UNIVERSIDAD DE ZARAGOZA

CENTRO POLITÉCNICO SUPERIOR

Análisis de prestaciones de un sistema de videoconferencia comercial

Proyecto Fin de Carrera

Ingeniería de Telecomunicación

Especialidad Telemática

Carlos Fernández Barberá

Director: José María Saldaña Medina

Ponente: Julián Fernández Navajas

Dpto. Ingeniería Electrónica y Comunicaciones, Julio de 2013

Agradecimientos

En primer lugar mis agradecimientos a José Mª y Julián, director y ponente, por su gran ayuda y apoyo en todo momento, desde el principio hasta el final de este proyecto.

También agradecer a Agustín, Luis Sequeira y Luis Casadesus por la ayuda que me han prestado siempre que lo he necesitado.

Por último, mis agradecimientos a mi familia y amigos que siempre han estado ahí apoyándome.

Resumen

"Análisis de prestaciones de un sistema de videoconferencia comercial"

En la actualidad existen diversas alternativas para establecer una comunicación por videoconferencia. Aunque existen soluciones gratuitas, los usuarios de entornos empresariales demandan mejores prestaciones. Es aquí donde aparecen alternativas propietarias con el principal objetivo de garantizar mayor calidad.

El objetivo del presente proyecto es el análisis de la alternativa propietaria "*Vidyo*", novedosa por el uso de la tecnología adaptativa de capas de vídeo *(Adaptive Video Layering, AVL)* junto con la alternativa más utilizada y extendida en el mundo de la videoconferencia, *Skype*. El uso de la alternativa *Vidyo* ha sido posible gracias al acuerdo de colaboración con la empresa ORBE S.L.

En primer lugar se implementó la puesta en marcha de dos plataformas de videoconferencia, realizando la instalación y configuración de los clientes en diferentes sistemas operativos y todo ello para un entorno controlado de laboratorio.

En segundo lugar se realizó la configuración y el montaje de los diferentes escenarios de red más adecuados para obtener unas pruebas consistentes y robustas. Hemos empleado varias herramientas, entre ellas una desarrollada en un proyecto anterior, que nos ha permitido realizar y automatizar baterías de pruebas, para así realizar el calibrado de los elementos del sistema y determinar que los escenarios son adecuados.

En tercer lugar, hemos realizado las pruebas técnicas encaminadas a medir las reacciones de las aplicaciones ante los cambios en las condiciones de la red. Estas pruebas se han realizado introduciendo limitaciones de ancho de banda, pérdidas y retardos en el entorno de laboratorio. Todo ello se ha realizado mediante la programación de "scripts" automatizados en diferentes lenguajes.

Por último, mediante el análisis de los resultados obtenidos, hemos extraído conclusiones sobre las características y funcionamiento de las dos soluciones de videoconferencia, estableciendo comparaciones entre ambas plataformas.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

En los últimos años, la popularización de los dispositivos electrónicos inteligentes *(tablet, smartphone…)*, el crecimiento de Internet y el desarrollo de servicios multimedia han llevado a una serie de cambios en los hábitos de comunicación de la sociedad. Además, hoy en día nuestra movilidad es mucho mayor que hace unos años, y esto contribuye a aumentar nuestras necesidades de comunicación. En este ámbito, se han desarrollado nuevas formas de comunicación alternativas al teléfono fijo (basado en conmutación de circuitos), que utilizan conmutación de paquetes. En algunos casos se utilizan las conexiones de datos de redes móviles (GSM, UMTS, etc.) para establecer comunicaciones por medio de Internet. Un claro ejemplo es el uso de aplicaciones de mensajería instantánea a través de *smartphone*. Otro ejemplo es la transmisión de voz a través de Internet (*Voice over Internet Protocol,* VoIP). Los sistemas de videoconferencia suponen otro paso adelante, y han pasado de ser algo exclusivo de ámbitos profesionales a utilizarse masivamente por particulares.

En el ámbito particular, la videoconferencia por Internet presenta algunas ventajas claras frente al teléfono: en primer lugar, la posibilidad de ver al interlocutor proporciona una mayor sensación de cercanía y contacto, y además en muchos casos supone un ahorro, al poder realizarse sin coste. En ámbitos profesionales, las videoconferencias pueden proporcionar una comunicación más fluida con otras oficinas o con personal desplazado, e incluso pueden ser útiles para presentar productos a los clientes, ahorrando así el coste económico y medioambiental que suponen los viajes [Bie09], aportando el valor añadido de la sostenibilidad.

Estos factores han hecho que en la actualidad existan numerosas soluciones comerciales de videoconferencia, que van desde las diseñadas para ámbitos corporativos, desarrolladas por las grandes firmas de telecomunicaciones como Cisco, Alcatel, Nortel, Avaya, Radvision, etc., hasta soluciones que solamente requieren instalar una aplicación, como *Skype*, *Msn* o *Google Talk*.

1.1. Problemática

La problemática a la que se enfrentan estos sistemas es garantizar la calidad de servicio utilizando la red Internet. Al ser una red compartida por todos y sin garantías de retardo máximo (se suele decir que Internet es una red *best effort* [XN99]*,* el problema al que se tienen que enfrentar estas soluciones, si quieren tener éxito, es el de garantizar un buen servicio con las tecnologías de acceso y transporte disponibles en el mercado.

En el ámbito doméstico, especialmente en los países occidentales, hoy en día la mayoría de los usuarios disponen de un acceso ADSL, una línea asimétrica con un gran ancho de banda de bajada, pero un enlace de subida más limitado. También existe la posibilidad de conectarse a través de redes inalámbricas (por ejemplo, basadas en Wi-Fi). Los accesos asimétricos están pensados en principio para proporcionar un acceso a Internet para consulta y descarga de información, donde el usuario sólo se limita a hacer peticiones a los servidores, por lo que descarga mucha más información de la que envía a Internet. Esto supone un problema para los sistemas de videoconferencia, donde la comunicación debe

ser simétrica y se necesita un ancho banda similar en ambos sentidos. Existen soluciones con líneas dedicadas que garantizan la calidad de la comunicación, pero conllevan un gran coste que muchos usuarios no se pueden permitir. Por tanto están apareciendo soluciones capaces de garantizar la calidad utilizando la red pública y los accesos residenciales.

La red pública presenta un cierto grado de imprevisibilidad, pues las condiciones de la red varían a lo largo del tiempo por muchos factores [KPL+09]: las fluctuaciones del tráfico y la congestión de la red pueden provocar cambios en los retardos, variaciones del retardo (*jitter*), del ancho de banda y también producir pérdidas de paquetes, lo que afecta directamente a las comunicaciones y a su calidad de servicio. Los sistemas de videoconferencia deben por tanto proporcionar calidad teniendo también en cuenta la variabilidad de las condiciones en la red.

En el presente proyecto analizaremos y estudiaremos el comportamiento de algunas herramientas de videoconferencia, centrándonos en las técnicas que utilizan para proporcionar calidad en un medio hostil como son las redes públicas actuales. En concreto, veremos cómo adaptan su tráfico según el estado de la red. Para ello algunas recurren al uso de *codec* por capas, que dividen la información en una capa principal que se debe transmitir siempre, y otras secundarias que añaden un mayor nivel de detalle a la imagen.

Por ello se ha seleccionado en primer lugar *Vidyo*, una solución multipunto que utiliza una arquitectura adaptativa de capas de vídeo para ofrecer videoconferencias de baja latencia y con alta definición, cuyo uso ha sido posible gracias al acuerdo de colaboración con la empresa ORBE S.L. La arquitectura adaptativa de capas de vídeo (AVL) optimiza dinámicamente la calidad del vídeo para cada terminal aprovechando la tecnología de compresión basada en la codificación por capas, en concreto el *codec* escalable de vídeo *(Scalabe Video Codec*, SVC) H.264.

A lo largo de la memoria compararemos esta solución propietaria con *Skype,* una aplicación de videoconferencia muy popular, especialmente en el ámbito privado. Compararemos el diseño de ambas y las diferentes soluciones que utilizan para proporcionar el servicio con calidad en las redes actuales. Pensamos que esta comparación será ilustrativa, porque permitirá ver cómo a través de mecanismos y arquitecturas de red diferentes, se puede llegar a proporcionar una calidad aceptable.

1.2. Objetivos

El objetivo principal de este proyecto es el de analizar los mecanismos para proporcionar calidad en sistemas de videoconferencia comercial. Se analizarán dos sistemas diferentes, con el principal objetivo de estudiar y comparar los mecanismos que usan para adaptarse a las condiciones de la red.

Las fases y tareas del presente trabajo, para conseguir el objetivo planteado, son las siguientes:

- Análisis y puesta en marcha de la solución comercial *Vidyo* en el laboratorio. La aplicación se instalará en máquinas con diferentes sistemas operativos (Windows 7, Linux Debian). También se instalará *Skype* en el mismo entorno.
- Desarrollo, configuración y montaje de los escenarios controlados de pruebas para la realización de las medidas.
- Calibración de todos los elementos y equipos del escenario para las pruebas.
- Configuración de las herramientas necesarias para llevar a cabo las medidas. Esto conlleva tanto la instalación y configuración de aplicaciones, así como el desarrollo de *script* para automatizar la captura de medidas y el procesado de los datos obtenidos.
- Estudio del comportamiento frente a limitaciones de ancho de banda, retardo y pérdidas de paquetes.
- Análisis de los resultados obtenidos.

1.3. Organización de la memoria

El presente proyecto está dividido en cinco capítulos. En el primero hacemos una introducción donde explicamos el porqué de este estudio, los objetivos y su problemática.

El segundo capítulo nos pone en situación sobre los sistemas de videoconferencia desde sus comienzos hasta el día de hoy. Describe las diferentes arquitecturas de red de los sistemas de videoconferencia y su funcionamiento a través de Internet.

El tercer capítulo describe el entorno de pruebas de laboratorio y las herramientas utilizadas para llevar a cabo el estudio.

El cuarto capítulo describe las pruebas realizadas y los resultados obtenidos.

Por último, en el quinto capítulo exponemos las conclusiones, tanto de los resultados obtenidos como del trabajo realizado durante este tiempo.

También se incluyen, como anexos a la memoria, otras secciones con información complementaria sobre el estudio realizado.

2. Estado del arte

En este capítulo situaremos el contexto del estudio realizado. Comenzaremos con una descripción de la evolución de los sistemas de videoconferencia en los últimos años, principalmente marcada por el uso de Internet. Posteriormente trataremos de los distintos modelos que se pueden utilizar para proporcionar el servicio, y terminaremos describiendo los mecanismos que permiten mejorar la transmisión de vídeo por Internet, como por ejemplo los *codec* por capas (escalables) o la adaptación del ancho de banda enviado.

2.1. Uso de Internet para videoconferencia

Los primeros intentos de desarrollar sistemas de videoconferencia utilizando las líneas convencionales de telefonía fracasaron por el insuficiente ancho de banda, las ineficientes técnicas de compresión de vídeo y el alto coste. Por ejemplo, AT&T en el año 1969 comercializó una solución de videotelefonía *Picturephone* (Fig.2.1) [Bell69], pero sólo consiguió vender unos pocos centenares en todo el mundo, por lo que las posibilidades de realizar una llamada eran casi inexistentes.

Fig.2.1. Videoteléfono AT&T (*Picturephone* 1969)

Con la llegada de las redes de transmisión digital, como RDSI, con un ancho de banda de 128 kbps, se hizo ya posible establecer videoconferencias, aunque el coste era todavía elevado. Una de las primeras compañías en comercializar este producto fue PictureTel Corp, fundada en 1984, y adquirida por Polycom en 2001. Estas comunicaciones se establecían en salas especiales, ya que se necesitaban equipos específicos y una línea dedicada, por lo que eran utilizadas solamente en el mundo de los negocios, o también en soluciones de telemedicina.

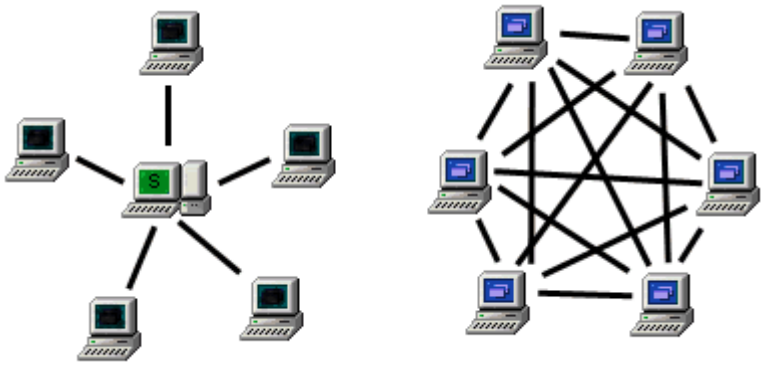
A partir de 1990 se desarrollan sistemas de videoconferencia basados en Internet, que incluían técnicas de compresión más eficaces, para poder así realizar videoconferencias desde el escritorio de un usuario particular. A partir del año 2000 se popularizó el uso de la videoconferencia con aplicaciones gratuitas de escritorio, que funcionaban a través de las líneas de conexión a Internet, como por ejemplo *Skype*. En el año 2005 aparecen los

primeros sistemas de videoconferencia de alta resolución, como por ejemplo los desarrollados por Polycom, Lifesize o Tandberg [Zdn05], [Jkc05a], [Jkc05b].

Actualmente se trabaja en el desarrollo de técnicas de codificación de vídeo más eficientes para conseguir videoconferencias de alta resolución no sólo desde un PC conectado a través de una línea fija, sino también desde dispositivos portátiles y usando también redes inalámbricas (GSM, UMTS, LTE, WiMAX, Wi-Fi).

2.2. Modelos para proporcionar el servicio

Podemos encontrar dos modelos básicos para proporcionar el servicio de videoconferencia: cliente-servidor y P2P *(peer to peer)*. La diferencia entre ellos radica en la manera y lugar donde se realizan algunas de las funciones. En el primer modelo existe un nodo central mientras que en el otro se usa un esquema descentralizado (Fig.2.2).



Estructura cliente-servidor Estructura P2P

Fig.2.2. Modelos de red para soportar videoconferencias.

2.2.1 Arquitectura basada en servidor

En estas arquitecturas centralizadas podemos encontrar dos tipos de nodos: por una parte los nodos servidor, que tienen gran capacidad de procesamiento y gran ancho de banda, para poder ofrecer contenidos a los nodos cliente. Éstos disponen de menor potencia de proceso y de una conexión más limitada, siendo esencialmente consumidores de los contenidos ofrecidos por el servidor.

Al incluir un control centralizado de los accesos y los recursos, la gestión de sistemas basados en servidor resulta más fácil. La escalabilidad se consigue mediante la adición de nuevos componentes en el servidor central. Al ser sistemas centralizados, se deben buscar mecanismos para incrementar su robustez, como el uso de servidores con redundancia en caso de fallo.

Este es el caso de *Vidyo,* la solución que vamos a estudiar, donde los elementos del sistema tienen unas denominaciones concretas [Vid13] como puede verse en la Fig.2.3.

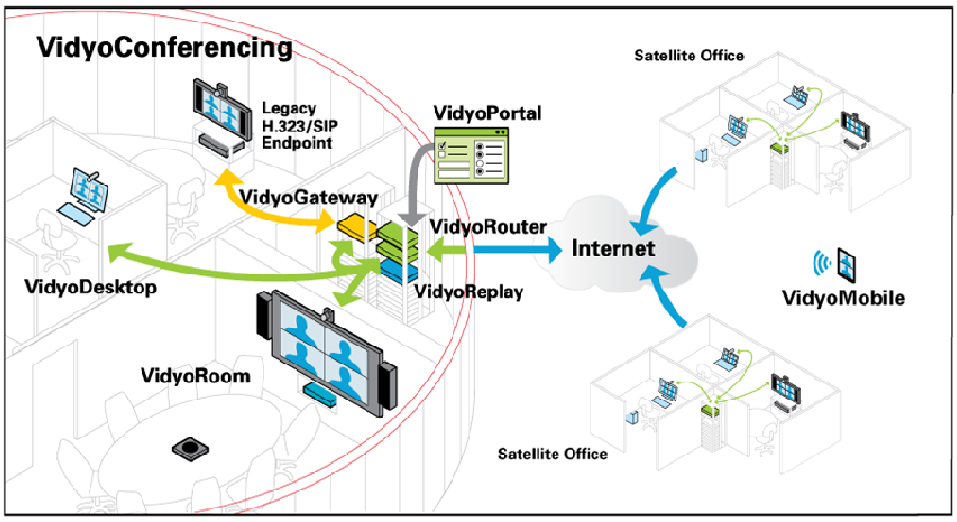


Fig.2.3. Elementos del sistema *Vidyo*

El equipo central (denominado *VidyoRouter*) se encarga de establecer las comunicaciones entre dos terminales. Por tanto, todas las comunicaciones pasan a través de él. Es el encargado de gestionar el ancho de banda necesario para que se produzca una comunicación de calidad. Es capaz de adaptar el tráfico a las capacidades de cada dispositivo terminal, enviando así a cada uno el formato y la calidad máxima que puede aceptar o decodificar, teniendo también en cuenta las limitaciones de la red. Por lo tanto, es este equipo quien se encarga de que la comunicación entre las personas que participan en la videoconferencia siga siendo fluida a pesar de los cambios en las condiciones de la red.

