo Random Netflow 1/100 vs DAG**

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 267858                    | 2701                           | 1,008370107%                 |
| Flujo 2 – 5Mbps  | 1                  | 26762                     | 252                            | 0,94163366 %                 |
| Flujo3 – 500kbps | 1                  | 2641                      | 18                             | 0,681560015%                 |
| Flujo 4 – 50kbps | 2                  | 241                       | 3                              | 1,244813278%                 |
| Flujo 5 – 5kbps  | 1                  | 27                        | 2                              | 7,407407407%                 |
| Flujo 6 – 500bps | 0                  | 3                         | 0                              | 0%                           |

Al analizar esta tabla podemos decir que Netflow detecta y muestrea correctamente los cuatro primeros flujos, en el quinto ya muestrea más de los deseado y el flujo 6 es indetectable por la poca cantidad de paquetes y la probabilidad de conteo.

### Muestreo 1 de cada 1000

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

Nfsen nos proporciona la siguiente información:

```
** nfdump -M /usr/local/nfSEN/profiles-data/live/upstream1 -T -r 2012/06/20/nfcapd.201206201805 -o extended -m -c 20
nfdump filter:
proto UDP and dst ip 147.83.118.237
Date flow start      Duration Proto      Src IP Addr:Port      Dest IP Addr:Port      Flags Tos  Packets      Bytes      pps      bps      Bpp Flows
2012-06-20 18:06:10.237    59.588 UDP      10.0.13.101:5501 -> 147.83.118.237:5501 .A.... 0     269      384132      4      51571    1428    1
2012-06-20 18:06:13.373    55.760 UDP      10.0.13.101:5502 -> 147.83.118.237:5502 .A.... 0     25      35700      0      5121    1428    1
2012-06-20 18:06:38.868    0.000 UDP      10.0.13.101:5503 -> 147.83.118.237:5503 .A.... 0     1      1428      0      0      1428    1
2012-06-20 18:06:53.281    0.000 UDP      10.0.13.101:5505 -> 147.83.118.237:5505 .A.... 0     1      1428      0      0      1428    1
2012-06-20 18:06:56.853    0.000 UDP      10.0.13.101:5504 -> 147.83.118.237:5504 .A.... 0     1      1428      0      0      1428    1
Summary: total flows: 5, total bytes: 424116, total packets: 297, avg bps: 56939, avg pps: 4, avg bpp: 1428
Time window: 2012-06-20 18:06:10 - 2012-06-20 18:07:09
Total flows processed: 5, Blocks skipped: 0, Bytes read: 288
Sys: 0.004s flows/second: 1250.0      Wall: 0.000s flows/second: 40983.6
```

**Fig. A6.4** Reporte de los 6 flujos en Nfsen con Random Netflow 1/1000

A continuación analizamos el porcentaje muestreado por Netflow respecto a lo capturado por la DAG:

**Tabla A6.4.** Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 267858                    | 269                            | 0,100426345%                 |
| Flujo 2 – 5Mbps  | 1                  | 26762                     | 25                             | 0,093416038%                 |
| Flujo3 – 500kbps | 1                  | 2641                      | 1                              | 0,037864445%                 |
| Flujo 4 – 50kbps | 1                  | 241                       | 1                              | 0,414937759%                 |
| Flujo 5 – 5kbps  | 1                  | 27                        | 1                              | 3,703703704%                 |
| Flujo 6 – 500bps | 0                  | 3                         | 0                              | 0%                           |

Al analizar esta tabla podemos decir que Netflow detecta y muestrea correctamente los dos primeros flujos, en el tercero muestrea menos, el cuarto y el quinto ya muestrean más de lo deseado, finalmente el flujo 6 es indetectable por la poca cantidad de paquetes y la probabilidad de conteo.

### Muestreo 1 de cada 65535

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

Nfsen nos proporciona la siguiente información:

```
** nfdump -M /usr/local/nfsen/profiles-data/live/upstream1 -T -r 2012/06/20/nfcapd.201206201820 -o extended -m -c 20
nfdump filter:
proto UDP and dst ip 147.83.118.237
Date flow start      Duration Proto      Src IP Addr:Port      Dst IP Addr:Port Flags Tos Packets    Bytes     pps     bps     Bpp Flows
2012-06-20 18:20:51.484   54.072 UDP 10.0.13.101:5501 -> 147.83.118.237:5501 .A..... 0      5    7140     0     1056   1428   1
Summary: total flows: 1, total bytes: 7140, total packets: 5, avg bps: 1056, avg pps: 0, avg bpp: 1428
Time window: 2012-06-20 18:20:51 - 2012-06-20 18:21:45
Total flows processed: 2, Blocks skipped: 0, Bytes read: 132
Sys: 0.004s flows/second: 500.0          Wall: 0.000Us flows/second: 16949.2
```

**Fig. A6.5** Reporte de los 6 flujos en Nfsen con Random Netflow 1/65535

A continuación analizamos el porcentaje muestreado por Netflow respecto a lo capturado por la DAG:

**Tabla A6.5.** Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 267858                    | 5                              | 0,001866661%                 |
| Flujo 2 – 5Mbps  | 0                  | 26762                     | 0                              | 0%                           |
| Flujo3 – 500kbps | 0                  | 2641                      | 0                              | 0%                           |
| Flujo 4 – 50kbps | 0                  | 241                       | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 27                        | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 3                         | 0                              | 0%                           |

Al analizar esta tabla podemos decir que Netflow detecta y muestrea correctamente sólo el primer flujo, los siguientes son indetectables por baja probabilidad de conteo.

## VII. 10 pruebas con Full/Random Netflow

### 10 pruebas con Full Netflow

#### Prueba 1/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.1.** Prueba 1 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1339286                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

### Prueba 2/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.2.** Prueba 2 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1339286                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

### Prueba 3/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.3.** Prueba 3 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 1339287                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

### Prueba 4/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.4.** Prueba 4 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1339286                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

### Prueba 5/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.5.** Prueba 5 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1339286                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

### Prueba 6/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.6.** Prueba 6 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 1339287                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

## Prueba 7/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.7.** Prueba 7 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1339286                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

## Prueba 8/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.8.** Prueba 8 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1339286                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |

|                  |    |    |    |      |
|------------------|----|----|----|------|
| Flujo 6 – 500bps | 14 | 14 | 14 | 100% |
|------------------|----|----|----|------|

### Prueba 9/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.9.** Prueba 9 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 1339287                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

### Prueba 10/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.10.** Prueba 10 - Muestreo Full Netflow vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 1339287                        | 100%                         |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 133810                         | 100%                         |
| Flujo3 – 500kbps | 1                  | 13201                     | 13201                          | 100%                         |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1201                           | 100%                         |
| Flujo 5 – 5kbps  | 1                  | 133                       | 133                            | 100%                         |
| Flujo 6 – 500bps | 14                 | 14                        | 14                             | 100%                         |

### Media y desviación de las pruebas

En esta tabla se muestra el porcentaje muestreado respecto a la DAG, para cada flujo y para cada prueba. Al final se calcula la media del porcentaje de cada flujo.

**Tabla A7.11.** Media y desviación con Full Netflow

| Número de Flujo | Full Netflow |          |          |          |          |          |          |          |          |           |            |
|-----------------|--------------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|------------|
|                 | 1/10 (%)     | 2/10 (%) | 3/10 (%) | 4/10 (%) | 5/10 (%) | 6/10 (%) | 7/10 (%) | 8/10 (%) | 9/10 (%) | 10/10 (%) | Media (%)  |
| 1-50M           | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 2-5M            | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 3-500k          | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 4-50k           | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 5-5k            | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |
| 6-500b          | 100          | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100      | 100       | <b>100</b> |

Como se observa, con Full Netflow se muestrean todos y cada uno de los paquetes que atraviesan el router.

## 10 pruebas con Full Netflow

### Muestreo 1 de cada 10

#### Prueba 1/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.12.** Prueba 1 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 134128                         | 10,01%                       |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 13171                          | 9,84%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 1332                           | 10,09%                       |
| Flujo 4 – 50kbps | 2                  | 1201                      | 126                            | 10,49%                       |
| Flujo 5 – 5kbps  | 6                  | 133                       | 11                             | 8,27%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

#### Prueba 2/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.13.** Prueba 2 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 133413                         | 10,01%                       |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 13868                          | 9,84%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 1341                           | 10,09%                       |
| Flujo 4 – 50kbps | 1                  | 1201                      | 125                            | 10,49%                       |
| Flujo 5 – 5kbps  | 8                  | 133                       | 16                             | 8,27%                        |
| Flujo 6 – 500bps | 1                  | 14                        | 1                              | 7,14%                        |

**Prueba 3/10**

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.14.** Prueba 3 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 133827                         | 9,99%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 13504                          | 10,09%                       |
| Flujo3 – 500kbps | 1                  | 13201                     | 1296                           | 9,81%                        |
| Flujo 4 – 50kbps | 1                  | 1201                      | 117                            | 9,74%                        |
| Flujo 5 – 5kbps  | 9                  | 133                       | 15                             | 11,27%                       |
| Flujo 6 – 500bps | 4                  | 14                        | 4                              | 28,57%                       |

**Prueba 4/10**

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.15.** Prueba 4 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 133533                         | 9,97%                        |

|                  |   |        |       |        |
|------------------|---|--------|-------|--------|
| Flujo 2 – 5Mbps  | 1 | 133810 | 13767 | 10,28% |
| Flujo3 – 500kbps | 1 | 13201  | 1320  | 9,99%  |
| Flujo 4 – 50kbps | 1 | 1201   | 129   | 10,74% |
| Flujo 5 – 5kbps  | 9 | 133    | 13    | 9,77%  |
| Flujo 6 – 500bps | 0 | 14     | 0     | 0%     |

### Prueba 5/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.16.** Prueba 5 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 134269                         | 10,02%                       |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 13039                          | 9,74%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 1318                           | 9,98%                        |
| Flujo 4 – 50kbps | 1                  | 1201                      | 119                            | 9,90%                        |
| Flujo 5 – 5kbps  | 8                  | 133                       | 15                             | 11,27%                       |
| Flujo 6 – 500bps | 3                  | 14                        | 3                              | 21,42%                       |

### Prueba 6/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.17.** Prueba 6 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 134394                         | 10,03%                       |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 12955                          | 9,68%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 1289                           | 9,76%                        |
| Flujo 4 – 50kbps | 2                  | 1201                      | 117                            | 9,74%                        |
| Flujo 5 – 5kbps  | 8                  | 133                       | 9                              | 6,76%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 7/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.18.** Prueba 7 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 133863                         | 9,99%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 13414                          | 10,02%                       |
| Flujo3 – 500kbps | 1                  | 13201                     | 1358                           | 10,28%                       |
| Flujo 4 – 50kbps | 1                  | 1201                      | 120                            | 9,99%                        |
| Flujo 5 – 5kbps  | 8                  | 133                       | 9                              | 6,76%                        |
| Flujo 6 – 500bps | 1                  | 14                        | 1                              | 7,14%                        |

### Prueba 8/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.19.** Prueba 8 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 134449                         | 10,03%                       |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 12877                          | 9,62%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 1327                           | 10,05%                       |
| Flujo 4 – 50kbps | 1                  | 1201                      | 100                            | 8,32%                        |
| Flujo 5 – 5kbps  | 7                  | 133                       | 10                             | 7,51%                        |
| Flujo 6 – 500bps | 1                  | 14                        | 1                              | 7,14%                        |

### Prueba 9/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.20.** Prueba 9 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo | Cantidad | Paquetes | Paquetes | Porcentaje |
|---------------|----------|----------|----------|------------|
|               |          |          |          |            |

|                  | de flujos | recibidos en DAG | muestreados por Nfsen | muestreado vs DAG |
|------------------|-----------|------------------|-----------------------|-------------------|
| Flujo 1 – 50Mbps | 1         | 1339287          | 133806                | 9,99%             |
| Flujo 2 – 5Mbps  | 1         | 133810           | 13494                 | 10,08%            |
| Flujo3 – 500kbps | 1         | 13201            | 1343                  | 10,17%            |
| Flujo 4 – 50kbps | 2         | 1201             | 111                   | 9,24%             |
| Flujo 5 – 5kbps  | 6         | 133              | 8                     | 6,01%             |
| Flujo 6 – 500bps | 2         | 14               | 2                     | 14,28%            |

## Prueba 10/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.21.** Prueba 10 - Muestreo Random Netflow 1/10 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 134046                         | 10,00%                       |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 13245                          | 9,89%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 1338                           | 10,13%                       |
| Flujo 4 – 50kbps | 1                  | 1201                      | 121                            | 10,07%                       |
| Flujo 5 – 5kbps  | 7                  | 133                       | 13                             | 9,77%                        |
| Flujo 6 – 500bps | 1                  | 14                        | 1                              | 7,14%                        |

## Media, desviación e intervalo de confianza de las pruebas

En esta tabla se muestran los paquetes muestreados por Netflow y se calcula la media, la desviación estándar y el intervalo de confianza total para el conjunto de las 10 pruebas realizadas:

**Tabla A7.22.** Estadísticas de las pruebas con Random Netflow 1/10

| Random Netflow (1 de cada 10) |            |            |            |            |            |            |            |            |            |             |             |                     |                              |
|-------------------------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|-------------|---------------------|------------------------------|
| Número de Flujo               | 1/10 (paq) | 2/10 (paq) | 3/10 (paq) | 4/10 (paq) | 5/10 (paq) | 6/10 (paq) | 7/10 (paq) | 8/10 (paq) | 9/10 (paq) | 10/10 (paq) | Media (paq) | Desviación estándar | Intervalo de confianza (95%) |
| 1-50M                         | 134128     | 133413     | 133827     | 133533     | 134269     | 134394     | 133863     | 134449     | 133806     | 134046      | 133972,80   | 330,04              | 204,55                       |
| 2-5M                          | 13171      | 13868      | 13504      | 13767      | 13039      | 12955      | 13414      | 12877      | 13494      | 13245       | 13333,40    | 317,41              | 196,73                       |
| 3-500k                        | 1332       | 1341       | 1296       | 1320       | 1318       | 1289       | 1358       | 1327       | 1343       | 1338        | 1326,20     | 20,21               | 12,53                        |
| 4-50k                         | 126        | 125        | 117        | 129        | 119        | 117        | 120        | 100        | 111        | 121         | 118,50      | 7,87                | 4,88                         |
| 5-5k                          | 11         | 16         | 15         | 13         | 15         | 9          | 9          | 10         | 8          | 13          | 11,90       | 2,73                | 1,69                         |
| 6-500b                        | 0          | 1          | 4          | 0          | 3          | 0          | 1          | 1          | 2          | 1           | 1,30        | 1,26                | 0,78                         |

El nivel de confianza y la amplitud del intervalo varían conjuntamente, de forma que un intervalo más amplio respecto a la media tendrá más posibilidades de acierto (mayor nivel de confianza), mientras que para un intervalo más pequeño, que ofrece una estimación más precisa, aumentan sus posibilidades de error.

En la siguiente tabla se encuentran los cómputos finales de paquetes y velocidad:

**Tabla A7.23.** Porcentaje de paquetes y velocidad con Random Netflow 1/10

| Computo (paquetes)              | Computo (Velocidad)         |
|---------------------------------|-----------------------------|
| <b>133972,80±204,55 (0,15%)</b> | <b>50,01Mbps±76,36kbps</b>  |
| <b>13333,40±196,73 (1,47%)</b>  | <b>4,97Mbps±73,44kbps</b>   |
| <b>1326,3±12,53 (0,94%)</b>     | <b>495,11kbps±4,678kbps</b> |
| <b>118,50±4,88 (4,12%)</b>      | <b>44kbps±1,82kbps</b>      |
| <b>11,90±1,69 (14,25%)</b>      | <b>4kbps±0,63kbps</b>       |
| <b>1,26±0,78 (60,49%)</b>       | <b>485,33bps±293,60bps</b>  |

Mirando los tantos por ciento podemos observar como la probabilidad de error va en aumento a medida que la velocidad del flujo baje, siendo para los cuatro primeros un resultado aceptable.

## Muestreo 1 de cada 100

### Prueba 1/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.24.** Prueba 1 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 13375                          | 0,99%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1358                           | 1,01%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 133                            | 1,09%                        |
| Flujo 4 – 50kbps | 5                  | 1201                      | 6                              | 0,49%                        |
| Flujo 5 – 5kbps  | 1                  | 133                       | 1                              | 0,75%                        |
| Flujo 6 – 500bps | 1                  | 14                        | 1                              | 7,14%                        |

## Prueba 2/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.25.** Prueba 2 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 13424                          | 1,00%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1316                           | 0,98%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 121                            | 0,91%                        |
| Flujo 4 – 50kbps | 8                  | 1201                      | 17                             | 1,41%                        |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 3/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.26.** Prueba 3 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 13381                          | 0,99%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1346                           | 1,00%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 135                            | 1,02%                        |
| Flujo 4 – 50kbps | 5                  | 1201                      | 11                             | 0,91%                        |
| Flujo 5 – 5kbps  | 2                  | 133                       | 2                              | 1,50%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 4/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.27.** Prueba 4 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 13451                          | 1,00%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1279                           | 0,95%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 136                            | 1,03%                        |
| Flujo 4 – 50kbps | 7                  | 1201                      | 9                              | 0,74%                        |
| Flujo 5 – 5kbps  | 1                  | 133                       | 1                              | 0,75%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 5/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.28.** Prueba 5 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 13409                          | 1,00%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1326                           | 0,99%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 126                            | 0,95%                        |
| Flujo 4 – 50kbps | 7                  | 1201                      | 15                             | 1,24%                        |
| Flujo 5 – 5kbps  | 1                  | 133                       | 1                              | 0,75%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 6/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.29.** Prueba 6 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo | Cantidad | Paquetes | Paquetes | Porcentaje |
|---------------|----------|----------|----------|------------|
|               |          |          |          |            |

|                  | de flujos | recibidos en DAG | muestreados por Nfsen | muestreado vs DAG |
|------------------|-----------|------------------|-----------------------|-------------------|
| Flujo 1 – 50Mbps | 1         | 1339287          | 13375                 | 0,99%             |
| Flujo 2 – 5Mbps  | 1         | 133810           | 1352                  | 1,01%             |
| Flujo3 – 500kbps | 1         | 13201            | 140                   | 1,06%             |
| Flujo 4 – 50kbps | 6         | 1201             | 9                     | 0,74%             |
| Flujo 5 – 5kbps  | 1         | 133              | 1                     | 0,75%             |
| Flujo 6 – 500bps | 0         | 14               | 0                     | 0%                |

### Prueba 7/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.30.** Prueba 7 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 13355                          | 0,99%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1362                           | 1,01%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 146                            | 1,10%                        |
| Flujo 4 – 50kbps | 5                  | 1201                      | 11                             | 0,91%                        |
| Flujo 5 – 5kbps  | 2                  | 133                       | 2                              | 1,50%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 8/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.31.** Prueba 8 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 13426                          | 1,00%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1295                           | 0,96%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 142                            | 1,07%                        |
| Flujo 4 – 50kbps | 9                  | 1201                      | 13                             | 1,08%                        |
| Flujo 5 – 5kbps  | 1                  | 133                       | 1                              | 0,75%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 9/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.32.** Prueba 9 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 13433                          | 1,00%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1293                           | 0,96%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 138                            | 1,04%                        |
| Flujo 4 – 50kbps | 7                  | 1201                      | 9                              | 0,74%                        |
| Flujo 5 – 5kbps  | 3                  | 133                       | 3                              | 2,25%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 10/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.33.** Prueba 10 - Muestreo Random Netflow 1/100 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 13444                          | 1,00%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1289                           | 0,96%                        |
| Flujo3 – 500kbps | 1                  | 13201                     | 122                            | 0,92%                        |
| Flujo 4 – 50kbps | 7                  | 1201                      | 20                             | 1,66%                        |
| Flujo 5 – 5kbps  | 2                  | 133                       | 2                              | 1,50%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Media, desviación e intervalo de confianza de las pruebas

En esta tabla se muestran los paquetes muestreados por Netflow y se calcula la media, la desviación estándar y el intervalo de confianza total para el conjunto de las 10 pruebas realizadas:

**Tabla A7.34.** Estadísticas de las pruebas Random Netflow 1/100

| Random Netflow (1 de cada 100) |            |            |            |            |            |            |            |            |            |             |             |                     |                              |
|--------------------------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|-------------|---------------------|------------------------------|
| Número de Flujo                | 1/10 (paq) | 2/10 (paq) | 3/10 (paq) | 4/10 (paq) | 5/10 (paq) | 6/10 (paq) | 7/10 (paq) | 8/10 (paq) | 9/10 (paq) | 10/10 (paq) | Media (paq) | Desviación estándar | Intervalo de confianza (95%) |
| 1-50M                          | 13375      | 13424      | 13381      | 13451      | 13409      | 13375      | 13355      | 13426      | 13433      | 13444       | 13407,30    | 31,72               | 19,66                        |
| 2-5M                           | 1358       | 1316       | 1346       | 1279       | 1326       | 1352       | 1362       | 1295       | 1293       | 1289        | 1321,60     | 29,88               | 18,52                        |
| 3-500k                         | 133        | 121        | 135        | 136        | 126        | 140        | 146        | 142        | 138        | 122         | 133,90      | 8,02                | 4,97                         |
| 4-50k                          | 6          | 17         | 11         | 9          | 15         | 9          | 11         | 13         | 9          | 20          | 12,00       | 4,05                | 2,51                         |
| 5-5k                           | 1          | 0          | 2          | 1          | 1          | 1          | 2          | 1          | 3          | 2           | 1,40        | 0,80                | 0,50                         |
| 6-500b                         | 1          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0           | 0,10        | 0,30                | 0,19                         |

Podemos observar como el intervalo de confianza para los tres primeros flujos es muy óptimo, pero para los tres siguientes el margen de error aumenta considerablemente. Por tanto aceptamos cualquier valor entre los rangos.

En la siguiente tabla se encuentran los cálculos finales de paquetes y velocidad:

**Tabla A7.35.** Porcentaje de paquetes y velocidad con Random Netflow 1/100

| Computo (paquetes)            | Computo (Velocidad)         |
|-------------------------------|-----------------------------|
| <b>13407,30±19,66 (0,14%)</b> | <b>50,05Mbps±73,39kbps</b>  |
| <b>1321,60±18,52 (1,40%)</b>  | <b>4,93Mbps±69,14kbps</b>   |
| <b>133,90±4,97 (3,71%)</b>    | <b>499,89kbps±18,55kbps</b> |
| <b>12,00±2,51 (20,91%)</b>    | <b>44,8kbps±9,37kbps</b>    |
| <b>1,40±0,50 (35,41%)</b>     | <b>5,22kbps±1,85kbps</b>    |
| <b>0,10±0,19 (185%)</b>       | <b>373,33bps±6,94bps</b>    |

Mirando los tantos por ciento podemos observar como la probabilidad de error va en aumento a medida que la velocidad del flujo baje, siendo para los tres primeros un resultado aceptable.

## Muestreo 1 de cada 1000

### Prueba 1/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.36.** Prueba 1 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1333                           | 0,09%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 141                            | 0,105%                       |
| Flujo3 – 500kbps | 8                  | 13201                     | 14                             | 0,10%                        |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 2/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.37.** Prueba 2 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1373                           | 0,10%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 107                            | 0,07%                        |
| Flujo3 – 500kbps | 4                  | 13201                     | 6                              | 0,04%                        |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1                              | 0,08%                        |
| Flujo 5 – 5kbps  | 1                  | 133                       | 1                              | 0,75%                        |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 3/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.38.** Prueba 3 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 1346                           | 0,10%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 127                            | 0,09%                        |
| Flujo3 – 500kbps | 7                  | 13201                     | 13                             | 0,09%                        |
| Flujo 4 – 50kbps | 2                  | 1201                      | 2                              | 0,16%                        |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 4/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.39.** Prueba 4 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1333                           | 0,09%                        |
| Flujo 2 – 5Mbps  | 2                  | 133810                    | 138                            | 0,10%                        |
| Flujo3 – 500kbps | 8                  | 13201                     | 16                             | 0,12%                        |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 5/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.40.** Prueba 5 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1361                           | 0,10%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 115                            | 0,08%                        |
| Flujo3 – 500kbps | 8                  | 13201                     | 11                             | 0,08%                        |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1                              | 0,08%                        |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 6/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.41.** Prueba 6 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 1322                           | 0,09%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 154                            | 0,11%                        |
| Flujo3 – 500kbps | 7                  | 13201                     | 11                             | 0,08%                        |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 7/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.42.** Prueba 7 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1330                           | 0,09%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 146                            | 0,10%                        |
| Flujo3 – 500kbps | 7                  | 13201                     | 12                             | 0,09%                        |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 8/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.43.** Prueba 8 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1358                           | 0,10%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 116                            | 0,08%                        |
| Flujo3 – 500kbps | 7                  | 13201                     | 13                             | 0,09%                        |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1                              | 0,08%                        |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 9/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.44.** Prueba 9 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339287                   | 1349                           | 0,10%                        |
| Flujo 2 – 5Mbps  | 2                  | 133810                    | 129                            | 0,09%                        |
| Flujo3 – 500kbps | 4                  | 13201                     | 11                             | 0,083%                       |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 10/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.45.** Prueba 10 - Muestreo Random Netflow 1/1000 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 1311                           | 0,09%                        |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 159                            | 0,11%                        |
| Flujo3 – 500kbps | 8                  | 13201                     | 16                             | 0,12%                        |
| Flujo 4 – 50kbps | 1                  | 1201                      | 1                              | 0,08%                        |
| Flujo 5 – 5kbps  | 0                  | 133                       | 2                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Media, desviación e intervalo de confianza de las pruebas

En esta tabla se muestran los paquetes muestreados por Netflow y se calcula la media, la desviación estándar y el intervalo de confianza total para el conjunto de las 10 pruebas realizadas:

**Tabla A7.46.** Estadísticas para las pruebas Random Netflow 1/1000

| Número de Flujo | Random Netflow (1 de cada 1000) |            |            |            |            |            |            |            |            |             | Desviación estándar | Intervalo de confianza (95%) |       |
|-----------------|---------------------------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|---------------------|------------------------------|-------|
|                 | 1/10 (paq)                      | 2/10 (paq) | 3/10 (paq) | 4/10 (paq) | 5/10 (paq) | 6/10 (paq) | 7/10 (paq) | 8/10 (paq) | 9/10 (paq) | 10/10 (paq) |                     |                              |       |
| 1-50M           | 1333                            | 1373       | 1346       | 1333       | 1361       | 1322       | 1330       | 1358       | 1349       | 1311        | 1341,60             | 18,19                        | 11,27 |
| 2-5M            | 141                             | 107        | 127        | 138        | 115        | 154        | 146        | 116        | 129        | 159         | 133,20              | 16,48                        | 10,21 |
| 3-500k          | 14                              | 6          | 13         | 16         | 11         | 11         | 12         | 13         | 11         | 16          | 12,30               | 2,76                         | 1,71  |
| 4-50k           | 0                               | 1          | 2          | 0          | 1          | 0          | 0          | 1          | 0          | 1           | 0,60                | 0,66                         | 0,41  |
| 5-5k            | 0                               | 1          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 2           | 0,30                | 0,64                         | 0,40  |
| 6-500b          | 0                               | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0           | 0,00                | 0,00                         | 0     |

Podemos observar como el intervalo de confianza para los dos primeros flujos es muy óptimo, pero para los cuatro siguientes el margen de error aumenta considerablemente. Por tanto aceptamos cualquier valor entre los rangos.

