UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

I.T. Telecomunicación (Sonido e Imagen)

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

“Desarrollo de una aplicación basada en Java para testear el rendimiendo de una red de datos""”

TRABAJO FINAL DE CARRERA

Autor/es:
Carlos Barambones Ferrara

Director/es:
Jaime Lloret Mauri
Juan Ramon Diaz Santos

GANDIA, 2013

Quería agradecer a todo el Departamento de Telemática de la EPSG por el apoyo que me han brindado durante estos últimos meses para finalizar mi proyecto y, en especial, agradecer toda la atención que me han dado a los profesores Jaime Lloret Mauri y Juan Ramón Díaz Santos, para ir resolviendo las múltiples dudas que iban surgiendo, y por su insistencia por mejorar y perfeccionar mi Proyecto Fin de Carrera.

Agradecer también el apoyo incondicional de mi familia, que siempre me han apoyado en los momentos más difíciles para que siguiera esforzándome al máximo y no dejara de luchar por el proyecto. Y por último agradecer a mi chica todo lo que ha aguantado estos últimos meses junto a mí, siempre con una brillante sonrisa para animarme, ya que sin ella no habría podido hacerlo.

Gracias a todos.

TABLA DE CONTENIDOS

1- INTRODUCCIÓN 9
1.1 - OBJETIVOS 13
1.2 – PRECEDENTES DEL PROYECTO 14
1.3 - ESTRUCTURA DEL PROYECTO 15
2 - HERRAMIENTAS DEL MERCADO 16
3 – INTERFAZ GRÁFICA 21
4 – PROGRAMACIÓN DE LA HERRAMIENTA 25
4.1 - DESCRIPCIÓN DE LA AQUITECTURA DEL SISTEMA 25
4.2 - DIAGRAMA LOGICO Y FUNCIONAMIENTO INTERNO 28
4.3 - CABECERAS DEL PROTOCOLO 34
5 - MEDIDAS REALIZADAS 39
5.1 - ESCENARIO DE ESTUDIO 1 42
5.2 - ESCENARIO DE ESTUDIO 2 47
6 - CONCLUSIONES 53
6.1 - CUMPLIMIENTO DEL OBJETIVO 53
6.2 - CONCLUSIONES SOBRE EL PROYECTO 54
6.3 - PROBLEMAS ENCONTRADOS 55
6.4 - APORTACIONES PERSONALES 58
6.5 – FUTURAS LINEAS DE ESTUDIO 58
7 - BIBLIOGRAFÍA 59

ÍNDICE DE FIGURAS

Figura 1. Interfaz Gráfica 22
Figura 2. Modo de Envío 23
Figura 3. Campos de la herramienta 23
Figura 4. Lista desplegable FPS 24
Figura 5. Lista desplegable Audio 24
Figura 6. Cliente-Servidor 26
Figura 7. Diagrama lógico de la aplicación 28
Figura 8. Cabecera Video 34
Figura 9. Cabecera Audio 37
Figura 10. Escenario de Prueba 39
Figura 11. Configuración en el PC1 (Servidor) 39
Figura 12. Configuración en el PC0 (Cliente) 40
Figura 13. Medidas Obtenidas Prueba 41
Figura 14. Escenario de Medida 1 42
Figura 15. Gráfica Medidas 1 44
Figura 16. Escenario de Medida 2 47
Figura 17. Gráfica Medidas 2 49
Figura 18. Gráfica Medidas 3 52

ÍNDICE DE TABLAS

Tabla 1. Tabla de Direcciones 1 43
Tabla 2. Medidas Excel 1 46
Tabla 3. Tabla de Direcciones 2 47
Tabla 4. Medidas Excel 2 51

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

IP: Internet Protocol

RTP: Real Time Protocol

RIP: Routing Information Protocol

IGRP: Interior Gateway Routing Protocol

OSPF: Open Shortest Path First

VOIP: Voice over IP

XLS: Microsoft Excel file format

ICMP: Internet Control Message Protocol

API: Application Programming Interface

QoS: Quality of Service

MTU: Maximum Transfer Unit

VoD: Video on Demand

IPTV: Internet Protocol Television

WiFi: Wireless Fidelity

RMON: Remote Network MONitoring

IETF: Internet Engineering Task Force

IBM: International Business Machines

NTP: Network Time Protocol

BGP: Border Gateway Protocol

GPL: General Public License

WEP: Wired Equivalent Privacy

ARP: Address Resolution Protocol

VLAN: Virtual Local Area Network

OSI: Open System Interconnection

DNS: Domain Name System

IGMP: Internet Group Management Protocol

DLPI: Data Link Provider Interface

BSD: Berkeley Software Distribution

STCP: Stream Control Transmission Protocol

IDS: Intrusion Detection System

IPS: Intrusion Prevention System

1- INTRODUCCIÓN

Estamos viviendo en una época en la que los sistemas de comunicación como la televisión o la telefonía están migrando a un formato digital. La digitalización de estos sistemas proporciona una optimización en las tasas binarias requeridas para su funcionamiento, propiciando la oferta de estos contenidos en otras plataformas como Internet. Internet proporciona una interactividad y unas condiciones económicas al usuario inigualables por otras plataformas convencionales.

Existen ya varios servicios que ofertan contenido de video digital mediante Internet, como VoD o IPTV, y también servicios de telefonía, mediante la tecnología VoIP.

Los códec más utilizados para la codificación de video digital son H.262 [1], H.264 [2] y próximamente H.265 [3]. Estos códec utilizan diversas técnicas para reducir la tasa binaria necesaria para reproducir el video como la redundancia espacial (Suavizando las transiciones de color y luminosidad), temporal (Codificando solo los pixeles en movimiento) y psico-visual (Dando preferencia la codificación del brillo a la del color o codificando con más bits las señales de altas frecuencias que las de bajas frecuencias).

Respecto a la telefonía, se está produciendo un aumento del uso de VoIP respecto a la telefonía analógica convencional. Esto es debido principalmente al menor coste económico que ofrece VoIP ya que en Internet el usuario paga por tener una conexión de determinadas características con la que puede hacer un uso ilimitado (en términos de tiempo de uso), a diferencia de la telefonía analógica que se paga por el tiempo de utilización de la red telefónica. De hecho ya hace tiempo que existen programas para el ordenador que ofrecen llamadas telefónicas totalmente gratuitas, como Skype.

Los códec más utilizados en la transmisión de VoIP son G.711 [4], G.729 [5], G.726 [6] y G.728 [7], los cuales se valen de técnicas de redundancia auditiva, como el enmascaramiento frecuencial y temporal, para conseguir una menor bitrate, incluso en algunos casos dejando sin codificar también los silencios en la comunicación.

Estos servicios son solo el comienzo de una era en la que todo el contenido multimedia se obtendrá desde la red. Es por ello que las herramientas que permitan el testeo de la red, informando de la calidad de servicio (QoS) que esta ofrece, serán indispensables en un futuro.

Las herramientas de monitorización de la red ofrecen al usuario información acerca del estado de esta, como información acerca de posibles fallos o congestión. En ocasiones es necesario probar que ocurre en la red cuando existe un fallo y cuánto tiempo tarda en recuperarse ante eventos inesperados, pero desde el punto de vista del usuario final. Para obtener esta información, los administradores suelen recurrir a herramientas de gestión de red como Remote Network MONitoring (RMON) [8] que fue desarrollado por el IETF, con el objetivo de ayudar en la monitorización y análisis de los protocolos en la red de área local. Pero esta solución no permite hacer pruebas del tráfico que se distribuye dentro de ésta (por ejemplo, el tiempo que tarda en transmitirse la información entre dos dispositivos finales, el tiempo que un dispositivo final está sin recibir datos, etc.), sólo reciben la información de los dispositivos de la red. Sin embargo en ocasiones es necesario obtener los valores de los parámetros tal como se obtienen en el receptor.

Existen artículos relacionados con la toma de medidas de calidad sobre redes como “End-to-end Delay in Mobile Networks: Does the Traffic Pattern Matter?”, realizado por Markus Laner en Alemania [9], en el cual el autor realiza medidas de calidad, en este caso sobre redes móviles; o “Medida de la calidad de voz en redes IP” realizado por José Joskowicz y Rafael Sotelo en Uruguay sobre redes de VoIP [10].

Dada la necesidad de herramientas que permitan probar el rendimiento de una red desarrollada desde el punto vista del usuario final, en este proyecto presentamos una nueva herramienta que permite realizar test de rendimientos de la red, informar sobre el estado de la misma y medir los parámetros que afectan a la calidad de servicio.

Pero para la realización de una herramienta de estas características se requiere tener en cuenta cuales son los principales parámetros de calidad que afectan a nuestra red, para posteriormente, implementar su sistema de cálculo en el núcleo de la aplicación. Nuestra herramienta será capaz de medir Delay, Jitter y Paquete Perdidos:

El Delay o Latencia es el retardo que se produce desde el envío de un paquete hasta la recepción del mismo. Cuanto mayor capacidad tenga la red, menor delay ofrecerán los paquetes y por lo tanto la calidad de servicio será mejor.

El Jitter es la diferencia entre el retardo de un paquete y el consecutivo. Se mide en valor absoluto e indica la variación de calidad de la red debido a una posible congestión.

Los Paquetes Perdidos en una red influyen mucho en la calidad de servicio, produciendo "pixelado" en Video digital o una comunicación ininteligible en VoIP.

Todos estos parámetros mencionados se basan para su cálculo en el envío de paquetes a la red. Para ello se sirven de distintos protocolos como el protocolo IP (Capa de Red), el protocolo UDP (Capa de Transporte) y el protocolo RTP (Capa de Aplicación).

IP. Protocolo que proporciona el envío de datagramas no orientados a conexión desde un origen a un destino. Su cabecera incluye la dirección IP origen y destino [11].

UDP. Protocolo diseñado para el envío de datagramas sin la seguridad de recepción de estos en el destino. Su cabecera incluye la dirección del puerto origen y destino [12].

RTP. Protocolo diseñado para el envío de paquetes multimedia en la red. Destacar los campos de Timestamp y número de Secuencia de la cabecera, necesarios para la reconstrucción del archivo y su reproducción [13].

Para la realización de una herramienta de testeo, que permita el envío de trafico multimedia a la red, se deberá utilizar o simular los protocolos mencionados.

El lenguaje de programación basado en Java, proporciona recursos que se adaptan perfectamente a lo que necesitamos, como una interfaz dinámica y sencilla para el usuario en la que se pueden introducir graficas en tiempo real.

Existen API's como JFreeChart [14] que es un software open source para Java, el cual permite la creación de gráficos complejos de forma simple. Esto permite al usuario de la herramienta observar los datos obtenidos en tiempo real y de forma intuitiva.

Existen también otras API's como Java Excel API [15] que permite a los desarrolladores Java leer, escribir y modificar dinámicamente hojas de cálculo de Microsoft Excel. Esto es muy útil para registrar y almacenar datos obtenidos por la herramienta para un futuro uso de ellos.

Java también ofrece la posibilidad de envío de datagramas UDP mediante la librería Net y la utilización de los sockets. Los sockets permite el envío de datagramas a través de un puerto designado por el usuario previamente. Es una característica básica para una herramienta de estas características.

Y por último, destacar la capacidad de Java de realizar "Multithreading", que es la característica que permite realizar varias tareas simultáneas. Esta característica es fundamental ya que la herramienta requeriría enviar datagramas, recibir datagramas, actualizar tablas y dibujar las gráficas, todo al mismo tiempo. Esta característica hace uso del concepto de "Thread" como un flujo de ejecución dentro de un proceso, en la que los thread no pueden funcionar sin un proceso padre supervisándolos.

Por todas estas características mencionadas nos hemos decantado por Java como lenguaje de programación para nuestra herramienta de testeo de la red.

1.1 - OBJETIVOS

La proliferación de las redes de datos en entornos empresariales, administraciones públicas e incluso en los hogares, ha provocado que a día de hoy sea inconcebible estar en un lugar sin conectividad con otros dispositivos. Además, el hecho de existir múltiples fabricantes de electrónica de red con un gran abanico de modelos, hace que sea muy complicado realizar un test de rendimiento de red cuando ya está implementada.

Una de las mayores necesidades que hay en las redes de datos es la falta de herramientas que permitan comprobar el rendimiento de la red cuando está funcionando correctamente así como su evolución cuando existen fallos.

El objetivo principal del proyecto es desarrollar una herramienta que permita el estudio de los distintos parámetros QoS sobre una topología de red. El desarrollo de esta herramienta conlleva a alcanzar varios objetivos secundarios como:

- Permitir realizar el estudio de la red en tiempo real, mediante el uso de graficas estadísticas o tablas de datos que permitan al usuario ver de manera intuitiva la calidad de servicio en cada momento.

- Medir los distintos parámetros que afectan a la QoS (Jitter, Delay, Paquetes perdidos,…) de una red, soportando el envío tanto de tráfico de Video como el de Audio.

- Dar la posibilidad al usuario de almacenar los datos registrados en las tablas, en archivos .XLS de Microsoft Excel, para tener un fácil manejo de ellos en estudios posteriores.

Una vez alcanzado el objetivo principal, que es el del desarrollo de la herramienta, realizaremos también dos estudios para comprobar su buen funcionamiento:

- Observar, dada una topología, como varían los parámetros QoS si las condiciones de los enlaces entre los ordenadores implicados en el envío del tráfico se saturan.

- Medir, dada una topología, el tiempo de convergencia cuando una ruta principal entre los ordenadores falla y el sistema elige una ruta secundaria entre ambos para el envío de los datos. Al mismo tiempo se compararan las medidas de calidad obtenidos, ya que la ruta principal y la ruta secundaria tendrán distinto número de saltos.

Estos serán los objetivos a alcanzar marcados para este proyecto.