Como se aprecia en la Fig.2.3., el *VidyoRouter* permite establecer videoconferencias entre dispositivos específicos, como los que puede haber en una sala especial para conferencias, ordenadores fijos o portátiles, e incluso dispositivos móviles, conectados a Internet a través de una red inalámbrica.

El equipo pasarela (denominado *VidyoGateway*) es el encargado de traducir otros protocolos de videoconferencia (H.323, SIP) al protocolo de *Vidyo* para su adaptación, y viceversa, de modo que la comunicación entre los diversos sistemas sea posible.

Para participar en una videoconferencia utilizando plenamente el protocolo de *Vidyo* y usando un terminal convencional conectado a Internet, el usuario debe instalar un software (denominado *VidyoDesktop*) en su terminal (Fig.2.4). En él podemos configurar diversos parámetros como la velocidad de transmisión, velocidad de recepción, resolución de vídeo, etc. Permite realizar llamadas directas a usuarios de la lista de contactos o realizar llamadas multipunto creando una sala virtual. Está disponible para la mayoría de sistemas operativos actuales (Windows, Linux, Android, Apple iOS).

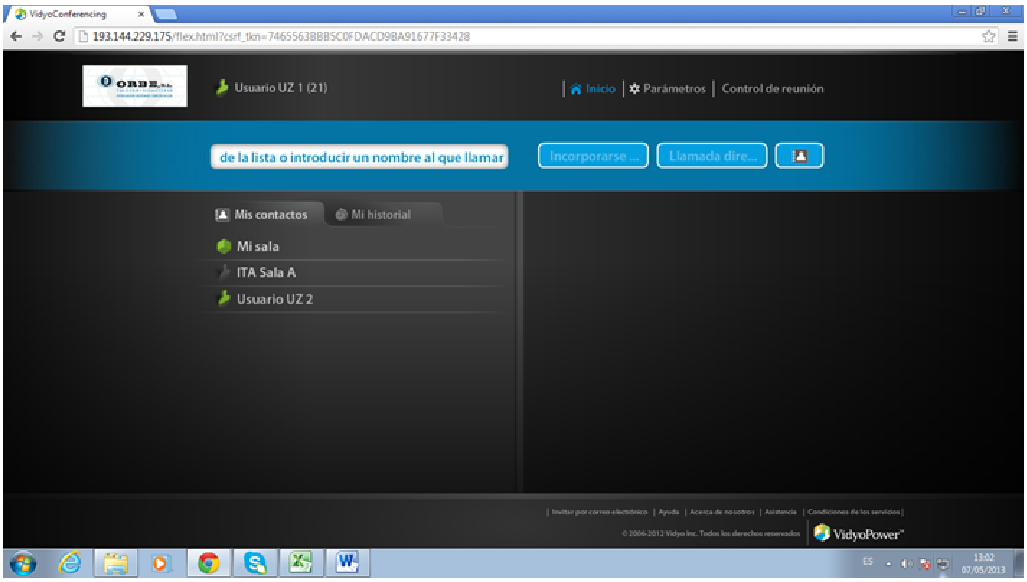


Fig.2.4. Aplicación de escritorio de *VidyoDesktop* en Microsoft Windows.

En conclusión, podemos decir que la fortaleza del sistema *Vidyo* reside en la manera en que el *VidyoRouter* es capaz de determinar los parámetros de calidad de servicio de forma dinámica, y transmitir así a cada usuario las capas de vídeo que precise, atendiendo a las condiciones de la red y a las capacidades de su dispositivo de visualización. Para ello utiliza la tecnología de codificación de vídeo H.264/SVC.

2.2.2 Arquitectura P2P (peer to peer)

En las redes P2P no existen clientes y servidores fijos, sino un número de nodos que funcionan a la vez como clientes y como servidores para el resto de nodos de la red. No existe jerarquía, por lo que se puede añadir un nuevo nodo fácilmente permitiendo así el crecimiento de la red. Las características de estas redes son la escalabilidad, robustez, descentralización y la distribución de costes entre los nodos. Su inconveniente es que pueden llegar a ser redes difíciles de gestionar. Sin embargo, todas estas redes deben disponer de una parte centralizada, que incluya por ejemplo las funciones de autenticación de los usuarios, gestión de cuentas, etc.

La arquitectura P2P es muy usada en sistemas de tiempo real de *streaming* de audio y vídeo [ROA07]. Uno de los principales ejemplos de arquitecturas P2P en la actualidad es *Skype* [GDJ06]. Aunque la autenticación utiliza un modelo cliente-servidor, el resto de la señalización está descentralizada y distribuida entre los nodos. En la literatura [XR07] se han identificado dos tipos de participantes o nodos en la red *Skype*: los *supernodos*, que son aquellos que tienen una dirección pública; y los nodos normales, que son el resto (Fig.2.5). Los *supernodos* desempeñan la función de retransmitir el tráfico a los nodos normales que se encuentran detrás de un *firewall* o usan direcciones de red locales, por lo que deben recurrir al protocolo NAT *(Network Address Translation)*. Los *supernodos* forman una red P2P entre sí.

En lo que se refiere a la seguridad, podría parecer que las comunicaciones en las redes P2P son más vulnerables, pero en el caso de *Skype* se utiliza un sistema de cifrado robusto. Otra ventaja de no usar un servidor central es que se obtiene un menor retardo en la

comunicación, ya que el tráfico se envía directamente entre los clientes, sin pasar por ningún otro elemento de la red.

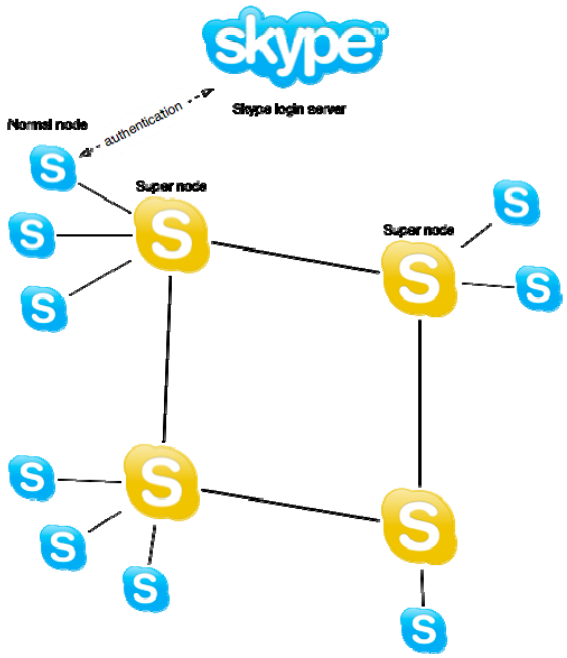


Fig.2.5. Red P2P de *Skype*.

2.3. Mecanismos de adaptación a los cambios en la red

Como hemos visto, los sistemas de videoconferencia utilizan diferentes métodos para adaptarse a la variabilidad de las condiciones de la red. En esta sección resumiremos dos de esos mecanismos: la codificación por capas y la adaptación del tráfico generado.

a) Codificación por capas de vídeo H.264/SVC

SVC es una técnica que divide el flujo de vídeo en múltiples resoluciones, niveles de calidad y velocidades de fotogramas. Está especialmente indicado para aplicaciones donde el ancho de banda no está garantizado (vídeo por Internet, videollamadas y comunicaciones inalámbricas). SVC fue diseñado inicialmente para soportar comunicaciones de vídeo en un solo sentido a través de la red conmutada de paquetes [Dav06]. *Vidyo* es la primera compañía que aplica la tecnología SVC en ambos sentidos de la comunicación.

SVC es la extensión de escalabilidad añadida al *codec* H.264/AVC *(Advanced Video Coding)* [OMG13]. AVC y su extensión SVC fueron estandarizados en un esfuerzo conjunto del *Joint Video Team* perteneciente al ITU-T VCEG y al ISO/IEC MPEG. El término "escalable" hace referencia a la estructura de SVC, gracias a la cual es posible escalar un vídeo, codificado con determinados parámetros, para una representación reducida sin la intervención de un nuevo proceso de codificación. SVC proporciona tres tipos de escalabilidad:

- En la dimensión del tiempo, permitiendo distintas frecuencias de muestreo en imágenes por segundo (FPS),

- En la dimensión del espacio, relativa al tamaño de la imagen,

- En la dimensión de la calidad, permitiendo distintos valores de la relación señal/ruido (SNR).

Cada una de las posibles combinaciones de estas tres características en un vídeo codificado con SVC se denomina *capa*.

El propósito de cualquier algoritmo de compresión de vídeo es explotar la redundancia espacial y temporal de información para que la calidad de vídeo sea aceptable en el otro extremo [Dav06]. Mientras que un codificador de vídeo no escalable genera un único flujo de bits comprimido, un codificador escalable comprime el vídeo en una secuencia de múltiples flujos de bits o capas (Fig.2.6). A la primera de las capas comprimidas se le llama *capa base*. La capa base se puede codificar de forma independiente y proporciona un nivel relativamente bajo de calidad de vídeo. Las demás capas adicionales son *capas de mejora*, y proporcionan información adicional, mejorando la calidad de la secuencia de vídeo. Las capas de mejora pueden ser decodificadas sólo junto con la capa de base. El flujo de bits completo consiste por tanto en la capa de base y un número de capas de mejora.

Como se aprecia en la Fig.2.6, si se garantiza la correcta recepción de la capa base, el usuario final podrá ver el vídeo con un nivel mínimo de calidad. Las capas de mejora se transmitirán sólo si resulta posible, y añadirán un mayor nivel de detalle a la imagen. Una opción es enviar la capa base por un canal más fiable, y el resto de capas por otro con mayor probabilidad de error. También se puede usar este sistema para ir añadiendo o quitando capas en función del ancho de banda disponible en cada momento, o de las capacidades del dispositivo del usuario receptor del vídeo (resolución de la pantalla, capacidad de procesado, etc.).

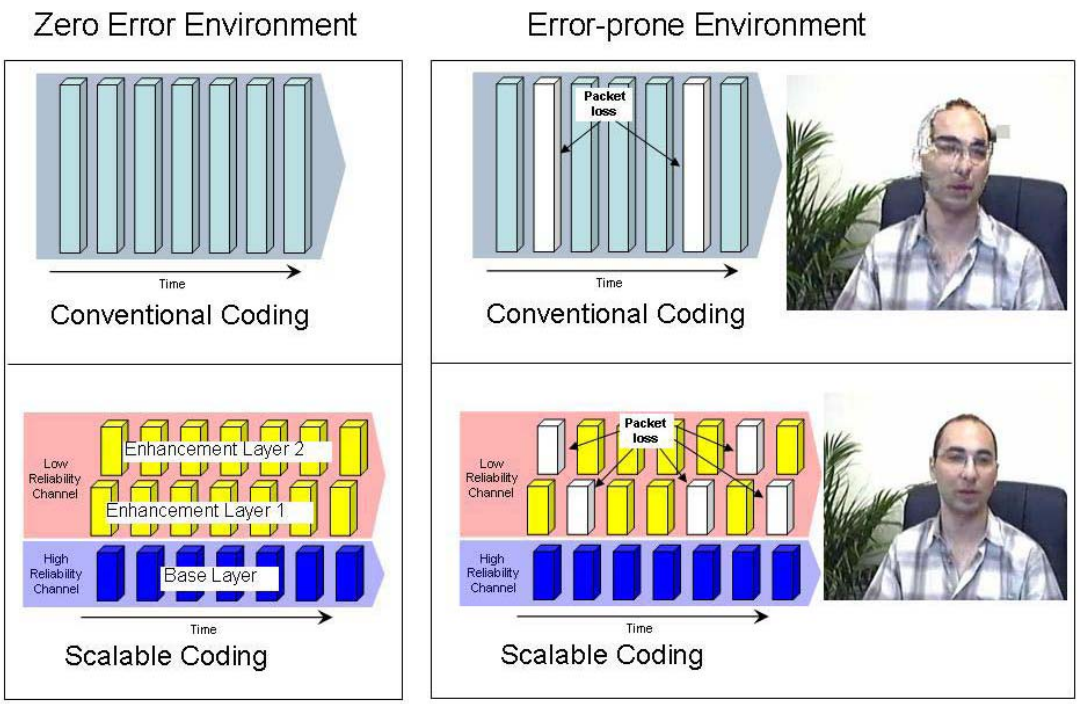


Fig.2.6. Principios de la codificación SVC

b) Adaptación del tráfico

Además de utilizar un *codec* robusto y escalable, los programas de videoconferencia pueden recurrir a medidas reactivas frente a los cambios de la red. En la literatura, algunos artículos han estudiado estas medidas, principalmente en *Skype*. En [BMR08] y [RMM08] se estudió esta aplicación, teniendo en cuenta que el código fuente no está disponible, por lo que mediante pruebas experimentales se obtuvieron algunas conclusiones respecto a su comportamiento de adaptación al medio hostil en el que debe funcionar.

Algunos de los mecanismos de adaptación que se identificaron son:

- En la capa de transporte *Skype* utiliza normalmente UDP, que es el protocolo más adecuado para servicios de tiempo real. Sin embargo, cuando no es posible utilizarlo a causa del NAT o de las políticas de *firewall, Skype* es capaz de funcionar sobre TCP.

- Durante los 20 segundos iniciales de la comunicación, *Skype* aplica redundancia en sus mensajes para determinar el estado de la red, enviando paquetes de tamaño grande y pequeño de manera alterna. Pasado este tiempo, y si las condiciones de la red son buenas, deja de mandar los paquetes de tamaño grande. Esto hace que el ancho de banda enviado durante los primeros segundos sea mayor que el generado después.

- Frente a las variaciones de la red, *Skype* aplica diferentes técnicas para adaptarse al medio. Cuando detecta pérdidas, reacciona retransmitiendo paquetes, lo que se traduce en la aparición de paquetes de tamaño grande. Sin embargo, cuando detecta que el ancho de banda disponible es insuficiente, *Skype* reacciona bajando la tasa de transmisión al mismo tiempo que aumenta el tiempo entre paquetes.

Vemos por tanto que el éxito de *Skype* no es casual. Todos estos mecanismos le permiten funcionar en muy diversas circunstancias adaptándose a las condiciones de la red, y proporcionar calidad de servicio en entornos muy variados.

3. Escenarios y sistema de pruebas

3.1 Escenarios

Se ha diseñado un esquema de red que permite la realización de pruebas para escenarios similares a los presentes en soluciones de videoconferencia reales. El esquema se muestra en la Fig.3.1. En un extremo hemos situado a un usuario con una cámara, reproduciendo un vídeo de fútbol americano[1], y en el otro se sitúa un usuario observador, que solamente recibe vídeo. Por lo tanto, sólo habrá tráfico de vídeo en un sentido, que es lo que necesitamos para realizar las medidas y analizar los resultados. Los dos terminales están situados en nuestro laboratorio (Lab. 2.03, Edificio *Ada Byron)*. En el caso de *Vidyo,* ambos usuarios utilizan para comunicarse el servidor de la videoconferencia, que está ubicado en otro lugar de la red, concretamente en la red de investigación de Aragón (RIA).

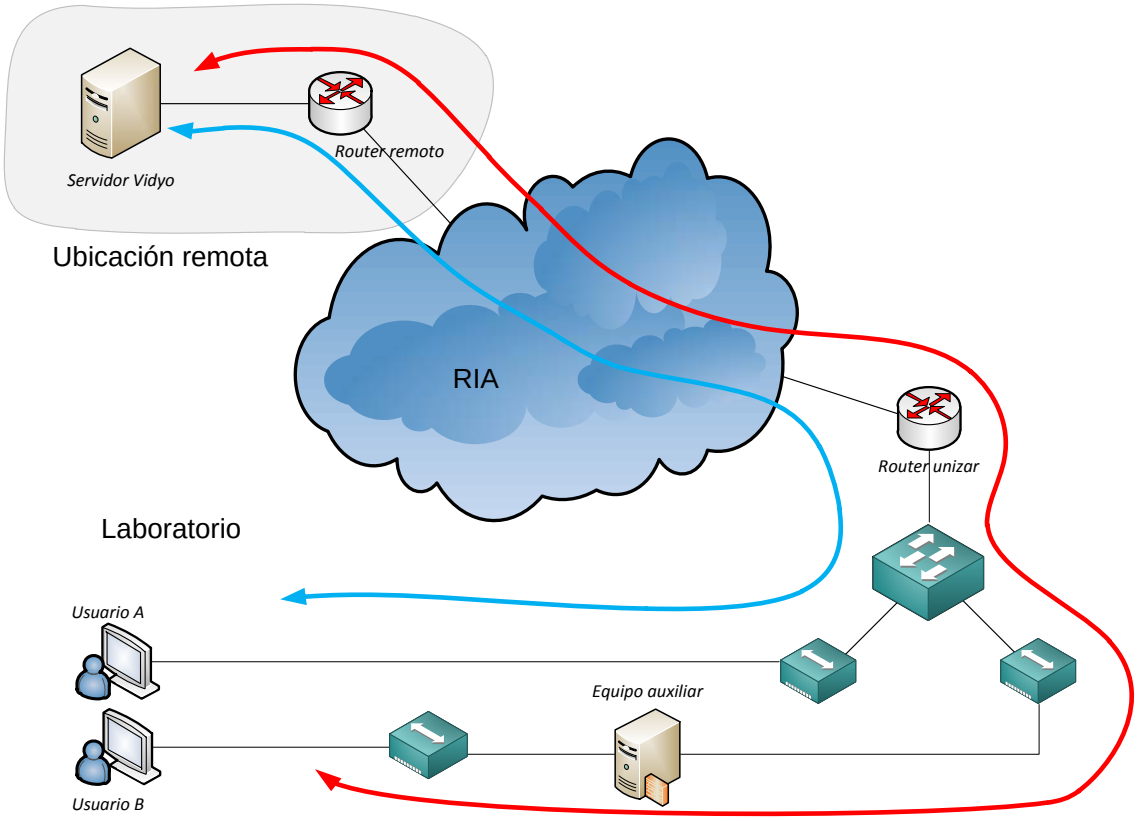


Fig.3.1. Esquema de la comunicación de videoconferencia.