En la siguiente tabla se encuentran los cálculos finales de paquetes y velocidad:

**Tabla A7.47.** Porcentajes de paquetes y velocidad con Random Netflow 1/1000

| Computo (paquetes)           | Computo (Velocidad)         |
|------------------------------|-----------------------------|
| <b>1341,60±11,27 (0,84%)</b> | <b>50,08Mbps±420,87kbps</b> |
| <b>133,20±10,21 (7,66%)</b>  | <b>4,97Mbps±381,30kbps</b>  |
| <b>12,30±1,71 (13,90%)</b>   | <b>459,20kbps±63,82kbps</b> |
| <b>0,60±0,41 (68,52%)</b>    | <b>22,40kbps±15,34kbps</b>  |
| <b>0,30±0,40 (132,28%)</b>   | <b>11,20kbps±14,81kbps</b>  |
| <b>0±0 (0%)</b>              | <b>0bps±0bps</b>            |

Mirando los tantos por ciento podemos observar como la probabilidad de error va en aumento a medida que la velocidad del flujo baja, siendo para los dos primeros un resultado aceptable.

## Muestreo 1 de cada 65535

### Prueba 1/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.48.** Prueba 1 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 3                  | 1339286                   | 20                             | 0,0014%                      |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 2                              | 0,0014%                      |
| Flujo3 – 500kbps | 1                  | 13201                     | 1                              | 0,0075%                      |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 2/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.49.** Prueba 2 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 4                  | 1339286                   | 19                             | 0,0014%                      |
| Flujo 2 – 5Mbps  | 2                  | 133810                    | 3                              | 0,0022%                      |
| Flujo3 – 500kbps | 1                  | 13201                     | 1                              | 0,0022%                      |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 3/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.50.** Prueba 3 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 3                  | 1339287                   | 21                             | 0,0015%                      |
| Flujo 2 – 5Mbps  | 2                  | 133810                    | 2                              | 0,0014%                      |
| Flujo3 – 500kbps | 0                  | 13201                     | 0                              | 0%                           |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 4/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.51.** Prueba 4 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 3                  | 1339286                   | 20                             | 0,0014%                      |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1                              | 0,0007%                      |
| Flujo3 – 500kbps | 1                  | 13201                     | 1                              | 0,0075%                      |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

**Prueba 5/10**

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.52.** Prueba 5 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 3                  | 1339286                   | 21                             | 0,0015%                      |
| Flujo 2 – 5Mbps  | 2                  | 133810                    | 2                              | 0,0014%                      |
| Flujo3 – 500kbps | 0                  | 13201                     | 0                              | 0%                           |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

**Prueba 6/10**

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.53.** Prueba 6 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 3                  | 1339287                   | 20                             | 0,0014                       |

|                  |   |        |   |         |
|------------------|---|--------|---|---------|
| Flujo 2 – 5Mbps  | 2 | 133810 | 3 | 0,0022% |
| Flujo3 – 500kbps | 0 | 13201  | 0 | 0%      |
| Flujo 4 – 50kbps | 0 | 1201   | 0 | 0%      |
| Flujo 5 – 5kbps  | 0 | 133    | 0 | 0%      |
| Flujo 6 – 500bps | 0 | 14     | 0 | 0%      |

### Prueba 7/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.54.** Prueba 7 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por NfSEN | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 1                  | 1339286                   | 22                             | 0,0016%                      |
| Flujo 2 – 5Mbps  | 0                  | 133810                    | 0                              | 0%                           |
| Flujo3 – 500kbps | 0                  | 13201                     | 0                              | 0%                           |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 8/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.55.** Prueba 8 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por NfSEN | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 5                  | 1339286                   | 18                             | 0,0013%                      |
| Flujo 2 – 5Mbps  | 4                  | 133810                    | 5                              | 0,0037%                      |
| Flujo3 – 500kbps | 0                  | 13201                     | 0                              | 0%                           |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

### Prueba 9/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.56.** Prueba 9 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 3                  | 1339287                   | 21                             | 0,0015%                      |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1                              | 0,0007%                      |
| Flujo3 – 500kbps | 1                  | 13201                     | 1                              | 0,0075%                      |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Prueba 10/10

En esta prueba se ha comprobado si el conteo de Netflow es correcto con respecto a los datos recogidos por la DAG.

**Tabla A7.57.** Prueba 10 - Muestreo Random Netflow 1/65535 vs DAG

| Tipo de flujo    | Cantidad de flujos | Paquetes recibidos en DAG | Paquetes muestreados por Nfsen | Porcentaje muestreado vs DAG |
|------------------|--------------------|---------------------------|--------------------------------|------------------------------|
| Flujo 1 – 50Mbps | 3                  | 1339286                   | 21                             | 0,0015%                      |
| Flujo 2 – 5Mbps  | 1                  | 133810                    | 1                              | 0,0007%                      |
| Flujo3 – 500kbps | 1                  | 13201                     | 1                              | 0,0075%                      |
| Flujo 4 – 50kbps | 0                  | 1201                      | 0                              | 0%                           |
| Flujo 5 – 5kbps  | 0                  | 133                       | 0                              | 0%                           |
| Flujo 6 – 500bps | 0                  | 14                        | 0                              | 0%                           |

## Media, desviación e intervalo de confianza de las pruebas

En esta tabla se muestran los paquetes muestreados por Netflow y se calcula la media, la desviación estándar y el intervalo de confianza total para el conjunto de las 10 pruebas realizadas:

**Tabla A7.58.** Estadísticas para las pruebas Random Netflow 1/65535

| Random Netflow (1 de cada 65535) |            |            |            |            |            |            |            |            |            |             |             |                     |                              |
|----------------------------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|-------------|---------------------|------------------------------|
| Número de Flujo                  | 1/10 (paq) | 2/10 (paq) | 3/10 (paq) | 4/10 (paq) | 5/10 (paq) | 6/10 (paq) | 7/10 (paq) | 8/10 (paq) | 9/10 (paq) | 10/10 (paq) | Media (paq) | Desviación estándar | Intervalo de confianza (95%) |
| 1-50M                            | 20         | 19         | 21         | 20         | 21         | 20         | 22         | 18         | 21         | 21          | 20,30       | 1,10                | 0,68                         |
| 2-5M                             | 2          | 3          | 2          | 1          | 2          | 3          | 0          | 5          | 1          | 1           | 2,00        | 1,34                | 0,83                         |
| 3-500k                           | 1          | 1          | 0          | 1          | 0          | 0          | 0          | 0          | 1          | 1           | 0,50        | 0,50                | 0,31                         |
| 4-50k                            | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0           | 0,00        | 0,00                | 0                            |
| 5-5k                             | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0           | 0,00        | 0,00                | 0                            |
| 6-500b                           | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0           | 0,00        | 0,00                | 0                            |

Podemos observar como el intervalo de confianza para el primer flujo es óptimo, pero para los cuatro siguientes el margen de error aumenta considerablemente. Por tanto aceptamos cualquier valor entre los rangos.

En la siguiente tabla se encuentran los cómputos finales de paquetes y velocidad:

**Tabla A7.59.** Porcentaje de paquetes y velocidad con Random Netflow 1/65535

| Computo (paquetes)        | Computo (Velocidad)          |
|---------------------------|------------------------------|
| <b>20,30±0,68 (3,35%)</b> | <b>49,66Mbps±25,45kbps</b>   |
| <b>2,00±0,83 (41,57%)</b> | <b>4,89Mbps±31,04kbps</b>    |
| <b>0,50±0,31 (61,97%)</b> | <b>1223,32kbps±11,56kbps</b> |
| <b>0±0 (0%)</b>           | <b>0kbps±0kbps</b>           |
| <b>0±0 (0%)</b>           | <b>0kbps±0kbps</b>           |
| <b>0±0 (0%)</b>           | <b>0bps±0bps</b>             |

Mirando los tantos por ciento podemos observar como la probabilidad de error va en aumento a medida que la velocidad del flujo baja, siendo sólo para el primero un resultado aceptable. Cabe destacar desigualdad y falseo de resultados para los siguientes flujos, en el tercero por ejemplo: se estima un cómputo de velocidad alcanzada de 1223,32kbps, cuando en la generación se inyectan 500kbps, este es el resultado de un mal muestreo.

#### Comparativa final con intervalos de confianza

A continuación veremos los intervalos de confianza de los diferentes flujos después de realizar 10 pruebas para cada uno de ellos y para cada tipo de muestreo:

**Tabla A7.60.** Comparativa final con intervalos de confianza

| Número de flujo | Intervalos de confianza (95%)   |                               |                              |                           |
|-----------------|---------------------------------|-------------------------------|------------------------------|---------------------------|
|                 | Tipo de muestreo                |                               |                              |                           |
|                 | 1 de 10 (paquetes)              | 1 de 100 (paquetes)           | 1 de 1000 (paquetes)         | 1 de 65535 (paquetes)     |
| 1-50M           | <b>133972,80±204,55 (0,15%)</b> | <b>13407,30±19,66 (0,14%)</b> | <b>1341,60±11,27 (0,84%)</b> | <b>20,30±0,68 (3,35%)</b> |
| 2-5M            | <b>13333,40±196,73 (1,47%)</b>  | <b>1321,60±18,52 (1,40%)</b>  | <b>133,20±10,21 (7,66%)</b>  | <b>2,00±0,83 (41,57%)</b> |
| 3-500k          | <b>1326,3±12,53 (0,94%)</b>     | <b>133,90±4,97 (3,71%)</b>    | <b>12,30±1,71 (13,90%)</b>   | <b>0,50±0,31 (61,97%)</b> |
| 4-50k           | <b>118,50±4,88 (4,12%)</b>      | <b>12,00±2,51 (20,91%)</b>    | <b>0,60±0,41 (68,52%)</b>    | <b>0±0 (0%)</b>           |

|        |                            |                           |                            |                 |
|--------|----------------------------|---------------------------|----------------------------|-----------------|
| 5-5k   | <b>11,90±1,69 (14,25%)</b> | <b>1,40±0,50 (35,41%)</b> | <b>0,30±0,40 (132,28%)</b> | <b>0±0 (0%)</b> |
| 6-500b | <b>1,26±0,78 (60,49%)</b>  | <b>0,10±0,19 (185%)</b>   | <b>0±0 (0%)</b>            | <b>0±0 (0%)</b> |

Podemos observar como para los últimos tres flujos, los intervalos de confianza son bajos con respecto a su media lo que hace obtener errores máximos y mínimos muy pequeños, es decir un aumento de la posibilidad de error.

De esta manera podemos afirmar que aceptamos como válido con un 95% de acierto cualquier valor que esté entre los rangos del intervalo. Aunque por otro lado los tantos por ciento superiores a 5 no son de plena confianza para el muestreo, siendo estos los de menor velocidad.

En este caso se muestra la comparativa relacionada con la velocidad:

**Tabla A7.61.** Porcentajes finales de paquetes y velocidad

| Número de flujo | Intervalos de confianza (95%) |                             |                             |                              |
|-----------------|-------------------------------|-----------------------------|-----------------------------|------------------------------|
|                 | Tipo de muestreo              |                             |                             |                              |
|                 | 1 de 10                       | 1 de 100                    | 1 de 1000                   | 1 de 65535                   |
| 1-50M           | <b>50,01Mbps±76,36kbps</b>    | <b>50,05Mbps±73,39kbps</b>  | <b>50,08Mbps±420,87kbps</b> | <b>49,66Mbps±25,45kbps</b>   |
| 2-5M            | <b>4,97Mbps±73,44kbps</b>     | <b>4,93Mbps±69,14kbps</b>   | <b>4,97Mbps±381,30kbps</b>  | <b>4,89Mbps±31,04kbps</b>    |
| 3-500k          | <b>495,11kbps±4,678kbps</b>   | <b>499,89kbps±18,55kbps</b> | <b>459,20kbps±63,82kbps</b> | <b>1223,32kbps±11,56kbps</b> |
| 4-50k           | <b>44kbps±1,82kbps</b>        | <b>44,8kbps±9,37kbps</b>    | <b>22,40kbps±15,34kbps</b>  | <b>0kbps±0kbps</b>           |
| 5-5k            | <b>4kbps±0,63kbps</b>         | <b>5,22kbps±1,85kbps</b>    | <b>11,20kbps±14,81kbps</b>  | <b>0kbps±0kbps</b>           |
| 6-500b          | <b>485,33bps±293,60bps</b>    | <b>373,33bps±6,94bps</b>    | <b>0bps±0bps</b>            | <b>0bps±0bps</b>             |

Aquí podemos apreciar como el intervalo de confianza respecto a la velocidad va aumentando a medida que disminuye la velocidad. Por lo tanto se requiere llegar a un compromiso de elección de probabilidad.

## VIII. Creación de trazas básicas

### Creación de trazas con longitud fija

El script básico de generación de paquetes es el siguiente:

```
//INICIO SCRIPT

packets {

packet eth_II first_packet{

    src_addr 00:30:48:76:1d:5b;
    dst_addr 00:07:b3:5f:9c:e0;
    protocol 0x800;
    payload "
        45 00 05 94 00 00 40 00 3f 11 14 af 93 53 76 f2
        0a 00 0d 65 15 7d 15 7c 05 78 9a 40" + dummy(1400);
    fcs_size 4;
}

}

traffic {

group my_traffic_group {
    send first_packet 10;
}
}

//FIN SCRIPT
```

Al crear un paquete, la capa de transporte agrega la cabecera UDP (con puerto origen y destino), la capa de red debe añadir la cabecera IP (con dos direcciones IP entre otros campos) y la capa de enlace debe añadir los diferentes campos de la trama MAC (con dos direcciones MAC, longitud/tipo y FCS).

En nuestro caso, a la hora de programarlo lo hemos hecho en base al script de ejemplo que proporciona el manual de *daggen*. A continuación se explican los diferentes campos programados en el script, y su resultado en formato Wireshark:

*Eth\_II* es el tipo de paquete que se genera y *first\_packet* es un nombre arbitrario que hay que ponerle al paquete.

*Src\_addr* y *dst\_addr* son las direcciones MAC origen (DAG) y destino respectivamente (Fa0/0 Cisco).

*Protocol* es el valor del campo Longitud/Tipo de la trama MAC, el valor es 0x800 que significa protocolo IP.

| No.  | Time  | Source           | Destination | Protocol | Info  | Delta Time |
|--|---|------------------|-------------|----------|---|------------|
| 1  | 0.0000000000                                    | 147.83.118.242   | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.0000000000 |            |
| 2  | 0.000000336                                     | 147.83.118.242   | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000336  |            |
| 3  | 0.000000673                                     | 147.83.118.242   | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000337  |            |
| Frame 1: 1446 bytes on wire (11568 bits), 1446 bytes captured (11568 bits)                       |   |                  |             |          |   |            |
| Extensible Record Format   |   |                  |             |          |   |            |
| Ethernet II, Src: Supermic_76:1d:5b (00:30:48:76:1d:5b), Dst: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0) |   |                  |             |          |   |            |
| Destination: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0)  |   |                  |             |          |   |            |
| Source: Supermic_76:1d:5b (00:30:48:76:1d:5b)  |   |                  |             |          |   |            |
| Type: IP (0x0800)  |   |                  |             |          |   |            |
| Frame check sequence: 0xcf025a5d [correct]   |   |                  |             |          |   |            |
| Internet Protocol, Src: 147.83.118.242 (147.83.118.242), Dst: 10.0.13.101 (10.0.13.101)          |   |                  |             |          |   |            |
| User Datagram Protocol, Src Port: fcp-addr-srvr2 (5501), Dst Port: fcp-addr-srvr1 (5500)         |   |                  |             |          |   |            |
| Data (1392 bytes)  |   |                  |             |          |   |            |
| 0000   | 00 07 b3 5f 9c e0 00 30 48 76 1d 5b 08 00 45 00 | .....0 Hv. [..E. |             |          |   |            |
| 0010   | 05 94 00 00 40 00 3f 11 14 af 93 53 76 f2 0a 00 | ...@.? ...5V...  |             |          |   |            |
| 0020   | 0d 65 15 7d 15 7c 05 78 9a 40 18 59 40 76 d4 5b | .e.]. .x @.v. [  |             |          |   |            |
| 0030   | ec 30 b2 1a 6e 1b bb 7d 0a a2 67 ef 91 d5 aa a7 | .0..n.} ..g....  |             |          |   |            |

Fig. A8.1 Capa 2 de un paquete generado

Seguidamente se vuelca en el script el *payload* de la trama en formato hexadecimal, donde cada valor forma parte de los campos de la cabecera IP, configurados con los valores por defecto más las IP's de origen (DAG) y destino (servgenmonII) correspondientes.

| No.  | Time  | Source           | Destination | Protocol | Info  | Delta Time |
|--|---|------------------|-------------|----------|---|------------|
| 1  | 0.0000000000                                    | 147.83.118.242   | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.0000000000 |            |
| 2  | 0.000000336                                     | 147.83.118.242   | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000336  |            |
| 3  | 0.000000673                                     | 147.83.118.242   | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000337  |            |
| Frame 1: 1446 bytes on wire (11568 bits), 1446 bytes captured (11568 bits)                       |   |                  |             |          |   |            |
| Extensible Record Format   |   |                  |             |          |   |            |
| Ethernet II, Src: Supermic_76:1d:5b (00:30:48:76:1d:5b), Dst: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0) |   |                  |             |          |   |            |
| Destination: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0)  |   |                  |             |          |   |            |
| Source: Supermic_76:1d:5b (00:30:48:76:1d:5b)  |   |                  |             |          |   |            |
| Type: IP (0x0800)  |   |                  |             |          |   |            |
| Frame check sequence: 0xcf025a5d [correct]   |   |                  |             |          |   |            |
| Internet Protocol, Src: 147.83.118.242 (147.83.118.242), Dst: 10.0.13.101 (10.0.13.101)          |   |                  |             |          |   |            |
| version: 4   |   |                  |             |          |   |            |
| Header length: 20 bytes  |   |                  |             |          |   |            |
| Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)                              |   |                  |             |          |   |            |
| Total Length: 1428   |   |                  |             |          |   |            |
| Identification: 0x0000 (0)   |   |                  |             |          |   |            |
| Flags: 0x02 (don't Fragment)   |   |                  |             |          |   |            |
| Fragment offset: 0   |   |                  |             |          |   |            |
| Time to live: 63   |   |                  |             |          |   |            |
| Protocol: UDP (17)   |   |                  |             |          |   |            |
| Header checksum: 0x14af [correct]  |   |                  |             |          |   |            |
| Source: 147.83.118.242 (147.83.118.242)  |   |                  |             |          |   |            |
| Destination: 10.0.13.101 (10.0.13.101)   |   |                  |             |          |   |            |
| User Datagram Protocol, Src Port: fcp-addr-srvr2 (5501), Dst Port: fcp-addr-srvr1 (5500)         |   |                  |             |          |   |            |
| Data (1392 bytes)  |   |                  |             |          |   |            |
| 0000   | 00 07 b3 5f 9c e0 00 30 48 76 1d 5b 08 00 45 00 | .....0 Hv. [..E. |             |          |   |            |
| 0010   | 05 94 00 00 40 00 3f 11 14 af 93 53 76 f2 0a 00 | ...@.? ...5V...  |             |          |   |            |
| 0020   | 0d 65 15 7d 15 7c 05 78 9a 40 18 59 40 76 d4 5b | .e.]. .x @.v. [  |             |          |   |            |
| 0030   | ec 30 b2 1a 6e 1b bb 7d 0a a2 67 ef 91 d5 aa a7 | .0..n.} ..g....  |             |          |   |            |

Fig. A8.2 Capa 3 de un paquete generado

Como parte íntegra del payload está la cabecera UDP: con los puertos configurados anteriormente para MGGEN y aprovechados para esta prueba, el campo de longitud de UDP y su checksum.

| No.  | Time                 | Source               | Destination | Protocol | Info  | Delta Time |
|--|----------------------|----------------------|-------------|----------|---|------------|
| 1  | 0.0000000000         | 147.83.118.242       | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.0000000000 |            |
| 2  | 0.000000336          | 147.83.118.242       | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000336  |            |
| 3  | 0.000000672          | 147.83.118.242       | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000337  |            |
| Frame 1: 1446 bytes on wire (11568 bits), 1446 bytes captured (11568 bits)                       |                      |                      |             |          |   |            |
| Extensible Record Format   |                      |                      |             |          |   |            |
| Ethernet II, Src: Supermic_76:1d:5b (00:30:48:76:1d:5b), Dst: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0) |                      |                      |             |          |   |            |
| Internet Protocol, Src: 147.83.118.242 (147.83.118.242), Dst: 10.0.13.101 (10.0.13.101)          |                      |                      |             |          |   |            |
| User Datagram Protocol, Src Port: fcp-addr-srvr2 (5501), Dst Port: fcp-addr-srvr1 (5500)         |                      |                      |             |          |   |            |
| Source port: fcp-addr-srvr2 (5501)   |                      |                      |             |          |   |            |
| Destination port: fcp-addr-srvr1 (5500)  |                      |                      |             |          |   |            |
| Length: 1400   |                      |                      |             |          |   |            |
| Checksum: 0x9a40 [validation disabled]   |                      |                      |             |          |   |            |
| Data (1392 bytes)  |                      |                      |             |          |   |            |
| 0000   | 00:00:07.b3:5f:9c:e0 | 00:00:30:48:76:1d:5b | 00:00:45:00 |          | ..._.Hv.[..E.   |            |
| 0010   | 05:00:00:40:00:3f:11 | 14:af:93:53:76:f2    | 00:00:45:00 |          | ...@.?...Sv....   |            |
| 0020   | 0d:65:15:7d:15:7c    | 05:78:9a:40:18:59    | 40:76:d4:5b |          | e}].x}@.v.[   |            |
| 0030   | ec:30:b2:1a:6e:1b    | bb:7d:0a:a2:67:ef    | 91:d5:aa:a7 |          | .0..n..}...g....  |            |

**Fig. A8.3 Capa 4 de un paquete generado**

Finalmente se complementa la trama con un relleno (*dummy*) de 1392 bytes:

| No.  | Time              | Source                  | Destination | Protocol | Info  | Delta Time |
|--|-------------------|-------------------------|-------------|----------|---|------------|
| 1  | 0.0000000000      | 147.83.118.242          | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.0000000000 |            |
| 2  | 0.000000336       | 147.83.118.242          | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000336  |            |
| 3  | 0.000000672       | 147.83.118.242          | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000337  |            |
| [Ethernet Header]  |                   |                         |             |          |   |            |
| 0000 0000 = offset: 0  |                   |                         |             |          |   |            |
| 0000 0000 = reserved: 0  |                   |                         |             |          |   |            |
| [Ethernet II, Src: Supermic_76:1d:5b (00:30:48:76:1d:5b), Dst: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0)] |                   |                         |             |          |   |            |
| Destination: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0)  |                   |                         |             |          |   |            |
| Source: Supermic_76:1d:5b (00:30:48:76:1d:5b)  |                   |                         |             |          |   |            |
| Type: IP (0x0800)  |                   |                         |             |          |   |            |
| Frame check sequence: 0xcf025a5d [correct]   |                   |                         |             |          |   |            |
| [Internet Protocol, Src: 147.83.118.242 (147.83.118.242), Dst: 10.0.13.101 (10.0.13.101)]          |                   |                         |             |          |   |            |
| [User Datagram Protocol, Src Port: fcp-addr-srvr2 (5501), Dst Port: fcp-addr-srvr1 (5500)]         |                   |                         |             |          |   |            |
| [Data (1392 bytes)]  |                   |                         |             |          |   |            |
| Data: 18594076d45bec30b21a6e1bbb7d0aa267ef91d5aaa75a43...  |                   |                         |             |          |   |            |
| [Length: 1392]   |                   |                         |             |          |   |            |
| 0500   | f5:14:23:d8:5c:51 | fb:87:24:bf:df:86       | b3:78:82:6e |          | ,#.\Q..\$.x.n   |            |
| 0510   | 69:3f:55:b6:df:39 | 1e:38:98:65:85:6f       | 87:63:b1:7c |          | i?u..9.8.e.o.c.   |            |
| 0520   | 78:d5:54:d4:26:4f | 5b:4a:0e:3a:d0:c2:b2:53 | 30:1c       |          | x.T.40[...50.   |            |
| 0530   | 92:85:d2:71:bf:f0 | a0:57:55:2e:c6:dc       | 92:78:58:0a |          | ...q..W.U...XX.   |            |
| 0540   | 4d:ac:00:73:fb:39 | bd:0a:73:8e:cc:26       | e1:fc:42:73 |          | M..S..9..5..&.BS  |            |
| 0550   | 81:14:e4:40:04:8e | 98:5a:bc:58:36:4e       | d6:8f:58:23 |          | :@.Z.^6N.X#   |            |
| 0560   | 9b:36:96:37:6f:54 | 41:e3:e2:0d:09:43       | 09:4b:36:8a |          | ;6.70TA.....K6.   |            |
| 0570   | 5f:1a:cb:83:a8:63 | b5:c1:14:b3:98:83       | 0e:bb:be    |          | ...C.e.....   |            |
| 0580   | 42:52:f5:b2:a6:36 | 95:88:43:9e:4b:c4       | e9:81:d7:48 |          | BR...6..C.KL...H  |            |
| 0590   | 9b:a2:44:05:69:a9 | c6:5d:5c:5e:e0:68       | 1a:9e:ab    |          | ...D.i..]^.h...   |            |
| 05a0   | 6c:94:cf:02:5a:5d |                         |             |          | 1..Z.]  |            |

**Fig. A8.4 Carga del paquete**

Aunque Wireshark marca 1392 bytes los últimos 8 también forman parte del payload, ya que en el script le hemos dicho 1400 bytes.