1.2 – PRECEDENTES DEL PROYECTO

Para este proyecto, en un principio se buscó información sobre el desarrollo de herramientas similares en la Escuela Politécnica Superior de Gandía. Pero después de realizar la búsqueda, no se han encontrado precedentes de trabajos realizados con la finalidad del desarrollo de una herramienta para medir y registrar parámetros de la calidad de servicio que ofrece una red de datos.

Lo más relacionado que hemos encontrado son proyectos con la finalidad de estudio de calidad en redes de datos, como por ejemplo, “Configuración y test de rendimiento de una red telefónica inalámbrica “realizado por Tarek Tahrichi Pardo en el año 2009 o también “Evaluación de la latencia en los dispositivos de red”, realizado por Pedro Juan Marza también en el mismo año, ambos bajo la tutela del profesor Jaime Lloret Mauri de la Escuela Politécnica Superior de Gandía

1.3 - ESTRUCTURA DEL PROYECTO

En el Capítulo 2, titulado como "Herramientas existentes", veremos todos los productos s que hemos encontrado relacionados con nuestro proyecto, el origen de dichas herramientas y sus características.

El Capítulo 3, llamado "Interfaz Gráfica", nos hará una breve explicación sobre la interfaz gráfica de nuestra herramienta, las funciones que ofrece al usuario y sus características.

En el Capítulo 4, denominado "Programación de la herramienta", explicaremos las fórmulas utilizadas en el desarrollo de la aplicación con Java y expondremos un diagrama lógico debidamente explicado, para que el usuario pueda comprender como funciona internamente la aplicación. Dedicaremos también un apartado a las cabeceras que utilizaremos en la herramienta para el envío de paquetes.

En el Capítulo 5, titulado como "Medidas Realizadas" expondremos los diferentes estudios que propusimos para la realización de medidas en el laboratorio con la herramienta, los equipos utilizados y los resultados obtenidos argumentando si son como esperábamos.

En el Capítulo 6 desarrollaremos las "Conclusiones" obtenidas del proyecto, y expondremos tanto los problemas encontrados como los objetivos alcanzados. También se incluirán las aportaciones personales y las líneas de estudio futuras del proyecto.

Por último, el Capítulo 7 nos mostrara la "Bibliografía" utilizada en el proyecto.

2 - HERRAMIENTAS EXISTENTES

En la actualidad existen algunas herramientas que permiten generar tráfico en la red. Pero ninguna de estas tiene una aplicación en el cliente final que permita ver que ha ocurrido con el tráfico tras haber atravesado la red. El primer tipo de herramientas que hemos encontrado que se asemejan a la desarrollada por nosotros son los generadores de tráfico. A continuación se muestran algunas de las más comunes.

Mike Ricketts, ingeniero de software de IBM, dentro del proyecto Purple, creó SendIP [16]. SendIP es una herramienta con gran número de opciones, que se ejecuta en línea de comandos y permite enviar paquetes de red de manera arbitraria. Además, las opciones permiten especificar el contenido de cada encabezado de una NTP, BGP, RIP, TCP, UDP, ICMP o paquetes IPv4 e IPv6. Sólo se puede ejecutar en Linux y tiene licencia GPL. La gran desventaja es que la última vez que fue actualizada, fue en el 2003.

En 2003, W. Feng y otros presentaron TCPivo [17]. Es una herramienta que proporciona una alta velocidad de repetición de paquetes desde un archivo de rastreo. TCPivo es capaz de reproducir con precisión trazas de red a alta velocidad utilizando el hardware estándar de un PC. Este software es de código abierto y actualmente está desarrollado exclusivamente en Linux.

Rude&Crude es un conjunto de programas desarrollados en Linux que se distribuye bajo la licencia GPL V2 [18]. Rude es un programa pequeño y flexible, que genera tráfico en la red. Estos pueden ser recibidos y registrados en el otro lado de la red con el programa Crude. Actualmente, estos programas solo pueden generar y medir el tráfico UDP.

Scapy [19] es un programa que permite manipular paquetes. Es capaz de crear o decodificar paquetes de muchos protocolos, satisfacer peticiones y respuestas, y mucho más. Es capaz de realizar acciones más clásicas como escanear, traza de rutas, sondeo, pruebas sobre un solo destino, ataques o descubrimiento de la red. También es capaz de hacer otras acciones que la mayoría de otras herramientas no pueden manejar, como el envío de tramas no válidas, la inyección de tramas 802.11, o combinar técnicas como VLAN hopping + ARP envenenamiento de caché, VOIP decodificación de canal cifrado WEP, etc.

PKTgen [20] es una herramienta de pruebas de alto rendimiento incluido en el propio kernel de Linux. Ser parte del kernel es una de las mejores maneras de probar el proceso de transmisión del controlador de nuestra tarjeta de red. PKTgen también se puede utilizar para generar paquetes ordinarios con el objetivo de probar otros dispositivos de la red. Es una herramienta bastante utilizada para probar encaminadores o puentes que utilizan la pila protocolos de red de Linux, como por ejemplo generar altas tasas de paquetes con el objetivo de saturar los dispositivos.

Joel. E Sommers y otros, de la universidad de Wisconsin crearon un conjunto de 5 programas denominado Harpoon [21]. Harpoon es un generador de tráfico que trabaja en las capas de transporte y sesión (atendiendo al modelo de referencia OSI) que es capaz de medir el flujo de datos en la red. Éste utiliza un conjunto de parámetros de distribución (temporales y espaciales) que pueden extraerse automáticamente de las trazas Netflow para generar flujos. Se puede utilizar para generar tráfico de fondo, para una aplicación o protocolo de prueba, o para probar el hardware de conmutación de red. Harpoon está formado por una combinación de cinco modelos de distribución para las sesiones TCP: tamaño de fichero, tiempo de interconexión, rangos IP origen y destino, número de sesiones activas. Hay tres modelos de distribución para sesiones UDP: velocidad de bits constante, periódica y exponencial. Cada una de estas distribuciones se puede configurar manualmente o de manera automática.

Nemesis [22], desarrollada por Jeff Nathan, es una aplicación capaz de enviar la información que se quiera en una red utilizando TCP/IP. Esta aplicación es muy utilizada para probar y depurar los sistemas de detección de intrusiones de red, cortafuegos, etc. Es una herramienta habitual a la hora de auditar redes y servicios. Nemesis puede crear e inyectar ARP, DNS, ETHERNET, ICMP, IGMP, IP, OSPF, RIP, TCP y UDP. Ha sido desarrollado para Linux y Windows. La versión de Windows requiere la instalación previa de Winpcap , mientras la versión de Linux requiere libnet 1.0.2a.

Packet Excalibur [23] es un motor de paquetes de red multiplataforma, trabaja con entorno gráfico y scripts con extensibles descripciones de protocolo basados en texto. Es una herramienta de red diseñada para crear y recibir paquetes personalizados de la red. Además, es capaz de rastrear y detectar paquetes falsos (generador de paquetes) todo ello en una única interfaz gráfica. Esta herramienta es muy útil para auditar cortafuegos, encaminadores, o cualquier equipo de red.

packETH [24] es una herramienta gráfica generadora de paquetes de Ethernet. Permite crear y enviar cualquier posible paquete o secuencia de paquetes en nuestra red Ethernet. Admite los protocolos Ethernet II, Ethernet 802.3, 802.1Q QinQ, ARP, Ipv4, IPv6 el usuario puede definir la carga la capa de red, UDP, TCP, ICMP, ICMPv6, IGMP, podremos incluso retardar el envío de paquetes, numero de paquetes a enviar, etc. Las principales ventajas de esta herramienta son que es muy fácil de usar y soporta muchas características personalizadas.

Mike Frantzen y otros crearon un conjunto de herramientas, denominada ISIC-IP Stack Integrity Checker [25], para probar la estabilidad de una pila IP v4 e Ipv6 y sus pilas de componentes (TCP, UDP, ICMP etc). Para ello, se generan muchos paquetes aleatorios del protocolo objeto de estudio. De todo este flujo generado, el 50% de los paquetes generados puede tener opciones IP. El 25% de los paquetes puede ser fragmentos IP.

Sin embargo, los porcentajes son arbitrarios y la mayoría de los campos de paquetes tienen una tendencia totalmente configurable. Los paquetes se envían en contra del equipo de destino con el objetivo de testearlo. Sirve para detectar vulnerabilidades en el cortafuegos, observar si existe fuga de paquetes o para encontrar errores en la pila IP. ISIC también dispone de una utilidad para examinar las configuraciones del hardware implementado en nuestra red.

Netperf [26] es una herramienta que puede ser utilizada para medir el rendimiento de muchos tipos diferentes de redes. Permite realizar pruebas tanto para el rendimiento unidirecional, como medir la latencia de extremo a extremo. Las variables actualmente medibles por netperf incluyen TCP y UDP a través de sockets BSD para IPv4 e Ipv6, DLPI, Unix Domain Sockets y SCTP para IPv4 e Ipv6. Sólo está disponible para Linux.

Roel Jonkman, de la universidad de Kansas, creó la utilidad NetSpec [27]. Es una herramienta diseñada, y desarrollada en Linux, para simplificar el proceso de las pruebas rendimiento y funcionalidad de la red. NetSpec proporciona un marco bastante genérico que permite al usuario controlar múltiples procesos a través de múltiples hosts, todo ello controlado desde un punto central de control. Se compone de demonios que implementan las fuentes de tráfico además de diversas herramientas de medición pasiva. NetSpec utiliza un lenguaje de scripting que permite al usuario definir múltiples flujos de tráfico desde/hacia varios equipos de manera automática.

Bit-Twist [28] es un generador de paquetes Ethernet basado en libpcap que está diseñado para complementar tcpdump. Es capaz de regenerar su tráfico capturarlo en una red, los paquetes se generan a partir de un archivo de rastreo tcpdump (archivo. Pcap). Bit-Twist viene con un completo editor de archivo de rastreo para permitir cambiar el contenido del mismo. Es muy útil para probar cortafuegos, IDS, IPS, además de permitir resolver problemas diversos en la red.

A. Dainotti y otros, de la universidad de degli Studi di Napoli ''Federico II'' (Italia), han creado recientemente D-ITG (Distributed Internet Traffic Generator) [29]. D-ITG es una plataforma que puede generar tráfico tanto IPv4 como IPv6. Además, permite generar tráfico en las capas de red, transporte, y aplicación. Es multiplataforma (soporta Windows, Linux y OSX).

Como se puede ver en las anteriores herramientas mencionadas, la mayoría son generadores de tráfico o herramientas destinadas a la auditoria de seguridad en la red. Hemos comprobado que no existe ningún generador de tráfico desarrollado en Windows, que también tenga un receptor para poder analizar los paquetes recibidos y que permita registrar los parámetros de calidad de la red en bases de datos y genere graficas estadísticas con estos datos. Debido a esa falta de herramientas, nace la necesidad de desarrollo del actual proyecto.

3 – INTERFAZ GRÁFICA

En este capítulo describiremos las características que ofrece la interfaz gráfica de nuestra herramienta. El aspecto gráfico final de la herramienta es el que se muestra en la Figura 1, expuesta en la siguiente página.

Fue desarrollada desde un principio para que se mostraran las gráficas estadísticas de los parámetros de calidad en la mitad izquierda de la interfaz gráfica y en la mitad derecha se mostrara una tabla con los datos de los últimos diez paquetes recibidos, una ventana que informaría al usuario de los sucesos que acontecían en el programa, tanto correctos como incorrectos, y un panel de las diferentes opciones de envío de la herramienta junto con las botones de puesta en marcha y detención de la misma.

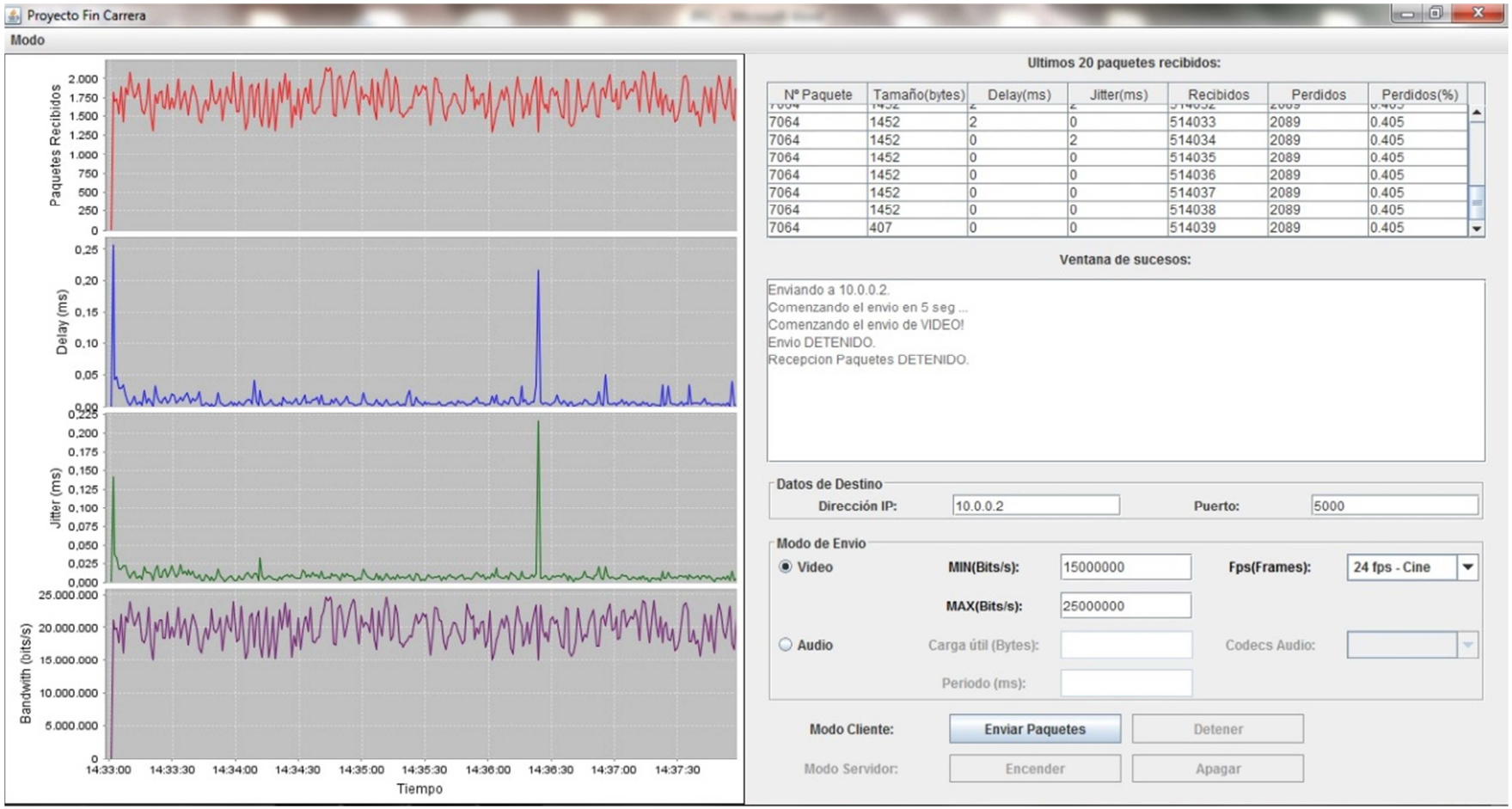

Figura 1. Interfaz Gráfica

La herramienta fue diseñada para que fuera reversible y se pudiera utilizar tanto para hacer de cliente como para hacer de servidor. La Figura 2 muestra una opción en la esquina superior izquierda de la herramienta en la que se puede seleccionar el modo de funcionamiento.