[1] Este vídeo, conocido como "football" es usado frecuentemente en artículos de investigación, como un ejemplo de vídeo con mucho movimiento (http://media.xiph.org/video/derf/)

Entre ambos usuarios y el servidor hemos situado otros elementos que nos facilitarán la realización de las pruebas. En primer lugar se incluyen dos *router*, correspondientes a la red de la Universidad y a la ubicación remota en RIA. También hemos incluido un conjunto de *hub*, que nos permitirán hacer las capturas en el laboratorio. Finalmente, se añade un equipo auxiliar para simular diferentes condiciones y limitaciones en la red. Esto da lugar a dos escenarios diferentes, en función de dónde se introduzcan las condiciones y limitaciones de red:

- El primer escenario, en el que estudiaremos el enlace de subida, lo denominaremos *escenario de emisión*. En este caso, el *usuario B* tiene la cámara, y genera el tráfico hacia el servidor de la videoconferencia. El equipo auxiliar se encargará de introducir las limitaciones al sistema.
- El segundo escenario nos servirá para estudiar el enlace de bajada, y lo denominaremos *escenario de recepción*. En este caso, el *usuario A* tiene la cámara y genera el tráfico. Las limitaciones se introducen en el tráfico que va desde el servidor de la videoconferencia hacia el *usuario B*, que actúa como observador.

3.2 Herramientas utilizadas en las pruebas

A continuación, describiremos las aplicaciones y herramientas necesarias para llevar a cabo las pruebas encaminadas a caracterizar el funcionamiento de este sistema en los escenarios descritos.

3.2.1. Proxy ARP

Hemos visto que para realizar las pruebas es necesario introducir un equipo auxiliar en medio de la comunicación, de modo que podamos modificar las condiciones de la red y, por tanto, de la videoconferencia. Para ello hemos optado por introducir un *Proxy* ARP (*Address Resolution Protocol*) para el usuario B, de tal forma que la presencia de este equipo le resulte transparente. De este modo conseguimos tener un elemento intermedio donde aplicar las diferentes limitaciones de red, y que también utilizaremos para capturar el tráfico. ARP es un protocolo muy utilizado en redes Ethernet, que traduce direcciones IP a direcciones MAC *(Medium Access Control)*. Un *Proxy* ARP responde a todas las peticiones ARP originadas o destinadas a las direcciones IP a quien representa, resolviendo su dirección MAC. El uso de un *Proxy* ARP es una solución factible, que no requiere un alto coste computacional ni equipos especiales. La configuración del *Proxy* ARP la podemos consultar en el anexo B. En ambos casos, el equipo *Proxy* se encontrará en la red local. El *Proxy* ARP utilizado es el que viene por defecto en la distribución Linux Debian (versión 2.6.32-5-amd64).

3.2.2. Traffic Control

La herramienta *Traffic Control (TC)* [Hu12], incluida dentro de la distribución de Linux Debian, permite conformar y ajustar el tráfico en una interfaz de red. El proceso es controlado a través de tres tipos de objetos: colas, clases y filtros.

TC soporta diferentes disciplinas de cola (denominadas *qdisc)*, que permiten gestionar la forma en que se envían y reciben los datos. Su funcionamiento se basa en reordenar, retrasar o descartar los paquetes. A su vez, las colas pueden tener clases, lo que permite tener varias subcolas. Además, cada clase puede estar dividida en subclases y así sucesivamente. Con esto se consigue clasificar los paquetes y aplicar criterios diferentes a cada tipo de tráfico. Para encolar los paquetes en las distintas colas se pueden utilizar filtros. La disciplina aplicada por defecto con *TC* es una cola sin clase denominada *qdisc pfifo_fast*, que se organiza en tres bandas, cada una con diferente prioridad y aplica un procedimiento FIFO *(First Input First Output)* en cada una (Fig.3.2).

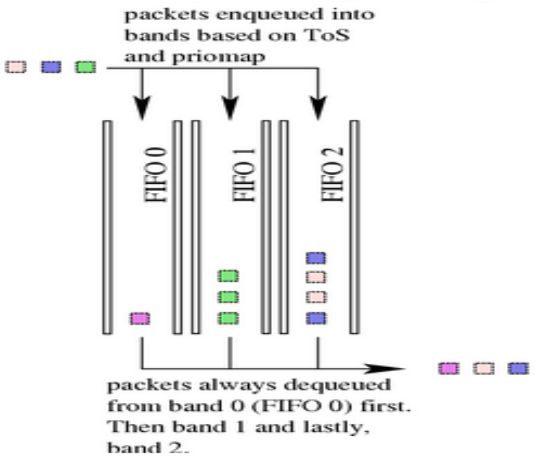


Fig.3.2. Esquema de las colas FIFO de *Traffic Control.*

En nuestro caso no es necesario el uso de clases, ya que simplemente se busca establecer un determinado ancho de banda, aplicar pérdidas o retardar los paquetes. Para ello, se ha seleccionado el filtro *Token Bucket Filter* (TBF) que se limita a dejar pasar paquetes a una tasa que no exceda la impuesta como parámetro.

Su funcionamiento utiliza el método *Leaky Bucket* (Fig.3.3), que consiste en un *buffer* llamado *bucket* que se llena constantemente de *token* a una tasa específica *(token rate)*. El tamaño del *buffer* se puede definir mediante el parámetro *burst* que indica el número de bytes de *token* que pueden almacenarse. Los datos entrantes se almacenan en otro *buffer,* a la espera de que haya *token* disponibles para ser procesados. Si los datos llegan a una tasa menor que los *token*, estos últimos se almacenaran en el *bucket* a la espera de que lleguen más datos. Esto permitirá ráfagas de datos a tasas mayores. Sin embargo, si la tasa de llegada de los datos es mayor que la tasa de *token,* habrá un momento en que los paquetes de datos comenzarán a ser descartados tras haber sobrepasado el límite de espera en el *buffer*, evitando así añadir grandes retardos, pero provocando pérdidas.

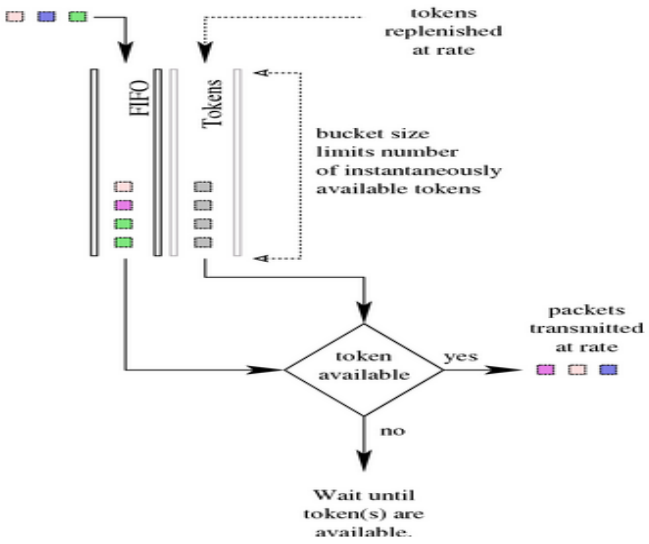


Fig.3.3. Esquema de la cola TBF de *Traffic Control*.

3.2.3. Capturas del tráfico

Durante todo el proceso, desde el inicio de la comunicación hasta su fin, debemos capturar el tráfico para su posterior procesado y análisis. Para ello utilizamos la herramienta de GNU/Linux *tcpdump* [Tcp13], que permite capturar el tráfico en una interfaz de red utilizando la librería de captura de paquetes *libpcap*, es decir, nos permite monitorizar no sólo los paquetes que vienen dirigidos a dicha interfaz, sino todos los que circulan por la red local (a esto se le denomina *modo promiscuo)*. A su vez, se pueden definir una serie de parámetros o filtros para seleccionar el tráfico que queremos capturar o monitorizar.

En nuestro caso capturaremos todo el tráfico en las interfaces de red indicadas y lo almacenaremos en ficheros, otra de las utilidades de *tcpdump*. Posteriormente, en la fase de procesado definiremos los parámetros y reglas, seleccionando así el tráfico que nos interesa de los ficheros que hemos obtenido anteriormente.

3.3 Equipos utilizados en las pruebas

Se enumeran a continuación los equipos que conforman los escenarios y el sistema de pruebas en nuestra red local, correspondiente al escenario de bajada (Fig.3.4). El esquema de red del escenario de subida consiste en intercambiar las máquinas miniPc2 y miniPc4. Otras características se detallan en el anexo E.

- MiniPc2 y MiniPc4 (Windows 7, procesador Intel core i3 CPU M370 2,4Ghz, tarjeta de red Realtek PCIe GBE Family Controller): Equipos donde corren las aplicaciones de videoconferencia.
- MiniPc3 (Linux Debian versión 2.6.32-5-amd64, procesador Intel core i3 CPU M370 2,4Ghz, tarjeta de red Realtek PCIe GBE Family Controller): *Proxy ARP*, equipo donde se aplican las limitaciones y se captura el tráfico.
- Hub1, Hub2 y Hub3: concentrador 3Com de 100Mbps (modelo 3C16611 SuperStack® II Dual Speed Hub 500 24-Port TP).

- Switch: Conmutador 3Com de 100Mbps (modelo 3CR17561-91 SuperStack®3 Switch 4500 26-Port).
- Tarjetas de red externas USB de D-Link (modelo DUB E100 usb 2.0 *fast Ethernet adapter*).
- Cámara de vídeo Logitech (modelo Quickcam Orbit AF).

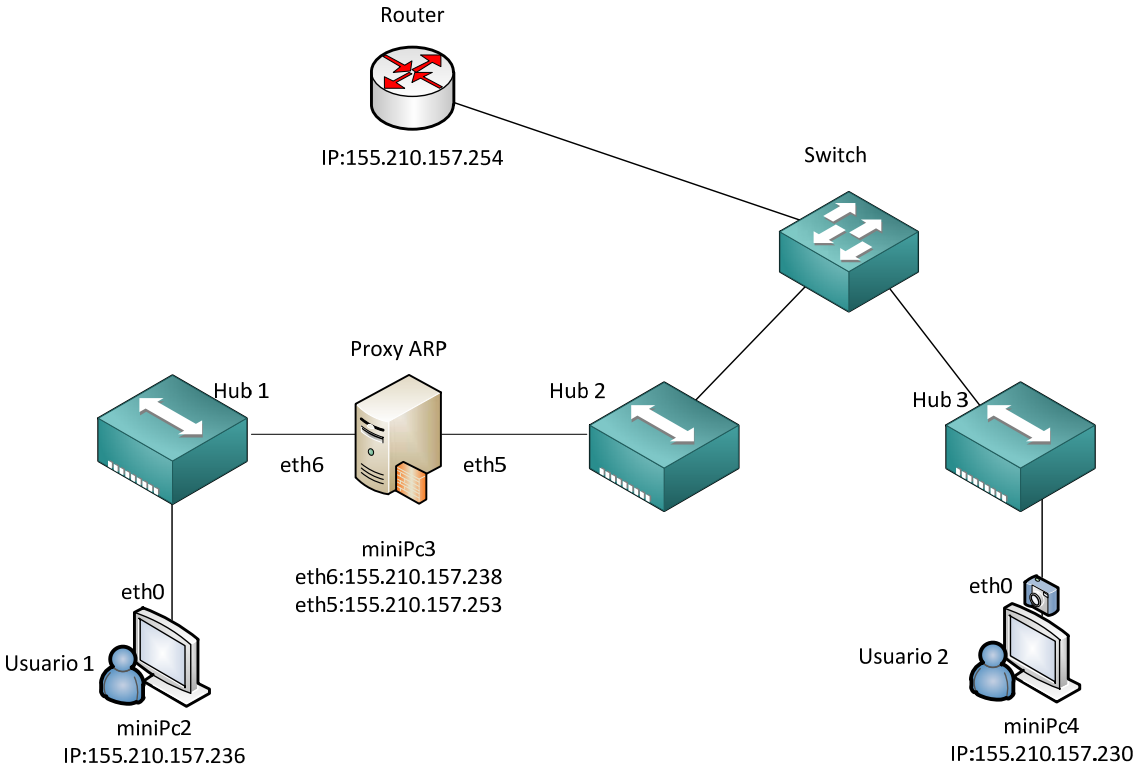


Fig.3.4. Esquema de la red local del laboratorio

La configuración de la red y su direccionamiento en los diferentes escenarios los podemos ver en las Tablas 3.1 (escenario de bajada) y 3.2 (escenario de subida).

Tabla 3.1. Direccionamiento IP, escenario con limitaciones en la bajada.

Equipo	Interfaz	Dirección IP
miniPc2	Eth0	155.210.157.236
miniPc3	Eth5	155.210.157.253
	Eth6	155.210.157.238
miniPc4	Eth0	155.210.157.230

Tabla 3.2. Direccionamiento IP, escenario con limitaciones en la subida.

Equipo	Interfaz	Dirección IP
miniPc2	Eth0	155.210.157.230
miniPc3	Eth5	155.210.157.253
	Eth6	155.210.157.238
miniPc4	Eth0	155.210.157.236

Para más detalle sobre la configuración a nivel de red de los equipos y las interfaces de red, podemos consultar el anexo B.

3.4 Calibración de las herramientas a utilizar en las pruebas

Como paso previo a la realización de las pruebas, es necesario caracterizar las herramientas de medida, y su correcto funcionamiento en el escenario. Nuestro objetivo es medir anchos de banda y otras propiedades del tráfico, por tanto, debemos comprobar que las herramientas son capaces de limitar y medir este parámetro con precisión. Por otra parte, debemos comprobar el correcto funcionamiento de todos los elementos de la red, y asegurar que no introducen imprecisiones en las medidas, y que no modifican las condiciones de las pruebas. Así podremos asegurar que las medidas y los resultados obtenidos son fiables.

3.4.1 Caracterización de la herramienta Proxy ARP.

En este caso debemos asegurar que el *Proxy* ARP cumple su función, sin añadir retardos adicionales a las pruebas. Para ello hemos utilizado otra herramienta desarrollada en nuestro grupo de investigación de la Universidad de Zaragoza llamada *ETG* (*E2E Traffic Generator*) [SFR+09], que permite generar tráfico de diferente tipo entre dos extremos. Por tanto, generaremos tráfico conocido desde una máquina origen a otra de destino pasando por la máquina donde está corriendo el *Proxy*, asegurando de esta forma su buen funcionamiento.

3.4.2 Caracterización de la herramienta TC (Traffic Control).

En este caso haremos pasar un tráfico de ancho de banda conocido a través de la herramienta *TC*, para comprobar que las limitaciones introducidas son las deseadas. También haremos uso de la herramienta *ETG* para generar el tráfico.

En primer lugar generamos tráfico uniforme con una tasa de 2 Mbps con origen en *miniPc2* y destino *miniPc4* atravesando el equipo *Proxy* donde está corriendo *TC* (Fig.3.5) y en segundo lugar generaremos tráfico a ráfagas entre 1,2 Mbps y 0,6 Mbps (Fig.3.6). En las

figuras se observa el tráfico de entrada y el tráfico de salida, una vez limitado por la herramienta. Se ha medido el ancho de banda medio cada segundo *(tick* de 1 seg.). A partir del primer minuto ponemos en funcionamiento la herramienta imponiendo diversas limitaciones al tráfico de entrada. En el caso del tráfico uniforme (Fig.3.5), vemos que *TC* es capaz de limitar el tráfico de salida con gran precisión. En el caso del tráfico a ráfagas (Fig.3.6), vemos que *TC,* mediante el uso de su *buffer,* puede "alisar" el tráfico, pasando a ser mucho más uniforme a la salida que a la entrada. Con estos ejemplos hemos comprobado el correcto funcionamiento de la herramienta para su aplicación en la medida de una comunicación de videoconferencia real.