Para finalizar correctamente la trama, hay que añadir un campo muy importante para que los routers y switches no descarten el paquete, el FCS (*fcs\_size*), con valor de 4 bytes y calculado por la DAG mientras se genera el archivo ERF:

| No. | Time        | Source         | Destination | Protocol | Info   | Delta Time |
|-----|-------------|----------------|-------------|----------|--|------------|
| 1   | 0.000000000 | 147.83.118.242 | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000000   |            |
| 2   | 0.000000336 | 147.83.118.242 | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000336   |            |
| 3   | 0.000000337 | 147.83.118.242 | 10.0.13.101 | UDP      | Source port: fcp-addr-srvr2 Destination port 0.000000337   |            |
|     |             |                |             |          | Loss counter: 0<br>wire length: 1446   |            |
|     |             |                |             |          | [Ethernet Header]<br>0000 0000 = offset: 0<br>0000 0000 = reserved: 0  |            |
|     |             |                |             |          | [Ethernet II, Src: Supermic_76:1d:5b (00:30:48:76:1d:5b), Dst: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0)]<br>Destination: Cisco_5f:9c:e0 (00:07:b3:5f:9c:e0)<br>Source: Supermic_76:1d:5b (00:30:48:76:1d:5b)<br>Type: IP (0x0800)  |            |
|     |             |                |             |          | Frame check sequence: 0xcf025a5d [correct]   |            |
|     |             |                |             |          | [Internet Protocol, Src: 147.83.118.242 (147.83.118.242), Dst: 10.0.13.101 (10.0.13.101)]  |            |
|     |             |                |             |          | [User Datagram Protocol, Src Port: fcp-addr-srvr2 (5501), Dst Port: fcp-addr-srvr1 (5500)]   |            |
|     |             |                |             |          | [Data (1392 bytes)]  |            |
|     |             |                |             |          | 0500 f5 14 23 d8 5c 51 fb 87 24 bf df 86 b3 78 82 6e ..#\.\Q.. \$....x.n<br>0510 69 3f 55 b6 df 39 1e 38 98 65 85 6f 87 63 b1 7c i?U..9.8 .e.o.c. <br>0520 78 d5 54 d4 26 4f 5b 4a 0e 3a d0 c2 b2 53 30 1c x.T.8o[J :....50.<br>0530 92 85 d2 71 bf f0 a9 57 55 2e c6 dc 92 78 58 0a ...q..W U....xx.<br>0540 4d ac de 73 fb 39 bd 0a 73 8e cc 26 e1 fc 42 73 M..s.9.. s.,&,Bs<br>0550 81 14 e4 40 04 8a 98 5a bc 5e 36 4e d6 8f 58 23 ;@...Z ^6N..X#<br>0560 3b 36 96 37 6f 54 41 e3 e2 0d 09 c3 09 4b 36 8a ;6.70TA.....K6.<br>0570 5f 1a cb 63 a8 63 bd 65 c1 f4 b3 98 83 0c bb be -.C.C.e .....<br>0580 42 52 f5 b2 a6 36 95 88 43 9e 4b 4c e9 81 d7 48 BR..6.. C.KL...H<br>0590 9b a2 ab 44 05 69 a9 c6 5d 5c 5e e0 68 1a 9e ab i..D.1.. ]\^A.h...<br>05a0 6c 94 cf 02 5a 5d 1..:.. |            |

Fig. A8.5 FCS del paquete generado

Ahora ya tenemos el paquete formado manualmente, con lo que hace falta especificar un patrón de tráfico (*traffic*) que queramos con la declaración *group*. Seguidamente con la función *send* escribimos en el archivo ERF el paquete creado tantas veces como se necesite, en este caso diez.

Como ya hemos comentado anteriormente, vamos a usar *daggen* para crear el archivo ERF, ya que la tarjeta DAG utiliza ese formato para transmitir datos:

```
servgenmonI:/home#daggen -f script_xavi_dummy.gen -o
script_xavi_dummy.erf -x my_traffic_group
```

Donde: *-f* es el script a utilizar, *-o* es el archivo ERF de salida y *-x* el tráfico que queremos utilizar de los declarado dentro del script.

Luego comprobamos con un *ls*, si se ha creado correctamente.

Una vez creado el archivo ERF, debemos asegurarnos que está alineado a 64 bits para que GenCBR lo puedan transmitir. Para ello, utilizamos dagbits:

```
servgenmonI:/home#dagbits -vvv align64 -f script_dummy.erf
dagbits: verbose: Reading input from script_xavi_dummy.erf
dagbits: verbose: feof true
dagbits: verbose: record size 0 < dag_record_size 16
-----
```

```
Records processed: 10 Bytes processed: 14640
align64: total failures 0
```

Alineado el archivo, ya es posible transmitirlo con GenCBR, el comando es el siguiente:

```
servgenmonI:/home#/GenCBR -d /dev/dag0 -f script_xavi_dummy.erf -r 1
-v -t 10 -T 100000000
caching file... done.
-----
```

```
total bytes sent to dag card: 124521984 (118.75 MiB)
approx. average speed: 99.51 Mbps (12.44 MiB/s)
```

Donde:  $-f$  es el archivo ERF a utilizar,  $-r$  es el método de la API,  $-v$  verbose para representar los datos en pantalla,  $-t$  la duración de la prueba y  $-T$  la velocidad de transmisión.

## Creación de trazas con longitud variable: Distribuciones

El potente software de la DAG nos ofrece la posibilidad de crear paquetes manualmente, pero su utilidad *daggen*, ofrece otras estructuras para la formación de paquetes.

En este caso vamos a generar 1000000 paquetes con tres tipos de distribuciones: uniforme y normal con dos tipos de desviación estándar, para posteriormente ver el histograma de los paquetes creados y verificar su precisión.

### Generación con distribución uniforme

El primer caso de uniformidad consta en crear paquetes aleatoriamente dentro de un rango previamente definido. Así que realizaremos un script muy parecido al anterior pero con alguna modificación:

```
//INICIO SCRIPT

packets {

    packet eth_II first_packet{
        src_addr 00:30:48:76:1d:5b;
        dst_addr 00:07:b3:5f:9c:e0;
        protocol 0x800;
        payload "
        45 00 05 94 00 00 40 00 3f 11 14 af 93 53 76 f2
        0a 00 0d 65 15 7d 15 7c 05 78 9a 40"
        + dummy(uniform(46,1400));
        fcs_size 4;
    }

}

traffic {

    group my_traffic_group {
        send first_packet 1000000;
    }
}

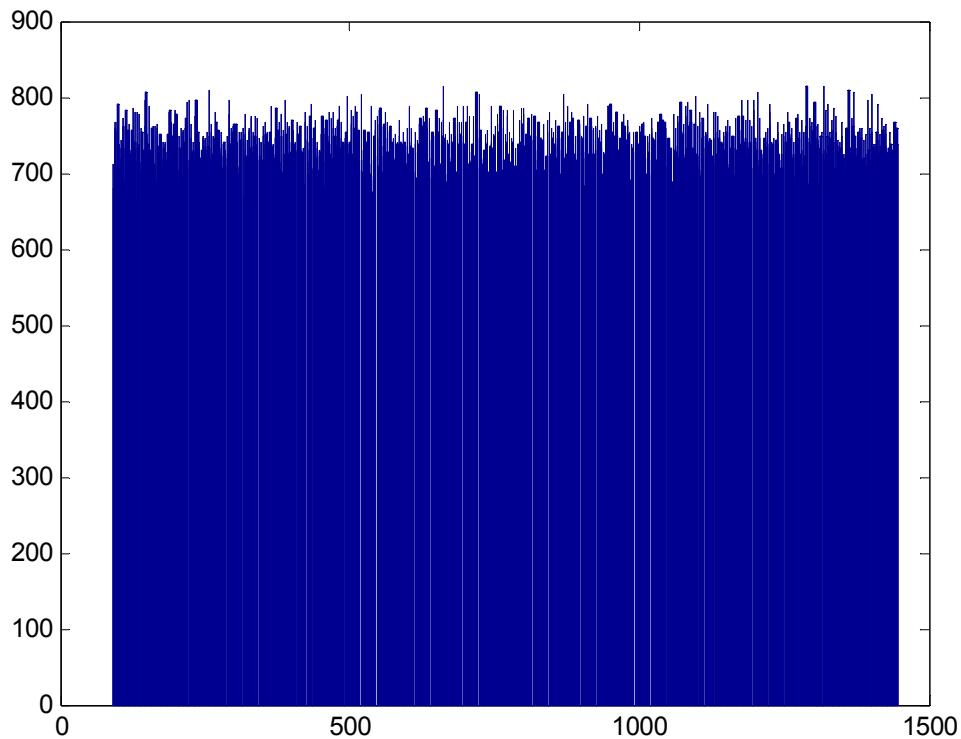
//FIN SCRIPT
```

Como se puede observar, se ha cambiado en contenido del *dummy* por la sintaxis *uniform(46,1400)* que hace que el programa escoja valores aleatoriamente entre 46 y 1400 bytes y así genere paquetes con diferente longitud.

El proceso de generación del archivo ERF es exactamente igual que el anterior caso, por lo que omitiremos la explicación.

Una vez generado exportaremos desde Wireshark a formato CSV para poder coger sólo la columna Length y realizar un histograma con Matlab.

A continuación veremos el histograma después de haber creado el archivo ERF:



**Fig. A8.6** Histograma con distribución uniforme de tamaños de paquete

Nos encontramos con un histograma muy acorde con lo que se esperaba, en donde los paquetes tienen todo tipo de longitudes y la cantidad sobre cada una de estas es muy similar.

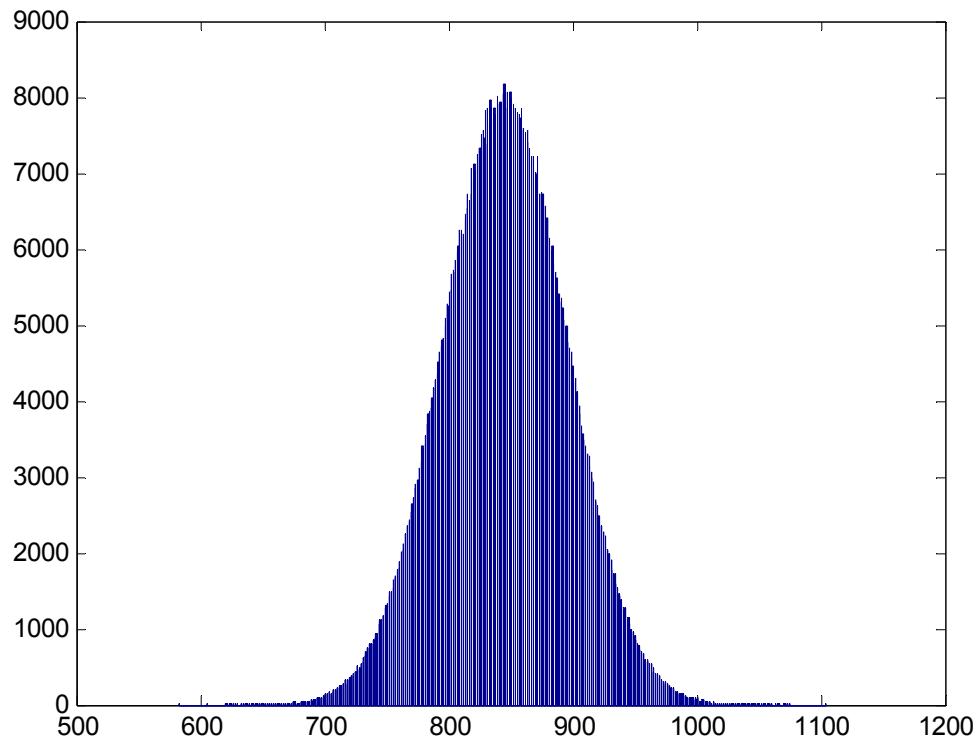
### Generación con distribución normal

En el segundo y tercer caso utilizamos una distribución normal, en la que crearemos paquetes en el rango entre una media y una desviación estándar para cada caso.

La idea es la misma que en el caso uniforme, sólo debemos cambiar el interior del *dummy* con la sintaxis *normal(x,y)*, donde *x* es la media y la *y* la desviación.

Realizaremos dos casos, uno con media 800 y distribución 50 y otro con media 800 y distribución 200.

El histograma para *normal(800,50)* es de la siguiente manera:

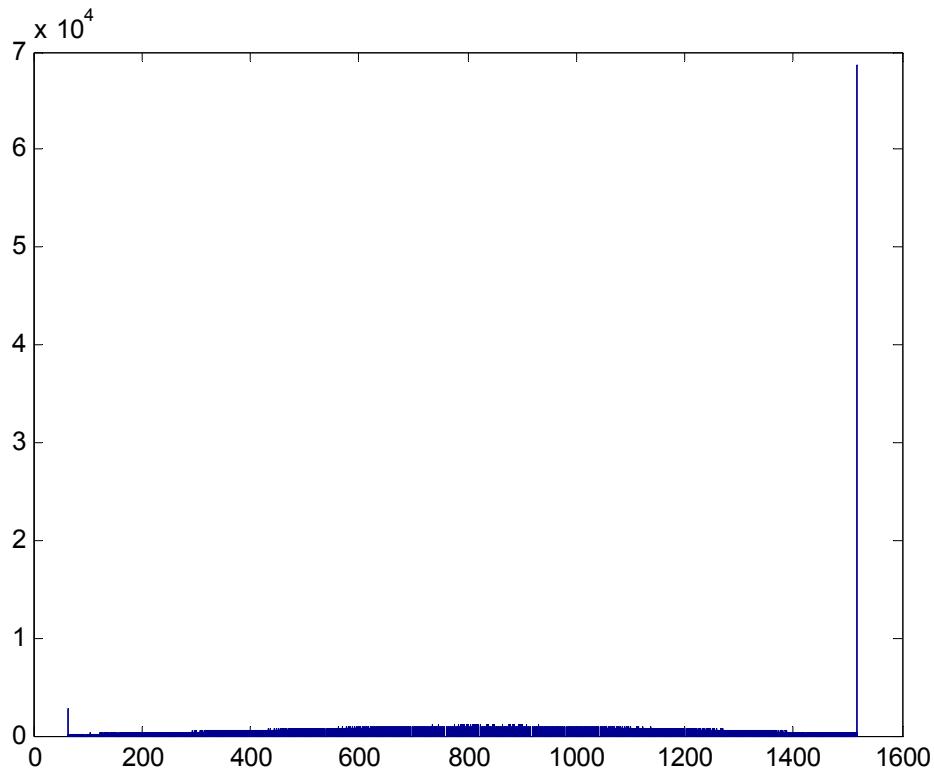


**Fig. A8.7** Histograma con distribución normal (800,50)

La generación es muy precisa, pero algo desviada de la media que se le había asignado.

El propio manual del programa *daggen* ya avisaba de que se podían producir desajustes en la generación de este tipo de distribución.

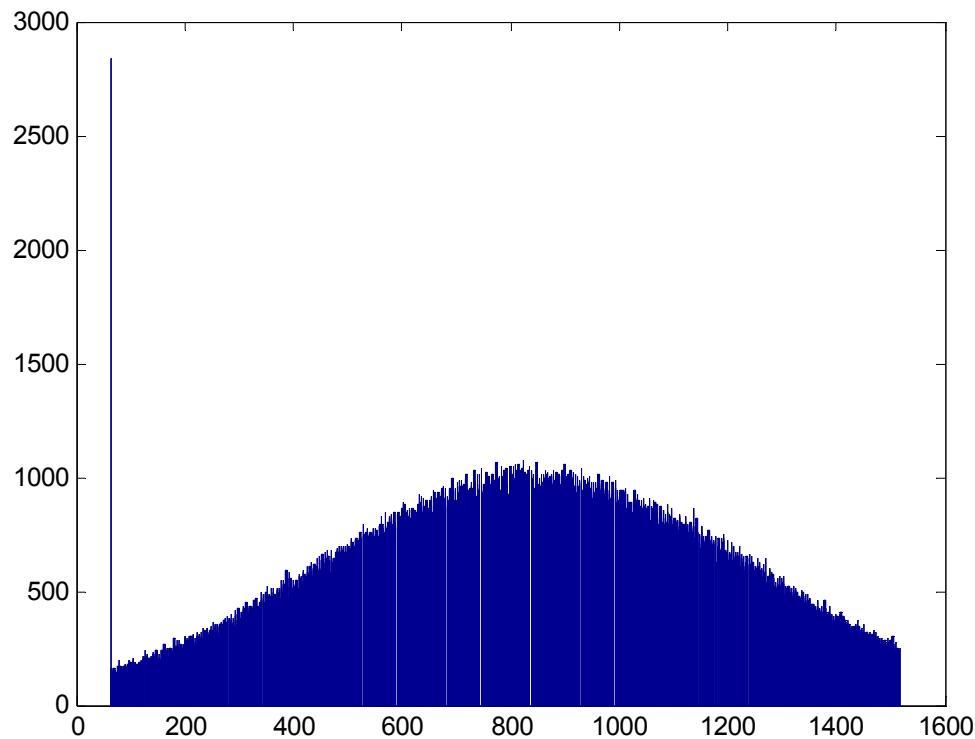
El histograma para  $\text{normal}(800, 200)$  es de la siguiente manera:



**Fig. A8.8** Histograma con distribución normal (800,200)

En este caso se puede observar como hay menor cantidad de paquetes por tipo de longitud, lo que hace tener una gráfica mucho más llana. Los extremos tienen picos debidos a que se han creado paquetes inferiores al mínimo (64) y superiores al máximo (1518), con lo que esos paquetes se sitúan en esos dos valores por ser cantidad mínima y máxima del estándar de Ethernet.

Si cortamos el valor máximo podemos apreciar mejor el histograma:



**Fig. A8.9** Histograma con distribución normal (800,200) filtrado

### Errores de generación

En los casos en que la generación de paquetes es con longitud aleatoria, hay un problema grave si se quiere que ese tráfico atraviese un router. Ésta generación crea un tamaño de *dummy* variable y la creación con el script se programa con una longitud fija de datagrama, con lo que para cada paquete estos dos valores serán diferentes. Esto hace que el router los compare y descarte los paquetes que no contengan la misma longitud en los dos campos. Así que, sólo pasarán los paquetes que coincidan con el valor 05 94 (1428 bytes) que está configurado estáticamente en el script.

El problema surge en que, al analizarlo con Netflow sólo se detectan estos paquetes y no se puede realizar un estudio correcto.

Se ha intentado a través de *daggen* la posibilidad de que el campo de longitud de UDP sea variable pero no es posible, ya que es un software cerrado y no es posible agregar sintaxis de programación estándar.

## IX. Tratado CESCA y comandos

La traza de Cesca se encuentra en formato PCAP con lo que ya está en el formato correcto para tratarlo con la suite de *tcpreplay*. Primeramente le pasaremos al archivo el *tcpprep* para crear el archivo *input.cache* y así dividir el tráfico entre dos nodos. El comando es el siguiente:

```
servgenmonI:/home# tcpprep -a -n bridge -i 20040219-12500-a.pcap -o
input.cache -v
```

Donde -a es el modo automático, -n el modo de parseo, -i el archivo input de entrada, -o el archivo de salida (cache) y -v verbose para obtener más información por pantalla.

A continuación hay que aplicar el *tcprewrite* para reescribir cabeceras MAC e IP a todos los paquetes de la traza. Para realizar el siguiente paso hay que tener en cuenta si se desea enviar la traza con la DAG o con la interfaz estándar eth0, ya que el comando cambia:

Para DAG:

```
servgenmonI:/home# tcprewrite --enet-dmac=00:07:B3:5F:9C:E0 --enet-
smac=00:30:48:76:1D:5B --infile=20040219-12500-a.pcap --
outfile=output_cesca.pcap --endpoints=147.83.118.242:10.0.13.101 --
cachefile=input.cache --skipbroadcast
```

Como comentamos en capítulos anteriores, si se desea enviar con la DAG nos debemos inventar una MAC y utilizar una IP válida del rango, así que usaremos las ya utilizadas anteriormente.

Las direcciones MAC las asignamos con los flags *--enet-smac* y *--enet-dmac*, origen y destino respectivamente. Con *--infile* indica el archivo de entrada (traza Cesca) y *--outfile* el archivo de salida resultante.

El flag *--endpoints* indica las IP's origen y destino separadas por dos puntos; *--cachefile* indica el uso del archivo cache generado previamente con *tcpprep* y finalmente *--skipbroadcast* sirve para poder descartar los paquetes broadcast de la traza.

Para eth0:

```
servgenmonI:/home# tcprewrite --enet-dmac=00:07:B3:5F:9C:E0 --enet-
smac=00:30:48:76:1D:5A --infile=20040219-12500-a.pcap --
outfile=output_cesca.pcap --endpoints=147.83.118.248:10.0.13.101 --
cachefile=input.cache --skipbroadcast
```

Este caso es el mismo que para la DAG, pero se deben cambiar las direcciones MAC e IP por las configuradas en la tarjeta Ethernet. De este modo evitamos que la tabla SAT del router encuentre ambigüedades por encontrar la misma MAC en diferentes puertos.

Al final el proceso obtenemos una traza PCAP con direcciones MAC e IP's cambiadas y más importante aún, con checksum recalculado.

Finalmente, tenemos un archivo PCAP que podemos emitir directamente con `tcpreplay` o transformarlo a ERF para enviarlo con la DAG.

Enviarlo con `tcpreplay` es de la siguiente manera:

```
servgenmonI:/home# tcpreplay -i eth0 output_cesca.pcap
```

Para la DAG es necesario realizar la conversión a ERF con la herramienta `dagconvert`:

```
servgenmonI:/home# dagconvert -v -T erf:pcap -i output_cesca.pcap -o cesca_dagconvert.erf
```

Donde `-v` indica verbose, `-T` el tipo de entrada y salida mediante dos puntos, `-i` el input y `-o` el output.

## X. Traza 100000 y un millón de paquetes

El script final a procesar por `daggen` es el siguiente:

```
//INICIO SCRIPT

packets {
    counters{

        a [66:1428] size 2;
        b [38:1400] size 2;
        c [6651:5289] size 2;
        d normal (6000,5) size 2;
        e normal (7000,5) size 2;
    }

    packet eth_II first_packet{
        src_addr 00:30:48:76:1d:5a;
        dst_addr 00:07:b3:5f:9c:e0;
        protocol 0x800;
        payload "45 00" + a + "00 00 40 00 3f 11" + c +
        "93 53 76 f2 0a 00 0d 65" + d + e + b + "9a 40"
        + dummy ([38:1400]);
        fcs_size 4;
    }
}

traffic {
    group my_traffic_group {
        send first_packet 100000;
    }
}

//FIN SCRIPT
```

Al tener que enviar las trazas con `tcpreplay`, la MAC origen debe ser la utilizada por la eth0 del PC emisor, por lo que requiere que el parámetro *c* (checksum de la cabecera IP) se recalcule y comprenda ese nuevo rango de valores. Finalmente como se ha comentado en el capítulo anterior, vamos a utilizar una distribución normal con desviación de 5 para los puertos (parámetros *d* y *e*).

La instrucción que genera el archivo PCAP es el siguiente:

```
servgenmonI:/home# daggen -f new_script_ports2_eth0_normal15.gen -o
new_script_ports2_eth0_normal15.pcap -x my_traffic_group -p
```

Donde: *-f* es el script a utilizar, *-o* es el archivo PCAP de salida, *-x* el tráfico que queremos utilizar de los declarado dentro del script y *-p* el flag para seleccionar el output a PCAP, ya que por defecto es ERF.

Para emitir el archivo usaremos:

```
servgenmonI:/home# tcpreplay -i eth0
new_script_ports2_eth0_normal15.pcap
```

Para las siguientes pruebas sólo se debe cambiar el parámetro *first\_packet* por el valor de paquetes que se deseé generar.

El estudio se basa en comparar los histogramas, las medias y las desviaciones estándares que ofrecen los resultados de Nfsen.