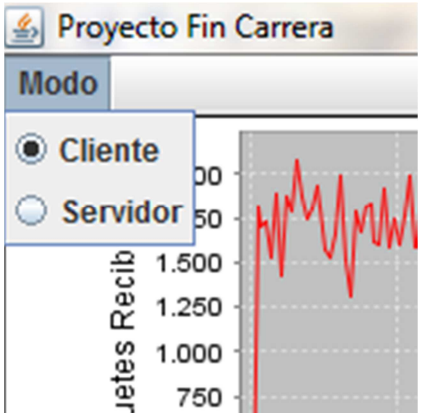

Figura 2. Modo de Envío

El caso mostrado en la Figura 1, es la del ordenador que actúa como cliente. En el caso del Servidor solo tiene que seleccionar el Puerto por el que escucha la llegada de los paquetes y darle al botón de "Encender" del modo Servidor. Dependiendo del tráfico que el usuario cliente desee enviar, se deben rellenar los siguientes campos marcados.

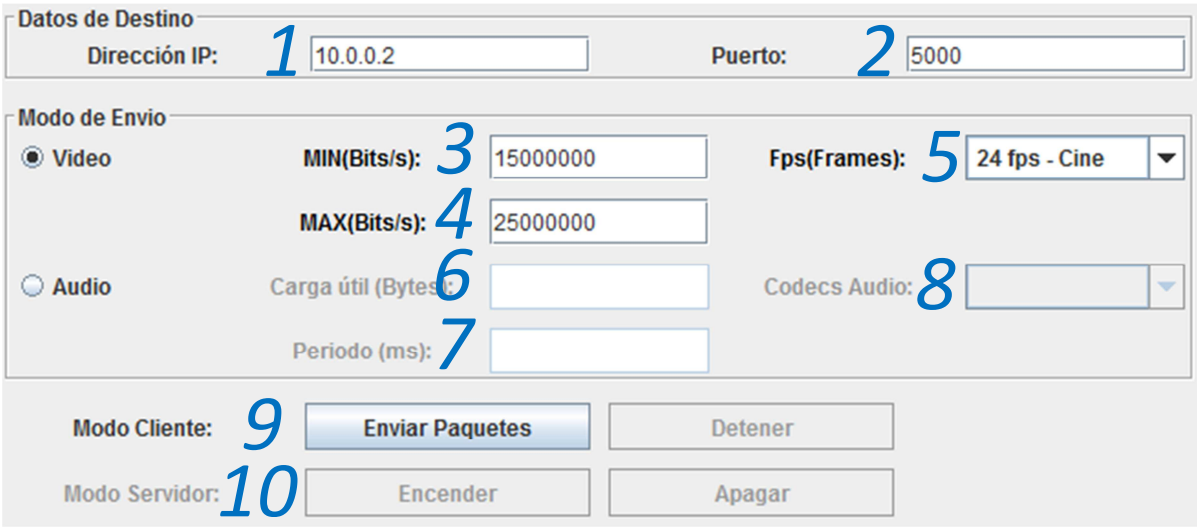

Figura 3. Campos de la herramienta

Por último se va a llevar a cabo una explicación de cada campo a rellenar indicado en la Figura 3, dependiendo del envío que el usuario desee con nuestra herramienta:

1. Direccion IP del ordenador destino.
2. Puerto del ordenador destino a donde se envian paquetes (Modo Cliente). Puerto propio que se utiliza para la recepción de paquetes. (Modo Servidor).
3. Bandwidth Maximo que se desea enviar en el Modo Video.
4. Bandwidth Minimo que se desea enviar en el Modo Video.
5. Frames por Segundo que se desea enviar en el Modo Video Es una lista desplegable donde se muestran los fps mas comunes. (Figura 4)

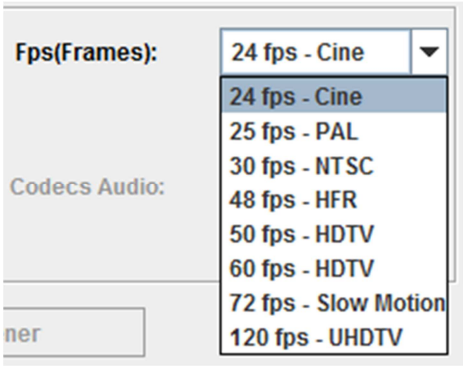

Figura 4. Lista desplegable FPS

6. Carga Util de los paquetes que se desean enviar en el Modo Audio.
7. Periodo de envio de los paquetes que se desean enviar en el Modo Audio.
8. Lista desplegable de los Codecs de Audio mas comunes para que el usuario pueda simularlos mas facilmente. (Figura 5)

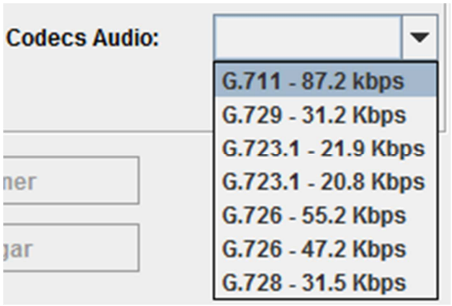

Figura 5. Lista desplegable Audio

9. Botones para comenzar y detener el envio de paquetes en el Modo Cliente
10. Botones para comenzar y detener la escucha de paquetes en el Modo Servidor.

4 – PROGRAMACIÓN DE LA HERRAMIENTA

En este capítulo se hará la explicación del sistema Cliente-Servidor utilizado en la aplicación para hacernos una idea básica de cómo funciona.

A continuación realizaremos un diagrama lógico para una explicación más profunda del funcionamiento interno y se expondrá el cálculo teórico realizado de los parámetros de calidad y el código utilizado en Java, explicando debidamente su sintaxis para su comprensión por parte del lector.

Y por último se explicaran las cabeceras utilizadas a nivel de aplicación y como se ha realizado el sistema de envío de paquetes con Java.

4.1 - DESCRIPCIÓN DE LA AQUITECTURA DEL SISTEMA

La aplicación tiene un funcionamiento básico que se puede resumir en el modelo Cliente-Servidor. El sistema Cliente-Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados Servidores, y los demandantes, llamados Clientes. La mayoría de los servicios de Internet son tipo de Cliente-Servidor. La acción de visitar un sitio web requiere una arquitectura cliente-servidor, ya que el servidor web sirve las páginas web al navegador (Cliente).

En nuestro caso, el cliente manda paquetes de datos hacia el ordenador destino (Servidor), el cual responde devolviendo los paquetes hacia el cliente, para así poder hacer el cálculo de los parámetros de calidad de la red y mostrarlos en nuestro ordenador (Cliente). Una muestra gráfica del proceso sería la mostrada por la Figura 6.

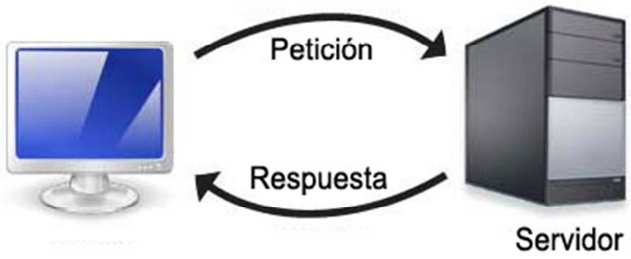

Figura 6. Cliente-Servidor

La aplicación se puede resumir en 5 funciones básicas dentro del código de Java:

-Ventana Principal de la aplicación:

Es la función "Principal" del programa. Es la que se encarga de dibujar el aspecto visual de la aplicación, la que enlaza las opciones introducidas por el usuario con las funciones específicas internas que deben activarse. También se encarga de activar y rellenar las gráficas estadísticas con los datos.

En esta función podemos seleccionar "Modo Cliente "(Predefinido) o "Modo Servidor". Estas opciones desactivan y activan determinadas opciones para el usuario en función de lo que desee hacer.

-Activar Cliente:

Dentro de "Modo Cliente", esta opción se encarga comenzar el envío de paquetes con las características introducidas por el usuario, ya sean Video o Audio. También activa una función que paralelamente está recibiendo los paquetes que el servidor nos está devolviendo, rellena las tablas de datos de la Ventana Principal y manda los datos registrados a la Ventana Principal para que dibuje las gráficas.

-Parar Cliente:

Se encarga de finalizar la función "Cliente", que se encarga del envío continuo de paquetes. También envía un paquete a la función que recibe los paquetes devueltos por el servidor, para que no siga esperando más paquetes y finalice la función.

-Activar Servidor:

Dentro de "Modo Servidor", se encarga de poner la aplicación en modo escucha, recibiendo los paquetes desde el cliente y devolviéndolos. Es un modo totalmente pasivo en la que el ordenador hace las funciones de recepción y reenvío de paquetes sin la ayuda del usuario.

-Parar Servidor:

Finaliza el "Modo Servidor", para que la aplicación no siga esperando más paquetes del cliente. Esto devuelve al usuario el control de la aplicación, pudiendo pasar de "Modo Servidor" a "Modo Cliente" si se desea.

Para que el sistema funcione, el ordenador cliente debe estar en "Modo Cliente" e indicar la Dirección IP y el Puerto del Servidor que se ha seleccionado para el envío. Por otro lado, el ordenador servidor debe estar en "Modo Servidor" e indicar el mismo Puerto que seleccionó el cliente como destino del envío de paquetes.

El servidor al fin y al cabo es un elemento pasivo el cual solo hace que devolvernos los paquetes que le lanzamos. El cliente es el elemento importante en el sistema, ya que envía y recibe paquetes desde el servidor, almacena y representa los datos obtenidos. El cliente también se encarga de la extracción de los datos en hojas de cálculo si el usuario lo desea.

4.2 - DIAGRAMA LOGICO Y FUNCIONAMIENTO INTERNO

La Figura 7 representa un diagrama lógico básico de las opciones de la aplicación. Este esquema se ha realizado para facilitar la comprensión del funcionamiento interno de la herramienta aunque faltan opciones que realiza internamente el programa que se explican más adelante.

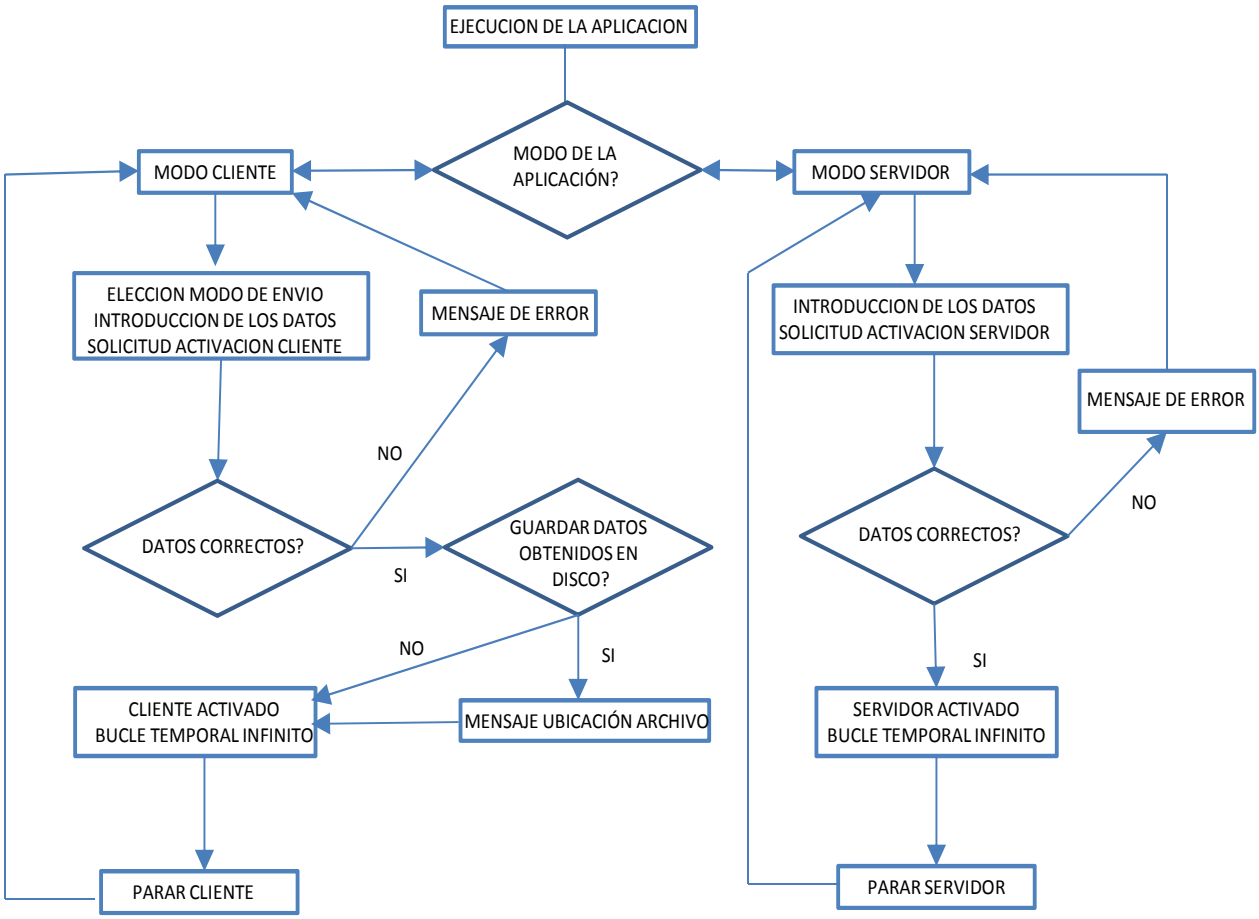

Figura 7. Diagrama lógico de la aplicación

Java, al lanzar la aplicación, ejecuta la función "Principal". Principal es una clase que extiende JFrame, permitiendo así, la creación de ventanas para poder introducir dentro del contenedor los botones, tablas, áreas de texto, graficas, etc. que se quieran introducir. También implementa JActionListener para poder realizar tareas en función de un evento determinado ya sea por clicar en algún botón o actualizar una gráfica con el paso de cada segundo. Dentro de esta función se ha diseñado toda la apariencia gráfica del programa que se debe mostrar.