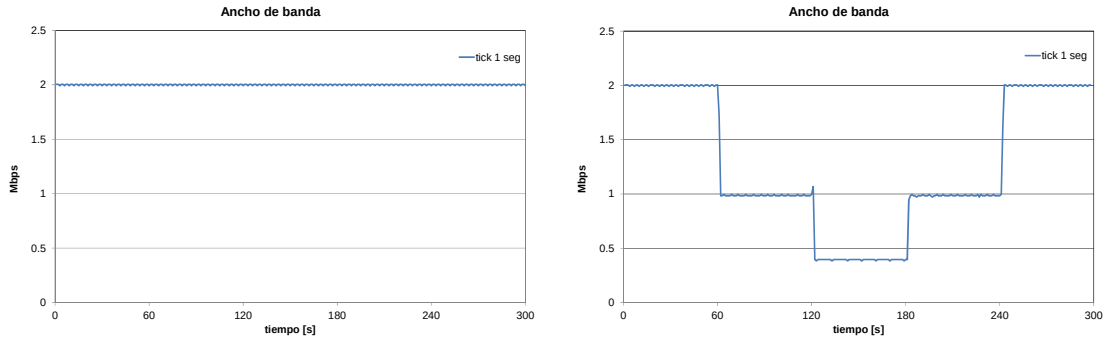


Fig.3.5. Tráfico de entrada y salida de *TC* (tráfico uniforme).

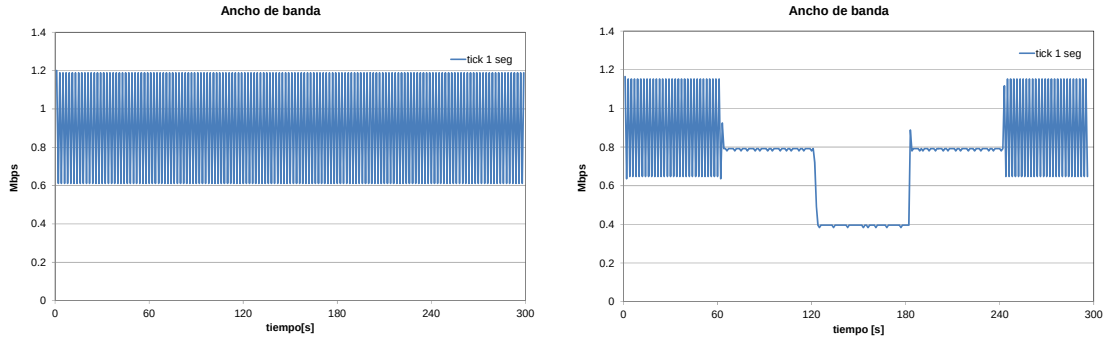


Fig.3.6. Tráfico de entrada y salida de *TC* (tráfico a ráfagas)

3.4.3 Caracterización del ancho de banda de la red

Para el establecimiento de una videoconferencia necesitamos unos requisitos mínimos de ancho de banda, que en conferencias de alta calidad pueden suponer hasta 2 Mbps en cada sentido de la comunicación. Por tanto, tenemos que comprobar que en nuestro entorno, sin introducir limitaciones, podemos garantizar estos anchos de banda. La herramienta que utilizamos en este caso se llama *IPERF* [Iperf] que nos permite medir el ancho de banda disponible entre dos extremos cualesquiera de una red. El funcionamiento es sencillo, y basta con ejecutar el servidor en un extremo y el cliente en otro. Los resultados de estas pruebas (Fig.3.7), como era de esperar, están muy por encima de los requisitos que necesitamos. El ancho de banda disponible en nuestra red está cerca de los 90 Mbps.

```
------------------------------------------------------------
Client connecting to 155.210.157.230, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 155.210.157.236 port 33257 connected with 155.210.157.230 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3] 0.0-60.0 sec   643 MBytes   89.8 Mbits/sec
```

Fig.3.7. Resultado del ancho de banda obtenido con la herramienta *IPERF*.

Una vez asegurado el correcto funcionamiento de las herramientas que intervienen en el entorno de pruebas, podremos proceder a la caracterización de las aplicaciones de videoconferencia, que consistirá en modificar las condiciones de la red mientras se realiza una comunicación.

3.5 Versatilidad del entorno de pruebas

En los apartados anteriores hemos visto la topología, los equipos y las herramientas a usar en el sistema de pruebas. Se ha buscado un diseño versátil, que permita numerosas pruebas, útiles para caracterizar el comportamiento de cualquier aplicación de videoconferencia.

En el desarrollo del proyecto, uno de los principales problemas al que nos hemos enfrentado han sido las fluctuaciones del ancho de banda disponible entre el laboratorio y el exterior, tanto en la conexión con la red de *Skype* como en la conexión con el servidor de *Vidyo* situado en la ubicación remota dentro de RIA. Durante la realización de las primeras medidas, nos dimos cuenta de que la congestión de la red de la Universidad debida al tráfico generado por sus usuarios, provocaba ciertas fluctuaciones en el ancho de banda disponible, que afectaban a las medidas. Para solucionar este problema se han debido realizar medidas en diferentes franjas horarias y días de la semana en los que los edificios estaban vacíos (fines de semana y festivos), con el objetivo de conseguir unas medidas fiables. Esta situación a la que nos hemos enfrentado es similar a la que podría aparecer en una empresa que quiera instalar una nueva solución de videoconferencia en su red: las condiciones óptimas para medir serán difíciles de alcanzar.

Tenemos un gran número de variables para las pruebas: los distintos parámetros para configurar las herramientas, las características de la videocámara, el vídeo a usar en la reproducción y también los propios parámetros de la aplicación de videoconferencia. Dado que no podemos abordar todo el abanico de pruebas posibles, hemos optado por mantener fijos algunos parámetros y hemos usado en las pruebas una serie de variables que hemos considerado significativas, y que nos pueden aportar suficiente información sobre el comportamiento de la aplicación a estudiar.

Los parámetros fijos en la realización de las pruebas en el caso de *Vidyo* son los siguientes:

- Velocidad de transmisión: en posición *Auto*, es decir, la aplicación cliente escoge la velocidad de transmisión de forma automática según las condiciones de la red.
- Velocidad de recepción: en posición *Auto*.
- Características del buffer de la herramienta *TC:* tamaño de *buffer* 104 KBytes y tamaño de *burst* 2 KBytes)
- Vídeo a transmitir (fútbol americano, como se ha explicado al principio del capítulo).

- Resolución de la videocámara: 800x450 (SVGA).
- Resolución de la aplicación cliente: Seleccionamos máxima resolución en las preferencias de la aplicación.

En el caso de *Skype* los parámetros de la videoconferencia son los mismos salvo los parámetros de velocidad de transmisión/recepción que no tenemos la posibilidad de modificar, en su caso será equivalente dado que *Skype* lo hace automáticamente.

4. Resultados

En este capítulo se presentarán las pruebas realizadas para medir y caracterizar las reacciones de los sistemas de videoconferencia frente a diferentes modificaciones en el estado de la red. En primer lugar presentaremos las pruebas del escenario con limitaciones en el enlace de bajada, y luego pondremos las limitaciones en el enlace de subida. Para cada escenario, se introducirán limitaciones en el ancho de banda disponible, así como pérdidas y retardos adicionales. Bajo cada una de las limitaciones realizaremos las medidas de ancho de banda generado, tiempo entre paquetes y tamaño de los paquetes.

4.1 Escenario con limitación en el enlace de bajada

Como se ha explicado en la Fig.3.1, en este escenario participan tres entidades principales: el *cliente emisor*, que genera el flujo de vídeo; el *servidor* que lo recibe y lo reenvía al *cliente observador*. En este caso las limitaciones son introducidas en el sentido que va desde el servidor *Vidyo* hacia el cliente observador.

4.1.1 Limitación del ancho de banda

a) *Vidyo*

En este apartado se presentan las pruebas encaminadas a observar la adaptación del sistema *Vidyo* frente a variaciones del ancho de banda disponible. Utilizaremos la herramienta *TC* para añadir diferentes limitaciones a la red, que variarán a lo largo del tiempo.

La Fig.4.1 muestra el ancho de banda instantáneo a lo largo de los 720 segundos que dura la prueba, en la que vamos introduciendo diferentes limitaciones de ancho de banda en la red, representadas por la línea verde. La línea gris representa el ancho de banda enviado por el *cliente emisor* al *servidor*. La línea azul representa el ancho de banda medio generado por el *servidor* cada segundo (*tick* = 1 seg.). Para poder observar el ancho de banda con menos fluctuaciones, se ha añadido también una línea roja, que representa el valor medio del ancho de banda cada 5 segundos (*tick* = 5 seg).

Como se puede ver, se ha limitado el ancho de banda disponible para el *cliente observador,* de forma escalonada, desde 1,2 hasta 0,4 Mbps con pasos de 0,2 Mbps cada 60 segundos. A partir del segundo 360 el ancho de banda disponible se vuelve a aumentar escalonadamente, para poder también observar la reacción de la aplicación cuando el ancho de banda aumenta.

A lo largo de toda la comunicación vemos (línea gris) cómo el *cliente emisor* transmite el tráfico hacia el servidor con un ancho de banda constante, es decir, está manteniendo el mismo nivel de calidad. El *servidor,* al detectar las limitaciones en el enlace de bajada (líneas verdes), adapta el tráfico para enviarlo al *cliente observador* (líneas azul y roja). Podemos extraer una primera conclusión: el servidor se da cuenta de que aparecen las limitaciones en el enlace de bajada, reaccionando con rapidez ante los cambios. Sin embargo, el *servidor* no solicita al *cliente emisor* que reduzca su tráfico. Este comportamiento puede justificarse debido a que en la videoconferencia puede haber otros potenciales participantes, y no todos tendrán las mismas limitaciones de ancho de banda. De este modo el *servidor* recibe el vídeo con toda la calidad que el *cliente emisor* es capaz de transmitir, y adapta luego el tráfico enviado a cada cliente, según sus características y la conexión.

Para entender mejor la adaptación de tráfico que hace *Vidyo,* hemos incluido otras figuras adicionales: la Fig.4.2 presenta el tiempo entre paquetes para cada paquete enviado, así como el tiempo medio entre paquetes. La Fig.4.3 presenta el tamaño de cada paquete, así como la evolución de su tamaño medio.

Durante los primeros 60 segundos no introducimos ningún tipo de limitación, así que observamos que el servidor *Vidyo* está generando 1,3 Mbps, con un tiempo medio entre paquetes en torno a los 7 ms, que corresponden a unos 150 paquetes por segundo, aunque este tiempo puede oscilar hasta valores de 30 o 40 ms. El tamaño medio de los paquetes es algo inferior a 1.100 bytes, con una gran cantidad de paquetes en torno a los 1.200 bytes, aunque aparecen paquetes de todos los tamaños.

Al aparecer la primera limitación, en el segundo 60, el *servidor* reduce el tráfico enviado al *cliente observador*, mientras la limitación se mantiene en niveles altos de ancho de banda (1,2 y 1 Mbps). Sin embargo, a partir del segundo 180, cuando la limitación es de 0,8 Mbps, se observa un comportamiento diferente, apareciendo unas oscilaciones en el ancho de banda, con un periodo de entre 16 y 20 segundos.

Este comportamiento se mantiene en el resto de tramos incluso cuando la limitación es menor, es decir, cuando la red se descongestiona. Observamos que el ancho de banda enviado oscila entre el límite establecido y su mitad. Este efecto se puede observar, por ejemplo, entre los segundos 180 y 240 (limitación en 0,8 Mbps), donde el ancho de banda generado varía entre 0,8 y 0,4 Mbps.

Finalmente cuando quitamos la limitación, en el segundo 600, observamos un pico de tráfico. Esto puede deberse a que el sistema está tratando de averiguar el ancho de banda disponible. Para ello, transmite cada vez más tráfico hasta que se da cuenta de que tiene suficiente ancho de banda, con lo que cambia otra vez de modo de funcionamiento, comportándose como lo hacía al principio de la comunicación (enviando 1,3 Mbps).

Este comportamiento nos sugiere que la aplicación tiene dos estados: uno en el que considera suficiente el ancho de banda disponible, y otro en el que el tráfico oscila: el servidor va aumentando progresivamente el tráfico que envía al *cliente observador*, y cuando detecta que es escaso, lo reduce paulatinamente hasta aproximadamente la mitad. Entonces, vuelve a subir para comprobar si las limitaciones continúan.

Este método implementado por el sistema resulta útil para reaccionar ante las frecuentes variaciones en las condiciones que se presentan en las redes de datos. Estas soluciones de videoconferencia deben implementar mecanismos para garantizar un funcionamiento mínimo, sean cuales sean las condiciones de la red, pues su éxito comercial dependerá en gran medida de esto.

Con respecto al resultado del tiempo entre paquetes (Fig.4.2), observamos que en condiciones normales se sitúa por debajo de los 10 ms, y va aumentando a medida que introducimos más limitación, y disminuye cuando la limitación es menor. Podemos decir por tanto que el número de paquetes por segundo aumenta y disminuye conforme lo hace el ancho de banda disponible

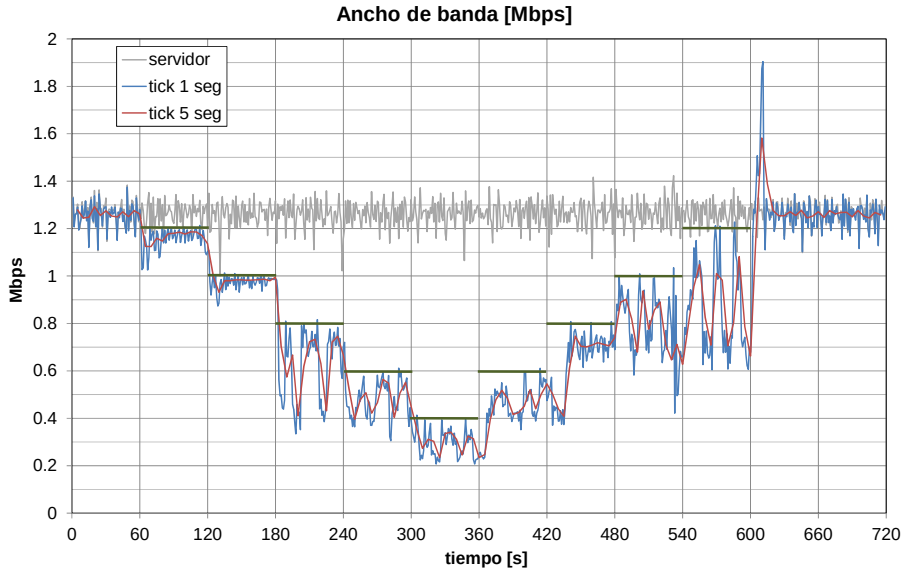


Fig.4.1. Evolución del ancho de banda a lo largo del tiempo para *Vidyo*

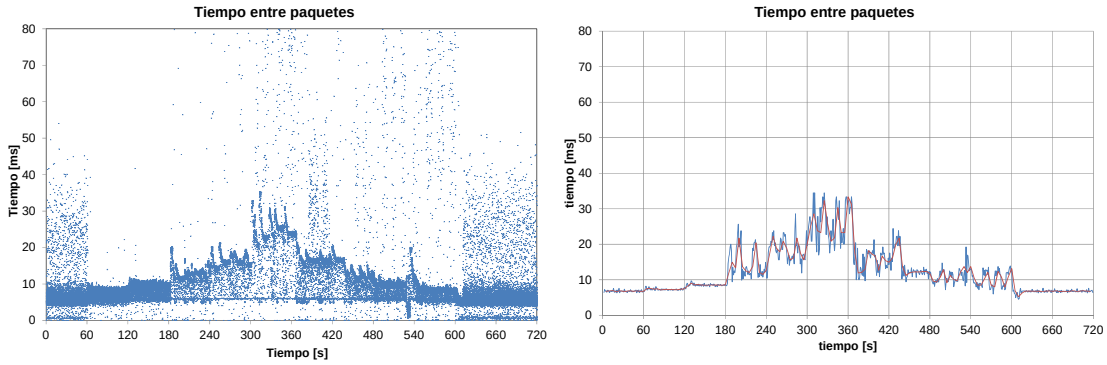


Fig.4.2. Tiempo entre paquetes para cada paquete; tiempo medio entre paquetes.