## Traza de 100000 paquetes

Se ha replicado la traza con `tcpreplay` y muestra el siguiente output:

```
Actual: 100000 packets (76284069 bytes) sent in 76.33 seconds. Rated:
999398.2 bps, 7.62 Mbps, 1310.31 pps

Statistics for network device: eth0
    Attempted packets:      1000000
    Successful packets:     1000000
    Failed packets:         0
    Retried packets (ENOBUFS): 0
    Retried packets (EAGAIN): 0
```

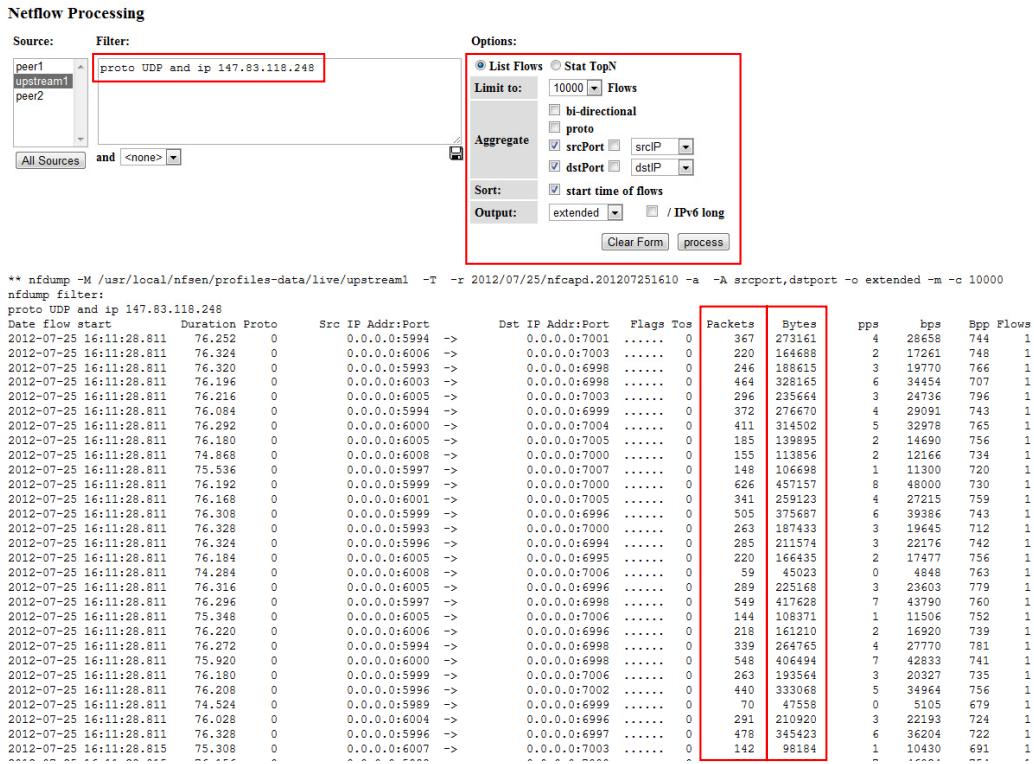
Con lo que nos aseguramos que ha enviado los 100000 paquetes sin errores.

Los flujos agregados por Nfsen verifican la cantidad y muestran información más detallada:

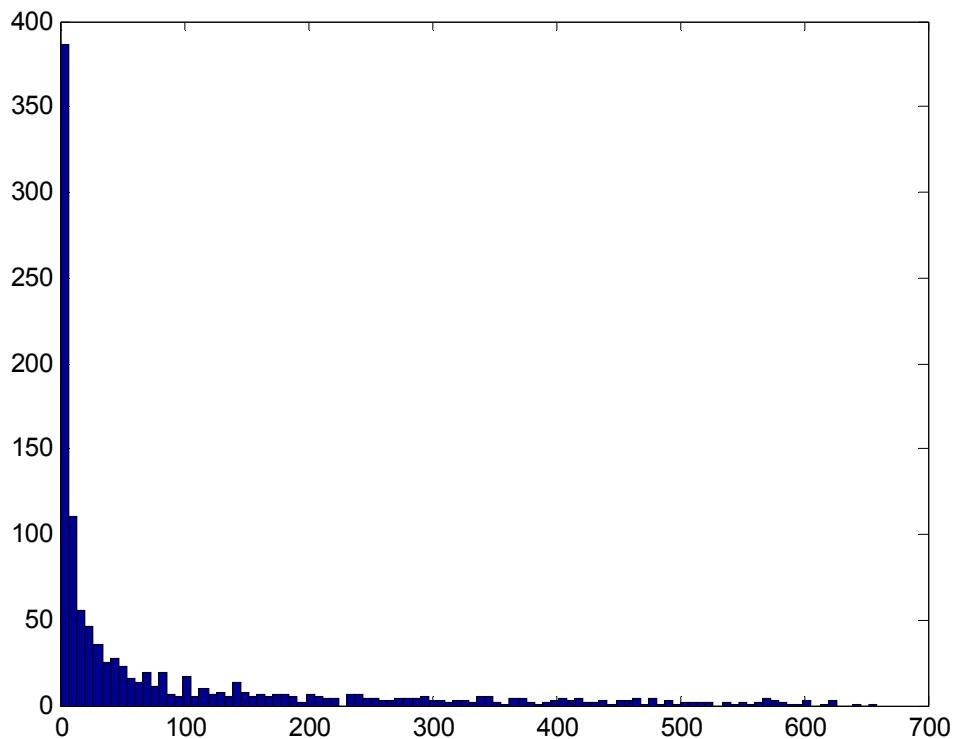
| Date  | flow | start        | Duration | Proto | Src IP    | Addr:Port | Dst IP    | Addr:Port | Flags | Tos    | Packets |
|---|------|--------------|----------|-------|-----------|-----------|-----------|-----------|-------|--------|---------|
| 2012-07-25  | 1462 | 16:11:28.811 | 76.332   | UDP   | 0.0.0.0:0 | ->        | 0.0.0.0:0 | .....     | 0     | 100000 |         |
| Summary: total flows: 1462, total bytes: 74.5 M, total packets: 100000, avg bps: 7.8 M, avg pps: 1310, avg bp |      |              |          |       |           |           |           |           |       |        |         |

**Fig. A10.1 Reporte Nfsen**

A continuación, si exportamos los flujos agregados por puerto origen y destino, es decir, todos los flujos que contengan los mismos números de puerto tanto origen y destino, serán agrupados; podremos realizar un histograma en base a los bytes y los paquetes de la transmisión:

**Fig. A10.2 Filtro Nfsen y columnas escogidas**

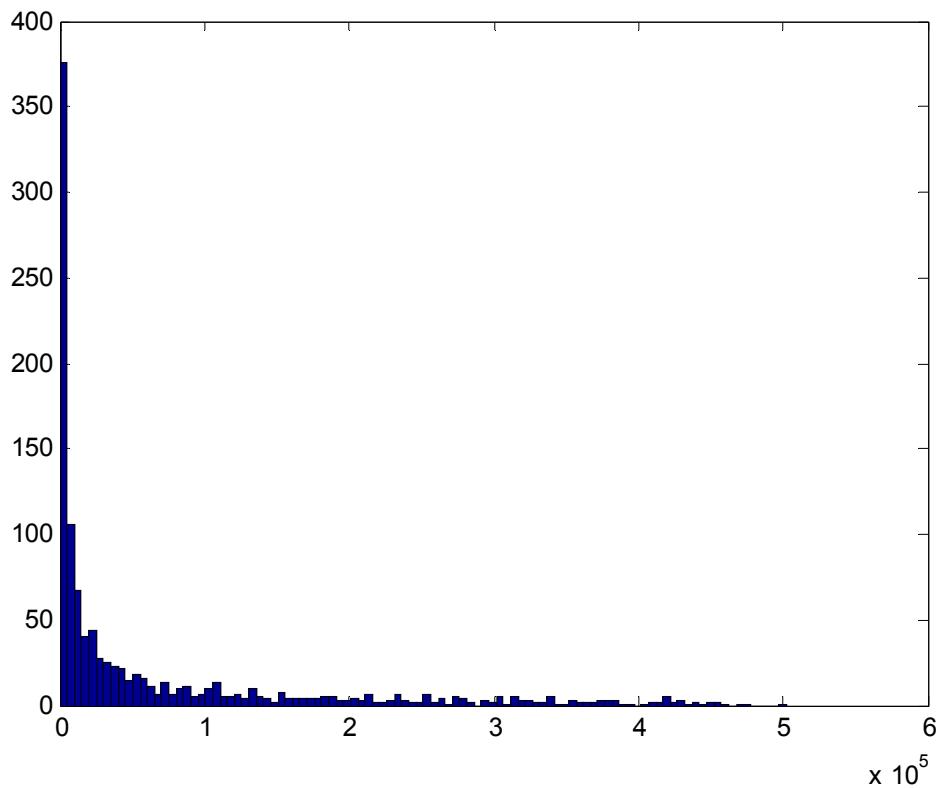
El histograma de paquetes queda de la siguiente manera:



**Fig. A10.3** Histograma de paquetes recibidos traza 100000

Podemos observar como hay mucha cantidad de flujos que tienen poca acumulación de paquetes, en cambio hay pocos flujos con cantidades elevadas de paquetes. Siendo la media 91,32 y la desviación estándar 144,23. Máximo 659 y mínimo 1.

El histograma de bytes queda de la siguiente manera:

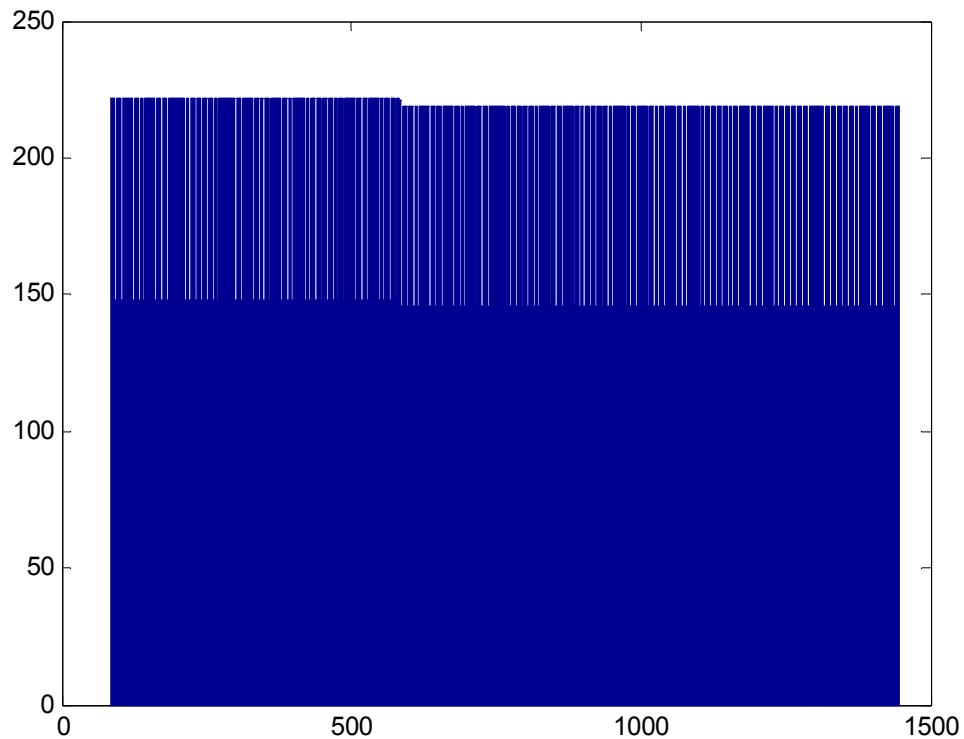


**Fig. A10.4** Histograma de bytes recibidos traza 100000

Se puede apreciar como en el caso de los bytes pasa algo muy parecido, en donde hay una gran cantidad de flujos que tienen pocos bytes, en cambio la gráfica baja drásticamente en cuanto los bytes aumentan, con lo que hay menos flujos que contengan tantos bytes. Su media es 68021,98 y la desviación estándar es 107493,26. Máximo 502505 y mínimo 66.

A continuación realizaremos el comportamiento de la variación de los tamaños de paquetes de la traza de 100000 paquetes. Así que exportaremos la traza a CSV y realizaremos el histograma de la columna Length.

El histograma es el siguiente:



**Fig. A10.5** Histograma de los tamaños de paquete traza 100000

Podemos ver como hay mucha uniformidad en cuanto a tamaño de paquete y cantidad de veces que aparece.

Ahora la idea es coger el flujo que ha obtenido más bytes y analizar el histograma de los paquetes que concentran esa comunicación, ya que el tamaño que aparece en Nfsen es la media. Para eso cogeremos el archivo nfcapd de la prueba realizada anteriormente (nfcapd.201207251610) y realizar un filtrado de puertos vía web:

**Netflow Processing**

Source: **peer1 upstream1 peer2** Filter: **proto udp and ip 147.83.118.248**

Options:

- List Flows  Stat TopN
- Limit to: **10000 Flows**
- bi-directional
- proto
- srcPort  srcIP
- dstPort  dstIP
- start time of flows
- Sort: **extended / IPv6 long**
- Output: **All Sources and <none>**

\*\* nfdump -M /usr/local/nfSEN/profiles-data/live/upstream1 -T -r 2012/07/25/nfcapd.201207251610 -a -A srcport,dstport -o extended -m -c 10000  
nfdump filter:  
proto udp and ip 147.83.118.248

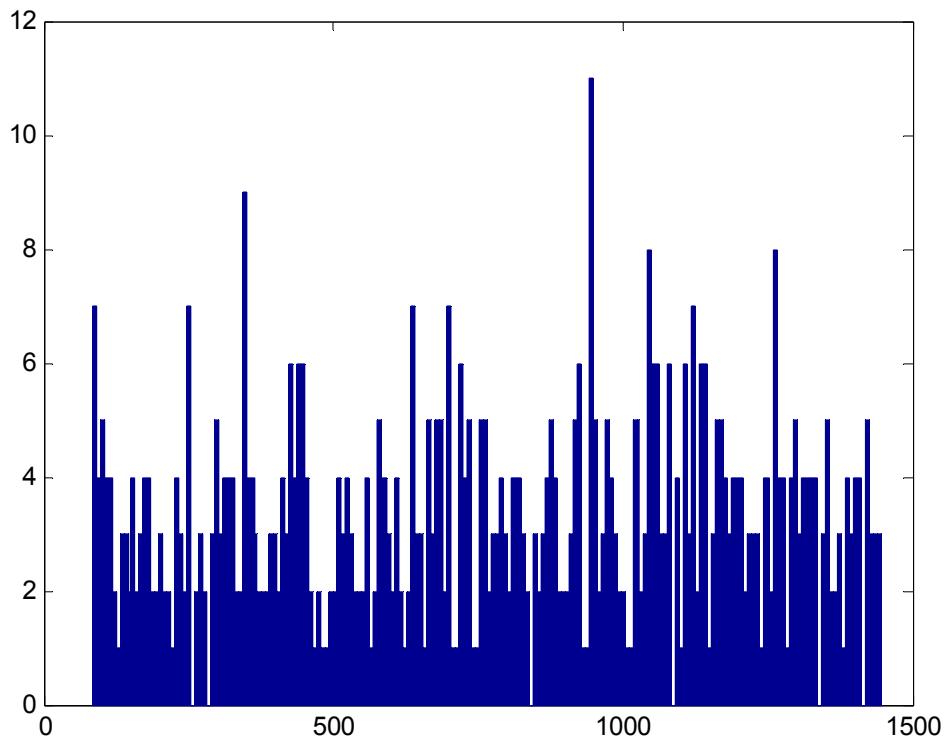
| Date       | flow         | start  | Duration | Proto        | Src IP | Addr:Port    | Dest IP | Addr:Port | Flags | Tos    | Packets | Bytes | pps | bps | Bpp | Flows |
|------------|--------------|--------|----------|--------------|--------|--------------|---------|-----------|-------|--------|---------|-------|-----|-----|-----|-------|
| 2012-07-25 | 16:11:28.811 | 76.252 | 0        | 0.0.0.0:5994 | ->     | 0.0.0.0:7001 | .....   | 0         | 367   | 273161 | 4       | 28658 | 744 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.324 | 0        | 0.0.0.0:6006 | ->     | 0.0.0.0:7003 | .....   | 0         | 220   | 164689 | 2       | 17261 | 748 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.320 | 0        | 0.0.0.0:5993 | ->     | 0.0.0.0:6998 | .....   | 0         | 246   | 188615 | 3       | 19770 | 766 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.196 | 0        | 0.0.0.0:6003 | ->     | 0.0.0.0:6998 | .....   | 0         | 464   | 328165 | 6       | 34454 | 707 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.216 | 0        | 0.0.0.0:6005 | ->     | 0.0.0.0:7003 | .....   | 0         | 296   | 235664 | 3       | 24736 | 796 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.084 | 0        | 0.0.0.0:5994 | ->     | 0.0.0.0:6999 | .....   | 0         | 372   | 276670 | 4       | 29091 | 743 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.292 | 0        | 0.0.0.0:6000 | ->     | 0.0.0.0:7004 | .....   | 0         | 411   | 314502 | 5       | 32978 | 765 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.180 | 0        | 0.0.0.0:6005 | ->     | 0.0.0.0:7005 | .....   | 0         | 185   | 139895 | 2       | 14690 | 756 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 74.868 | 0        | 0.0.0.0:6008 | ->     | 0.0.0.0:7000 | .....   | 0         | 155   | 113856 | 2       | 12166 | 734 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 75.536 | 0        | 0.0.0.0:5997 | ->     | 0.0.0.0:7007 | .....   | 0         | 148   | 106698 | 1       | 11300 | 720 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.192 | 0        | 0.0.0.0:5999 | ->     | 0.0.0.0:7000 | .....   | 0         | 626   | 457157 | 8       | 48000 | 730 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.168 | 0        | 0.0.0.0:6001 | ->     | 0.0.0.0:7005 | .....   | 0         | 341   | 259123 | 4       | 27215 | 759 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.308 | 0        | 0.0.0.0:5999 | ->     | 0.0.0.0:6996 | .....   | 0         | 505   | 375687 | 6       | 39386 | 743 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.328 | 0        | 0.0.0.0:5993 | ->     | 0.0.0.0:7000 | .....   | 0         | 263   | 187433 | 3       | 19645 | 712 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.324 | 0        | 0.0.0.0:5996 | ->     | 0.0.0.0:6994 | .....   | 0         | 285   | 211574 | 3       | 22176 | 742 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.184 | 0        | 0.0.0.0:6005 | ->     | 0.0.0.0:6995 | .....   | 0         | 220   | 166435 | 2       | 17477 | 756 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 74.284 | 0        | 0.0.0.0:6008 | ->     | 0.0.0.0:7006 | .....   | 0         | 59    | 45023  | 0       | 4848  | 763 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.316 | 0        | 0.0.0.0:6005 | ->     | 0.0.0.0:6996 | .....   | 0         | 289   | 225168 | 3       | 23603 | 779 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.296 | 0        | 0.0.0.0:5997 | ->     | 0.0.0.0:6998 | .....   | 0         | 549   | 417628 | 7       | 43790 | 760 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 75.348 | 0        | 0.0.0.0:6005 | ->     | 0.0.0.0:7006 | .....   | 0         | 144   | 108371 | 1       | 11506 | 752 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.220 | 0        | 0.0.0.0:6006 | ->     | 0.0.0.0:6996 | .....   | 0         | 218   | 161210 | 2       | 16920 | 739 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.272 | 0        | 0.0.0.0:5994 | ->     | 0.0.0.0:6998 | .....   | 0         | 339   | 264765 | 4       | 27770 | 781 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 75.920 | 0        | 0.0.0.0:6000 | ->     | 0.0.0.0:6998 | .....   | 0         | 548   | 406494 | 7       | 42833 | 741 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.180 | 0        | 0.0.0.0:5999 | ->     | 0.0.0.0:7006 | .....   | 0         | 263   | 193564 | 3       | 20327 | 735 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.208 | 0        | 0.0.0.0:5996 | ->     | 0.0.0.0:7002 | .....   | 0         | 440   | 330368 | 5       | 34964 | 756 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 74.524 | 0        | 0.0.0.0:5989 | ->     | 0.0.0.0:6999 | .....   | 0         | 70    | 47558  | 0       | 5105  | 679 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.028 | 0        | 0.0.0.0:6004 | ->     | 0.0.0.0:6996 | .....   | 0         | 291   | 210920 | 3       | 22193 | 724 | 1   |     |       |
| 2012-07-25 | 16:11:28.811 | 76.328 | 0        | 0.0.0.0:5996 | ->     | 0.0.0.0:6997 | .....   | 0         | 478   | 345423 | 6       | 36204 | 722 | 1   |     |       |
| 2012-07-25 | 16:11:28.815 | 75.308 | 0        | 0.0.0.0:6007 | ->     | 0.0.0.0:7003 | .....   | 0         | 142   | 98184  | 1       | 10430 | 691 | 1   |     |       |
| 2012-07-25 | 16:11:28.815 | 76.156 | 0        | 0.0.0.0:5998 | ->     | 0.0.0.0:7000 | .....   | 0         | 581   | 438126 | 7       | 46024 | 754 | 1   |     |       |
| 2012-07-25 | 16:11:28.815 | 70.608 | 0        | 0.0.0.0:5990 | ->     | 0.0.0.0:6991 | .....   | 0         | 38    | 30081  | 0       | 3408  | 791 | 1   |     |       |
| 2012-07-25 | 16:11:28.815 | 76.280 | 0        | 0.0.0.0:5999 | ->     | 0.0.0.0:6995 | .....   | 0         | 396   | 294023 | 5       | 30836 | 742 | 1   |     |       |

**Fig. A10.6 Captura filtraje de Nfsen**

Ahora simplemente hay que coger todos los datos y pasarlos a Excel y realizar una ordenación de la columna bytes de mayor a menor. Una vez realizado eso, obtendremos la pareja de puertos que han obtenido mayor bytes.

En este caso es la pareja 5999:6999, con lo que ya podemos filtrar la captura original con esos puertos (ip.port==5999 && ip.port==6999), exportarla a CSV y realizar el histograma de la columna Length con Matlab.

El histograma queda de la siguiente manera:



**Fig. A10.7** Histograma de los tamaños de paquete del flujo con mayor bytes

Podemos observar como hay bastante diversidad de tamaños para el flujo con mayor bytes, con lo que podemos concluir que no sigue una distribución normal. Tiene media 780,52 y su desviación media es 390,60.

### Traza de 1 millón de paquetes

Se ha replicado la traza con tcpreplay y muestra el siguiente output, con lo que nos aseguramos que ha enviado los 1000000 paquetes sin errores.

```
Actual: 1000000 packets (764796459 bytes) sent in 766.98 seconds.
Rated: 997153.1 bps, 7.61 Mbps, 1303.81 pps
Statistics for network device: eth0
    Attempted packets:          1000000
    Successful packets:         1000000
    Failed packets:             0
    Retried packets (ENOBUFS):  0
    Retried packets (EAGAIN):   0
```

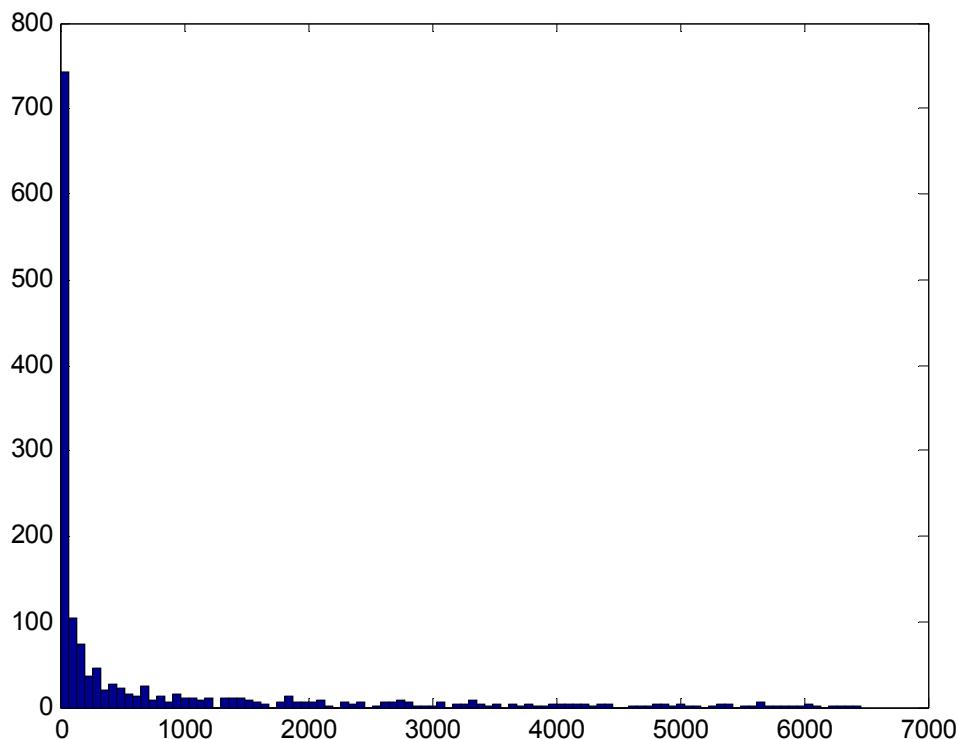
Los flujos agregados por Nfsen verifican la cantidad y muestran información más detallada:

```
Date flow start      Duration Proto      Src IP Addr:Port      Dst IP Addr:Port      Flags Tos  Packets
2012-07-25 18:08:00.880 766.968 UDP      0.0.0.0:0      ->      0.0.0.0:0      ..... 0 1.0 M
Summary: total flows: 8512, total bytes: 746.8 M, total packets: 1.0 M, avg bps: 7.8 M, avg pps: 1303, avg bpp
Time window: 2012-06-06 01:21:47 - 2012-07-25 18:24:46
```

**Fig. A10.8 Resultado la transmisión en Nfsen**

Por otro lado, si exportamos los flujos agregados por puerto origen y destino, es decir, todos los flujos que contengan los mismos números de puerto tanto origen y destino, serán agrupados; podremos realizar un histograma en base a los bytes y los paquetes de la transmisión.