Al lanzar la aplicación, esta se muestra al usuario y se queda inerte, esperando que este seleccione algunas de sus opciones. De forma predeterminada, al inicio de la aplicación se muestra seleccionado el Modo Cliente y dentro de este, Video, como forma de envío de paquetes.

Si nos decidimos por enviar este tipo de tráfico, debemos introducir la dirección IP y Puerto del destino correctamente. Un formato introducido que no sea correcto como por ejemplo introducir un formato diferente a "X.X.X.X", siendo X el rango desde 0 a 255 en el campo de dirección IP, o un numero para el puerto fuera del rango 0-65535 hará que la aplicación lance un error. La introducción de caracteres que no sean numéricos en los campos de Dirección IP, Puerto, Bandwitdh MAX y Bandwidth MIN y un Bandwidth MAX menor que un Bandwidth MIN, también lanzara un error. Otro error será dejar la casillas de los FPS sin seleccionar .Se debe seleccionar los FPS que se desean enviar para que el programa pueda enviar con un determinado periodo los distintos paquetes.

Si se introducen todos los datos correctamente, la aplicación lanzara un mensaje de "Guardar como…" para que el usuario seleccione una ruta y el nombre del archivo donde se guardaran los datos obtenidos. Al introducirla y darle a "Aceptar", se mostrara un mensaje emergente con ruta exacta especificada por el usuario. Si por el contrario no desea guardar los datos y presiona el botón de "Cancelar", se informara al usuario que los datos no se guardaran en ninguna ubicación.

Una vez pasado este punto, la función Principal lanzara los hilos "Cliente", el cual se encarga del envío continuo de paquetes; "RecibirPaquetes", encargado de recibir los paquetes devueltos por el Servidor; y pondrá en funcionamiento las gráficas que empezaran a actualizarse cada segundo.

El hilo Cliente se encarga de montar los paquetes con nuestras cabeceras personalizadas y realizar el bucle continuo de envío hasta que el usuario desee finalizar. Este hilo se sirve se las funciones de la librería NET de Java que permite la apertura se sockets y el envío de paquetes a través de ellos. Cuando se tiene los datos de cada paquete introducidos en un tipo byte[] , se introduce esta información en una variable tipo DatagramPacket y se envía. La sintaxis en Java es la siguiente:

```
DatagramPacket Paquete = new DatagramPacket(paqueteUDP,
paqueteUDP.length,InetAddress.getByName(CampoIPField.getText()),
Integer.parseInt(CampoPuertoField.getText()));

socketEnvio.send(Paquete);
```

DatagramPacket necesita solo la información de los datos del paquete (1º Campo), la longitud del paquete (2º Campo), la dirección IP destino (3º Campo) y el puerto destino (4º Campo). El comando "send" efectúa el envío del paquete a través del socket, acoplándole la cabecera UDP.

Estos paquetes serán recogidos por el ordenador destino el cual debe de estar en "Modo Servidor" y activado. Su única función es la de recibir los paquetes y reenviarlos al ordenador origen.

Paralelamente en el ordenador Cliente, el hilo "RecibirPaquetes" recibe los paquetes reenviados por el servidor. Este hilo es el que efectúa todos los cálculos respecto a los parámetros de calidad. Con cada paquete que le llega efectúa el cálculo del delay, jitter, paquetes perdidos y ancho de banda , incluye en una tabla los valores de los últimos 10 paquetes recibidos, va escribiendo con cada paquete recibido el Excel (En caso de haber seleccionado esta opción) y pasa a esos datos a la función "Principal"

para que esta realice un promedio del jitter y del delay de los paquetes recibidos por segundo, la suma total de paquetes recibidos por segundo y el respectivo ancho de banda que se consume por segundo y represente todos estos datos en las gráficas . Para el cálculo de dichos parámetros la función" RecibirPaquetes" utiliza la definición de cada uno de ellos. A continuación se explica la sintaxis utilizada en el código Java.

Delay:

```
delay = (tiempoLlegada-tiempoOrigen)/2;
```

Según su definición, el delay es el tiempo que tarda un paquete desde su lanzamiento de un ordenador origen hasta su recepción por el ordenador destino. En nuestro caso como el paquete realiza el viaje de ida y vuelta, es decir Cliente-Servidor-Cliente, tenemos que dividirlo por dos el tiempo obtenido.

Jitter:

Para el cálculo del jitter, según su definición es la diferencia entre el delay de un paquete y el delay del anterior paquete recibido. Por lo tanto nos servimos de la sintaxis anterior para el cacluclo del delay y añadimos algunos detalles:

```
delay = (tiempoLlegada-tiempoOrigen)/2;

if(recibidos>1){
if(NumeroPaquete==(paqueteAnterior+1) )
            jitter=Long.toString(Math.abs(delay-tiempoJitter));
      else
            jitter="ERROR";
}

tiempoJitter=delay;
```

Según este código, a la llegada del primer paquete se almacena su delay en la variable delay y como es el primer paquete no cumple la condición para el cálculo del jitter impuesta por el if, que obliga a recibir más de 1 paquete para su cálculo. Por lo tanto

nos saltamos todas las llaves if y en la última línea almacenamos el delay de este paquete para que el siguiente paquete lo utilice.

Cuando se recibe un segundo paquete, calculamos su delay y ahora si cumplimos la condición del if, por lo que entramos a ejecutar el código de esta llave. Existe otra condición dentro impuesta para que el cálculo del jitter se efectúe solo cuando son dos paquetes consecutivos. En el caso de pérdida de paquete no se debe calcular el jitter y por lo tanto se indicara como "ERROR" en las tablas de datos. Si los paquetes sí son consecutivos, entonces procedemos al cálculo del jitter restando el delay del paquete que acabamos de recibir con el delay almacenado del anterior paquete recibido. Esta diferencia se mide en valor absoluto y por eso le aplicamos "Math.abs" a esa resta.

Paquetes Perdidos:

Para el cálculo de los paquetes perdidos solo tenemos que extraer el número de paquete, incluido en la cabecera de cada paquete (que será el número de paquetes totales enviados) y restarle el número de paquetes recibidos que lo registramos con un contador. Para este cálculo nos servimos de la clase BigDecimal la cual nos deja operar con muchos decimales ideal para el porcentaje de paquetes perdidos. La sintaxis es la siguiente:

```
BigDecimal Resta = new BigDecimal(NumeroPaquete-recibidos);

BigDecimal perdidos = ((Resta.divide
(new BigDecimal(NumeroPaquete), 5,
RoundingMode.HALF_DOWN)).multiply
(new BigDecimal("100"))). stripTrailingZeros();
```

Primero realizamos la resta entre los paquetes totales y los recibidos y lo almacenamos como tipo "BigDecimal". Entonces le dividimos los paquetes totales, indicándole el número de decimales que queremos (5 en nuestro caso), que realice un redondeo "HALF_DOWN" (este comando redondea hacia abajo en caso de estar en medio justo de dos cifras, ej: 1,5), y esto lo multiplicamos por 100 para obtener el porcentaje.

Añadimos un último comando al final, “stripTrailingZeros()” para eliminar los ceros a la derecha del número obtenido.

A la vez que calcula todos estos parámetros, si el usuario lo ha seleccionado, el hilo “RecibirPaquetes” está escribiendo los parámetros en las filas de un archivo Microsoft Excel (.XLS) gracias a la librería JXL que hemos importado. A medida que va recibiendo los paquetes, mediante un bucle, va rellenando fila por fila con los parámetros obtenidos. Al archivo Excel se le ha intentado dar un toque más agradable centrando todos los datos de las celdas, aumentando el tamaño de las mismas y rellenando de color la primera fila que indica el tipo de datos de cada columna y las celdas del jitter que debido a la pérdida de paquetes ha dado ERROR. Este archivo solo se cerrara, permitiendo su lectura, cuando el usuario haya presionado el botón de “Parar Cliente”. Solo entonces el programa dejara de recorrer las filas del Excel y liberara el archivo para su posterior consulta.

La función “Principal” importa la librería JFreeChart, la cual permite el dibujo de las gráficas estadísticas. Esta función recibe todos los datos de los parámetros de calidad obtenidos en el hilo “RecibirPaquetes” y al cabo de cada segundo representan la suma de los paquetes recibidos, el bandwidth consumido, el promedio del delay por cada paquete y el promedio del jitter por cada paquete.

Como el hilo “RecibirPaquete” del cliente y el hilo “Servidor” del servidor esperan continuamente un paquete para continuar sus tareas, cuando el usuario decide presionar el botón de “Detener” (Modo Cliente) o “Apagar” (Modo Servidor), la aplicación debe mandar un paquete a los sockets que están escuchando ellos mismos para poder desbloquear esos hilos y finalizarlos completamente.

Y hasta aquí la explicación general de cómo funciona internamente nuestra herramienta. En el siguiente capítulo se llevara a cabo la explicación de las cabeceras utilizadas, la sintaxis utilizada para su utilización con Java y una explicación breve del código.

4.3 - CABECERAS DEL PROTOCOLO

Nuestra herramienta tiene la capacidad de generar y enviar paquetes a un destino y así medir los parámetros de calidad de la red. Pero todo esto no sería posible sin las cabeceras que le aplica la herramienta a los paquetes. En función del tipo de tráfico seleccionado, la herramienta aplicara la cabecera de Video o la de Audio.

-Video: Con esta casilla activada, se implantara una Cabecera personalizada a los paquetes para poder soportar una gran cantidad de paquetes en la red y poder medir los parámetros de calidad correctamente. En la Figura 8 se muestra la cabecera.

Figura 8. Cabecera Video

"Identificador": Identifica el tipo de tráfico del paquete recibido para poder aplicarle unas formulas u otras. El Byte=0 quiere decir tráfico de Video y el Byte=255 quiere decir tráfico de Audio. Con esto descartamos también cualquier otro paquete indeseado que llegue al socket sin alguno de estos dos identificadores.

"Nº de Paquete": Indica el número del paquete. Con 3 Bytes en la cabecera tiene una capacidad para registrar el numero de (2^{24}-1) paquetes, es decir, 16777215 paquetes.

"Tiempo Origen": Indica el tiempo en milisegundos en que salió el paquete del cliente. Al utilizar una función de Java que representa el tiempo en 13 dígitos, necesitamos truncar este tiempo para reducirlo. Con 3 Bytes tenemos la capacidad de registrar una diferencia de tiempo de 16777215 ms desde que se inicia la aplicación. Por eso decidimos truncar los tiempos, es decir, tomar un tiempo de referencia y quedarnos

con la diferencia de los tiempos posteriormente medidos y el de referencia. Así podemos captar medidas durante mucho más tiempo utilizando solo 3 Bytes.

“Dirección IP”: Indica la dirección IP propia del cliente, para cuando el servidor reciba cada paquete, sepa reenviarlo de vuelta a su origen. Ocupa un byte por cada campo de los 4 que tiene una dirección IP.

“Puerto”: Indica el puerto seleccionado por el usuario por donde se enviaran y recibirán los paquetes. Esta información la utilizara el servidor para reenviar los paquetes. Ocupa 2 Bytes debido a que el número de Puertos puede ser desde 0 hasta el 65534.

“Subpaquetes”: Indica el nº de Subpaquete dentro de cada Paquete. En video es probable que al hacer la división de la cantidad de bits/s que se quiere enviar por los FPS, de como resultado paquetes mayores que el MTU y por lo tanto conviene fraccionarlos en subpaquetes. Un paquete podrá ser fraccionado en 255 subpaquetes como máximo, con 1 Byte reservado en la cabecera.

“Paquete Final”: Indica si es el último subpaquete del paquete. Se indica como Byte=0 (Subpaquete normal dentro del paquete) y Byte=1(Subpaquete final del paquete). Esto se utilizara en el cálculo del jitter cuando se requiere saber si el último subpaquete recibido era el último del paquete o no para realizar el cálculo.