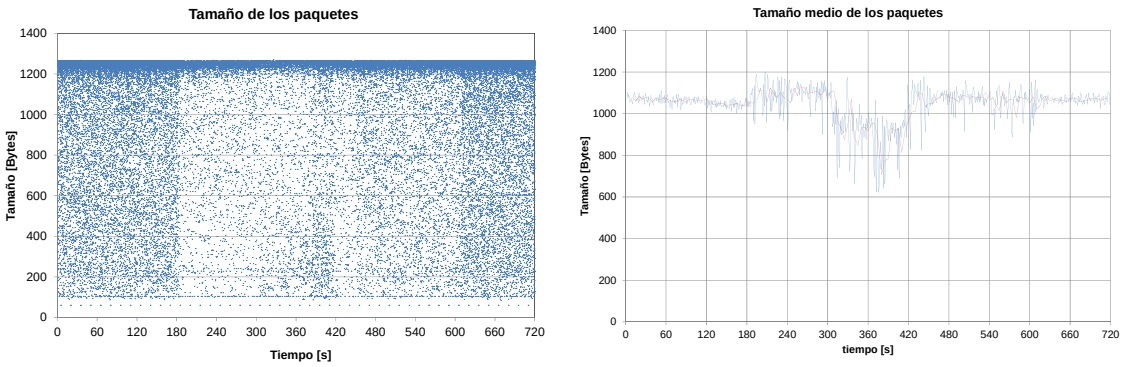


Fig.4.3. Tamaño de cada paquete; evolución del tamaño medio de los paquetes

Sin embargo, el tamaño de los paquetes (Fig.4.3) no sufre mucha variación a lo largo de la comunicación excepto en el tramo de 300 a 420 segundos, en el que el ancho de banda disponible es primero 0,4 Mbps y 0,6 Mbps después. En este caso el tamaño medio de los paquetes se sitúa por debajo de los 1.000 bytes, lo que podría deberse a un cambio de comportamiento porque el sistema ha quitado alguna capa en las imágenes de vídeo a causa del poco ancho de banda disponible.

Para observar con más claridad este comportamiento, la Fig.4.4 representa el histograma de la distribución del tamaño de los paquetes a lo largo del tiempo, solamente hasta el segundo 360. La Fig.4.5 representa el porcentaje acumulado.

Observamos que la distribución de los tamaños varía muy poco con el cambio del ancho de banda disponible, salvo en el caso que hemos comentado (300 a 360 seg), en el que hay un incremento de la cantidad de paquetes pequeños. Podemos observar (Fig.4.5) que cuando no hay limitaciones, el 30% de los paquetes son de tamaño menor de 1.200 bytes mientras que el 70% son de tamaño mayor, entre 1.200 bytes y 1.300 bytes. Esto quiere decir que el sistema intenta mandar mayoritariamente paquetes de tamaño grande, aumentando así la eficiencia.

En el caso en el que el ancho de banda disponible cae hasta los 0,4 Mbps (300 a 360 seg), hay un cambio notable en el tamaño medio de los paquetes. En este caso el porcentaje acumulado de paquetes mayores de 1.200 bytes es de casi el 50%. Por lo tanto el sistema será menos eficiente en este caso.

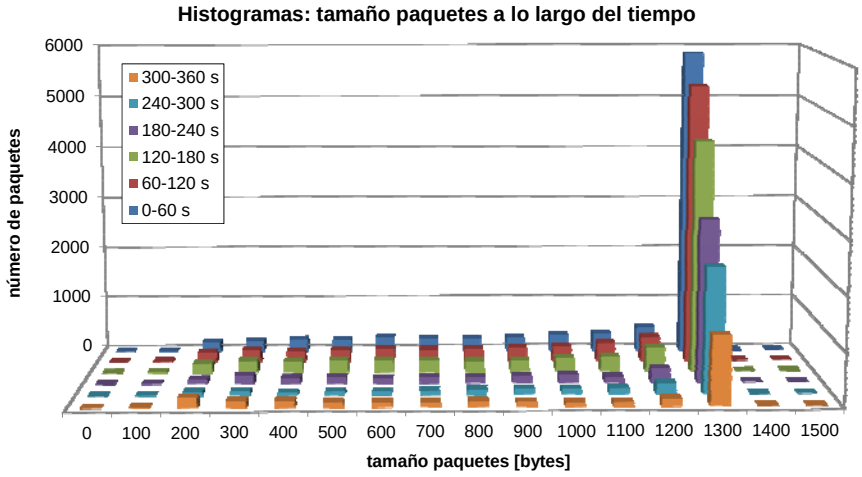


Fig.4.4. Histograma del tamaño de los paquetes a lo largo del tiempo.

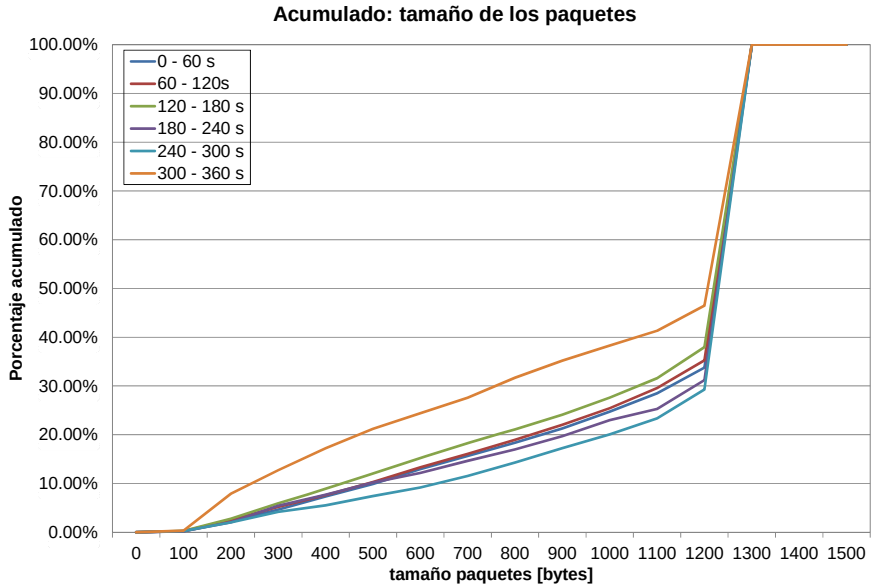


Fig.4.5. Histograma del porcentaje acumulado de paquetes a lo largo del tiempo.

b) *Skype*

Analizamos ahora el comportamiento de *Skype* ante las mismas limitaciones de ancho de banda. La Fig.4.6, que es correlativa a la Fig.4.1 obtenida con *Vidyo,* muestra el ancho de banda instantáneo generado por el *cliente emisor* a lo largo de los 720 segundos. Igual que antes, la línea azul representa el ancho de banda medio cada segundo, la línea roja el ancho de banda medio cada 5 segundos, y la línea verde la limitación de ancho de banda.

En la Fig.4.6 no hemos representado el ancho de banda generado, porque es exactamente igual que el recibido. Por tanto, a diferencia de *Vidyo,* en *Skype* el cliente emisor sí adapta su tráfico según a las condiciones de la red.

Por otro lado, vemos que el comportamiento de *Skype* es siempre el mismo en todos los tramos de la prueba (las fluctuaciones son aproximadamente de los mismos valores), adaptándose al ancho de banda disponible en la red. Se puede destacar también que a partir del segundo 600 aparece un pico que tiene una finalidad análoga al observado en *Vidyo*, es decir, explorar los límites de ancho de banda.

Con respecto a los resultados del tiempo entre paquetes (Fig.4.7) podemos decir que el tráfico sigue una distribución constante, con un máximo de 20 ms entre los paquetes. En las gráficas del tamaño de los paquetes (Fig.4.8), observamos varias franjas en las que se distribuyen los paquetes. La primera se sitúa entre 1.000 y 1.400 bytes, y corresponde al tráfico UDP de vídeo. Las demás corresponden a paquetes UDP de tamaños por debajo de 200 bytes, donde cabe destacar la presencia periódicos de paquetes de 31 bytes cada 10 segundos y de 45 bytes cada 50 segundos.

Como era de esperar, el número de paquetes por segundo (Fig.4.7) sigue la misma tendencia que el ancho de banda mostrado en la Fig.4.6, aumentando y disminuyendo conforme lo hace el ancho de banda disponible en la red. De la misma manera, la tendencia del tamaño medio de los paquetes a lo largo del tiempo es similar a las anteriores, como podemos observar en la Fig.4.8.

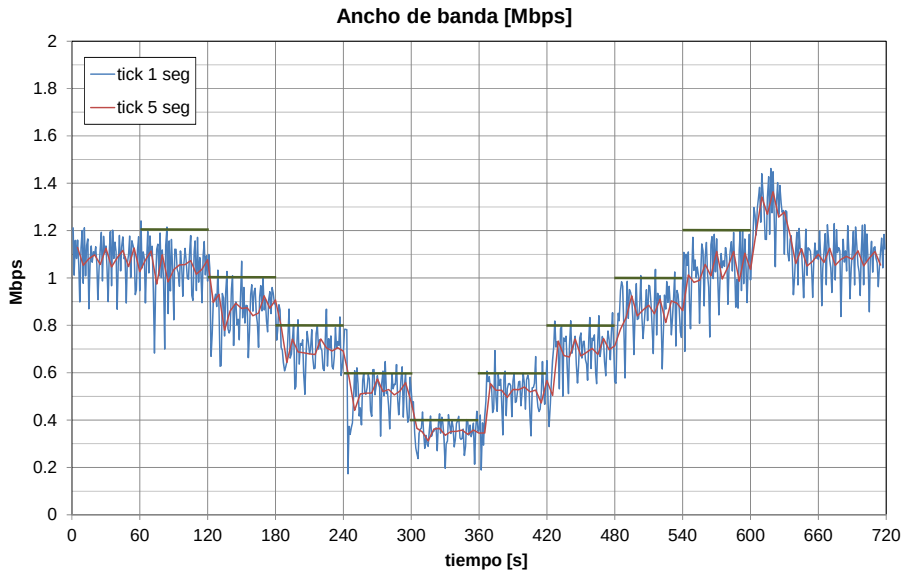


Fig.4.6. Evolución del ancho de banda para *Skype*.

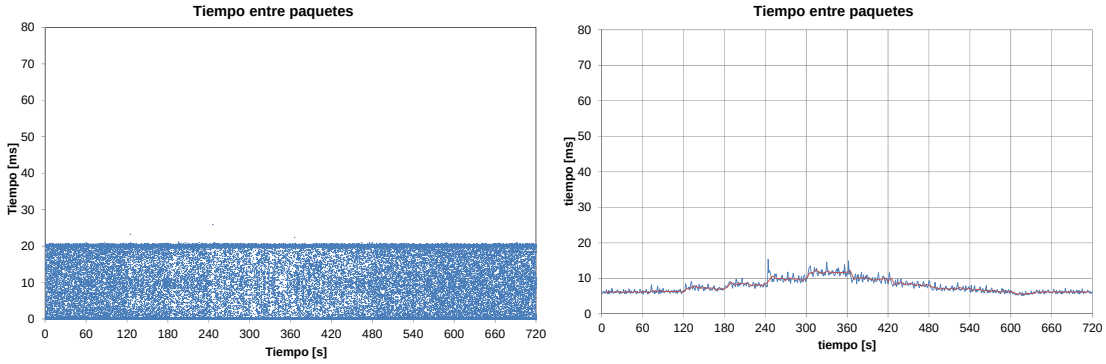


Fig.4.7. Tiempo entre paquetes para cada paquete; tiempo medio entre paquetes.

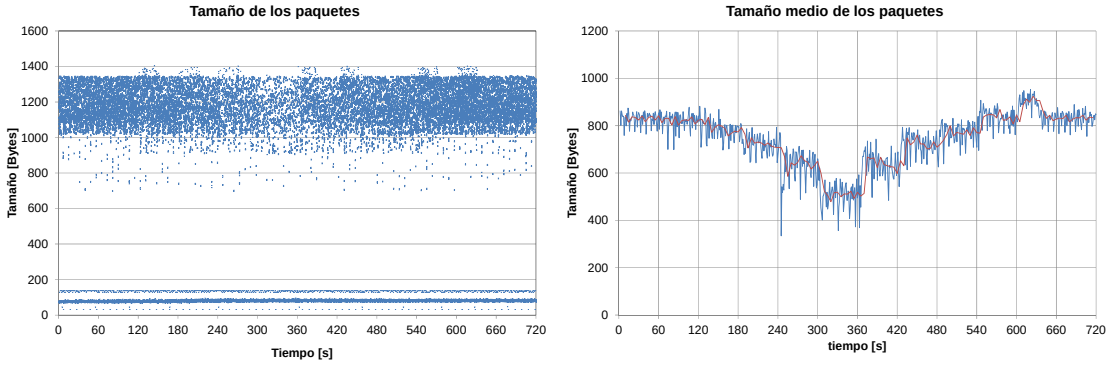


Fig.4.8. Tamaño de cada paquete; tamaño medio de los paquetes.

En los histogramas de la Fig.4.9 y 4.10 observamos mejor lo comentado anteriormente respecto al tamaño de los paquetes. Por un lado, aparece un grupo de paquetes pequeños, por debajo de 200 bytes, que se mantiene constante en número, independientemente de las condiciones de la red. Sin embargo, el otro grupo de paquetes, de tamaño grande, sí varía su distribución con las diferentes condiciones de la red. A causa de ello, los paquetes pequeños pueden llegar a ser casi el 60% del total (limitación en 0,4 Mbps). Por el contrario, en el tramo donde no existe limitación, los paquetes grandes son el 70% y los pequeños el 30%.

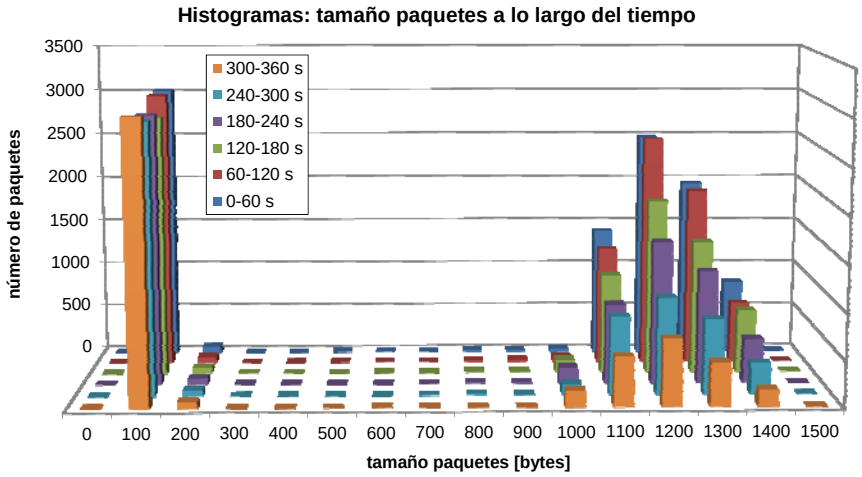


Fig.4.9. Histograma del tamaño de los paquetes a lo largo del tiempo.

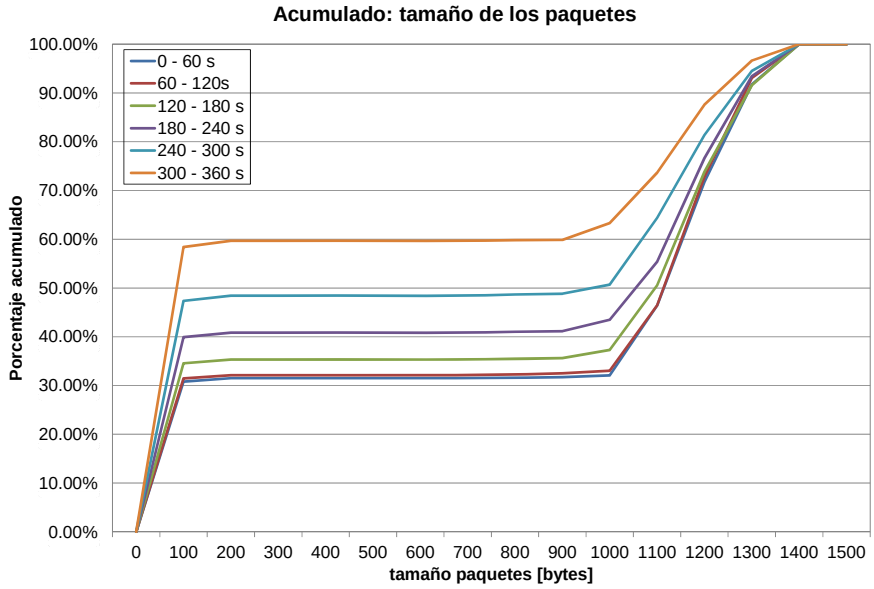


Fig.4.10. Histograma del porcentaje acumulado de paquetes a lo largo del tiempo.

4.1.2 Limitación mediante pérdidas

a) Vidyo

En este apartado se presentan las pruebas encaminadas a observar la adaptación del sistema *Vidyo* frente a pérdidas *(loss)* de paquetes en la red. Para ello, utilizaremos la herramienta *TC netem* para añadir diferentes porcentajes de pérdidas, que variarán a lo largo del tiempo.

Como en apartados anteriores, la Fig.4.11 muestra el ancho de banda instantáneo generado por *Vidyo* a lo largo de 720 segundos, mientras se van introduciendo diferentes porcentajes de pérdidas en la red, de forma escalonada (línea morada) desde el 1% hasta el 10% con incrementos del 1% cada 60 segundos.