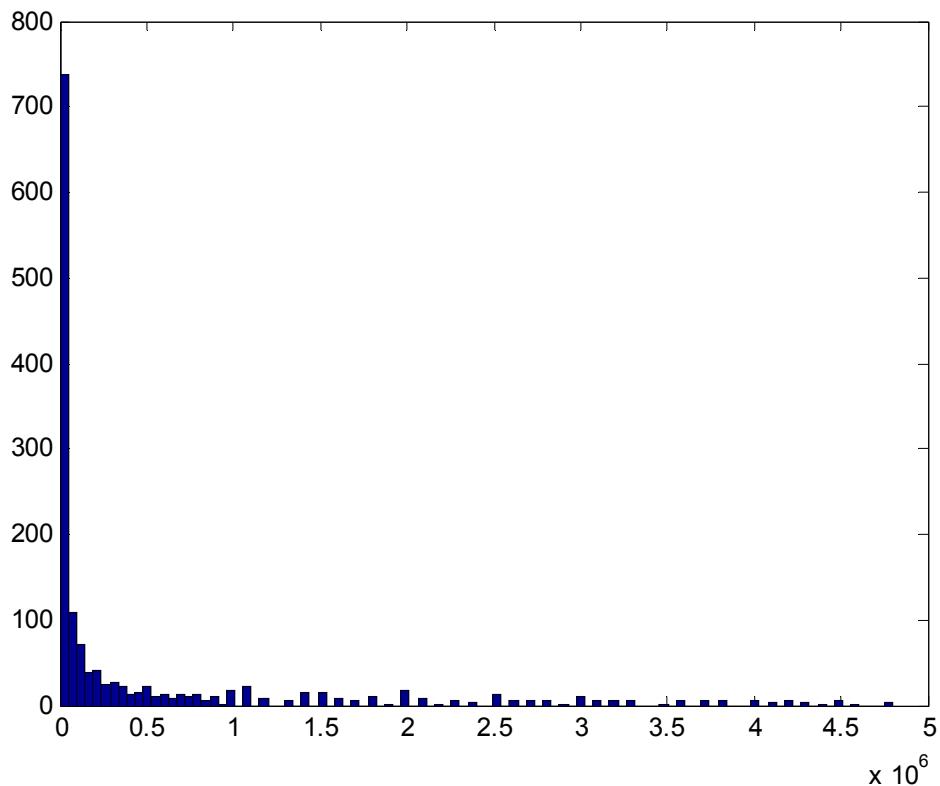
El histograma de paquetes queda de la siguiente manera:



**Fig. A10.9 Histograma de paquetes recibidos traza 1000000**

Podemos observar como hay mucha cantidad de flujos que tienen poca acumulación de paquetes, en cambio hay pocos flujos con cantidades elevadas de paquetes. Siendo la media 683,06 y la desviación estándar 1305,04. Máximo 6465 y mínimo 1.

El histograma de bytes queda de la siguiente manera:

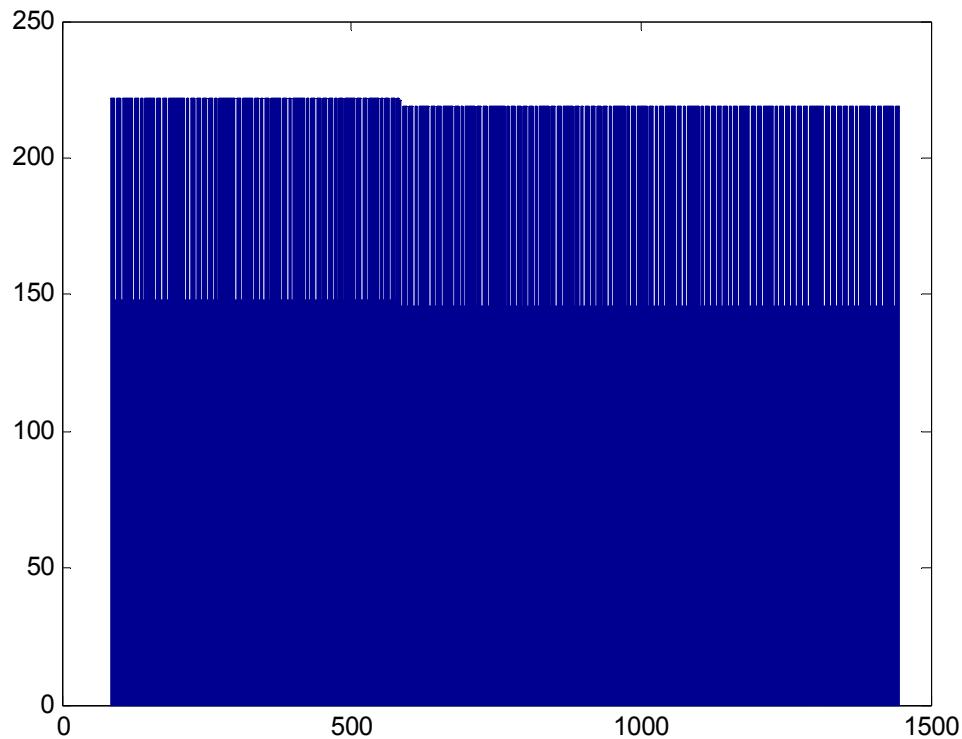


**Fig. A10.10** Histograma de bytes recibidos traza 1000000

Se puede apreciar como en el caso de los bytes pasa algo muy parecido, en donde hay una gran cantidad de flujos que tienen pocos bytes, en cambio la gráfica baja drásticamente en cuanto los bytes aumentan, con lo que hay menos flujos que contengan tantos bytes. Su media es 510494,56 y la desviación estándar es 975948,40. Máximo 4800000 y mínimo 85.

A continuación realizaremos el comportamiento de la variación de los tamaños de paquetes de la traza de 100000 paquetes. Así que exportaremos la traza a CSV y realizaremos el histograma de la columna Length.

El histograma es el siguiente:



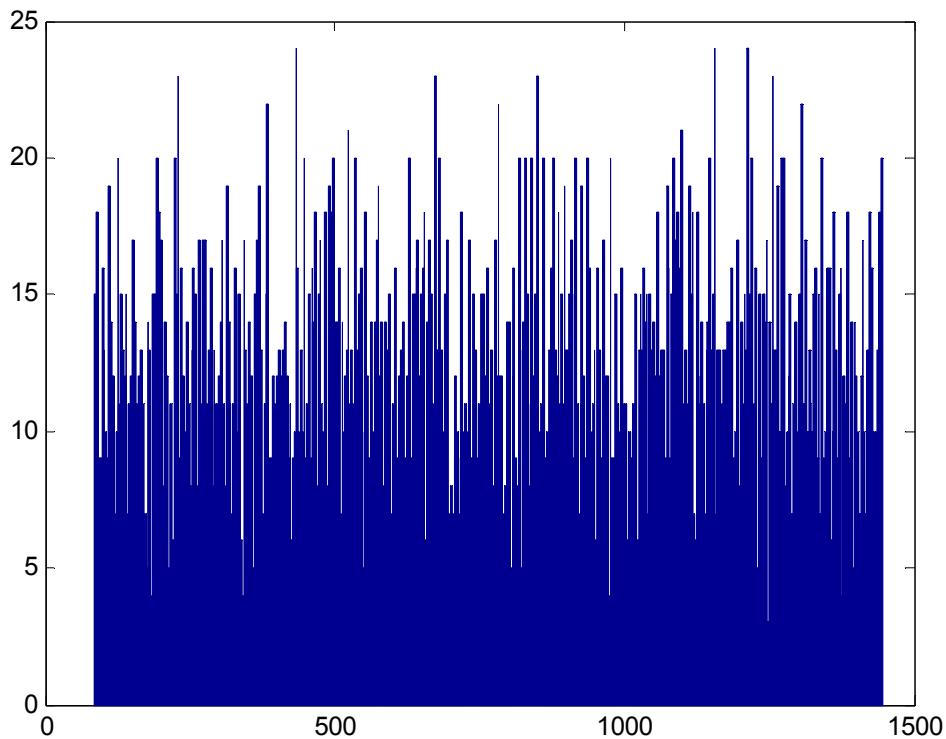
**Fig. A10.11** Histograma de tamaño de paquete traza de un millón

Podemos ver como hay mucha uniformidad en cuanto a tamaño de paquete y cantidad de veces que aparece.

Ahora la idea es coger el flujo que ha obtenido más bytes y analizar el histograma de los paquetes que concentran esa comunicación.

En este caso es la pareja 5999:6999, con lo que ya podemos filtrar la captura original con esos puertos (`ip.port==6000 && ip.port==6999`), exportarla a CSV y realizar el histograma de la columna Length con Matlab.

El histograma queda de la siguiente manera:



**Fig. A10.12** Histograma de los tamaños de paquete del flujo con mayor bytes

Podemos observar como hay bastante diversidad de tamaños para el flujo con mayor bytes, con lo que podemos concluir que no sigue una distribución normal. Tiene media 770,23 y su desviación media es 390,67.

## XI. Sintaxis cron para las pruebas Full/Random Netflow

*Cron* es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero *crontab*.

Para configurar el archivo sólo debemos introducir el comando: *crontab -e* y agregar las líneas deseadas. Es importante saber que sólo se ejecutarán las instrucciones siempre y cuando cumplan un AND booleano, es decir que solo resulta verdadero si los 5 campos son ciertos en el minuto especificado.

### Sintaxis

La sintaxis es sencilla y sólo es necesario configurar el qué y cuándo se quiere ejecutar.

```
----- minuto (0 - 59)
| .----- hora (0 - 23)
| | .----- día del mes (1 - 31)
| | | .---- mes (1 - 12) O jan,feb,mar,apr ... (los meses en inglés)
| | | | .--- día de la semana (0 - 6) (Domingo=0 ó 7) O sun,mon,tue,wed,thu,fri,sat (los días en inglés)
| | | |
* * * * comando para ser ejecutado
```

**Fig. A11.1 Sintaxis de cron**

Se verá más fácil con algunos ejemplos:

```
25 10 22 5 wed /usr/bin/who
```

Esta línea ejecutará el who, el día 22 de mayo a las 10:25 y que sea miércoles.

También es posible operar con asteriscos como comodín para comprender todos los valores del campo especificado:

```
25 10 * * wed /usr/bin/who
```

Esta línea ejecutará el who, todos los días de todos los meses que sean miércoles y sean las 10:25.

### Sincronización Cron con Full Netflow

La instrucción que nos interesa para nuestra batería de pruebas en modo full netflow es el siguiente:

```
1,6,11,16,21,26,31,36,41,46,51,56 13,14,15,16,17,18,19,20,21 30 7 1
/home/xavi/dag-2.5.5/tools/100k.sh
```

Como ya hemos comentado anteriormente enviaremos la instrucción al minuto siguiente de que Nfsen genere su archivo nfcapd.

Con lo que en el campo minutos, ponemos todos los posibles que hay en una hora, a continuación todas las horas que queremos que tarde en enviar las 100 instrucciones, el día 30 de julio y que sea lunes.

En resumen, la instrucción se ejecutará en el minuto 1,6,11... durante las horas 13,14... del día 30 de julio si es lunes.

El script 100k.sh sólo contiene la instrucción que ejecuta tcpreplay por la interfaz y el archivo deseado:

```
#!/bin/bash
/usr/local/bin/tcpreplay -i eth0 /home/xavi/dag-
2.5.5/tools/new_script_ports2_eth0_normal5.pcap
```

Es muy importante comprobar que las rutas de los archivos sean correctas, tanto la del programa (tcpreplay) como la del archivo (PCAP), ya que cron no sabe utilizar las instrucciones sin las rutas completas.

También cabe destacar que al ser un demonio, ninguna de las instrucciones ni los resultados aparecerán por pantalla.

### **Sincronización Cron con Random Netflow 1 de cada 10**

La instrucción que nos interesa para nuestra batería de pruebas en modo 1 de cada 10 es el siguiente:

```
1,6,11,16,21,26,31,36,41,46,51,56 0,1,2,3,4,5,6,7,8,9 31 7 2  
/home/xavi/dag-2.5.5/tools/100k.sh
```

Como ya hemos comentado anteriormente enviaremos la instrucción al minuto siguiente de que Nfsen genere su archivo nfcapd.

Por ello, en el campo minutos, ponemos todos los posibles que hay en una hora, a continuación todas las horas que queremos que tarde en enviar las 100 instrucciones, el día 31 de julio y que sea martes.

En resumen, la instrucción se ejecutará en el minuto 1,6,11... durante las horas 0,1,2... del día 31 de julio si es martes.

El script que se usa es el *100k.sh*, el mismo que en el caso anterior.

### **Sincronización Cron con Random Netflow 1 de cada 100**

La instrucción que nos interesa para nuestra batería de pruebas en modo 1 de cada 100 es el siguiente:

```
1,6,11,16,21,26,31,36,41,46,51,56 15,16,17,18,19,20,21,22,23 31 7 2  
/home/xavi/dag-2.5.5/tools/100k.sh
```

Como ya hemos comentado anteriormente enviaremos la instrucción al minuto siguiente de que Nfsen genere su archivo nfcapd.

Por ello, en el campo minutos, ponemos todos los posibles que hay en una hora, a continuación todas las horas que queremos que tarde en enviar las 100 instrucciones, el día 31 de julio y que sea martes.

En resumen, la instrucción se ejecutará en el minuto 1,6,11... durante las horas 15,16,17... del día 31 de julio si es martes.

El script que se usa es el *100k.sh*, el mismo que en el caso anterior.

## Sincronización Cron con Random Netflow 1 de cada 1000

La instrucción que nos interesa para nuestra batería de pruebas en modo 1 de cada 1000 es el siguiente:

```
1,6,11,16,21,26,31,36,41,46,51,56 10,11,12,13,14,15,16,17,18 1 8 3  
/home/xavi/dag-2.5.5/tools/100k.sh
```

Como ya hemos comentado anteriormente enviaremos la instrucción al minuto siguiente de que Nfsen genere su archivo nfcapd.

Por ello, en el campo minutos, ponemos todos los posibles que hay en una hora, a continuación todas las horas que queremos que tarde en enviar las 100 instrucciones, el día 1 de agosto y que sea miércoles.

En resumen, la instrucción se ejecutará en el minuto 1,6,11... durante las horas 10,11,12... del día 1 de agosto si es miércoles.

El script que se usa es el *100k.sh*, el mismo que en el caso anterior.

## Sincronización Cron con Random Netflow 1 de cada 65535

La instrucción que nos interesa para nuestra batería de pruebas en modo 1 de cada 65535 es el siguiente:

```
1,6,11,16,21,26,31,36,41,46,51,56 10,11,12,13,14,15,16,17,18 2 8 4  
/home/xavi/dag-2.5.5/tools/100k.sh
```

Como ya hemos comentado anteriormente enviaremos la instrucción al minuto siguiente de que Nfsen genere su archivo nfcapd.

Por ello, en el campo minutos, ponemos todos los posibles que hay en una hora, a continuación todas las horas que queremos que tarde en enviar las 100 instrucciones, el día 1 de agosto y que sea miércoles.

En resumen, la instrucción se ejecutará en el minuto 1,6,11... durante las horas 10,11,12... del día 2 de agosto si es jueves.

El script que se usa es el *100k.sh*, el mismo que en el caso anterior.

## XII. Procesado de archivo *nfcapd*

El filtrado es muy intuitivo vía web a través de Nfsen, pero en segundo plano se está invocando una completa instrucción de nfdump. Al tratarse de 100 pruebas para cada tipo de muestreo es necesario automatizar este proceso con un script. La idea es realizar el filtrado de los 100 archivos con las horas correspondientes a la transmisión y provocar la salida a archivos TXT.

Los archivos *nfcapd* tienen la siguiente nomenclatura: *nfcapd.AAAAMMDDHHMM*. Veámoslo con un ejemplo:

*Nfcapd.201205131540*, este archivo es el registro de Nfsen de las 15:40 del día 13 de mayo del 2012.

El script *extended.sh* implementado es válido sólo para archivos *nfcapd* que estén con un rango de hora entre las 10 y las 23 y es el siguiente:

```
#!/bin/bash

counter=1
hora=1325
minutos=25

while [ $counter -lt 101 ]
do
echo "Inicio de parseo del nfcapd[$hora]"

./nfdump -r nfcapd.20120730$hora -o extended 'proto UDP and ip
147.83.118.248' -A srcport,dstport -m -c 10000 > nfcapd-
full/nfcapd.20120730$hora.txt

let counter=counter+1
let hora=hora+5

let minutos=minutos+5

if [ $minutos -eq 60 ]; then
    let hora=hora+40
    let minutos=0
fi
done
```

Como se puede observar en el código, el comando ejecuta el programa *nfdump* con diferentes flags para el filtraje: con *-o* obtenemos salida extendida de datos, entre comillas simples se ubican los filtros por protocolo e ip (en nuestro caso sólo lo que contenga ip del pc generador y protocolo UDP), con *-A* realizamos la agregación de puertos, tanto origen como destino, con *-m* se ordenan por el instante de tiempo en que han empezado y con *-c* el número máximo de flujos a mostrar.

El problema del rango de horas es por culpa del sistema operativo ya que si ponemos una hora con ceros a la izquierda (por ejemplo cualquier hora entre las 00 y las 09), éste los ignorará y no los pondrá en el nombre del archivo.

Por otro lado, dependiendo de la hora en que empiecen las transmisiones de los diferentes tipos de muestreo hay que cambiar tanto la constante *hora* como la de *minutos*.

En este caso, si se realizan pruebas en el horario crítico, se ha utilizado el programa FlexibleRenamer para renombrar masivamente todos los archivos nfcapd con una nomenclatura válida para el script. Posteriormente al parseo, se utilizó de nuevo el programa para poner el nombre de los archivos correctamente.

### XIII. Software java de comparación

```
package prueba;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.StringWriter;

public class inicio {

    public static void main(String[] args) {
        int contCopia = 1325;
        int contACopia = 25;

        try{
            String linea = null;
            String linea2 = null;

            BufferedReader bo = null;//buffer original
            BufferedReader bc = null;
            BufferedWriter br = null;
            String[] resultPackets = new String[1096];
            String[] resultBytes = new String[1096];

            File archivoRpackets = new
File("F:/Pruebas_DAG_CBR/new_script/nfcapd-full/matlab/packets.txt");
```

```

File archivoRBytes = new
File("F:/Pruebas_DAG_CBR/new_script/nfcapd-full/matlab/bytes.txt");

for(int i=0; i<100; i++) {

    System.out.println(contCopia);

    File archivoO = new
File("F:\Pruebas_DAG_CBR\new_script\nfcapd.201207301325.txt");

    File archivoC = new
File("F:\Pruebas_DAG_CBR\new_script\nfcapd-full\nfcapd-full-
extended\nfcapd.20120730"+ contCopia +".txt");

    FileReader fileO = new
FileReader(archivoO);//abrir archivo

    bo = new BufferedReader(fileO);

    boolean copia = false;
    int cont = 0;
    boolean fin = false;

    while ((linea = bo.readLine()) != null) {

        if(linea.substring(0,7).equals("Summary")) {

            fin=true;
            System.out.println("sum: " +
linea.substring(0,7));
        }

        if(copia && !fin) {

            String srcO = linea.substring(49,61);
            String dstO = linea.substring(74,87);

            FileReader fileC = new
FileReader(archivoC);

            bc = new BufferedReader(fileC);

```

```
        boolean encontrado = false;

        while((linea2 = bc.readLine()) != null){

            if(linea2.length()>=87){

                String srcC =
linea2.substring(49,61);

                String dstC =
linea2.substring(74,87);

                if(src0.equals(srcC) &&
dst0.equals(dstC)){//compara ips y puertos origenes con los de la
copia

                    String packPackets =
linea2.substring(104,108);

                    String packBytes =
linea2.substring(110,117);

                    encontrado = true;

                    if(i==0){//la priemra vez que
entra inicializa el archivo

                        resultPackets[cont] =
"";

                        resultBytes[cont] = "";


                    }

                    resultPackets[cont] =
resultPackets[cont] + packPackets + ";";//acumula en array los
experimentos

                    resultBytes[cont] =
resultBytes[cont] + packBytes + ";"


                    cont++;



                break;
            }
        }
    }
}
```

```
        }

    }

    bc.close();

    fileC.close();

    if(!encontrado){

        if(i==0){

            resultPackets[cont] = "";//inicializa y quita el null

            resultBytes[cont] = "";

        }

        resultPackets[cont] = resultPackets[cont] + "X;";

        resultBytes[cont] = resultBytes[cont] + "X;";

        cont++;

    }

}

copia = true;//true par al aprimera linea

}

bo.close();

fileO.close();

contCopia = contCopia +5;

contACopia = contACopia+5;

if (contACopia==60){

    contCopia = contCopia+40;

    contACopia = 0;

}
```

```
        }

        FileWriter fileRpackets = new
FileWriter(archivoRpackets); //crea nuevo archivo

        FileWriter fileRBytes = new FileWriter(archivoRBytes);

        for(int i=0; i<1096; i++){//recorre string y escribe lo
leido

            fileRpackets.write(resultPackets[i] + "\n");

            fileRBytes.write(resultBytes[i] + "\n");

        }

        fileRpackets.close();

        fileRBytes.close();

    }

} catch(Exception e){//el catch de toda la vida

    StringWriter sw = new StringWriter();

    e.printStackTrace(new PrintWriter(sw));

    String stacktrace = sw.toString();

    System.out.println("Pues parece que peta: " +
stacktrace);

}

}

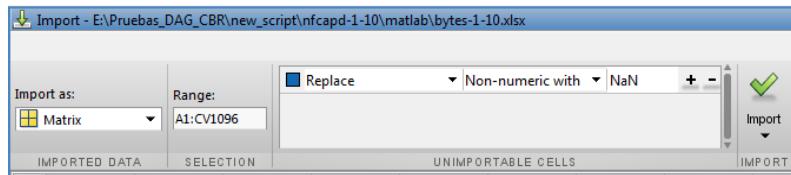
}
```

## XIV. Procesado de matrices en Matlab

Matlab interpreta las matrices con esta sintaxis: matriz(x,y), donde las x son las filas y las y las columnas. Al operar con ellas utilizaremos el comodín ":" para recorrer todas las posiciones de la coordenada deseada.

Primeramente hay que importar los vectores con los datos originales (*bytes\_o* y *packets\_o*) y las matrices de valores (*bytes\_Z* y *packets\_Z*; donde la *Z* es el muestreo).

Hay que tener en cuenta que las matrices pueden contener valores nulos ('x'), que deberemos cambiar a 'NaN' en el momento de la importación en Matlab, así ese valor no será cero sino nulo.



**Fig. A14.1 Filtro de reemplazo de X por NaN**

A continuación se realizan dos bucles para realizar todo este proceso de cálculo de errores y medias:

Cálculo para el archivo bytes (ejemplo para muestreo 1 de cada 10):

*Bucle con 1095 iteraciones en el que se aplica la fórmula 10.1 y crea una matriz de errores llamada errorB10*

```
for x=1:1095
errorB10(x,:)=abs(((bytes_o(x)/10)-
bytes_10(x,:))/((bytes_o(x)/10)));
end
```

*Bucle con 1095 iteraciones en el que se hace la media de cada flujo. Esta media no es sobre el total de experimentos, sino sólo por el número de experimentos en los que ha aparecido ese flujo.*

```
for x=1:1095
mediaB10(x,:)=mean(errorB10(x,find(not(isnan(errorB10(x,:))))));
end
```

La combinación del find, not e isnan devolverá la cantidad inversa de veces que aparece el 'NaN'. Es decir, si en toda la fila aparecen 35 NaN's esto nos dará el valor de 65 (número de veces detectado).

A partir de los datos obtenidos realizaremos los cálculos logarítmicos para graficar los diagramas de dispersión y su recta de tendencia. Estos datos nos ofrecen una información más detallada acerca de la dispersión y la correlación entre la media de errores y la cantidad de bytes o paquetes.

Si queremos buscar el pendiente logarítmico deberemos trabajar con los valores en esa escala, por lo que habrá que realizar el logaritmo de los valores originales de bytes y packets, junto con la media de errores obtenidos para cada tipo de muestreo. Para el ejemplo del cálculo de la recta de los bytes con muestreo 1 de cada 10 los comandos son los siguientes:

```
logx=log10(bytes_o);
logyb10=log10(mediaB10);
```

A partir de aquí, si se quiere representar la recta de tendencia se deberá obtener la media y la desviación de cada uno de los flujos con el programa tcpstat. Para ellos se ha realizado un script en base al orden de los puertos de la traza original originada por Nfsen y poder extraer los datos.