“Paquetes Totales”: Indica el nº de paquetes totales enviados. Esto es necesario para calcular el porcentaje exacto de paquetes perdidos.

“Datos de Video”: Carga útil aleatoria entre los márgenes dados por el cliente.

Con la opción de Video activada, solo se necesita indicar el rango de bitrate que se desea enviar introduciendo el valor máximo y el valor mínimo, junto con los FPS (Frames por Segundo) que se deseen. Los FPS se utilizan para introducir el periodo de envío entre cada paquete. A continuación se expone la sintaxis utilizada en Java y la utilización de las variables tipo byte para la formación de la cabecera junto al paquete:

```
paqueteUDP[0]=(byte)(255);

//Campo Identificador

paqueteUDP[1] = (byte)(Paquete & 0x000000ff);
paqueteUDP[2] = (byte)((Paquete & 0x0000ff00) >> 8);
paqueteUDP[3] = (byte)((Paquete & 0x00ff0000) >> 16);

//3 Campos que al combinarse representan el Número de Paquetes.

paqueteUDP[4] = (byte)(tiempoOrigen & 0x000000ff);
paqueteUDP[5] = (byte)((tiempoOrigen & 0x0000ff00) >> 8);
paqueteUDP[6] = (byte)((tiempoOrigen & 0x00ff0000) >> 16);

//3 Campos que al combinarse representan el Tiempo de Origen.

paqueteUDP[7] = (byte)(Ip[0] & 0x000000ff);
paqueteUDP[8] = (byte)(Ip[1] & 0x000000ff) ;
paqueteUDP[9] = (byte)(Ip[2] & 0x000000ff) ;
paqueteUDP[10] = (byte)(Ip[3] & 0x000000ff) ;

//4 Campos que al combinarse representan Dirección IP origen.

paqueteUDP[11] = (byte)
(Integer.valueOf(CampoPuertoField.getText()) & 0x000000ff);
paqueteUDP[12] = (byte)
((Integer.valueOf(CampoPuertoField.getText()) & 0x0000ff00)>>8);

//2 Campos que al combinarse representan el Puerto origen.


paqueteUDP[13] = (byte)((i+1) & 0x000000ff);

//Campo que indica el  Número de Subpaquetes
```

```
paqueteUDP[14]=(byte)(0);

//Campo de PaqueteFinal

paqueteUDP[15] = (byte)(totales & 0x000000ff);
paqueteUDP[16] = (byte)((totales & 0x0000ff00) >> 8);
paqueteUDP[17] = (byte)((totales & 0x00ff0000) >> 16);

//3 Campos que al combinarse representan los Paquetes totales
enviados.

for(int z=0;z<(1450);z){
      paqueteUDP[18+z]=(byte)(255);
}

//Bucle que rellena el resto de campos y que representan la
carga útil del paquete. En este caso se rellena con el MTU.
```

En el código anterior representa como rellena nuestra aplicación un paquete en binario. Podemos ir viendo cómo vamos introduciendo en cada posición del vector tipo byte paqueteUDP[ ], cada campo definido anteriormente en la cabecera.

Como bien se puede ver efectuamos una máscara a los todos los números que tiene más de 1 byte de espacio reservado en la cabecera, porque a la hora de almacenar, tendremos que dividir el numero introducido en los bytes disponibles.

-Audio: Con esta casilla activada, se implantara una Cabecera personalizada a los paquetes, más liviana que la cabecera de Video, para poder simular correctamente los códecs específicos de VoIP. En la Figura 9 se muestra la cabecera.

Figura 9. Cabecera Audio

Con la opción de Audio activada, solo se necesita indicar el Bitrate que se desea enviar, ya que el envío de Audio tiene un Bitrate constante; junto con el periodo de envío que se desee. La aplicación ofrece la opción de seleccionar, de una lista predefinida, los códec de VoIP más utilizados para simular su envío como G.711, G.729, G.723, G.726 y G.728.

La sintaxis es exactamente que la de la cabecera de Video quitando los campos que no se incluyen en la cabecera de Audio (SubPaquetes, PaqueteFinal y PaquetesTotales).

Los campos de las cabeceras, tanto de Audio como de Video se han diseñado trabajando en "Bytes", ya que supone un ahorro considerable de carga a los paquetes y permite simular las cabeceras RTP que se utilizan en la realidad.

Ya que la cabecera RTP utiliza 12 Bytes, el programa reajusta la carga útil para que simule perfectamente la cabecera RTP. En el caso de Video (Nuestra cabecera es de 18 Bytes), el programa le resta 6 Bytes a la carga útil del paquete indicada por el usuario y en el caso de Audio (Nuestra cabecera es de 13 Bytes), el programa le resta 1 Byte. Así la herramienta puede simular perfectamente un envío real de Audio o Video.

5 - MEDIDAS REALIZADAS

Una vez explicado los fundamentos internos de la herramienta, en este capítulo se procede a realizar una prueba simple de envío de tráfico a un ordenador destino para comprobar el buen funcionamiento de la herramienta.

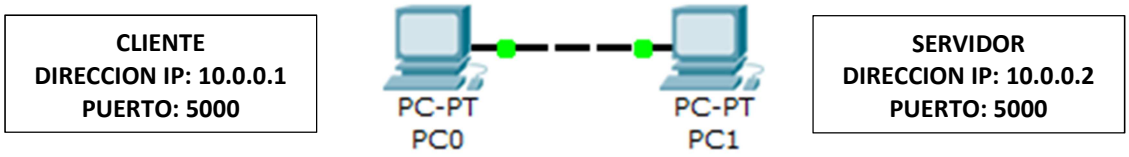

Figura 10. Escenario de Prueba

Se va a configurar, como muestra en la Figura 10, el ordenador PC0 en "Modo Cliente" , como muestra la Figura 11, y el ordenador PC1 en "Modo Servidor", como muestra la Figura 12. Se va a proceder a un envío de tráfico de Audio digital con el códec G.711 y se comprobara los parámetros de calidad obtenidos, que en este caso no debe existir prácticamente pérdida de calidad debido al enlace directo mediante FastEthernet.

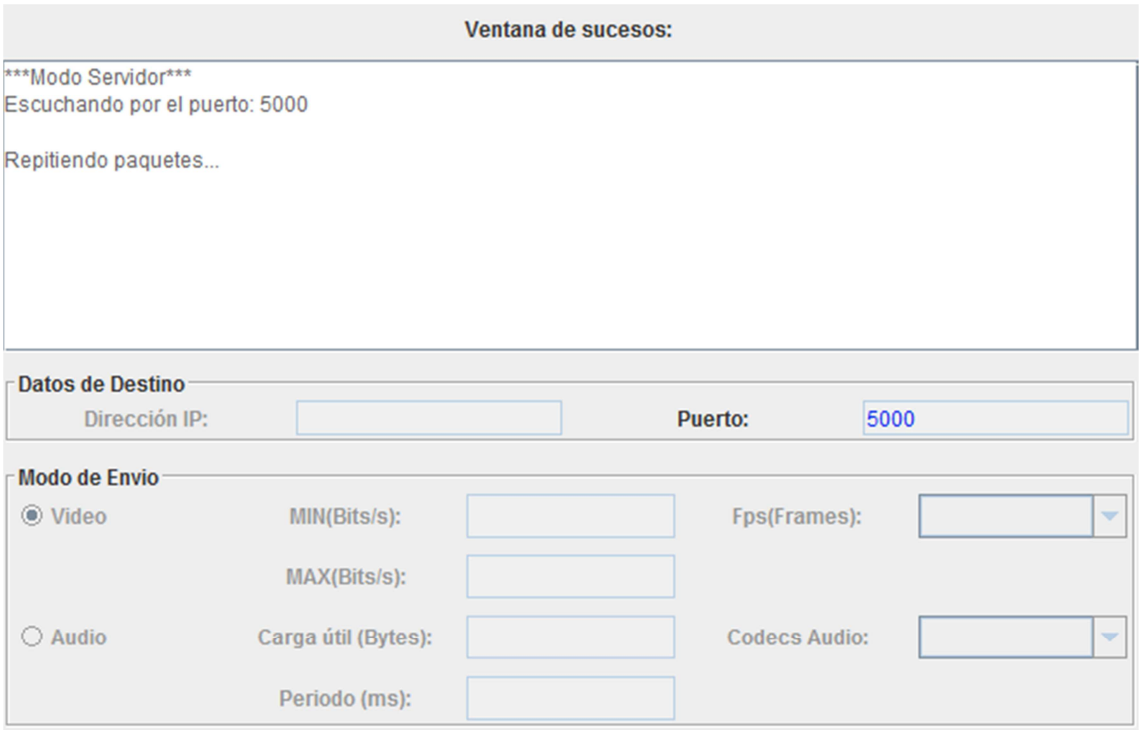

Figura 11. Configuración en el PC1 (Servidor)

Figura 12. Configuración en el PC0 (Cliente)

En las ilustraciones mostradas, podemos ver como solo hace falta introducir el valor del número de puerto en el servidor y clicar en "Activar" para que este e ponga en marcha. En este caso escucharemos la llegada de paquetes por el puerto 5000.

Por otro lado, podemos ver también la configuración introducida para que el cliente funcione perfectamente. En este caso, le indicamos la Dirección IP y el Puerto del ordenador destino a donde el usuario desea enviar el tráfico, en nuestro caso a la IP 10.0.0.2 y el puerto 5000. En el caso del cliente, el puerto introducido indica por donde se enviaran los paquetes y por donde se recibirán los paquetes reenviados por el servidor.

A continuación expondremos las medidas obtenidas y expuestas en las gráficas estadísticas por la herramienta para comprobar si son fiables o no. Lo normal es que el jitter y el delay que se obtenga sea en cada momento tienda a 0 y el número de paquetes recibido y el bandwidth constantes y coherentes con el códec introducido.

Si en las medidas observadas existiera una pérdida de paquetes, jjitter o un delay anormal, deberíamos sospechar que nuestra herramienta no funciona correctamente y deberíamos seguir trabajando en ella. La Figura 13 muestra los datos obtenidos.

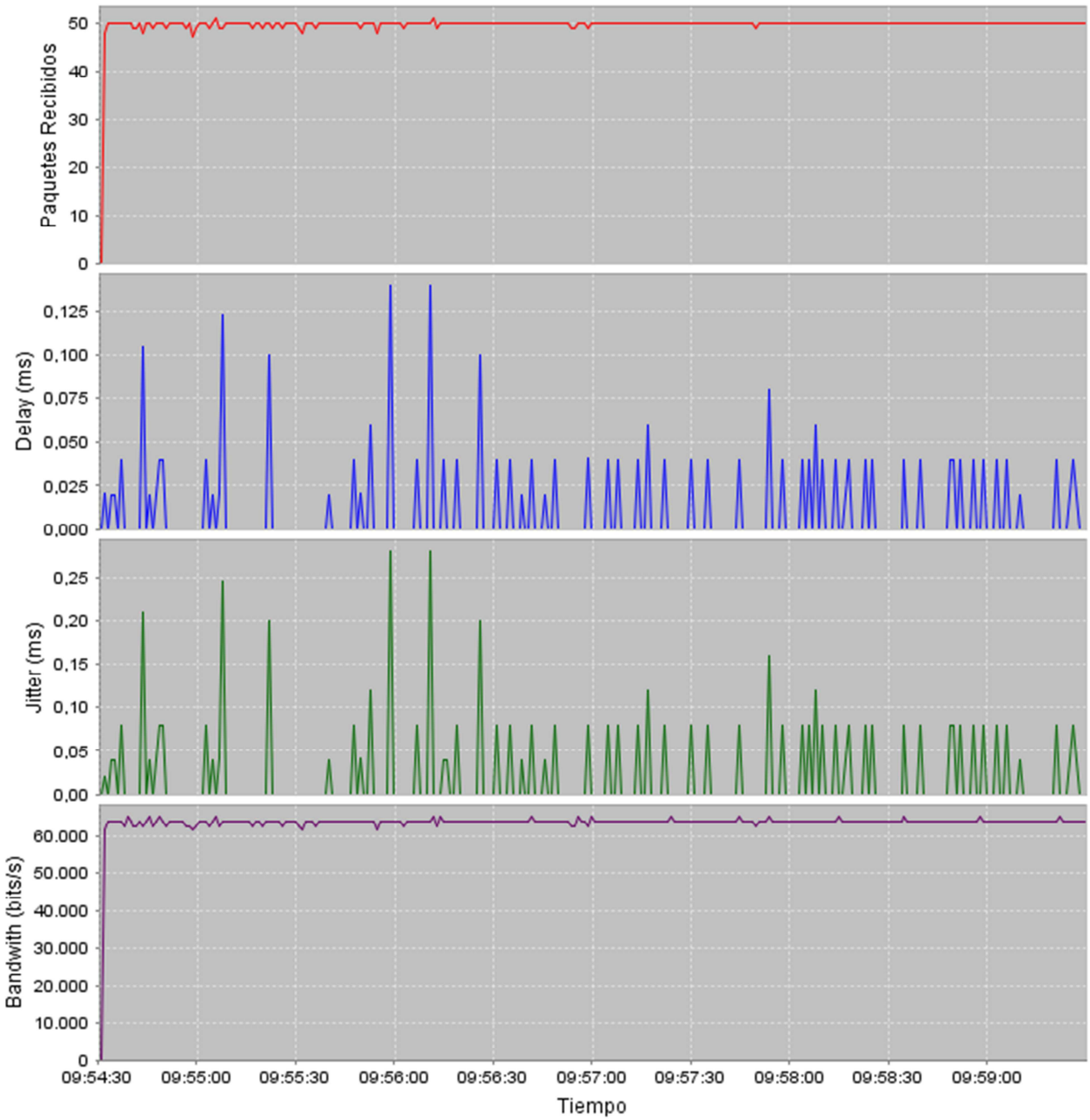

Figura 13. Medidas Obtenidas Prueba

Basándonos en estas medidas podemos observar que tanto el jitter como el delay son valores muy cercanos a 0 y por lo tanto son como esperábamos. Respecto al número de paquetes es constante y en torno a 50, confirmando los valores esperados, ya que al enviar un paquete cada 20 ms, al cabo de 1 s debemos haber recibido 50. Y por último para confirmar que nuestra herramienta funciona perfectamente, el bandwidth recibido es constante y correcto para el códec G.711, ya que al enviar una carga útil de 160 bytes en cada paquete, al cabo de 1 s se debe recibir 160 x 50 x 8 = 64 kbits/s.