A lo largo de toda la comunicación vemos cómo el ancho de banda generado por el servidor (líneas azul y roja) apenas sufre cambios, salvo una ligera y poco apreciable tendencia ascendente desde el segundo 60 hasta el 660, que es el intervalo donde hemos introducido pérdidas. Sin embargo, se observa que en este intervalo el tráfico enviado por el *emisor* (línea gris) no varía, por lo que se aprecia solamente una ligera diferencia entre este tráfico y el que el *servidor* envía al *cliente receptor*.

Finalmente, a partir del segundo 660, el ancho de banda disminuye al haber cesado las pérdidas en la red.

Este comportamiento nos sugiere que el tráfico de vídeo no se modifica, pero quizá el incremento de ancho de banda se deba a algún flujo de señalización, que requiere retransmisiones en caso de pérdidas.

Las Fig.4.12 y 4.13 nos confirman este comportamiento: por una parte, la media del tiempo entre paquetes se mantiene en torno a los 7 ms durante toda la comunicación, y por otra la media del tamaño de los paquetes se mantiene sin mucha oscilación entre 1.000 y 1.200 bytes. No se observa tampoco ninguna variación en la distribución de los tamaños o tiempos de cada paquete en función de las pérdidas.

Podemos concluir, por tanto, que *Vidyo* no modifica el tráfico aunque haya pérdidas en la red. El programa confía en la potencia de su codificación de vídeo para recuperar la imagen y mostrar un contenido aceptable para el receptor a pesar de las pérdidas.

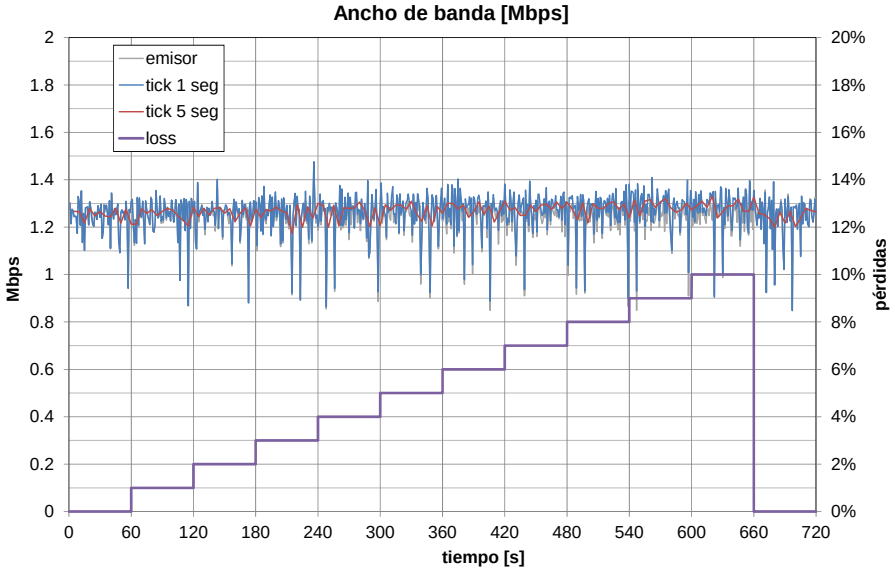


Fig.4.11. Evolución del ancho de banda a lo largo del tiempo para *Vidyo*

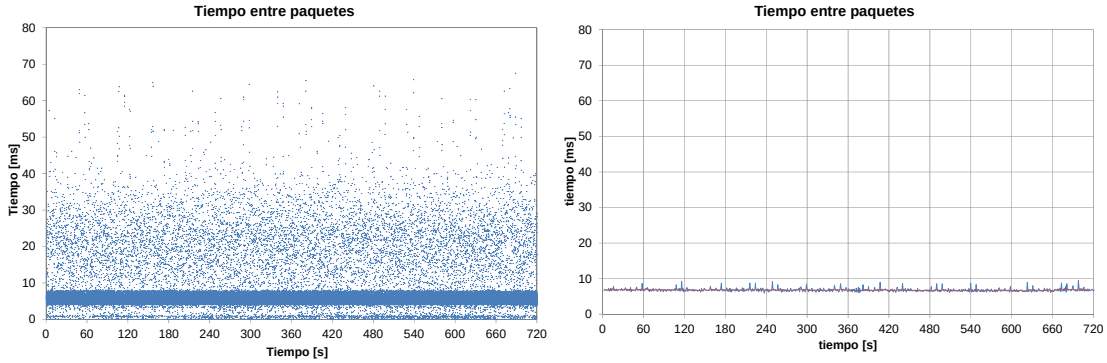


Fig.4.12. Tiempo entre paquetes; tiempo medio entre paquetes.

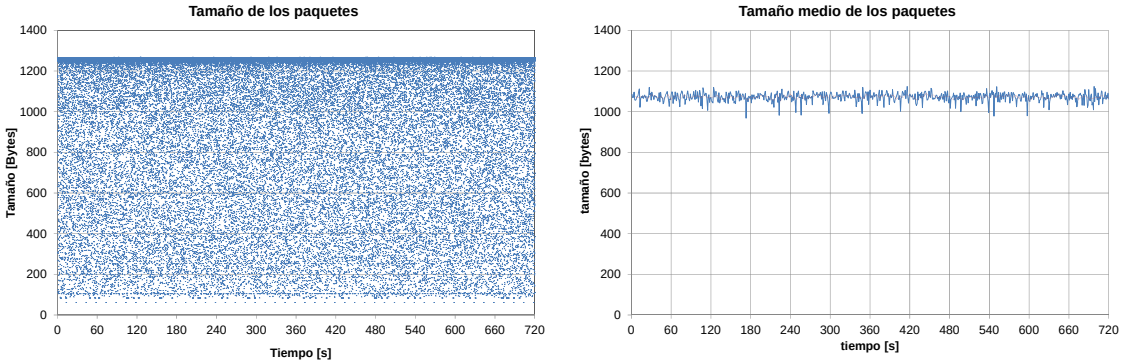


Fig.4.13. Tamaño de los paquetes; tamaño medio de los paquetes.

b) Skype

De la misma manera, analizamos el comportamiento de la solución *Skype* ante los mismos efectos de pérdidas a lo largo del tiempo. Recordemos que en este caso no hay servidor, sólo un cliente emisor y otro receptor. La Fig.4.14 muestra el ancho de banda instantáneo generado por el cliente *emisor* a lo largo de los 720 segundos. En este caso, podemos ver cómo *Skype* sí modifica el tráfico enviado: el ancho de banda aumenta con las pérdidas en la red. Este efecto se observa desde el segundo 60, con un valor de pérdidas del 1% y un ancho de banda de 1,3 Mbps, hasta el segundo 540 con un valor de 9% de pérdidas y un ancho de banda de casi 2,4 Mbps. Como se dijo en [BMM+08], *Skype* implementa diferentes mecanismos para hacer frente a las condiciones de la red. En caso de detectar pérdidas, *Skype* recurre a la transmisión redundante de los bloques de datos en los mensajes. Observamos que *Skype* aumenta la frecuencia de envío, lo que se corresponde con los resultados del trabajo citado. El cuanto al tiempo entre paquetes (Fig. 4.15) sigue una tendencia descendente conforme aumentan las pérdidas introducidas. Además, el tamaño de los paquetes (Fig. 4.16) varía desde el momento en que aparecen pérdidas en el sistema: en el segundo 60 comienzan a aparecer paquetes de tamaño más grande a lo habitual, alrededor de 1.400 bytes. Esto hace que el tamaño medio de los paquetes vaya en aumento, alcanzando su valor máximo (1.100 bytes) en el intervalo de 380 a 420 segundos.

Este comportamiento se mantiene mientras las pérdidas están por debajo del 7% (segundo 420). A partir de este momento, *Skype* cambia su comportamiento, y observamos que desaparecen los paquetes más grandes (en torno a 1.400 bytes), lo que hace que el tamaño medio disminuya, aunque se mantiene en torno a 1.000 bytes hasta el segundo 540.

Sin embargo, a partir del segundo 540, cuando la limitación de pérdidas introducida alcanza el valor de 9%, observamos un nuevo cambio en el comportamiento de la aplicación: el

ancho de banda generado cae por debajo de 1 Mbps. Podemos interpretar que *Skype* decide no seguir aumentando el tráfico, por estimar que el valor de las pérdidas es demasiado alto para la videoconferencia. En el segundo 600, cuando introducimos un valor de pérdidas del 10%, se acentúa este comportamiento de *Skype:* el tráfico baja por debajo de 0,8 Mbps. Al minuto siguiente quitamos las limitaciones y vemos como el ancho de banda se restablece a su valor inicial de 1,1 Mbps.

Podemos concluir, por tanto, que *Skype* hace frente a las pérdidas con diferentes mecanismos, disminuyendo el tiempo entre paquetes, añadiendo redundancia y retransmisiones de los paquetes hasta llegar a un cierto límite de pérdidas, que situamos en torno al 9%, a partir del cual decide disminuir el tráfico.

Por motivos de espacio, no mostraremos en este caso los histogramas correspondientes a cada intervalo.

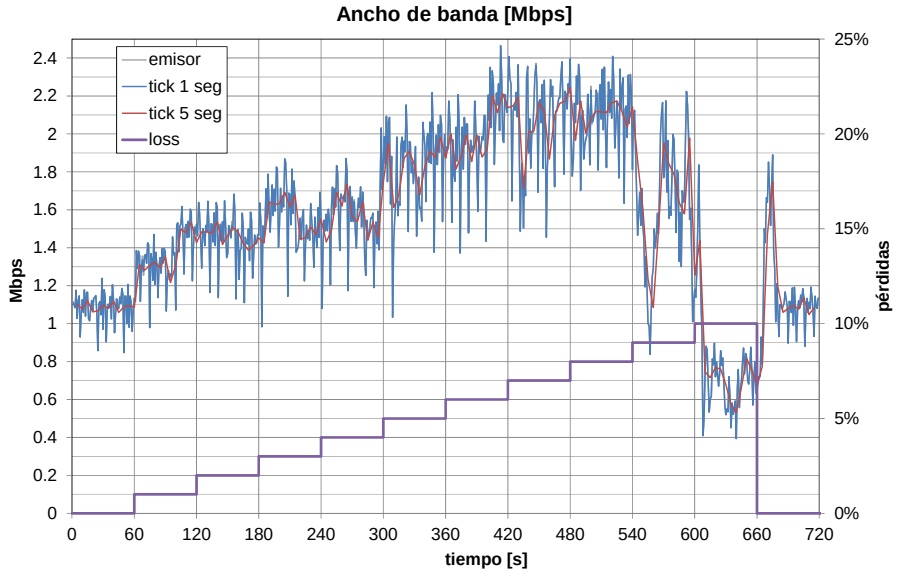


Fig.4.14. Evolución del ancho de banda a lo largo del tiempo para *Skype*.

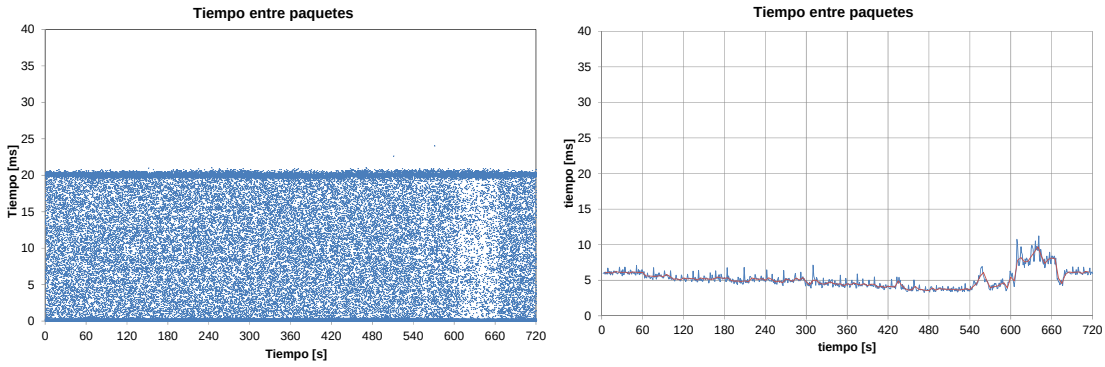


Fig.4.15. Tiempo entre paquetes; tiempo medio entre paquetes.

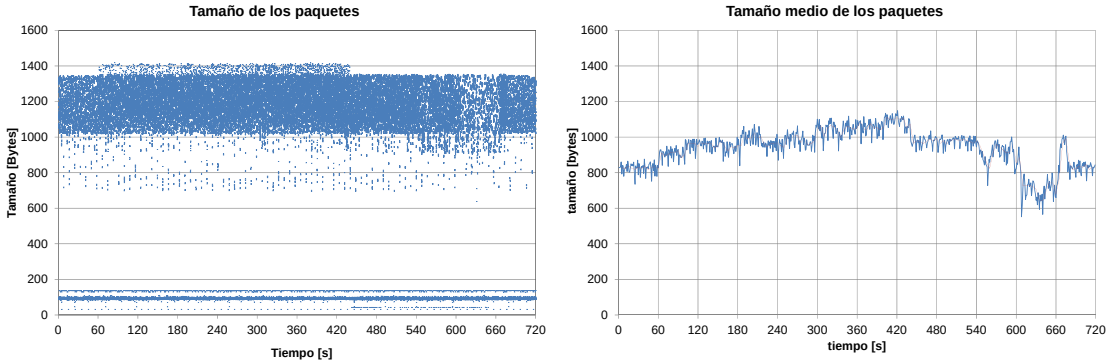


Fig.4.16. Tamaño medio de los paquetes; tamaño medio de los paquetes.

En los histogramas de las Figuras 4.17 y 4.18 observamos más claramente lo comentado anteriormente respecto al tamaño de los paquetes. Por un lado observamos un grupo de paquetes pequeños por debajo de 200 bytes. Por otro lado, el otro grupo de paquetes de tamaño grande, aunque siguen la misma distribución a lo largo de la comunicación, sí varían su distribución en los distintos intervalos, donde las condiciones de la red son diferentes. En condiciones normales los paquetes pequeños significan el 30% de los paquetes y los paquetes grandes el 70%, pero conforme aumentan las pérdidas en la red esta relación varía, llegando a ser 15/75.

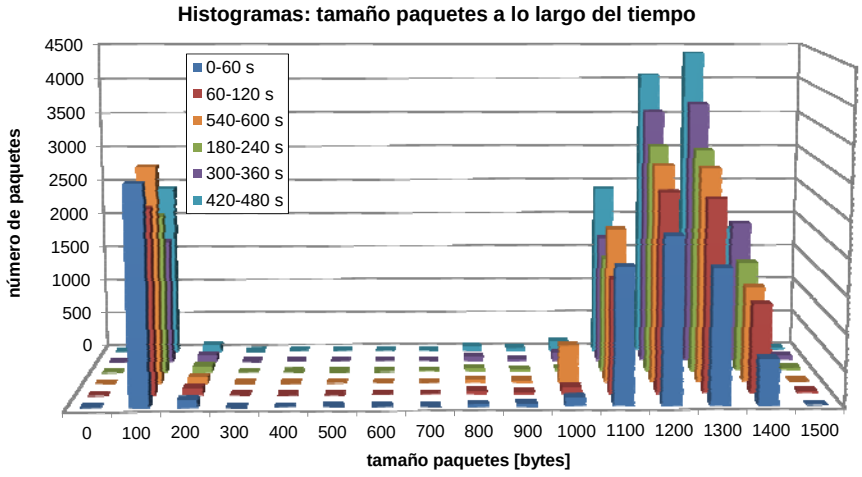


Fig.4.17. Histograma del tamaño de los paquetes.

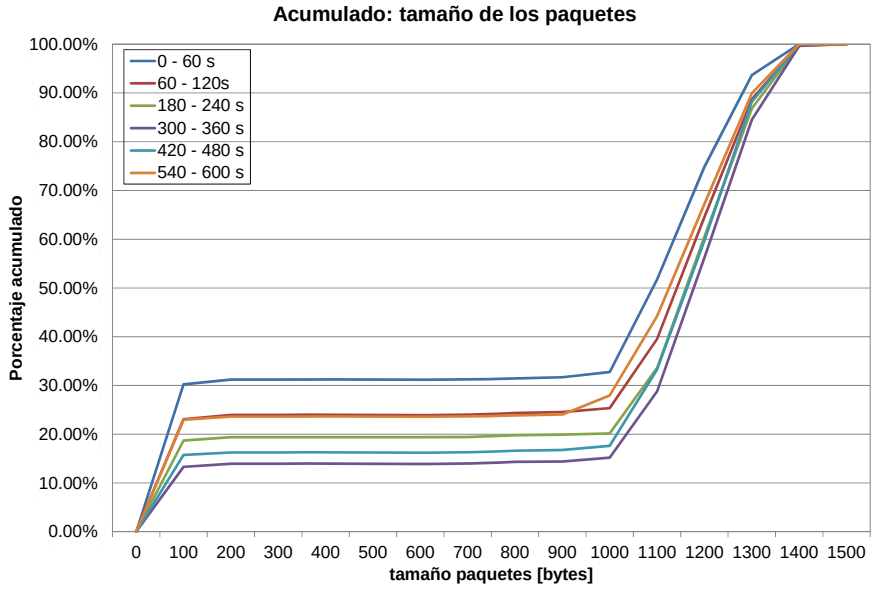


Fig.4.18. Histograma del porcentaje acumulado de paquetes.