Los puertos (origen y destino respectivamente) almacenados y ordenados desde Nfsen son pasados a un txt llamado *puertos.txt* y con la siguiente forma:

| puertos.txt: Bloc de notas |         |
|----------------------------|---------|
| Archivo                    | Edición |
| 6006                       | 6996    |
| 5999                       | 7006    |
| 6000                       | 6998    |
| 5994                       | 6998    |
| 6005                       | 6995    |
| 6008                       | 7006    |
| 6005                       | 7003    |
| 5994                       | 6999    |
| 5993                       | 7000    |
| 6003                       | 6998    |
| 5989                       | 6999    |
| 5996                       | 6997    |
| 6006                       | 7003    |
| 6005                       | 7005    |
| 5997                       | 7007    |
| 5999                       | 6996    |
| 6008                       | 7000    |
| 6005                       | 6996    |
| 6004                       | 6996    |
| 6000                       | 7004    |
| 5993                       | 6998    |
| 5994                       | 7001    |
| 6001                       | 7005    |
| 5996                       | 7002    |
| 5997                       | 6998    |
| 6005                       | 7006    |
| 5998                       | 6993    |
| 6002                       | 7002    |
| 6013                       | 6995    |
| 6002                       | 6995    |
| 6004                       | 6999    |
| 5999                       | 6995    |
| 5990                       | 6991    |
| 6000                       | 7001    |
| 6001                       | 6999    |
| 5998                       | 7002    |
| 6000                       | 6994    |
| 5998                       | 7000    |
| 5989                       | 7003    |

**Fig. A14.2** Contenido archivo *puertos.txt*

A continuación se automatiza el proceso para extraer la media y la desviación de cada flujo que ofrece el programa tcpstat con un script llamado *script.sh*.

```

#!/bin/bash

k=0
for i in $(cat puertos.txt)
do
    array[$k]=$i
    k=$((k+1))
done

puerto_o=0
puerto_d=1
counter=1

while [ $counter -lt 1096 ]
do
echo "Calculando media y desviacion de longitud de paquetes del
flujo$counter con puertos ${array[$puerto_o]}:${array[$puerto_d]}"

./tcpstat -r new_script_ports2_eth0_normal5.pcap 100 -f "port
${array[$puerto_o]} && port ${array[$puerto_d]}" -o "%a\t%d\n" >>
append

let puerto_o=puerto_o+2
let puerto_d=puerto_d+2
let counter=counter+1
done

```

El funcionamiento de este script es el siguiente: pasamos los puertos a un array para poder trabajar con ellos y seguidamente ejecutamos tantas veces como sea necesaria (1095) la instrucción que invoca el programa:

```

./tcpstat -r new_script_ports2_eth0_normal5.pcap 100 -f "port
${array[$puerto_o]} && port ${array[$puerto_d]}" -o "%a\t%d\n" >>
append

```

Donde: `-r` es el archivo fuente de la información, `100` es la cantidad en segundos para el que se desea obtener los datos (como la transmisión del archivo PCAP dura 76,33 segundos, acordamos 100 para abarcar toda la transmisión), `-f` es el filtro que se aplica, aquí es donde se deben poner cada uno de los puertos origen y destino de cada flujo a partir de los valores del array (sigue la misma nomenclatura que los filtros de Wireshark), `-o` el output de salida (`%a` es la media y `%d` la desviación) y finalmente una redirección (`>>`) a un archivo `append`. Obteniendo un archivo de este tipo:

| 1  | 743.50 | 402.17 |
|----|--------|--------|
| 2  | 739.98 | 393.94 |
| 3  | 745.78 | 399.63 |
| 4  | 785.02 | 397.45 |
| 5  | 760.52 | 410.01 |
| 6  | 767.10 | 374.81 |
| 7  | 800.16 | 389.12 |
| 8  | 747.74 | 375.45 |
| 9  | 716.67 | 390.62 |
| 10 | 711.25 | 376.93 |
| 11 | 683.40 | 381.64 |
| 12 | 726.64 | 396.01 |
| 13 | 752.58 | 364.58 |
| 14 | 760.19 | 399.48 |
| 15 | 724.93 | 390.23 |
| 16 | 747.93 | 404.16 |
| 17 | 738.55 | 405.44 |
| 18 | 783.13 | 391.23 |
| 19 | 728.81 | 403.08 |
| 20 | 769.21 | 391.65 |
| 21 | 770.73 | 376.55 |
| 22 | 748.31 | 380.10 |
| 23 | 763.89 | 380.28 |
| 24 | 760.97 | 396.06 |
| 25 | 764.71 | 399.13 |
|    | ...    | ...    |

**Fig. A14.3 Contenido archivo *append***

Ahora ya es posible calcular el coeficiente de variación y la recta de tendencia con los siguientes comandos de Matlab:

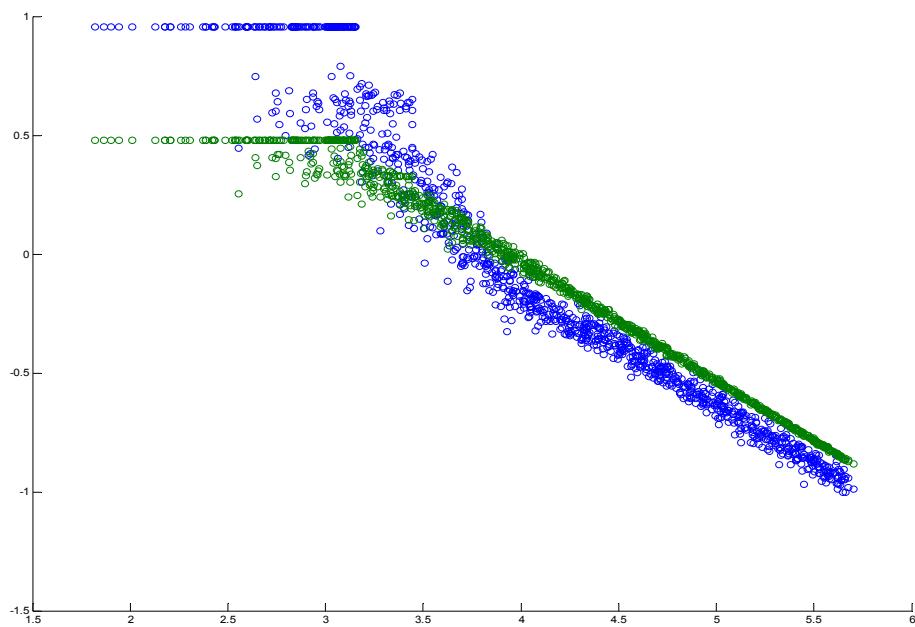
```
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectab10(i,1)=log10(sqrt(((1-
0.1)/(0.1*packets_o(i,1))))*coeff(i,1));
end
```

Una vez tengamos los valores, podremos graficar las dos representaciones juntas: el diagrama de dispersión logarítmico de los bytes originales en función de la media de errores junto con el diagrama de dispersión teórico.

```
scatter(logx,logyb10)
hold on
scatter(logx,rectab10)
```

Con `hold on` fijamos el gráfico para poder dibujar encima de nuevo.

El resultado es el siguiente:



**Fig. A14.4** Gráfica logarítmica de bytes con probabilidad 1 de cada 10 con recta

## XV. Diagramas de dispersión

### Diagramas de dispersión con el 100% de los flujos detectados

#### 100 experimentos con probabilidad 1 de cada 10

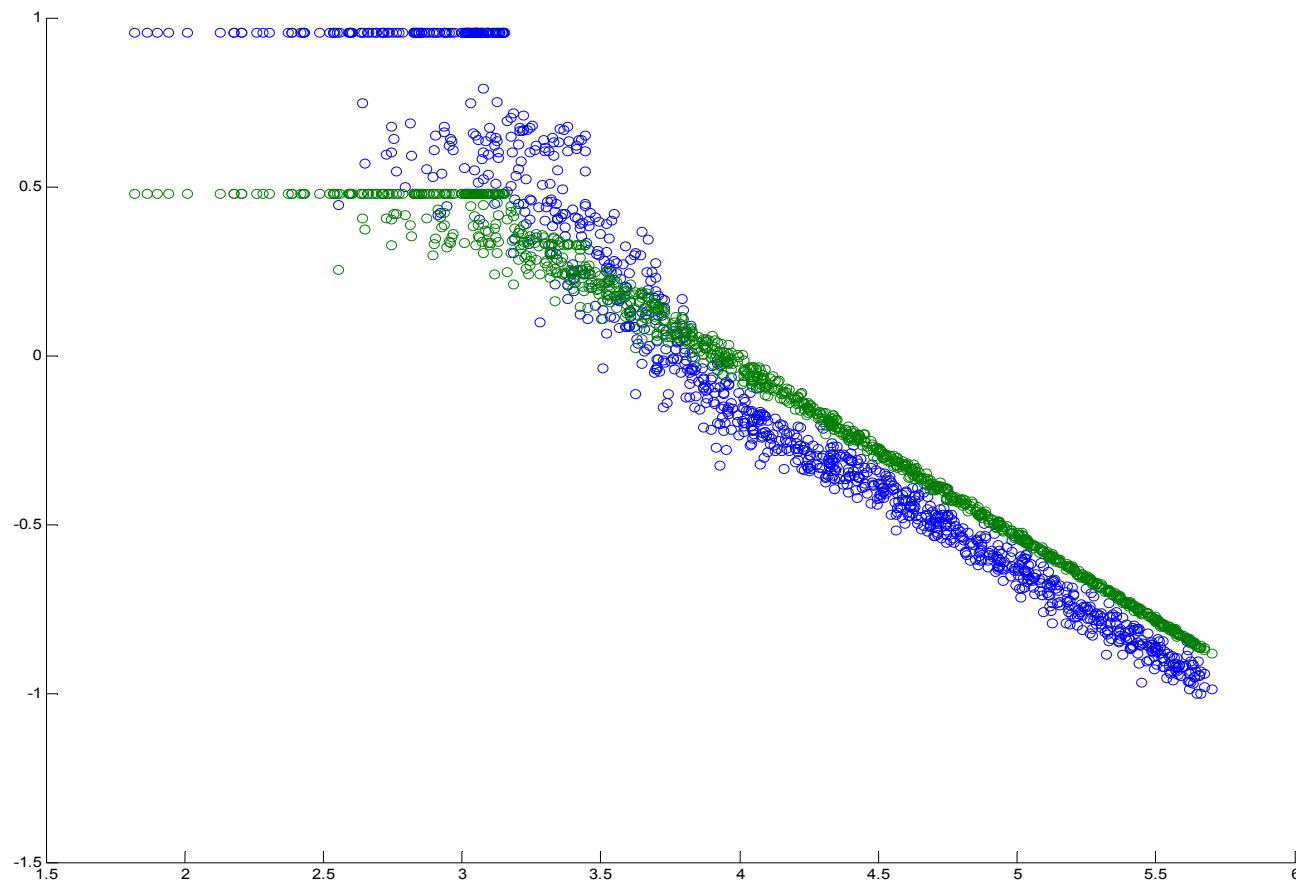
Los comandos utilizados para graficar el diagrama de dispersión de los experimentos son los siguientes:

Para los bytes:

```
for x=1:1095
errorB10(x,:)=abs(((bytes_o(x)/10)-
bytes_10(x,:))/((bytes_o(x))/10));
end

for x=1:1095
mediaB10(x,:)=mean(errorB10(x,find(not(isnan(errorB10(x,:))))));
end

logx=log10(bytes_o);
logyb10=log10(mediaB10);
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectab10(i,1)=log10(sqrt(((1-
0.1)/(0.1*packets_o(i,1))))*coeff(i,1));
end
scatter(logx,logyb10)
hold on
scatter(logx,rectab10)
```



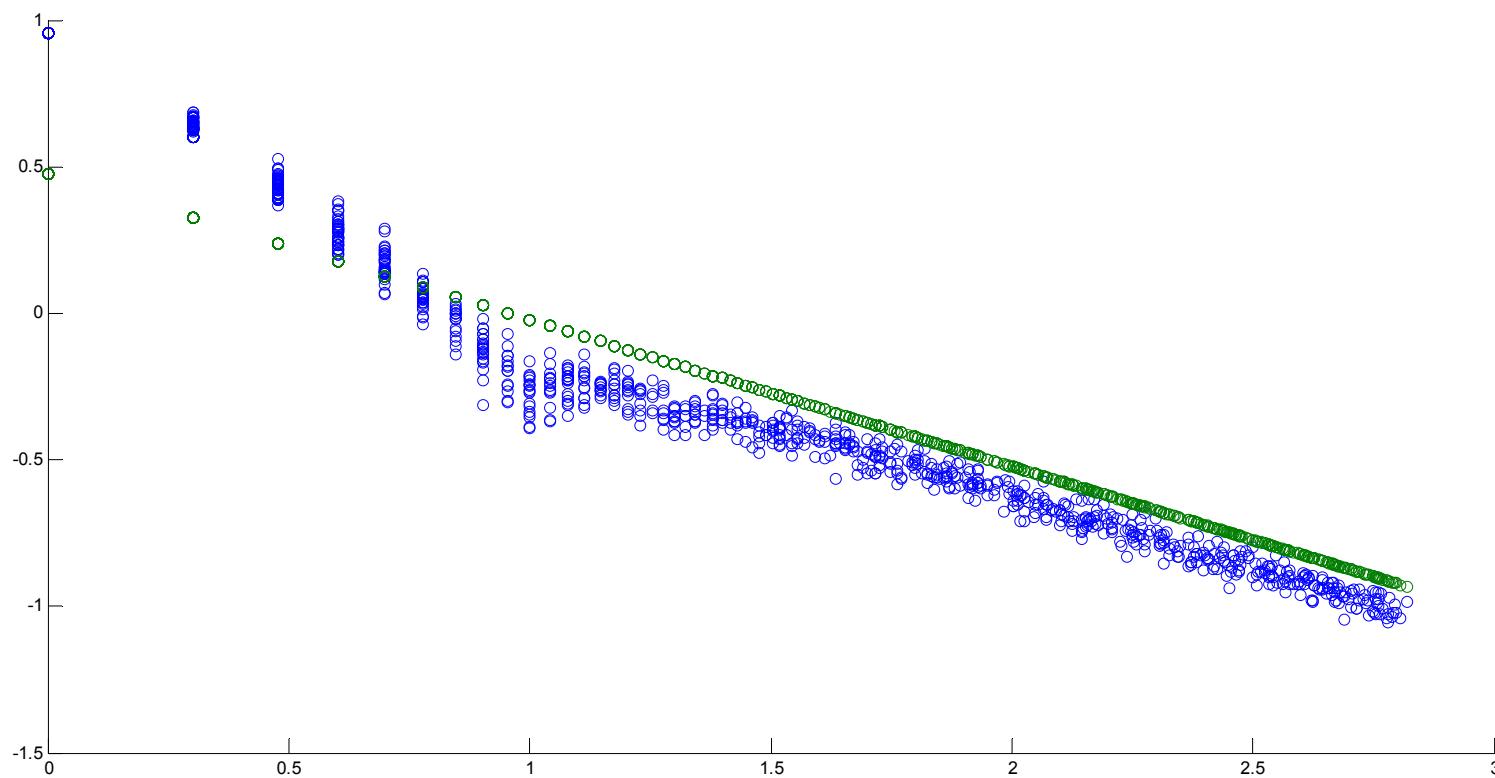
**Fig. A15.1** Gráfica logarítmica de bytes con probabilidad 1 de cada 10 con recta

Para los packets:

```
for x=1:1095
restaP10(x,:)=abs(((packets_o(x)/10)-packets_10(x,:))/((packets_o(x))/10));
end

for x=1:1095
mediaP10(x,:)=mean(restaP10(x,find(not(isnan(restaP10(x,:))))));
end

logxP=log10(packets_o);
logyp10=log10(mediaP10);
for i=1:1095
rectap10(i,1)=log10(sqrt(((1-0.1)/(0.1*packets_o(i,1)))));
end
scatter(logxP,logyp10)
hold on
scatter(logxP,rectap10)
```



**Fig. A15.2** Gráfica logarítmica de packets con probabilidad 1 de cada 10 con recta

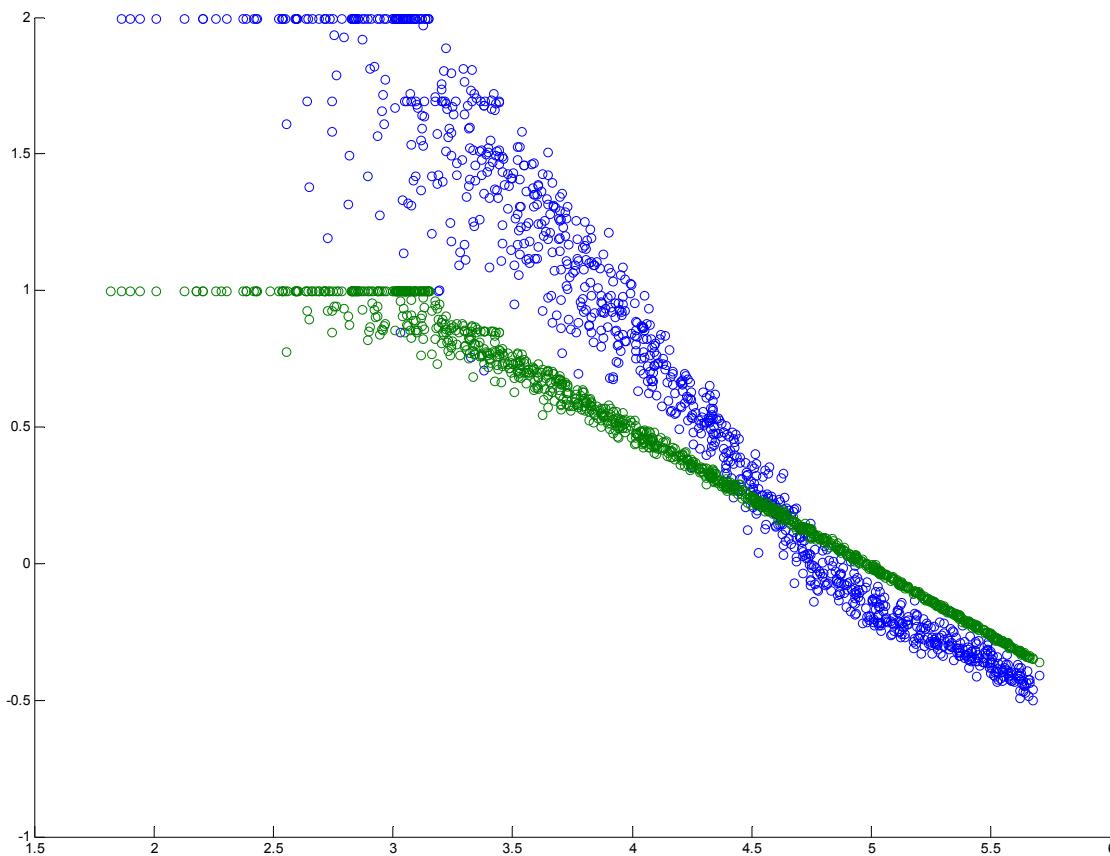
**100 experimentos con probabilidad 1 de cada 100**

Para los bytes:

```
for x=1:1095
errorB100(x,:)=abs(((bytes_o(x)/100)-bytes_100(x,:))/((bytes_o(x))/100));
end

for x=1:1095
mediaB100(x,:)=mean(errorB100(x,find(not(isnan(errorB100(x,:))))));
end

logx=log10(bytes_o);
logyb100=log10(mediaB100);
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectab100(i,1)=log10(sqrt(((1-0.01)/(0.01*packets_o(i,1))))*coeff(i,1));
end
scatter(logx,logyb100)
hold on
scatter(logx,rectab100)
```



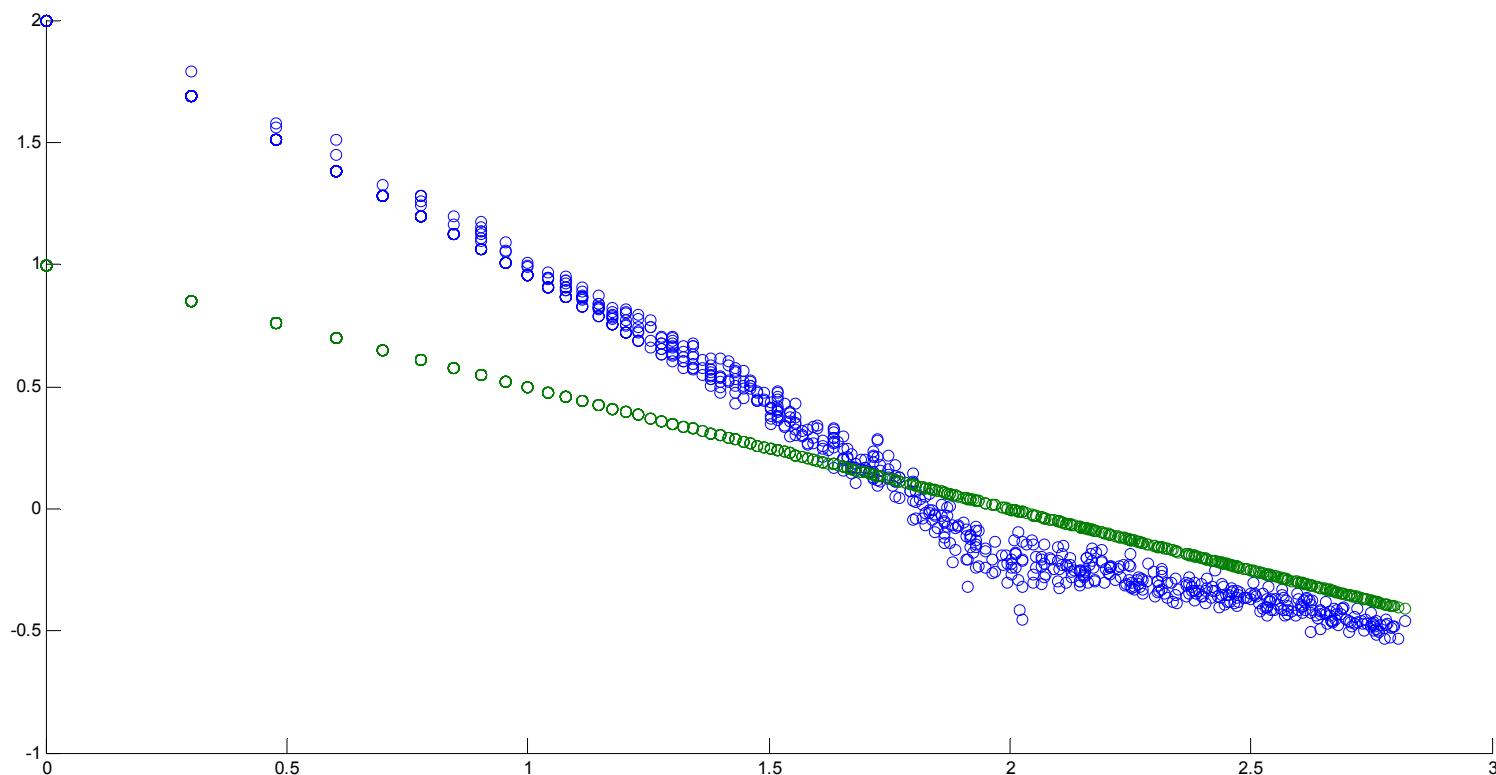
**Fig. A15.3** Gráfica logarítmica de bytes con probabilidad 1 de cada 100 con recta

Para los packets:

```
for x=1:1095
errorP100(x,:)=abs(((packets_o(x)/100)-packets_100(x,:))/((packets_o(x))/100));
end

for x=1:1095
mediaP100(x,:)=mean(errorP100(x,find(not(isnan(errorP100(x,:))))));
end

logxP=log10(packets_o);
logyp100=log10(mediaP100);
for i=1:1095
rectap100(i,1)=log10(sqrt(((1-0.01)/(0.01*packets_o(i,1)))));
end
scatter(logxP,logyp100)
hold on
scatter(logxP,rectap100)
```



**Fig. A15.4** Gráfica logarítmica de packets con probabilidad 1 de cada 100 con recta

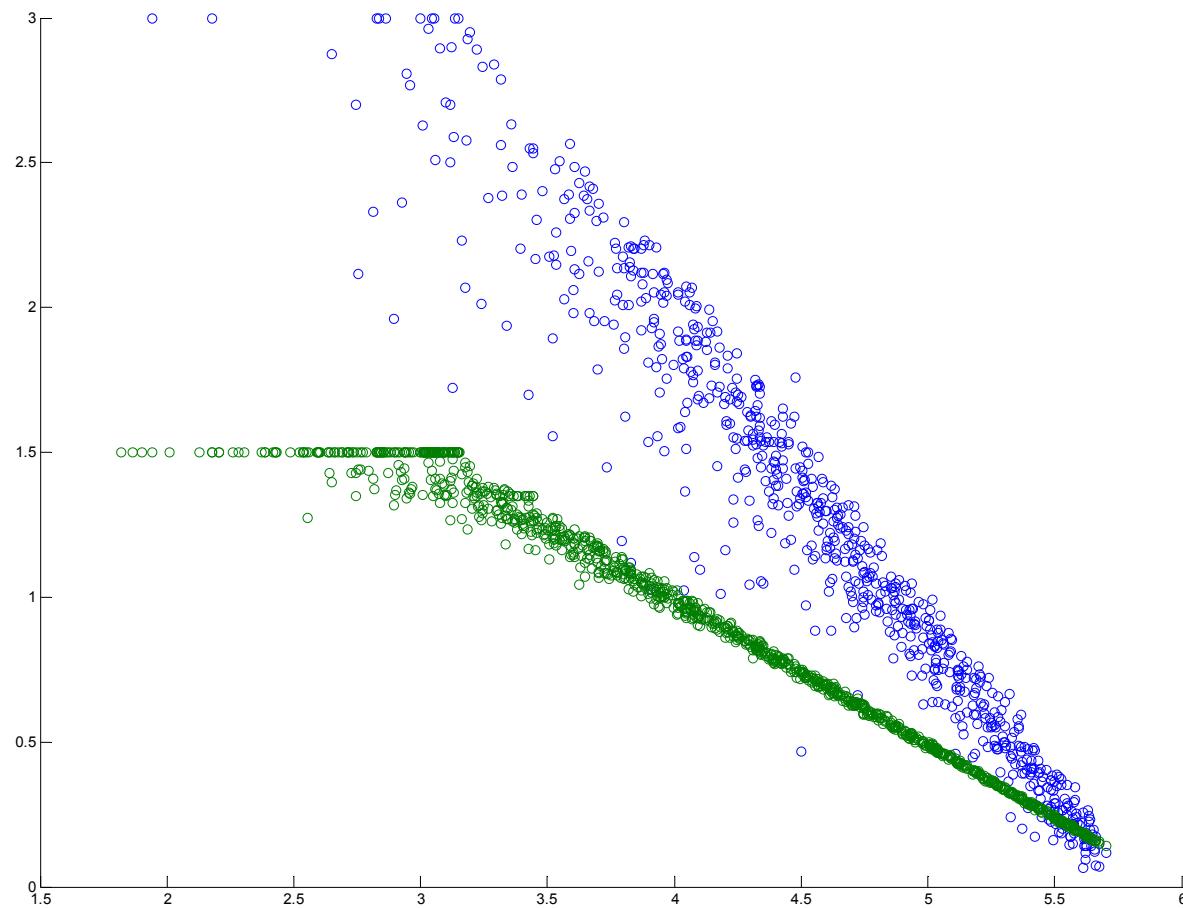
**100 experimentos con probabilidad 1 de cada 1000**