5.1 - ESCENARIO DE ESTUDIO 1

Una vez comprobado que la herramienta funciona correctamente, vamos a proponer dos escenarios distintos para realizar medidas en el laboratorio El primer escenario propuesto es el que se muestra en la Figura 14.

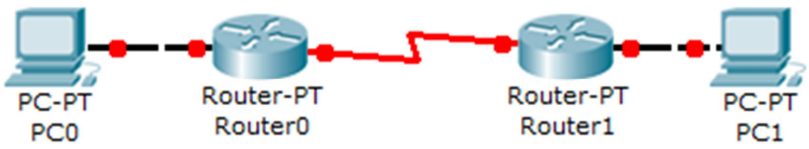

Figura 14. Escenario de Medida 1

Con la topología propuesta aquí arriba se pretende variar la velocidad en el enlace Serial y ver efectivamente como se deteriora o mejora la calidad de servicio en función de la velocidad introducida. A continuación detallaremos los equipos utilizados en la medida:

- 2 Ordenadores de Sobremesa con el Sistema Operativo Windows XP.
- 2 Router 1841 de CISCO SYSTEMS con el IOS versión 15.1.
- 2 Cables FastEthernet.
- 1 Cable Serial.

Las direcciones introducidas en este escenario se muestran en la Tabla 1.

DISPOSITIVO	FastEthernet 0/0	Serial 0/0
PC0	10.0.0.1	X
Router0	10.0.0.2	10.0.1.1
PC1	10.0.2.2	X
Router1	10.0.2.1	10.0.1.2

Tabla 1. Tabla de Direcciones 1

Una vez descrito el escenario de las medidas y sus características se procederá al envío de una señal de Video de entre 40 kbit/s y 50 kbit/s a 24 FPS con el enlace del serial a 64 kbit/s. Este hecho se corresponde, en la Figura 15, con la Zona 1. Existe un cierto delay en torno a 40 ms, un jitter variable menos de 4 ms, los paquetes recibidos son los correctos en torno a 24 por segundo (24 fps seleccionado), y el bandwidth a nivel de aplicación esta entre 40 kbit/s y 50 kbit/s. Hasta aquí todo correcto ya que en el enlace serial hemos introducido que se puede enviar hasta 64 kbit/s a nivel de capa Ethernet.

Al cabo de un rato, en la Zona 2, se procede a bajar la velocidad del clock rate a la mitad, 32kbits/s produciendo un “cuello de botella” en la interfaz. Podemos observar como comienza un aumento del delay de los paquetes hasta alcanzar los 7s, el jitter se encuentra en torno a 20 ms y tanto los paquetes recibidos como el bandwidth se han reducido a la mitad.

La Zona 3 no muestra ningún cambio en la configuración del sistema. Solo se quería hacer reflexionar que en las gráficas aparentemente el jitter desaparece, cosa que no es cierta. En realidad, en ese momento en el router se ha sobrepasado la capacidad de la cola de espera y este ha empezado a desechar paquetes. Produciendo que, tanto el

porcentaje como el número de paquetes perdidos, empieza a aumentar. El jitter gráficamente baja, pero si se compara con las tablas de datos, se puede ver que realmente se está dejando de calcular el jitter de muchos paquetes debido a la pérdida de muchos otros y por lo tanto en las tablas está indicado como "ERROR".

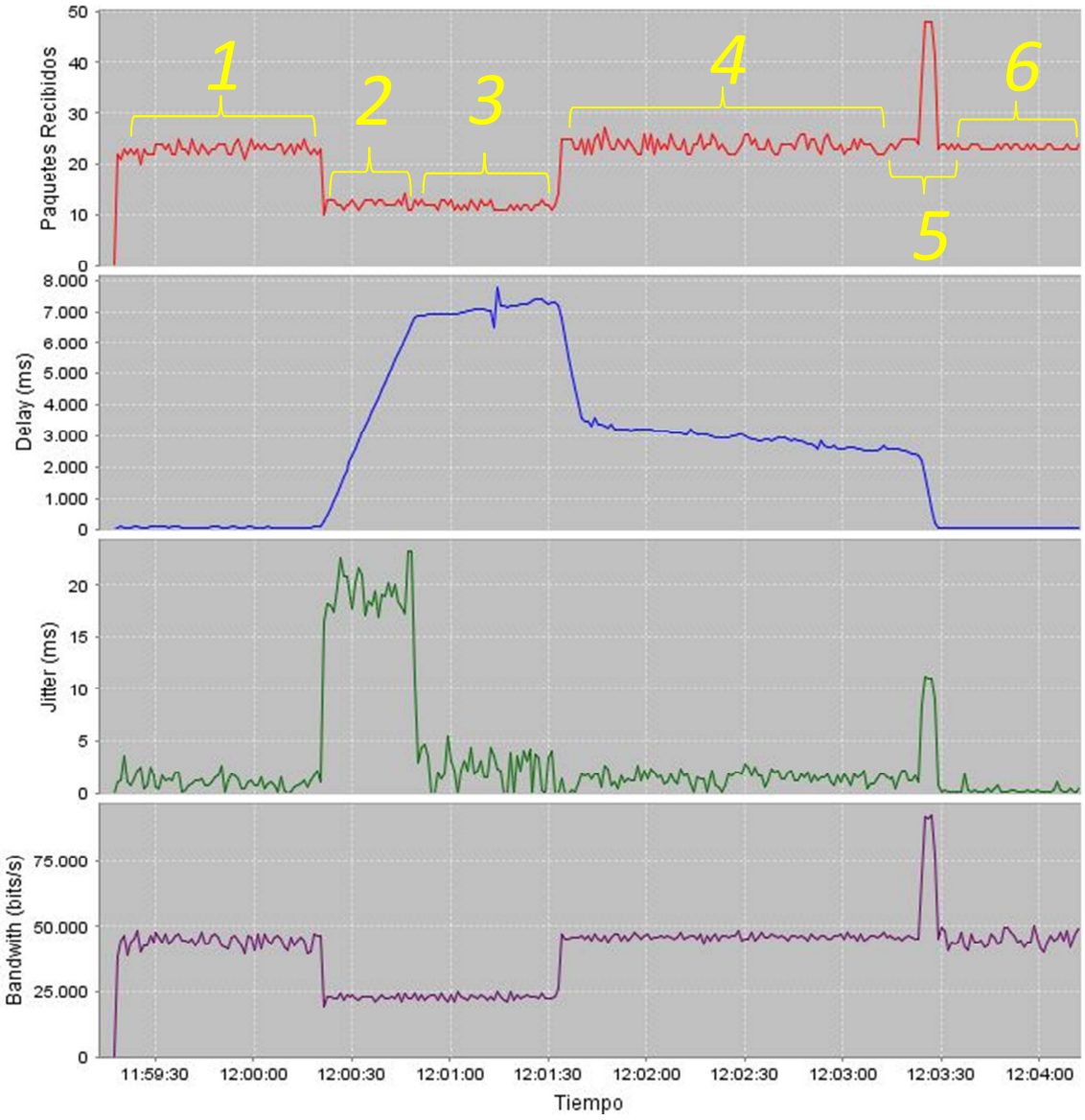

Figura 15. Gráfica Medidas 1

En la Zona 4 se restablece la velocidad del clock rate a 64 kbit/s. Se produce un aumento del número de paquetes recibidos y del bandwidth hasta alcanzar los valores del principio. Podemos ver como el jitter se restablece también a valores normales pero el delay de los paquetes pasa a ser en torno a 4s y descendiendo muy lentamente. Esto es debido a que el router está intentando poco a poco, descongestionar la cola de espera.

Para acelerar el proceso, en la Zona 5, volvemos a aumentar el clockrate hasta alcanzar los 128 kbit/s. Esto produce aumento momentáneo de los paquetes recibidos y del bandwidth, y un decrecimiento absoluto del delay de los paquetes. Al aumentar el clockrate del enlace a 128 kbit/s hemos forzado a que el router vacíe de golpe su cola de espera llena de paquetes.

Por último, en la Zona 6 muestra el estado de reposo alcanzado por el sistema al haber vaciado por completo la cola de espera del router.

A continuación se mostrara en la Tabla 2, para finalizar el estudio de esta topología, cómo se refleja en el archivo extraído de Microsoft Excel, la transición entre la Zona 2 y la Zona 3 de la gráfica. Se ve claramente que en el campo donde se calcula el jitter pasa a valer “ERROR” y los paquetes perdidos y su porcentaje empieza a aumentar.

Con estas últimas medidas damos por finalizado el estudio de esta topología utilizando la herramienta de testeo desarrollada.

Nº Paquete	Tamaño(bytes)	Delay(ms)	Jitter(ms)	Recibidos	Perdidos	Perdidos(%)
1789	256	6523	23	1787	2	,112
1790	256	6546	23	1788	2	,112
1791	256	6569	23	1789	2	,112
1792	256	6592	23	1790	2	,112
1793	256	6615	23	1791	2	,112
1794	256	6639	24	1792	2	,111
1795	256	6662	23	1793	2	,111
1796	256	6685	23	1794	2	,111
1797	256	6708	23	1795	2	,111
1798	256	6731	23	1796	2	,111
1799	256	6754	23	1797	2	,111
1800	256	6777	23	1798	2	,111
1801	229	6797	20	1799	2	,111
1802	229	6816	19	1800	2	,111
1803	229	6835	19	1801	2	,111
1804	229	6854	19	1802	2	,111
1806	229	6852	ERROR	1803	3	,166
1808	229	6850	ERROR	1804	4	,221
1810	229	6848	ERROR	1805	5	,276
1812	229	6846	ERROR	1806	6	,331
1813	229	6866	20	1807	6	,331
1815	229	6864	ERROR	1808	7	,386
1817	229	6862	ERROR	1809	8	,44
1819	229	6860	ERROR	1810	9	,495
1821	229	6858	ERROR	1811	10	,549
1823	229	6856	ERROR	1812	11	,603
1825	222	6853	ERROR	1813	12	,658
1826	222	6871	18	1814	12	,657
1828	222	6868	ERROR	1815	13	,711
1830	222	6865	ERROR	1816	14	,765
1830	222	6865	ERROR	1816	14	,765
1832	222	6862	ERROR	1817	15	,819
1834	222	6848	ERROR	1818	16	,872
1835	222	6866	18	1819	16	,872
1837	222	6863	ERROR	1820	17	,925
1839	222	6860	ERROR	1821	18	,979
1841	222	6857	ERROR	1822	19	1,032
1843	222	6854	ERROR	1823	20	1,085
1845	222	6851	ERROR	1824	21	1,138
1846	222	6869	18	1825	21	1,138
1848	222	6866	ERROR	1826	22	1,19
1850	229	6864	ERROR	1827	23	1,243
1852	229	6862	ERROR	1828	24	1,296
1854	229	6860	ERROR	1829	25	1,348
1855	229	6879	19	1830	25	1,348
1857	229	6878	ERROR	1831	26	1,4
1859	229	6874	ERROR	1832	27	1,452
1861	229	6872	ERROR	1833	28	1,505
1862	229	6891	19	1834	28	1,504
1864	229	6889	ERROR	1835	29	1,556
1866	229	6887	ERROR	1836	30	1,608
1868	229	6885	ERROR	1837	31	1,66
1870	229	6884	ERROR	1838	32	1,711
1871	229	6903	19	1839	32	1,71
1873	225	6900	ERROR	1840	33	1,762
1875	225	6898	ERROR	1841	34	1,813
1877	225	6895	ERROR	1842	35	1,865
1879	225	6893	ERROR	1843	36	1,916
1881	225	6866	ERROR	1844	37	1,967

Tabla 2. Medidas Excel 1

5.2 - ESCENARIO DE ESTUDIO 2

En el siguiente escenario realizaremos una serie de medidas introduciendo ciertas condiciones en la topología. En la Figura 16 se describe la nueva topología:

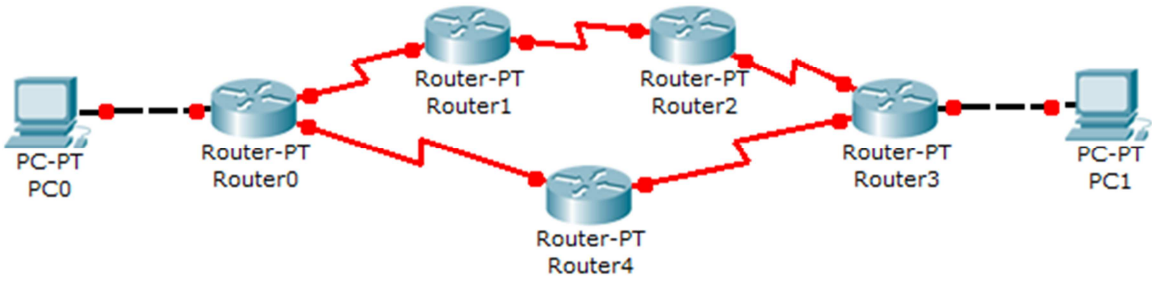

Figura 16. Escenario de Medida 2

Se ha diseñado el sistema con dos rutas posibles para el envío de tráfico de paquetes en las que el propio sistema elegirá la Ruta Principal y la Ruta Secundaria.