4.1.3 Limitación mediante retardos

a) Vidyo

En este apartado se presentan las pruebas encaminadas a observar la adaptación del sistema *Vidyo* frente a las variaciones del retardo de la red. Para ello, utilizaremos la herramienta *TC* para añadir diferentes valores de retardo, que variarán a lo largo del tiempo.

Como en apartados anteriores, se muestra (Fig.4.19) el ancho de banda instantáneo generado por *Vidyo* a lo largo de 480 segundos, mientras vamos introduciendo diferentes valores de retardo adicionales, de forma escalonada, desde 50 hasta 300 ms, con incrementos de 50 ms cada 60 segundos. En el instante 420 segundos se eliminan los retardos adicionales.

En los resultados no observamos un cambio apreciable en el ancho de banda generado (Fig. 4.19). El tamaño de los paquetes no se modifica (Fig. 4.21), pero sí es apreciable un pico y una ligera reducción posterior en el tiempo entre paquetes, en los segundos posteriores a cada cambio en el retardo (Fig. 4.20). Una posible explicación para este comportamiento es que *Vidyo* intenta adaptar el tráfico a esta nueva situación, interpretando que es algo transitorio, causado por una congestión temporal. Finalmente, como el retardo se mantiene, vuelve al comportamiento inicial después de unos 10 segundos.

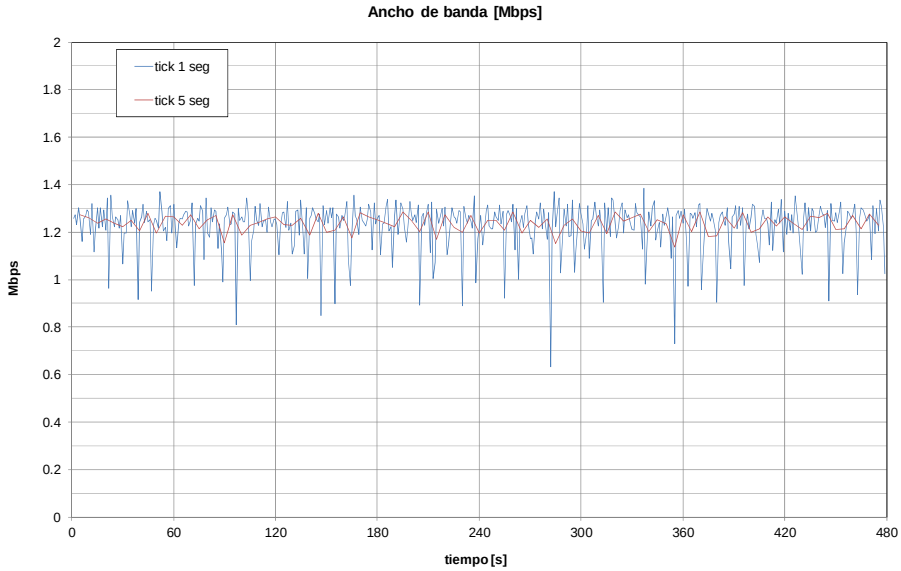


Fig.4.19. Evolución del ancho de banda para *Vidyo*

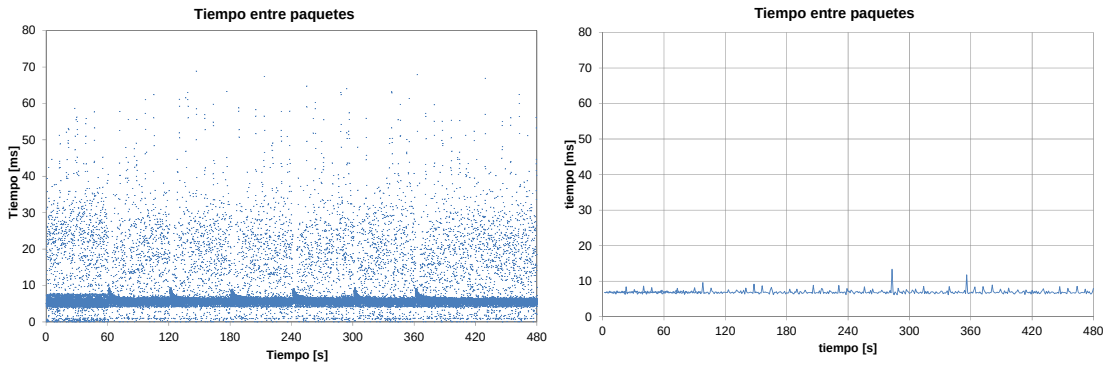


Fig.4.20. Tiempo entre paquetes; tiempo medio entre paquetes.

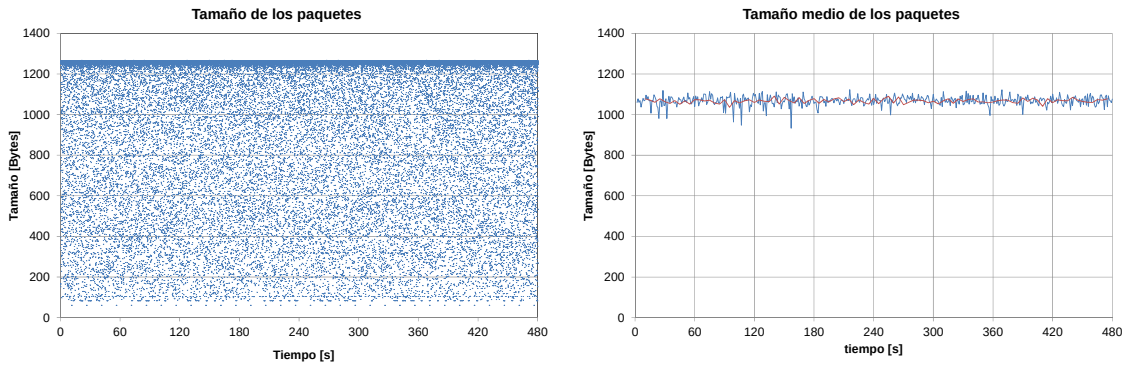


Fig.4.21. Tamaño de los paquetes; tamaño medio de los paquetes.

b) Skype

Se ha repetido el test presentado en a), pero en este caso analizamos el comportamiento de *Skype* ante los mismos efectos de retardo a lo largo del tiempo. En la Fig. 4.22 vemos que cuando aplicamos la limitación de 50 ms en el segundo 60, *Skype* detecta que han aparecido problemas en la red, y se comporta de forma similar a cuando había pérdidas, es decir, aumenta el ancho de banda generado. Aproximadamente 30 segundos más tarde se da cuenta que nada puede hacer ante el retardo, y continúa mandando el tráfico de forma normal. Al final de la comunicación (420 segundos), cuando eliminamos las limitaciones, se vuelve a comportar de la misma manera, para comprobar su funcionamiento en las nuevas condiciones de la red. Este cambio en el ancho de banda está causado por un aumento en el tamaño de los paquetes (*Skype* introduce más redundancia), como se ve en la Fig. 4.24, y no del tiempo entre paquetes, que se mantiene estable (Fig. 4.23).

Podemos concluir por tanto que *Skype* tampoco tiene ningún mecanismo para afrontar este efecto, aunque se ve un cambio de comportamiento que se puede deber a una exploración de las nuevas condiciones de la red. Esto contrasta con los resultados obtenidos en [BMM+08], en los que se afirmaba que *Skype* no reacciona de ninguna manera al retardo. Este artículo se hizo hace unos años, y las características de *Skype* pueden haber mejorado en las últimas versiones.

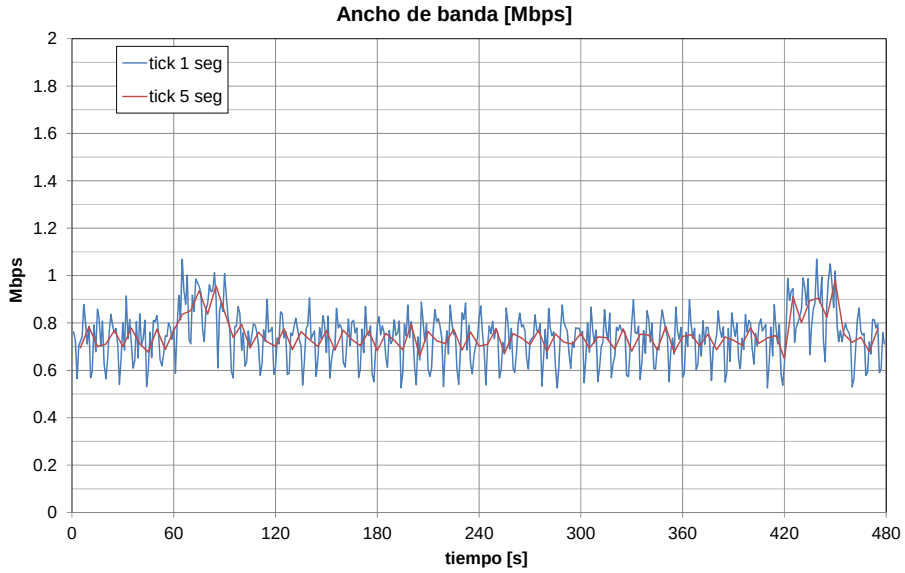


Fig.4.22. Evolución del ancho de banda para *Skype*.

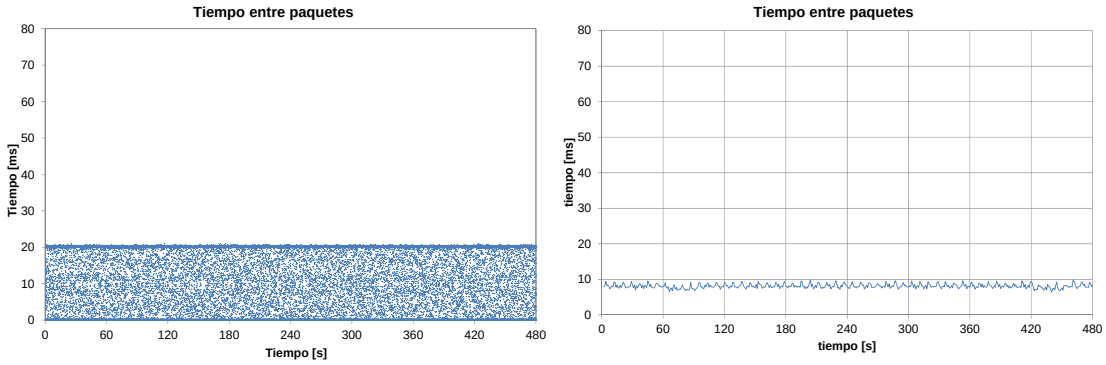


Fig.4.23. Tiempo entre paquetes; tiempo medio entre paquetes.

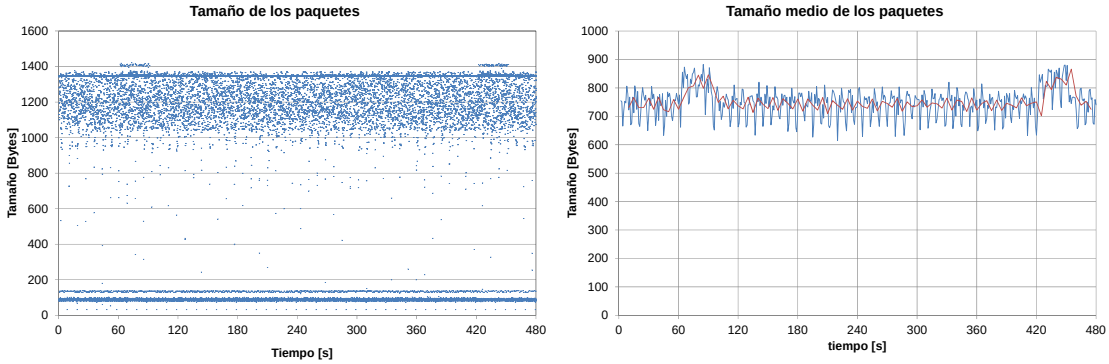


Fig.4.24. Tamaño medio de los paquetes; tamaño medio de los paquetes.

4.2 Escenario con limitación en el enlace de subida

En la sección anterior se han estudiado los mecanismos de los sistemas de videoconferencia para reaccionar ante las limitaciones en el enlace de bajada. Pasamos ahora a estudiar las reacciones, cuando las limitaciones aparecen en el enlace de subida. Como los comportamientos son similares en muchos casos, sólo comentaremos los efectos que varíen o sean significativos con respecto al escenario anterior; además, sólo estudiaremos la aplicación *Vidyo,* porque, como ya se ha explicado, en el caso de *Skype*, al no tener un servidor central, el efecto de las limitaciones es el mismo independientemente de si aparecen en el enlace de subida o en el de bajada.

4.2.1 Limitación del ancho de banda

Como se ve en la Fig.4.25, la adaptación a las condiciones de la red es muy buena, con un comportamiento muy parecido al caso de las limitaciones en el enlace de bajada. Pero en este caso el servidor simplemente repite lo que recibe y manda al receptor este tráfico limitado. Por eso la línea gris y la línea azul se superponen en la gráfica.

A diferencia de lo que ocurría con las limitaciones en la bajada, observamos que el tamaño de las oscilaciones es prácticamente el mismo en todos los tramos de la comunicación. Cabe destacar que, cuando quitamos la limitación (segundo 600), en este caso no aparece un pico en el ancho de banda como se veía en el escenario anterior, podemos decir que el sistema tiene un comportamiento menos *agresivo*. Por último, debido a una caída en la red, el ancho de banda sufrió en este caso una bajada brusca alrededor del segundo 680.

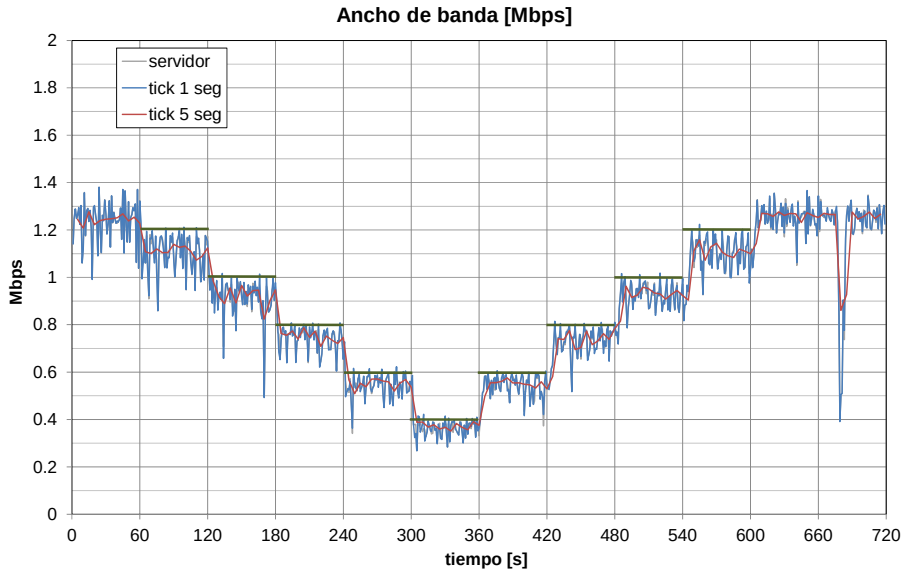


Fig.4.25. Evolución del ancho de banda para *Vidyo*

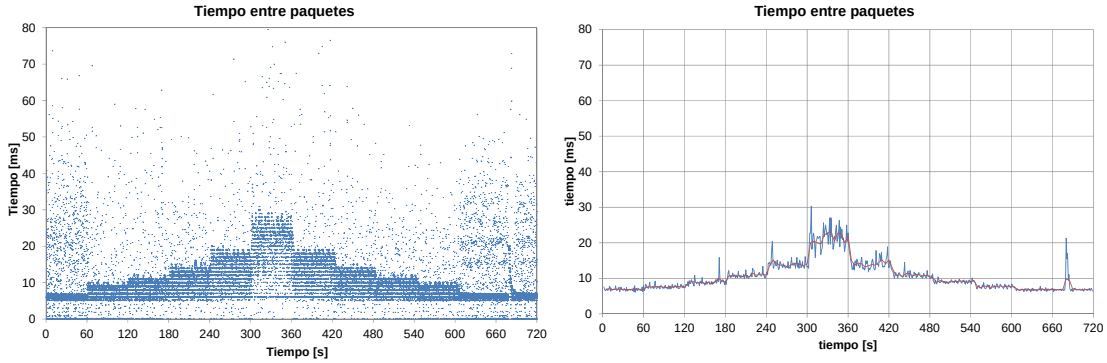
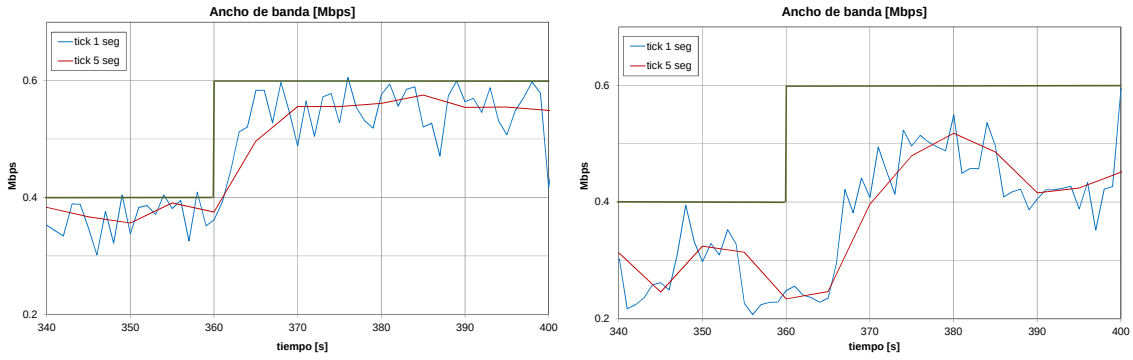


Fig.4.26. Tiempo entre paquetes para cada paquete; tiempo medio entre paquetes.