Para los bytes:

```
for x=1:1095
errorB1000(x,:)=abs(((bytes_o(x)/1000)-bytes_1000(x,:))/((bytes_o(x))/1000));
end

for x=1:1095
mediaB1000(x,:)=mean(errorB1000(x,find(not(isnan(errorB1000(x,:))))));
end

logx=log10(bytes_o);
logyb1000=log10(mediaB1000);
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectab1000(i,1)=log10(sqrt(((1-0.001)/(0.001*packets_o(i,1))))*coeff(i,1));
end
scatter(logx,logyb1000)
hold on
scatter(logx,rectab1000)
```



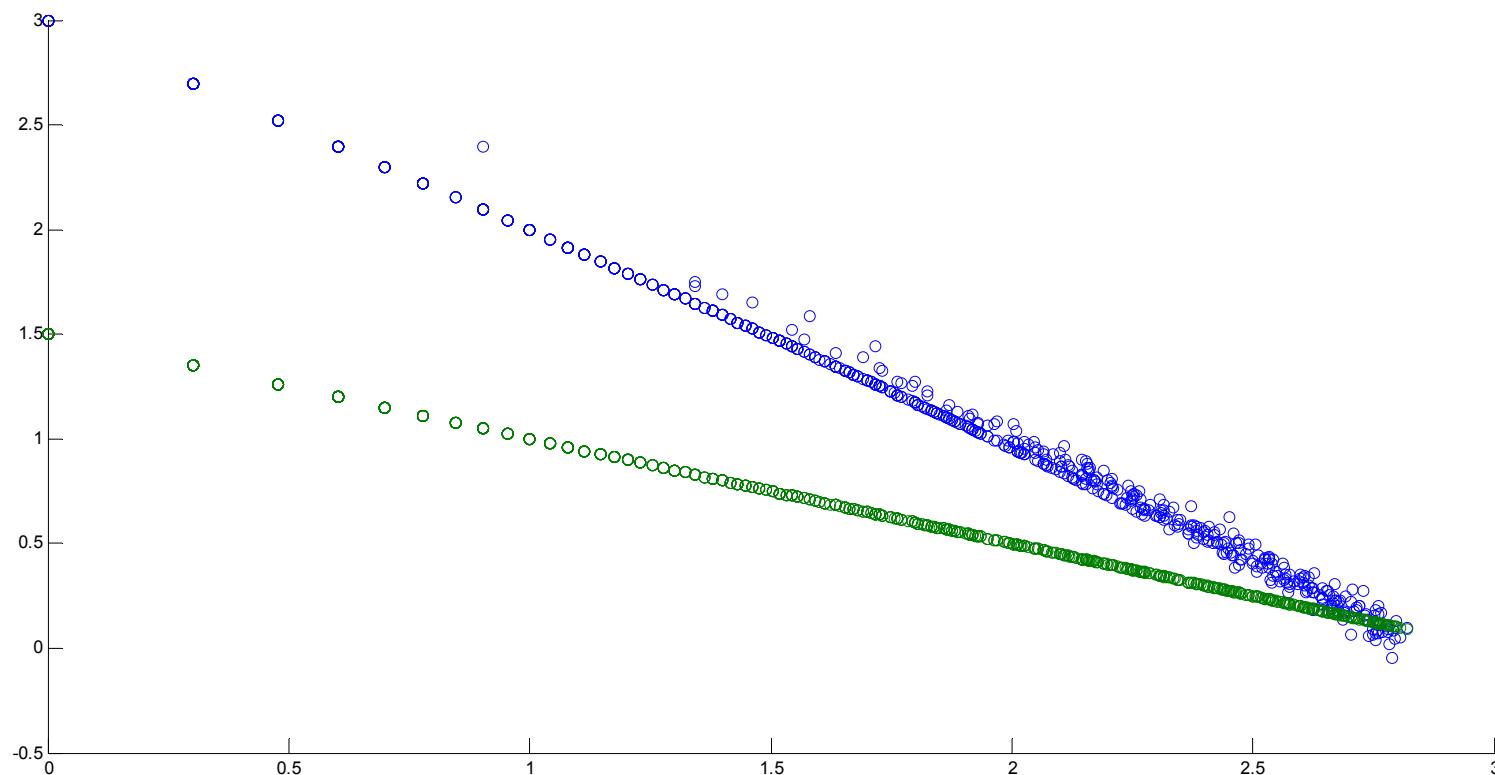
**Fig. A15.5** Gráfica logarítmica de bytes con probabilidad 1 de cada 1000 con recta

Para los packets:

```
for x=1:1095
errorP1000(x,:)=abs(((packets_o(x)/1000)-packets_1000(x,:))/((packets_o(x))/1000));
end

for x=1:1095
mediaP1000(x,:)=mean(errorP1000(x,find(not(isnan(errorP1000(x,:))))));
end

logxP=log10(packets_o);
logyp1000=log10(mediaP1000);
for i=1:1095
rectap1000(i,1)=log10(sqrt(((1-0.001)/(0.001*packets_o(i,1)))));
end
scatter(logxP,logyp1000)
hold on
scatter(logxP,rectap1000)
```



**Fig. A15.6** Gráfica logarítmica de packets con probabilidad 1 de cada 1000 con recta

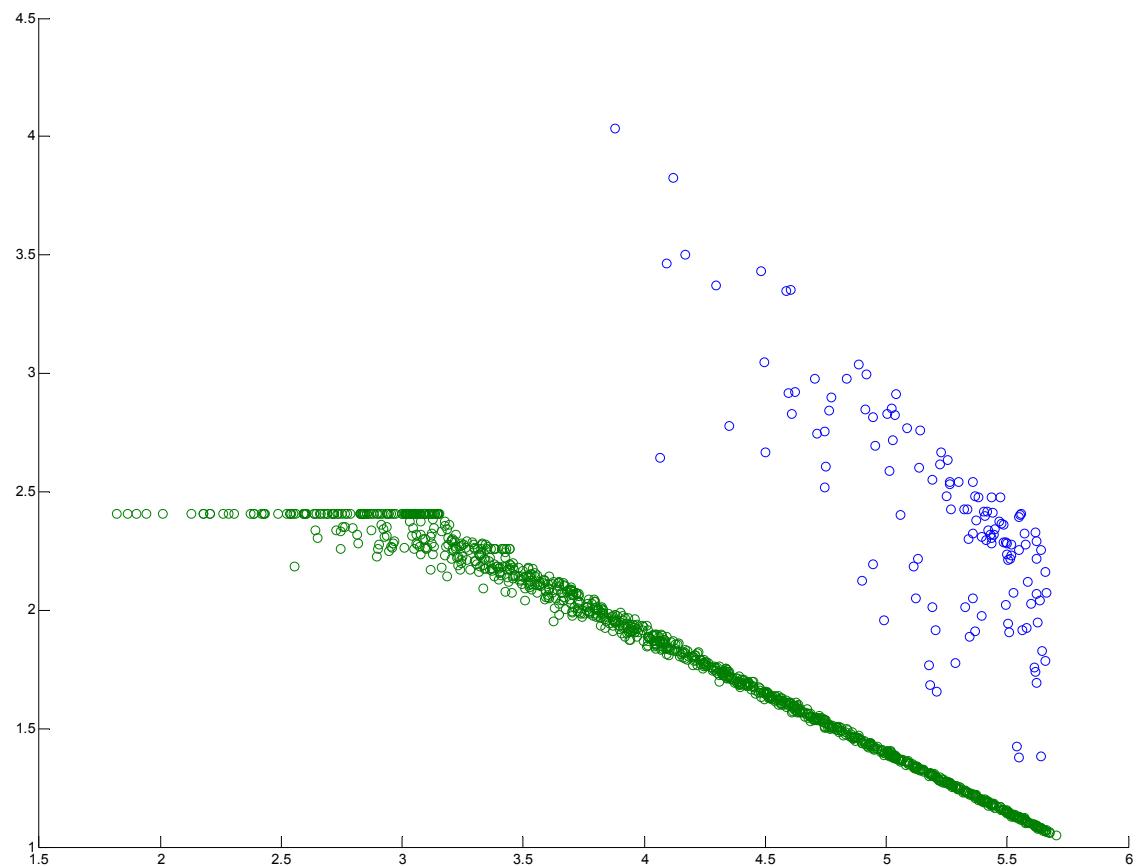
**100 experimentos con probabilidad 1 de cada 65535**

Para los bytes:

```
for x=1:1095
errorB65535(x,:)=abs(((bytes_o(x)/65535)-bytes_65535(x,:))/((bytes_o(x)/65535)));
end

for x=1:1095
mediaB65535(x,:)=mean(errorB65535(x,find(not(isnan(errorB65535(x,:))))));
end

logx=log10(bytes_o);
logyb65535=log10(mediaB65535);
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectab65535(i,1)=log10(sqrt(((1-(1/65535))/((1/65535)*packets_o(i,1))))*coeff(i,1));
end
scatter(logx,logyb65535)
hold on
scatter(logx,rectab65535)
```



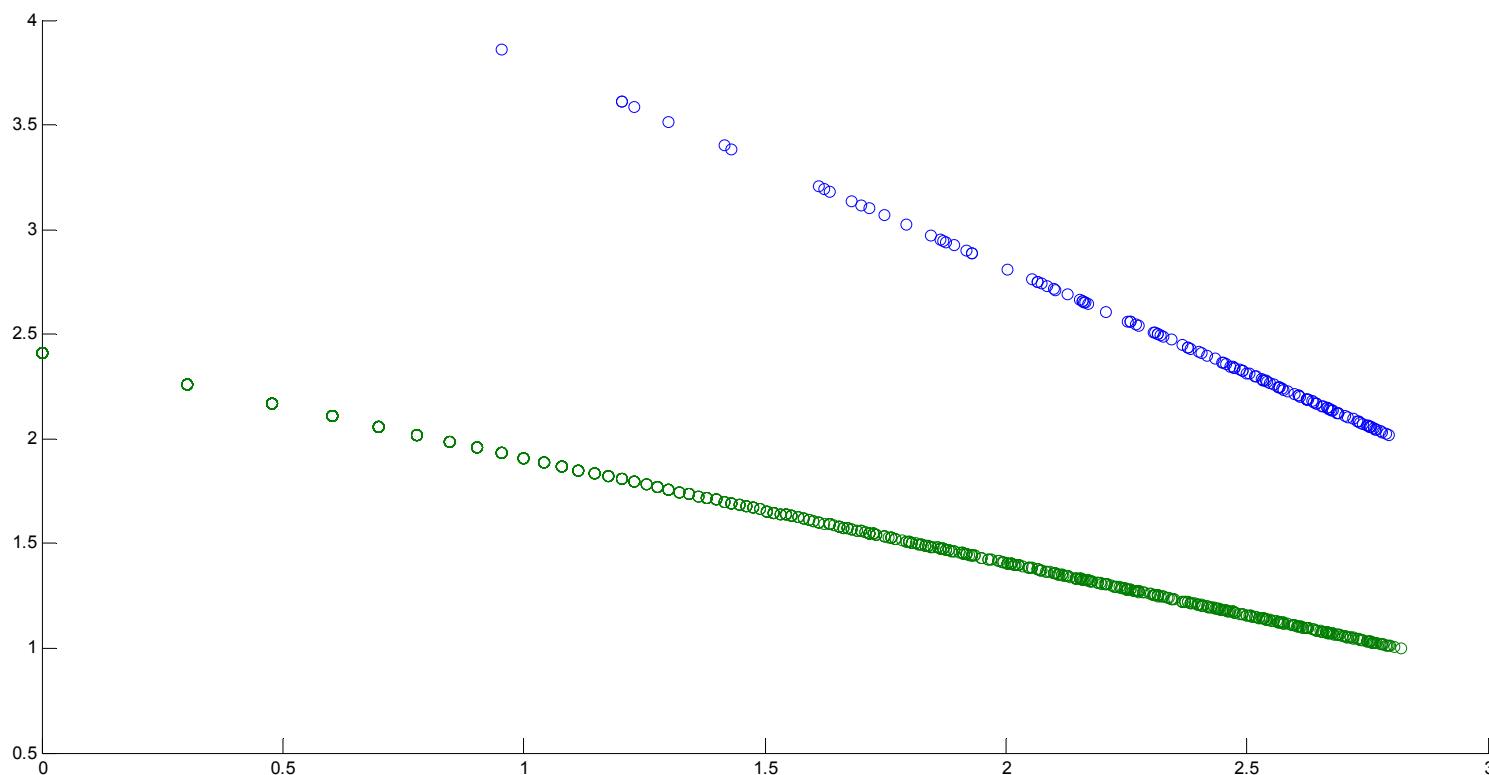
**Fig. A15.7** Gráfica logarítmica de bytes con probabilidad 1 de cada 65535 con recta

Para los packets:

```
for x=1:1095
errorP65535(x,:)=abs(((packets_o(x)/65535)-
packets_65535(x,:))/((packets_o(x))/65535));
end

for x=1:1095
mediaP65535(x,:)=mean(errorP65535(x,find(not(isnan(errorP65535(x,:))))));
end

logxP=log10(packets_o);
logyp65535=log10(mediaP65535);
for i=1:1095
rectap65535(i,1)=log10(sqrt(((1-(1/65535))/((1/65535)*packets_o(i,1)))));
end
scatter(logxP,logyp65535)
hold on
scatter(logxP,rectap65535)
```



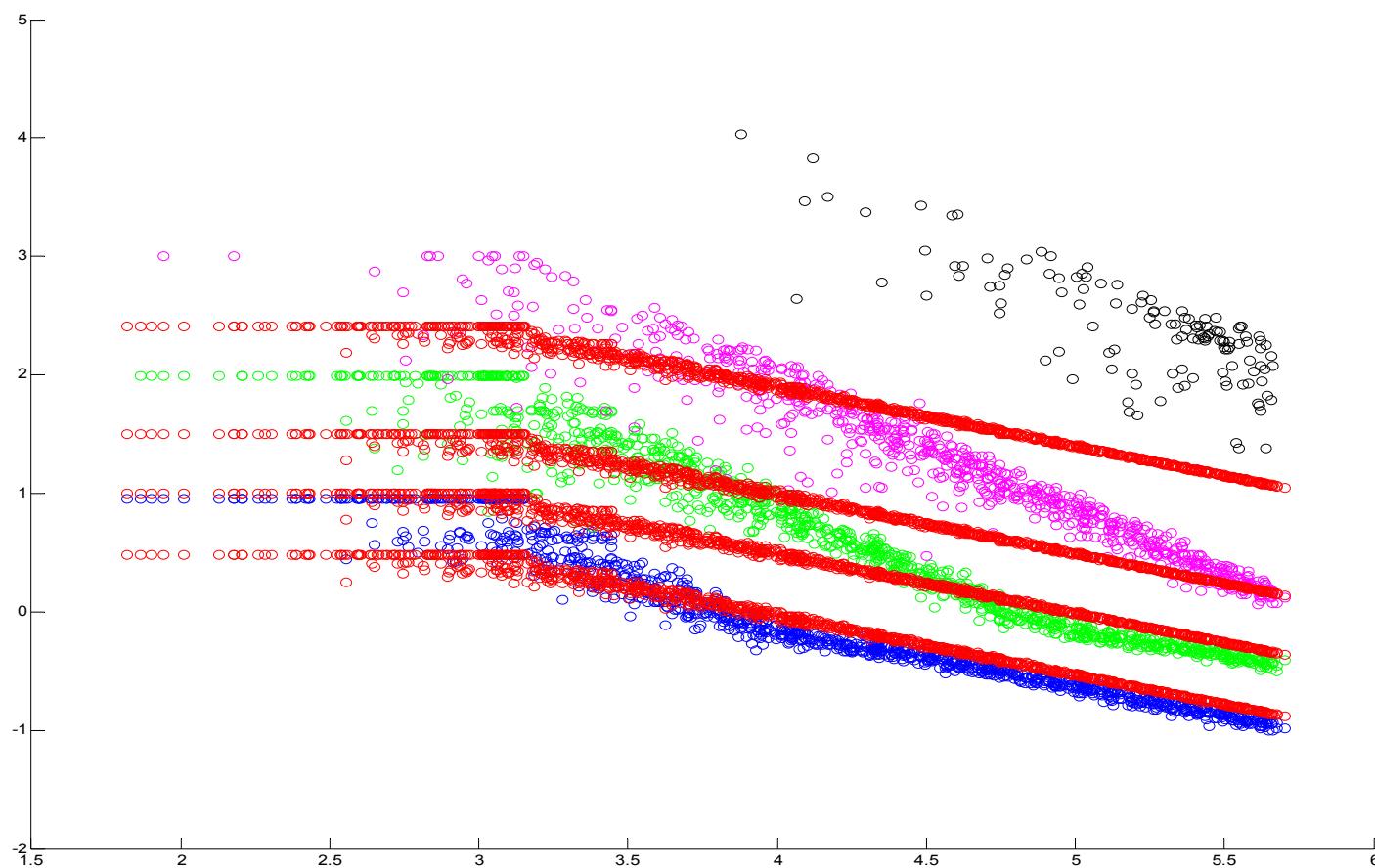
**Fig. A15.8** Gráfica logarítmica de packets con probabilidad 1 de cada 65535 con recta

## Gráfica conjunta

Ahora realizaremos una gráfica conjunta con las cuatro gráficas tanto de bytes como de packets.

Para los bytes:

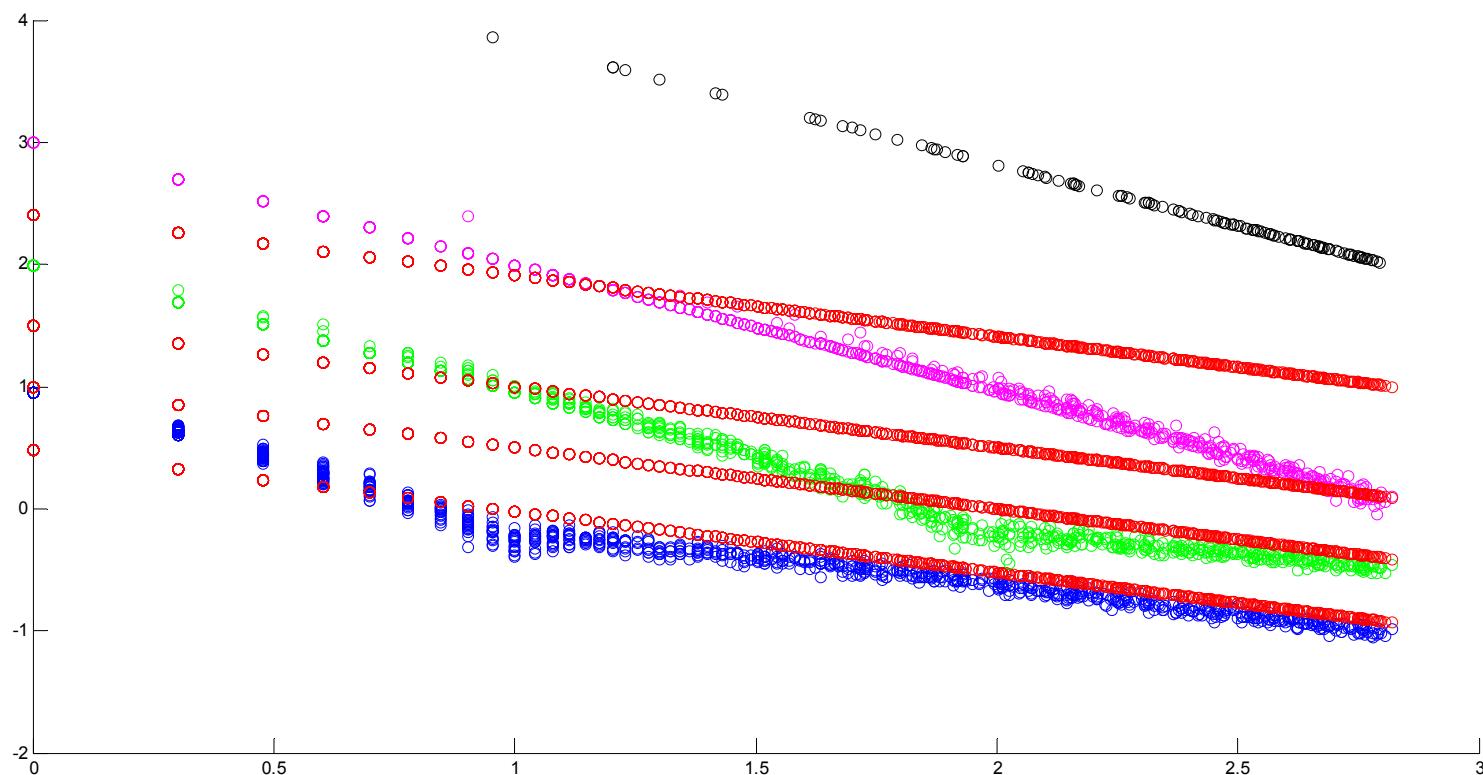
```
scatter(logx, logyb10, [], 'blue')
hold on
scatter(logx, rectab10, [], 'red')
scatter(logx, logyb100, [], 'green')
scatter(logx, rectab100, [], 'red')
scatter(logx, logyb1000, [], 'magenta')
scatter(logx, rectab1000, [], 'red')
scatter(logx, logyb65535, [], 'black')
scatter(logx, rectab65535, [], 'red')
```



**Fig. A15.9** Gráfica logarítmica de bytes con todas las probabilidades

Para los packets:

```
scatter(logxP,logyp10,[],'blue')
hold on
scatter(logxP,rectap10,[],'red')
scatter(logxP,logyp100,[],'green')
scatter(logxP,rectap100,[],'red')
scatter(logxP,logyp1000,[],'magenta')
scatter(logxP,rectap1000,[],'red')
scatter(logxP,logyp65535,[],'black')
scatter(logxP,rectap65535,[],'red')
```



**Fig. A15.10** Gráfica logarítmica de packets con todas las probabilidades

## Diagramas de dispersión con el 80% de los flujos detectados

El estudio del error puede ser más afinado si nos centramos sólo en aquellos flujos que son detectados el 80% de las veces. Esto sólo pasa en las probabilidades 100, 1000 y 65535, ya que en 10 se detectan todos los flujos.