Para esta topología realizaremos 7 redes con las siguientes direcciones, mostradas en la Tabla 3, asignadas a cada interfaz del dispositivo correspondiente.

DISPOSITIVO	FastEhternet 0/0	Serial 0/0	Serial 0/1
PC0	10.0.0.1	X	X
Router0	10.0.0.2	10.0.1.1	10.0.5.1
Router1	X	10.0.1.2	10.0.2.1
Router2	X	10.0.2.2	10.0.3.1
Router3	10.0.4.1	10.0.3.2	10.0.6.2
PC1	10.0.4.2	X	X
Router4	X	10.0.5.2	10.0.6.1

Tabla 3. Tabla de Direcciones 2

Nuestros objetivos con esta topología son los siguientes:

- Enviar tráfico de un ordenador origen a un ordenador destino para obtener los parámetros de calidad de la ruta Principal designada por el sistema, que será la que pasa por el Router 4.
- Provocar un Fallo en uno de los enlaces por los que pasa la Ruta Principal y medir el tiempo de convergencia de la red para utilizar la Ruta Secundaria, teniendo en cuenta que en el sistema estará implementado el protocolo de enrutamiento RIP versión 2.
- Medir los parámetros de calidad de la Ruta Secundaria seleccionada por el sistema, compararlos con los obtenidos en Ruta Principal y confirmar que la calidad de servicio empeora debido a la mayor cantidad de saltos de la Ruta Secundaria frente a la Ruta Principal.

Para llevar a cabo estos objetivos, se han utilizado en esta topología los siguientes equipos:

- 2 Ordenadores de Sobremesa con el Sistema Operativo Windows XP.
- 5 Router 1841 de CISCO SYSTEMS con el IOS versión 15.1.
- 2 Cables FastEthernet.
- 5 Cables Serial.

Para comenzar las medidas realizaremos un envió, con la topología tal cual la hemos descrito, de un tráfico de Audio con el códec G.729 para registrar los parámetros de calidad de la Ruta Principal. Los enlaces de los seriales estarán configurados a 64kbits/s.

Los datos obtenidos en las gráficas son los siguientes:

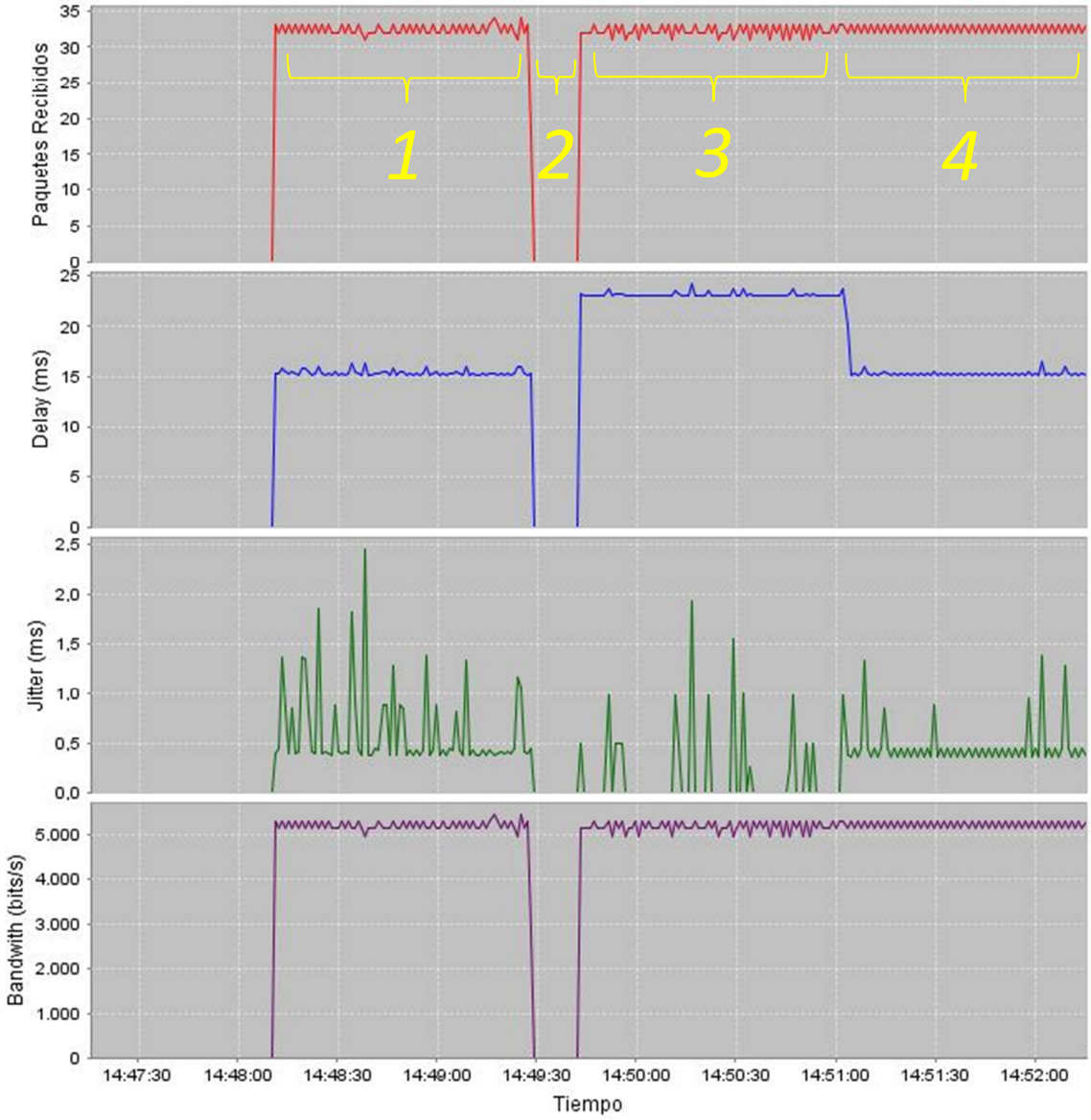

Figura 17. Gráfica Medidas 2

Podemos ver en la Figura 17, como en la Zona 1 iniciamos el envío de Audio con el códec G.729, y registramos un delay de un 15 ms, un jitter variable entre 0 y 2 ms, y un bandwidth de unos 5,3 kbit/s, parámetros propios del sistema que pertenecen a la Ruta Principal, que es la que pasa por el Router 4.

Al inicio de la Zona 2, provocamos un fallo en el enlace entre el router 4 y el router 3, desconectando el enlace serial. Vemos que momentáneamente los paquetes recibidos y el bandwidth es 0, debido a que el sistema tarda unos segundos en recalcular el envío por una Ruta Secundaria.

En la Zona 3, se ve como el sistema ha finalizado el intercambio de información y ha elegido una nueva Ruta Principal. Mediante las gráficas se puede observar que el sistema ha tardado en torno a 15 s en la elección. Este dato es propio para el protocolo de enrutamiento RIP versión 2 y la topología del sistema.

Podemos observar también como el delay ha aumentado hasta alcanzar los 23 ms, debido a que la Ruta Secundaria (ahora Ruta Principal) se compone de 5 saltos entre el ordenador origen y el ordenador destino, a diferencia de la Ruta Principal anterior que se componía de 4 saltos. Este salto de más, determina exactamente ese delay superior que corresponde al procesamiento interno del router extra en la nueva ruta.

En la Zona 4 podemos ver como el delay vuelve a los valores del principio debido a que se arregla el fallo provocado en la Zona 2 (conectando de nuevo el enlace serial entre el router 4 y el router 3). Esta transición se efectúa de manera mucho más rápida que cuando se produjo el fallo.

A continuación mostraremos los datos almacenados por el documento Excel en la Tabla 4, en los que veremos en la transición desde la Zona 2 y la Zona 3 como aumenta de golpe el número de paquetes perdidos, registrado también con un “ERROR” en el jitter del primer paquete recibido cuando se ha reestablecido la ruta, debido al tiempo que necesita el sistema para reestablecer una Ruta Principal y continuar con el envío de los paquetes.

Nº Paquete	Tamaño(bytes)	Delay(ms)	Jitter(ms)	Recibidos	Perdidos	Perdidos(%)
2464	20	15	0	2464	0	,
2465	20	15	0	2465	0	,
2466	20	16	1	2466	0	,
2467	20	15	1	2467	0	,
2468	20	15	0	2468	0	,
2469	20	15	0	2469	0	,
2470	20	15	0	2470	0	,
2471	20	16	1	2471	0	,
2472	20	15	1	2472	0	,
2473	20	15	0	2473	0	,
2474	20	15	0	2474	0	,
2475	20	15	0	2475	0	,
2476	20	16	1	2476	0	,
2477	20	15	1	2477	0	,
2478	20	15	0	2478	0	,
2479	20	15	0	2479	0	,
2480	20	15	0	2480	0	,
2481	20	16	1	2481	0	,
2482	20	15	1	2482	0	,
2952	20	23	ERROR	2483	469	15,888
2953	20	23	0	2484	469	15,882
2954	20	23	0	2485	469	15,877
2955	20	23	0	2486	469	15,871
2956	20	23	0	2487	469	15,866
2957	20	23	0	2488	469	15,861
2958	20	31	8	2489	469	15,855
2959	20	23	8	2490	469	15,85
2960	20	23	0	2491	469	15,845
2961	20	23	0	2492	469	15,839
2962	20	23	0	2493	469	15,834
2963	20	23	0	2494	469	15,829
2964	20	23	0	2495	469	15,823
2965	20	23	0	2496	469	15,818
2966	20	23	0	2497	469	15,813
2967	20	23	0	2498	469	15,807
2968	20	23	0	2499	469	15,802
2969	20	23	0	2500	469	15,797
2970	20	23	0	2501	469	15,791
2971	20	23	0	2502	469	15,786
2972	20	23	0	2503	469	15,781
2973	20	23	0	2504	469	15,775
2974	20	23	0	2505	469	15,77
2975	20	23	0	2506	469	15,765
2976	20	23	0	2507	469	15,759
2977	20	23	0	2508	469	15,754
2978	20	23	0	2509	469	15,749
2979	20	23	0	2510	469	15,744
2980	20	23	0	2511	469	15,738
2981	20	23	0	2512	469	15,733
2982	20	23	0	2513	469	15,728
2983	20	23	0	2514	469	15,722
2984	20	23	0	2515	469	15,717
2985	20	23	0	2516	469	15,712
2986	20	23	0	2517	469	15,707
2987	20	23	0	2518	469	15,701
2988	20	23	0	2519	469	15,696
2989	20	23	0	2520	469	15,691

Tabla 4. Medidas Excel 2

Por último se expone en la Figura 18 el mismo procedimiento pero esta vez enviando un tráfico de Video entre 30 kbit/s y 50 kbit/s a 24 fps. Se pueden observar el mismo comportamiento que en las zonas señaladas en la gráfica de envío de audio con el códec G.729. Las diferencias obvias, por el distinto tipo de tráfico, son la cantidad de paquetes recibidos y el bandwidth recibido. La única diferencia es que, en este caso, se observa que el delay propio del sistema con este tipo de tráfico a través de la Ruta Principal es de en torno a 75ms y una vez seleccionada la Ruta Secundaria pasa a ser de uno 125 ms. Pero el comportamiento de los parámetros resulta el mismo.

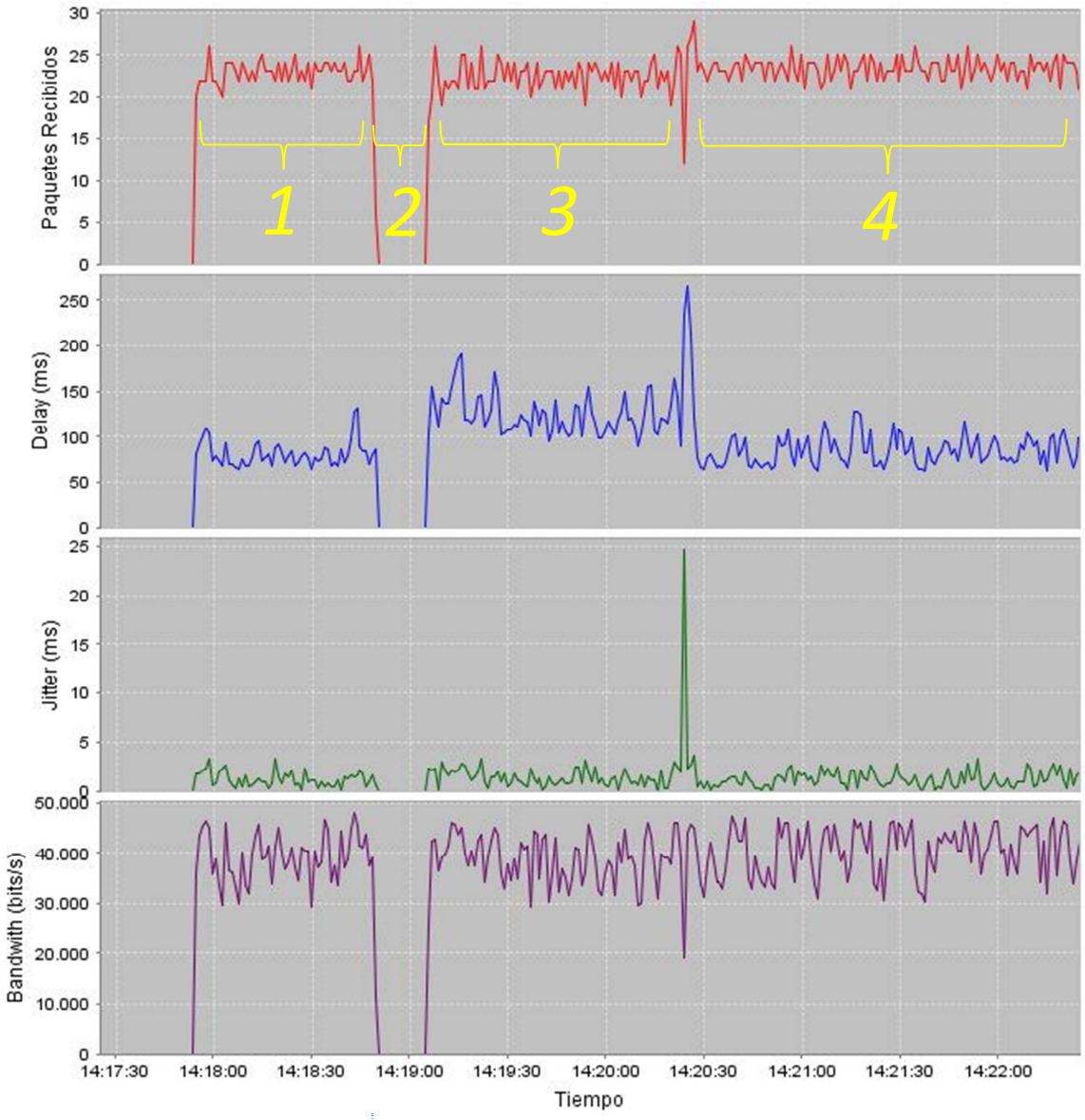

Figura 18. Gráfica Medidas 3

6 - CONCLUSIONES

En este último capítulo vamos a exponer primero si se han cumplido los objetivos propuestos para este proyecto desde un principio. Después comentaremos las conclusiones que hacemos al haber realizado la toma de medidas con la herramienta en varios escenarios y con distintos objetivos. El apartado siguiente haremos un resumen de lo que este proyecto nos ha aportado personalmente, tanto en la experiencia como el conocimiento adquirido. Y por último comentaremos las posibles líneas futuras a seguir a partir de ahora con la herramienta de testeo de la red finalmente desarrollada.