La manera más sencilla de realizarlo es con Excel, generando una condición para cada flujo en busca de las 'X' y realizando una resta de 100 al resultado obtenido. De este modo obtendremos la cantidad de experimentos detectados respecto a 100.

|   | CX1   |       |       |       |       |       |       |       |    |    |     |
|---|-------|-------|-------|-------|-------|-------|-------|-------|----|----|-----|
|   | CO    | CP    | CQ    | CR    | CS    | CT    | CU    | CV    | CW | CX | CY  |
| 1 | 12444 | 15585 | 10203 | 10626 | 20503 | 18216 | 15140 | 20116 |    | 0  | 100 |
| 2 | 20963 | 11410 | 15691 | 21179 | 12202 | 17881 | 13996 | 21918 |    | 0  | 100 |
| 3 | 38936 | 43710 | 39757 | 36496 | 44828 | 47306 | 44117 | 41373 |    | 0  | 100 |
| 4 | 20240 | 28972 | 34621 | 23432 | 22451 | 21594 | 27088 | 17927 |    | 0  | 100 |
| 5 | 13103 | 19278 | 15999 | 14951 | 20294 | 23693 | 15901 | 14671 |    | 0  | 100 |
| 6 | 7824  | 5853  | 1851  | 5466  | 3471  | 6196  | 3461  | 3274  |    | 0  | 100 |
| 7 | 31361 | 15888 | 17008 | 22713 | 25993 | 26462 | 19509 | 23395 |    | 0  | 100 |

|   | CY1   |       |       |       |       |       |       |       |    |    |     |
|---|-------|-------|-------|-------|-------|-------|-------|-------|----|----|-----|
|   | CO    | CP    | CQ    | CR    | CS    | CT    | CU    | CV    | CW | CX | CY  |
| 1 | 12444 | 15585 | 10203 | 10626 | 20503 | 18216 | 15140 | 20116 |    | 0  | 100 |
| 2 | 20963 | 11410 | 15691 | 21179 | 12202 | 17881 | 13996 | 21918 |    | 0  | 100 |
| 3 | 38936 | 43710 | 39757 | 36496 | 44828 | 47306 | 44117 | 41373 |    | 0  | 100 |
| 4 | 20240 | 28972 | 34621 | 23432 | 22451 | 21594 | 27088 | 17927 |    | 0  | 100 |
| 5 | 13103 | 19278 | 15999 | 14951 | 20294 | 23693 | 15901 | 14671 |    | 0  | 100 |
| 6 | 7824  | 5853  | 1851  | 5466  | 3471  | 6196  | 3461  | 3274  |    | 0  | 100 |
| 7 | 31361 | 15888 | 17008 | 22713 | 25993 | 26462 | 19509 | 23395 |    | 0  | 100 |

Fig. A15.11 Resta de X detectadas con 100

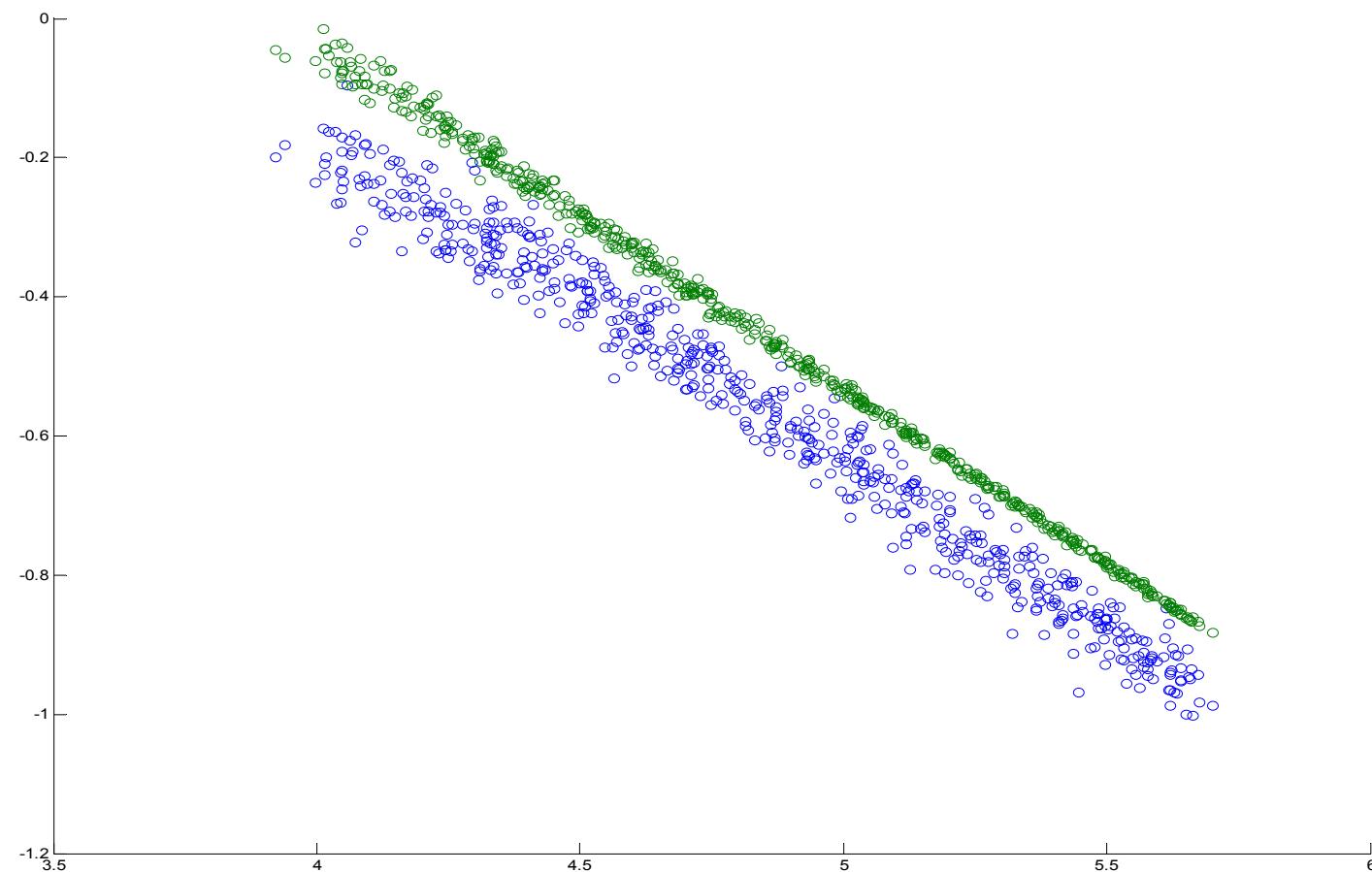
Posteriormente con Matlab, importaremos y buscaremos las posiciones de los que son superiores a 80%, calcularemos de nuevo los logaritmos sólo con los flujos superiores al 80% y los valores del *polyfit* ya que cambiarán. Finalmente se graficará:

```
ochenta10=find(repe_10>=80);
ochenta100=find(repe_100>=80);
```

**100 experimentos con probabilidad 1 de cada 10**

Para los bytes:

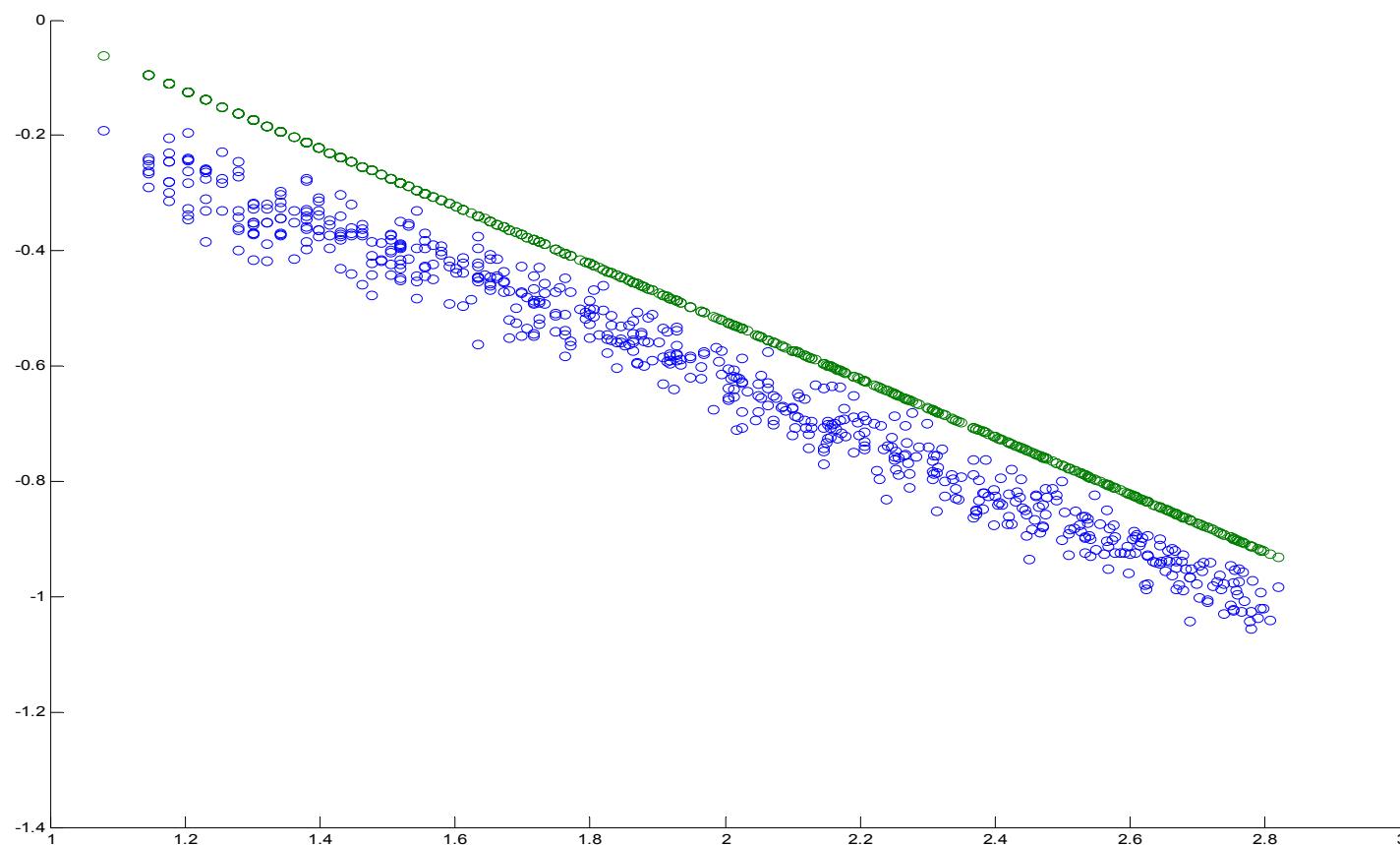
```
logx1080=log10(bytes_o(ochenta10));
logyb1080=log10(mediaB10(ochenta10));
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectab10(i,1)=log10(sqrt(((1-0.1)/(0.1*packets_o(i,1))))*coeff(i,1));
end
scatter(logx1080,logyb1080)
hold on
scatter(logx1080,rectab10(ochenta10))
```



**Fig. A15.12** Gráfica logarítmica de bytes con probabilidad 1 de cada 10 con recta (80%)

Para los packets:

```
logxp1080=log10(packets_o(ochenta10));
logyp1080=log10(mediaP10(ochenta10));
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectap10(i,1)=log10(sqrt((1-0.1)/(0.1*packets_o(i,1))));
end
scatter(logxp1080,logyp1080)
hold on
scatter(logxp1080,rectap10(ochenta10))
```

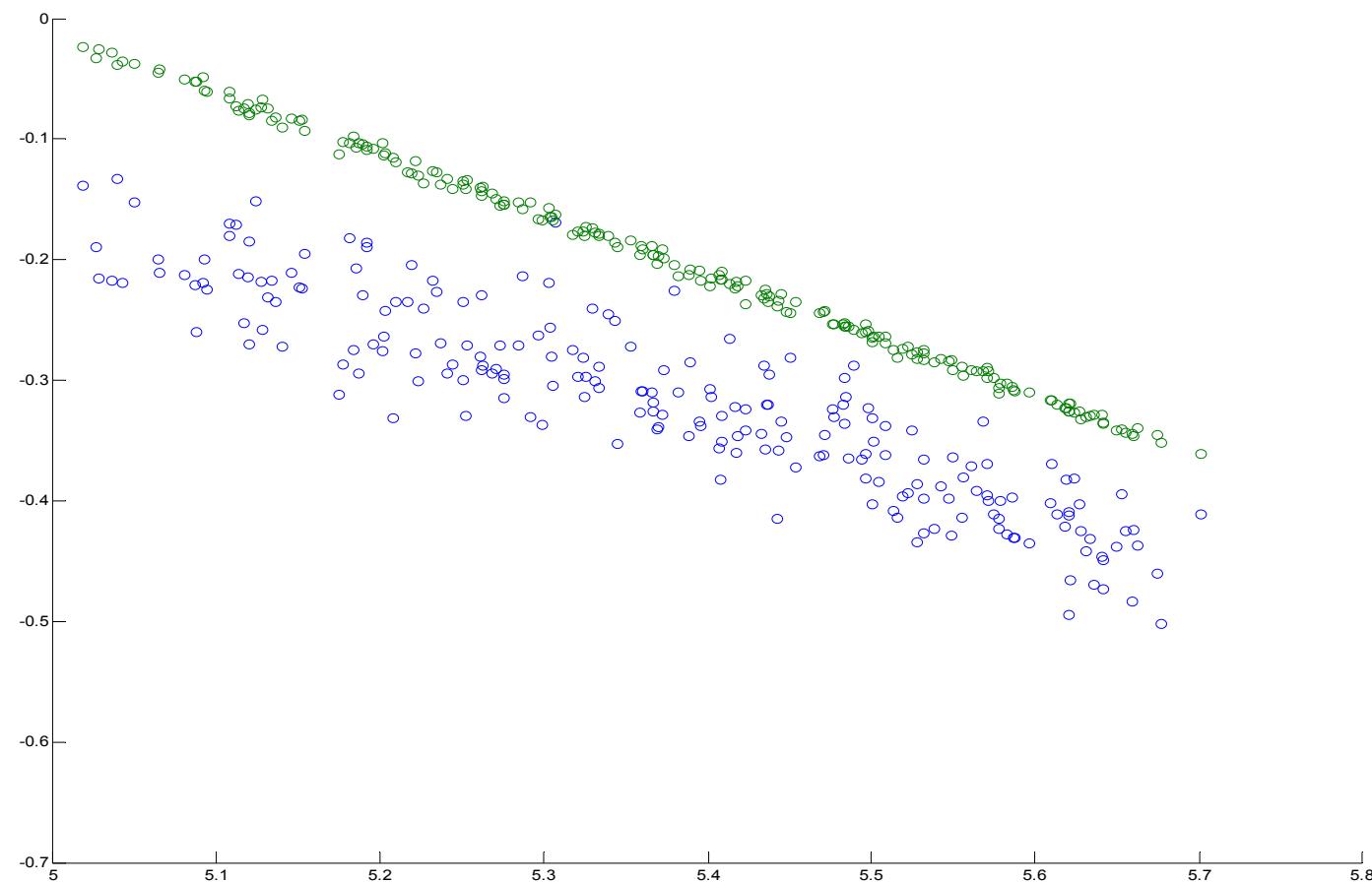


**Fig. A15.13** Gráfica logarítmica de packets con probabilidad 1 de cada 10 con recta (80%)

**100 experimentos con probabilidad 1 de cada 100**

Para los bytes:

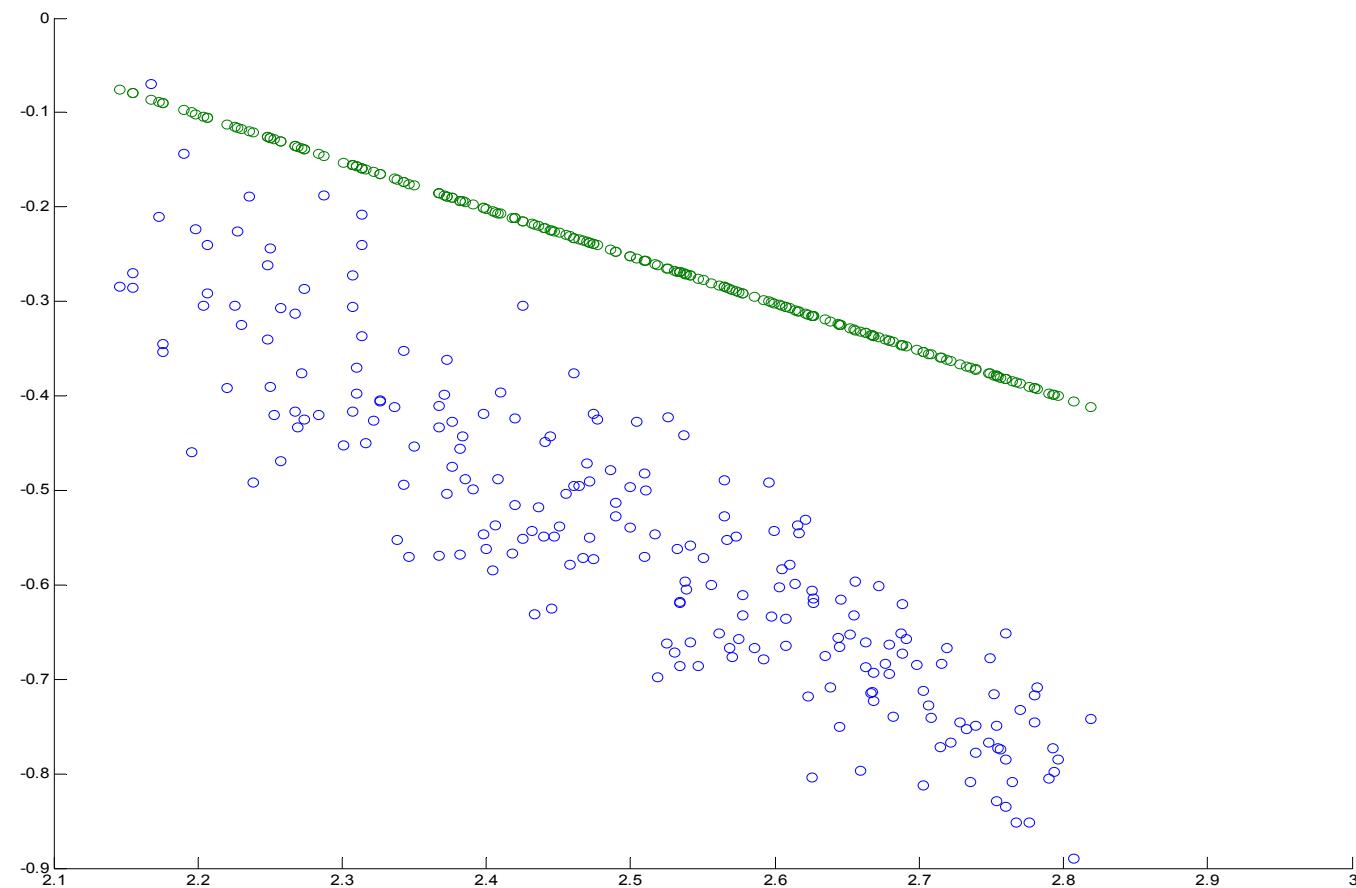
```
logx10080=log10(bytes_o(ochenta100));
logyb10080=log10(mediaB100(ochenta100));
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectab100(i,1)=log10(sqrt(((1-0.01)/(0.01*packets_o(i,1))))*coeff(i,1));
end
scatter(logx10080,logyb10080)
hold on
scatter(logx10080,rectab100(ochenta100))
```



**Fig. A15.14** Gráfica logarítmica de bytes con probabilidad 1 de cada 100 con recta (80%)

Para los packets:

```
logxp10080=log10(packets_o(ochenta100));
logyp10080=log10(mediaP100(ochenta100));
for i=1:1095
coeff(i,1)=sqrt(((append(i,2)).^2/(append(i,1).^2))+1);
rectap100(i,1)=log10(sqrt((1-0.01)/(0.01*packets_o(i,1))));
end
scatter(logxp10080,logyp10080)
hold on
scatter(logxp10080,rectap100(ochenta100))
```



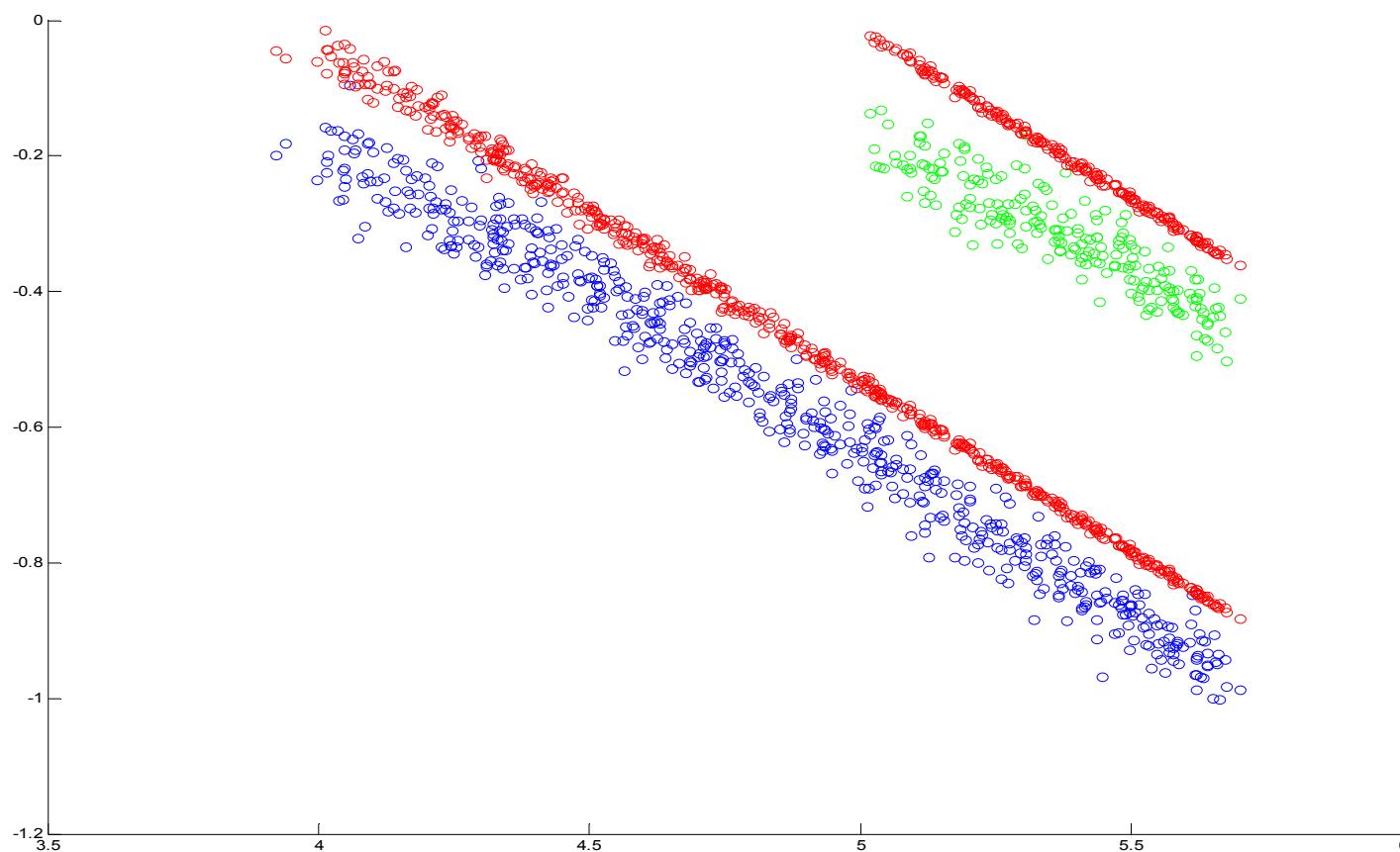
**Fig. A15.15** Gráfica logarítmica de packets con probabilidad 1 de cada 100 con recta (80%)

Los experimentos con probabilidad 1 de cada 1000 y 1 de cada 65535 no es posible graficarlos ya que no existen flujos con una presencia de más del 80%.

### Gráfica conjunta

Para los bytes:

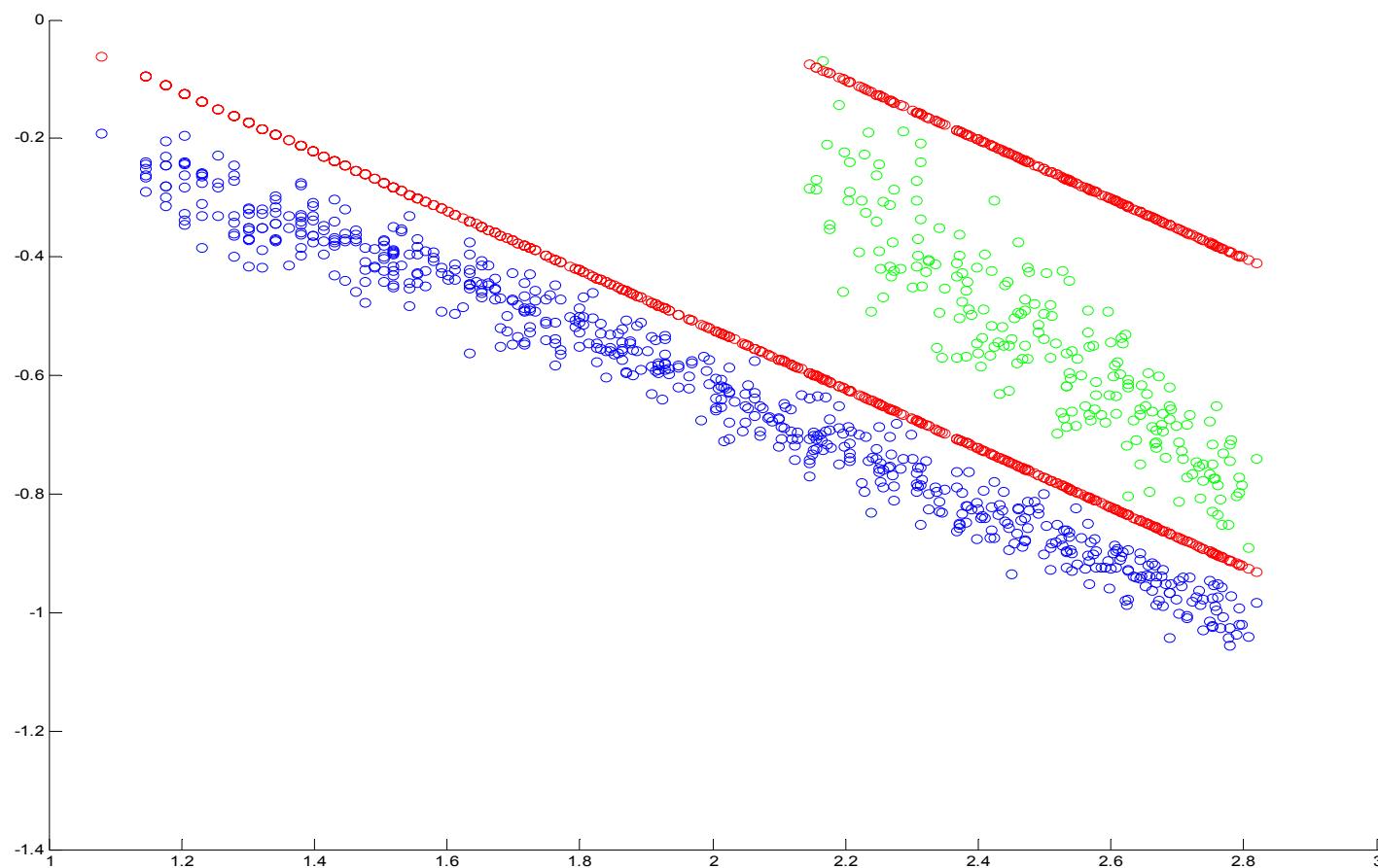
```
scatter(logx1080,logyb1080,[],'blue')
hold on
scatter(logx1080,rectab10(ochenta10),[],'red')
scatter(logx10080,logyb10080,[],'green')
scatter(logx10080,rectab100(ochenta100),[],'red')
```



**Fig. A15.16** Gráfica logarítmica de bytes con todas las probabilidades (80%)

Para los packets:

```
scatter(logxp1080,logyp1080,[],'blue')
hold on
scatter(logxp1080,rectap10(ochenta10),[],'red')
scatter(logxp10080,logyp10080,[],'green')
scatter(logxp10080,rectap100(ochenta100),[],'red')
```



**Fig. A15.17** Gráfica logarítmica de packets con todas las probabilidades (80%)