6.1 - CUMPLIMIENTO DEL OBJETIVO

El Primer objetivo alcanzado con éxito, fue el de crear una herramienta que fuera capaz de calcular los parámetros de calidad de una red de datos. Esta debía ser capaz de mandar paquetes de datos para realizar el cálculo necesario para estos parámetros

El Segundo objetivo conseguido fue implementar graficas estadísticas que se fueran actualizando en cada momento con los parámetros de calidad anteriormente calculados. Esto proporcionaría una idea inmediata al usuario del estado de la red.

El Tercer objetivo logrado fue el de permitir almacenar esos datos calculados para su posible extracción en archivos .XLS propiedad de Microsoft Excel para un futuro uso.

Estos objetivos conforman el objetivo principal que era el de crear un herramienta basada en Java, para el cálculo de los parámetros de calidad de una red de datos.

Se propusieron otros objetivos como el estudio de distintos escenarios de red mediante la herramienta, que fueron totalmente alcanzados.

6.2 - CONCLUSIONES SOBRE EL PROYECTO

Con la finalización del desarrollo de la herramienta presentada en este proyecto, queremos decir como conclusión que en este proyecto hemos presentado una herramienta que nos permite medir diversos parámetros de calidad sobre la red de manera intuitiva utilizando el sistema operativo Windows y, comparándola con las herramientas ya existentes, podemos decir que la mayor contribución está en la creación de la primera herramienta con interfaz gráfica para medir el rendimiento de red que ofrece resultados estadísticos representados en gráficas y permite almacenar los datos registrados en hojas de cálculo para su posterior estudio o utilización en futuros trabajos.

Sobre todo es un aporte más hacia el sector de la investigación telemática, un mundo en constante avance y cambio, necesitado de herramientas a la altura de las necesidades que las nuevas tecnologías llevan consigo. Poco a poco el mundo telemático se hará más amplio y complejo, y necesitamos que las herramientas sean claras e intuitivas para poder realizar estudios de manera más eficiente.

También concluir que gracias a las pruebas realizadas, explicando el contenido de estas y debatiendo los resultados obtenidos, que la herramienta descrita en el proyecto puede considerarse como óptima para la medición de los parámetros de calidad en la red. A partir de ahora se deben marcar objetivos de optimización de la herramienta y de comparación de los datos obtenidos, respecto a los obtenidos por otras herramientas similares, para comprobar el grado de fiabilidad que reflejan las medidas capturadas.

6.3 - PROBLEMAS ENCONTRADOS

Durante el desarrollo de la herramienta nos hemos ido encontrando con dificultades las cuales hemos tenido que dar solución para poder finalizar con éxito nuestra herramienta. A continuación se citaran los problemas más importantes encontrados.

Problema 1

En un principio se ideo el sistema de funcionamiento de la aplicación para que el cliente solo realizara la función de envío de paquetes y el servidor realizara la función de recibir los paquetes y la medición de los parámetros de calidad y su posterior representación en las gráficas estadísticas.

Este sistema tuvo que rechazarse debido a que en la toma de tiempos de cada ordenador para calcular los parámetros de calidad, se observaba que los relojes no eran síncronos. De hecho se ideo e implemento un sistema de sincronización al inicio del envío de los paquetes. Aunque esta solución fue inútil debido a que no solo los tiempos entre los ordenadores no estaban sincronizados, sino que el desfase temporal continuaba desajustándose unos 10 ms en las medidas realizadas, aproximadamente cada 5 s transcurridos.

Solución 1

Se ideo entonces el sistema Cliente-Servidor-Cliente para que la toma de tiempos se realizara en el mismo ordenador. Es decir, adoptar un sistema en el que el ordenador origen y destino fueran el mismo. Solo habría que adoptar unos pequeños ajustes en las fórmulas para calcular los parámetros. Debido a que el camino seguido por los paquetes seria el mismo en la ida y en la vuelta ya que, aunque hubieran diferentes rutas a seguir, anularíamos la distribución de carga en los dispositivos enrutadores de la topología y tomaríamos con éxito nuestras medidas.

Problema 2

Al realizar medidas de pruebas, nos dimos cuenta que en los tiempo de jitter se metía una constante de 11 ms a los tiempos medidos propios del jitter, la cual no podíamos explicar su origen.

Solución 2

La solución de este problema fue que al realizar las pruebas en un portátil con conexión Ethernet (que es la que queríamos utilizar en el envío) y la antena WiFi activada, el sistema no podía seleccionar correctamente por donde enviar los paquetes. Apagando la antena WiFi desaparecía el problema.

Problema 3

Vimos que al enviar paquetes muy grandes (mayores que el MTU) se producía un desajuste considerable en las medidas.

Solución 3

Aunque la división en subpaquetes se ocupa también la capa IP, decidimos que la herramienta realizara la división en subpaquetes para no saturar el sistema y realizar medidas más fiables.

Problema 4

Un error que se cometió en la programación de la aplicación, fue implementando la tabla que mostraba los parámetros de calidad de los paquetes recibidos. Esta hacia que, al cabo de un rato, la memoria del sistema se saturara al acumular en dicha tabla cientos de valores. La saturación del sistema obviamente afectaba al envío de los paquetes y a su recepción, produciendo efectos que no eran los esperados.

Solución 4

Se limitó dicha tabla a mostrar solo los últimos 10 paquetes recibidos.

Problema 5

En algunas medidas realizadas y repasándolas en el archivo de Microsoft Excel, se podía ver como cuando había perdida de paquetes, en ocasiones no marcaba el jitter como "ERROR". Esto era un fallo ya que la herramienta tenía programada que cuando hubiera pérdida de paquetes, y por lo tanto no se recibieran paquetes consecutivos, no se debía calcular el jitter.

Solución 5

Nos dimos cuenta que cuando se dividían en subpaquetes los paquetes grandes, el último subpaquete era imposible de identificar. Y por eso, cuando se recibía el primer subpaquete del paquete siguiente, se calculaba el jitter hubiera o no hubiera pérdida de paquetes. Para solucionarlo, se implementó un campo en la cabecera llamado "PaqueteFinal" que indicaba si el paquete era o no el último, en cuyo caso sí se debía calcular el jitter al recibir el primer subpaquete del paquete siguiente.

Problema 6

Al principio del desarrollo de la herramienta, se implementó la cabecera como variables "String", ya que facilitaba mucho su manejo. Entonces vimos que la cabecera ocupaba unos 45 bytes, más grande incluso que algunos códec de VoIP como G.729, imposibilitando así su simulación.

Solución 6

Se desarrolló las cabeceras en binario mediante las variables tipo "Byte", ocupando así 13 bytes en el caso de Audio y 18 bytes en el caso de Video.

Y aquí termina la lista de los problemas más importantes que nos hemos encontrado desarrollando la herramienta. Por supuesto que nos hemos encontrado muchas más dificultades, pero estas eran de menor importancia y hemos considerado no hacer constancia de ellas.

6.4 - APORTACIONES PERSONALES

Gracias a este proyecto he podido aprender mucho sobre cómo funciona una red de datos internamente y más aún cómo funciona el envío multimedia dentro de la red, ya que me ha obligado a leer bastante documentación y teoría al respecto.

Pero sobretodo me ha dado la oportunidad de aprender a trabajar en el lenguaje de programación Java. Gracias a la cantidad de libros leídos y a la información disponible en Internet he podido asimilar el concepto de programación orientado a objetos y con ello alcanzar el objetivo final que era el desarrollo de la herramienta de este proyecto.

Y por último y no menos importante, la realización de un Proyecto Fin de Carrera te proporciona una mejoría en las habilidades de organización del trabajo, la recopilación de la información y la exposición correcta en un documento del trabajo realizado.

6.5 – FUTURAS LINEAS DE ESTUDIO

Para finalizar, decir que una vez desarrollada la herramienta, ahora queda abierto un abanico de posibilidades como trabajar más sobre la herramienta o la realización de nuevos estudios.

Trabajo de optimización de la herramienta, como el uso de menos recursos del sistema de la herramienta o mejorar las cabeceras utilizadas en ella; de mejoría de la herramienta , como permitir el envío de archivos de video y audio y permitir su reproducción a su llegada, a la vez que se observa la pérdida de calidad en la red mediante los datos y asociarlos a la reproducción deficiente del archivo; y la realización de diversos estudios como la comparación de tiempos de convergencia de diversos protocolos de enrutamiento o la comparación con otras herramientas similares para comprobar la fiabilidad de sus medidas.

7 - BIBLIOGRAFÍA

[1] Página web sobre H.262, disponible en: http://www.itu.int/rec/T-REC-H.262-201303-P!Amd1/en

[2] Página web sobre H.264, disponible en: http://www.itu.int/rec/T-REC-H.264-201304-I/es

[3] Página web sobre H.265, disponible en: http://www.itu.int/rec/T-REC-H.265-201304-I

[4] Página web sobre G.711, disponible en: http://www.itu.int/rec/T-REC-G.711-198811-I/es

[5] Página web sobre G.729, disponible en: http://www.itu.int/rec/T-REC-G.729-201206-I/es

[6] Página web sobre G.726, disponible en: http://www.itu.int/rec/T-REC-G.726-199012-I/en

[7] Página web sobre G.728, disponible en: https://www.itu.int/rec/T-REC-G.728-201206-I/en

[8] Página web sobre RMON,disponible en http://www.rfc-editor.org/rfc/rfc2819.txt

[9] M. Laner et al. End-to-end Delay in Mobile Networks: Does the Traffic Pattern Matter? En ISWCS'13, Ilmenau, Alemania, 2013.

[10] José Joskowicz, Rafael Sotelo,"Medida de la calidad de voz en redes IP", Memoria de Trabajos de Difusión Científica y Técnica, 2007, pp 12-23

[11] Página web sobre Internet Protocolo, disponible en: http://www.rfc-es.org/rfc/rfc0791-es.txt

[12] Página web sobre User Datagram Protocol, disponible en: http://www.rfc-es.org/rfc/rfc0768-es.txt

[13] Página web sobre Real Time Protocol, disponible en: http://www.ietf.org/rfc/rfc3550.txt

[14] Página web sobre JFreeChart, disponible en: http://www.jfree.org/jfreechart/

[15] Página web de Java Excel API, disponible en: http://jexcelapi.sourceforge.net/

[16] Página web de SendIP, disponible en: http://www.earth.li/projectpurple/progs/sendip.html

[17] W. Feng, A. Goel, A. Bezzaz, W. Feng, J. Walpole, "TCPivo: A High- Performance Packet Replay Engine", ACM SIGCOMM 2003 Workshop on Models, Methods, and Tools for Reproducible Network Research (MoMeTools), August 2003.

[18] Página web de Rude&Crude, disponible en: http://rude.sourceforge.net/

[19] Página web de Scapy Project, disponible en: http://www.secdev.org/projects/scapy/doc/index.html

[20] Daniel Turrull Torrents, Open Source Traffic Analyzer. Master of Science Thesis. Stockholm, Sweden. June, 2010. Disponible en http://people.kth.se/~danieltt/pktgen/docs/DanielTurull-thesis.pdf

[21] Joqel Sommers, Hyungsuk Kim and Paul Barford, Harpoon: a flow-level traffic generator for router and network tests, ACM SIGMETRICS Performance Evaluation Review, Volume 32 Issue 1, June 2004. Pages 392-392

[22] Página web de Nemesis, disponible en: http://nemesis.sourceforge.net/

[23] Página web de Packet Excalibur, disponible en http://freecode.com/projects/packetexcalibur

[24] Página web de PackETH., disponible en: http://packeth.sourceforge.net/packeth/Home.html

[25] Página web de ISIC -- IP Stack Integrity Checker, disponible en http://isic.sourceforge.net/

[26] Página web de Netperf, disponible en: http://www.netperf.org/netperf/

[27] Página web de NetSpec: A Tool for Network Experimentation and Measurement, disponible en: http://www.ittc.ku.edu/netspec/

[28] Página web de Bit-Twist, disponible en: http://bittwist.sourceforge

[29] A. Dainotti, A. Botta, A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios", Computer Networks, 2012, Volume 56, Issue 15, pp 3531-3547