Con respecto al tiempo entre paquetes (Fig.4.26), observamos unas franjas horizontales en las que se acumulan los tiempos entre paquetes. Las franjas aparecen cada milisegundo, lo que nos hace pensar que el programa está usando un reloj que le hace enviar con esta periodicidad. Es posible que este efecto también exista en el escenario de limitación en bajada, pero si fuera así no lo podríamos apreciar porque entre el servidor y la máquina que captura hay una red que introduce una variación (*jitter*) en el tiempo para atravesarla, lo que hace que el tiempo entre paquetes se distribuya, enmascarando este posible efecto. Para poder observarlo en el escenario de bajada deberíamos capturar el tráfico junto al servidor.

Finalmente presentamos unas gráficas para comparar la velocidad de adaptación del ancho de banda en los dos escenarios (limitaciones en la subida o en la bajada). Para ello, hemos hecho un *zoom* de los segundos 340 a 400 de las Fig.4.25 y 4.1, que se representan en la Fig. 4.27. Vemos cómo la adaptación en el escenario con limitaciones en la subida (Fig. 4.27, a) es más rápida: el sistema empieza a subir el ancho de banda poco después del segundo 360. En cambio, si las limitaciones están en la bajada (Fig. 4.27, b), pasan unos 6 segundos hasta que el ancho de banda empieza a subir (segundo 366). Este comportamiento podría explicarse por el diferente periodo que se observa en las oscilaciones del ancho de banda: en la Fig.4.25 el periodo es mucho menor que en la 4.1, por lo que el sistema reacciona antes ante el cambio en el ancho de banda en el primer caso.



(a) (b)

Fig.4.27. a) Ancho de banda con limitaciones en la subida; b) ancho de banda con limitaciones en la bajada

4.2.2 Limitación mediante pérdidas

El comportamiento ante el efecto de pérdidas es muy parecido al que observábamos en el escenario de bajada. El ancho banda (Fig.4.28) experimenta una tendencia ligeramente ascendente hasta que las pérdidas cesan, en el segundo 660. Cabe destacar que el tamaño medio de los paquetes esta vez es algo inferior, situándose por debajo de 1.000 bytes (Fig.4.30). El tiempo medio entre paquetes es de 8ms durante toda la comunicación (Fig.4.29).

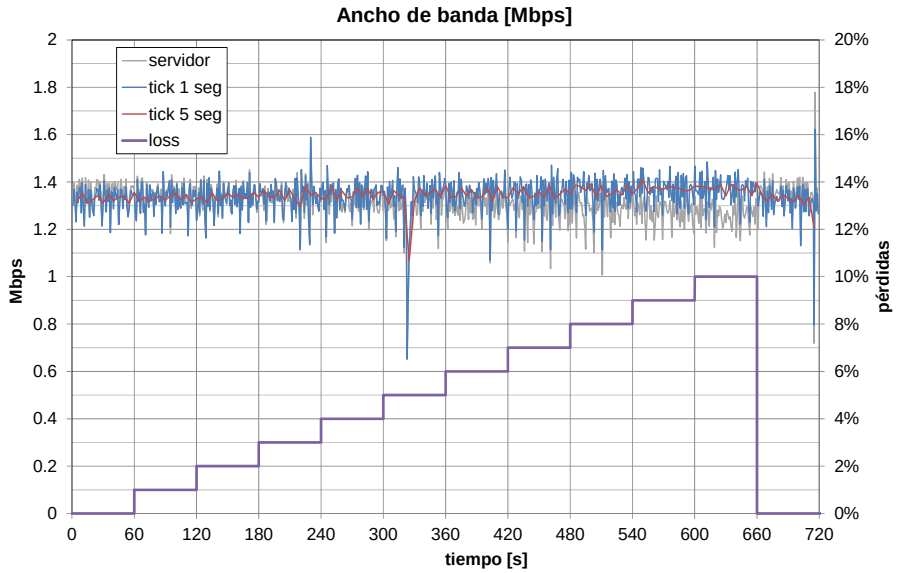


Fig.4.28. Evolución del ancho de banda para *Vidyo*.

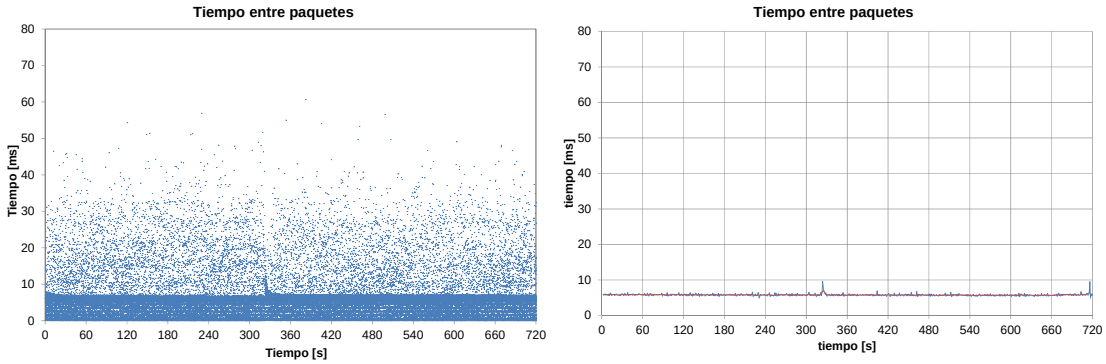


Fig.4.29. Tiempo entre paquetes; tiempo medio entre paquetes.

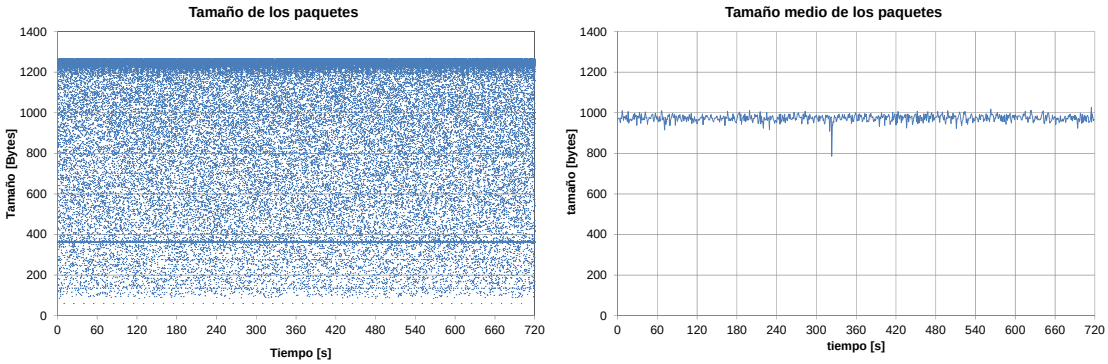


Fig.4.30. Tamaño de los paquetes; tamaño medio de los paquetes.

4.2.3 Limitación mediante retardos

También en este caso, al aplicar retardos en lar red, observamos el mismo comportamiento que en el escenario de bajada. El ancho de banda se mantiene constante, cerca de 1,2 Mbps (Fig.4.31). El tiempo medio entre paquetes es de 8 ms (Fig.4.32) y el tamaño medio los paquetes es de 1.100 bytes (Fig.4.33).

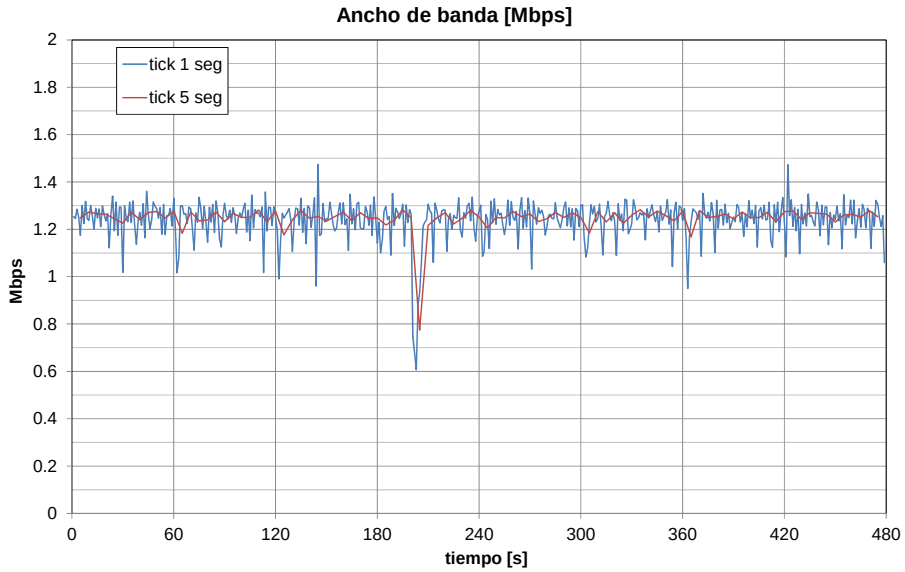


Fig.4.31. Evolución del ancho de banda para *Vidyo*.

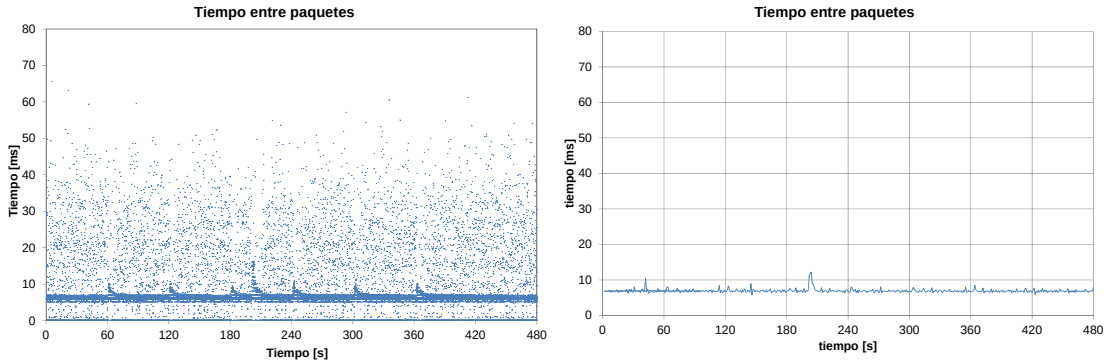


Fig.4.32. Tiempo entre paquetes; tiempo medio entre paquetes.

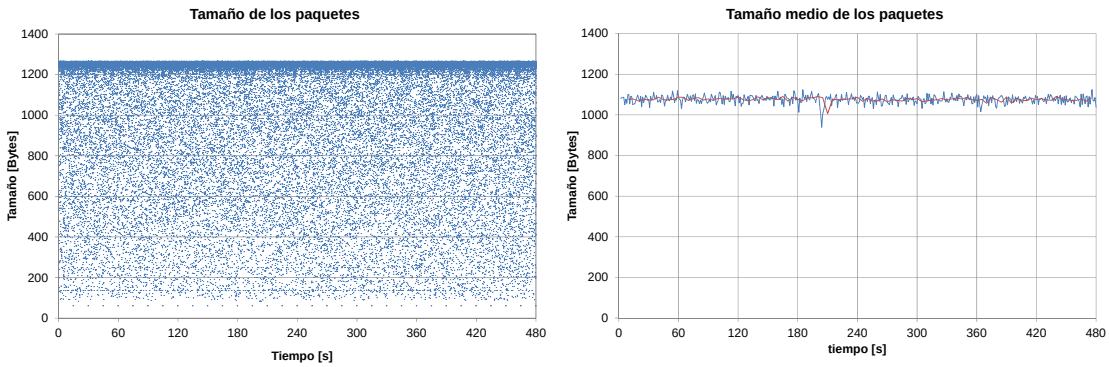


Fig.4.33. Tamaño de los paquetes; tamaño medio de los paquetes.

5. Conclusiones y líneas futuras

5.1 Conclusiones

Después de exponer el trabajo realizado en este proyecto, estamos ahora en disposición de presentar un resumen de las conclusiones, incluyendo tanto la metodología de las pruebas como los resultados obtenidos.

En primer lugar, se ha conseguido crear un entorno de laboratorio útil y versátil, que nos ha permitido la realización de un amplio abanico de pruebas, para así caracterizar el comportamiento de las aplicaciones ante distintos parámetros de la red.

El entorno de trabajo incluye una serie de dispositivos de red que permiten capturar los distintos flujos de tráfico, teniendo en cuenta que el flujo generado por el emisor no es siempre el mismo que llega al receptor. El servidor de videoconferencia puede adaptar el tráfico a las características del dispositivo final, y también al estado de la red.

También incluimos en el entorno de trabajo un emulador, que permitía modificar artificialmente las condiciones de la red, introduciendo limitaciones adicionales de ancho de banda, retardos o pérdidas. Al comienzo del trabajo se debió comprobar que el funcionamiento de esta herramienta era el esperado, y que introducía correctamente las limitaciones en la red.

Aunque en el trabajo nos hemos centrado en *Vidyo* y *Skype,* el entorno de laboratorio podría utilizarse para someter a análisis a otras aplicaciones del mismo tipo.

Medimos en primer lugar la reacción de ambas herramientas frente a las limitaciones de ancho de banda. Se comprobó que ambas soluciones reaccionan rápidamente ante las modificaciones del ancho de banda disponible, utilizando diferentes comportamientos para adaptarse. En *Vidyo* se distinguen dos comportamientos diferentes, en función del ancho de banda disponible. Por debajo de un umbral se producen grandes oscilaciones en el tráfico, que pueden deberse a que el sistema trata de encontrar el límite del ancho de banda que puede utilizar. Pero cuando se alcanza el límite, el sistema lo detecta y reduce su tráfico. Por su parte, *Skype* mantiene el mismo comportamiento a largo de toda la prueba.

Las limitaciones se introdujeron tanto en el enlace de subida como en el de bajada. En el caso de *Vidyo* pudimos ver cómo la adaptación al medio es más lenta cuando las limitaciones están en el enlace de bajada. Sin embargo, *Skype,* al ser una solución *peer-to-peer* sin servidor, no muestra ninguna diferencia entre los dos escenarios.

Posteriormente se caracterizaron las reacciones de ambos sistemas frente a las pérdidas en la red. En el caso de *Vidyo* sólo observamos un ligero aumento en el ancho de banda respecto al tráfico del *emisor* mientras que en *Skype,* sí se experimentan más variaciones, observando un aumento notable del tráfico conforme aumentan las pérdidas hasta un determinado umbral a partir del cual el tráfico cae.

La última de las condiciones de la red que se modificó fue el retardo extremo a extremo. Usando el emulador, introdujimos retardos adicionales, comprobando que ambas aplicaciones apenas reaccionan ante este efecto.

Globalmente, una vez visto el comportamiento de ambas aplicaciones frente a las variaciones de la red, podemos decir que tanto *Vidyo* como *Skype* son soluciones muy competentes. *Vidyo* es una solución propietaria enfocada al entorno empresarial, que permite realizar videoconferencias de alta calidad confiando en la codificación de vídeo por capas. Su adaptación al medio se basa por tanto en la robustez de su codificación, modificando el ancho de banda y eliminando o añadiendo capas a la imagen en función de las condiciones de la red y de las características de los terminales de los usuarios.

Por su parte, *Skype* es una solución gratuita enfocada a usuarios particulares. Su adaptación al medio consiste en aplicar redundancia y modificar los parámetros de tamaño y tiempo entre paquetes.

Podemos concluir que tanto *Vidyo* como *Skype* intentan hacer frente a las limitaciones que se encuentran en el medio, aunque de manera diferente. Esto es debido a que *Vidyo* parte de la ventaja de tener un servidor dedicado y de usar un *codec* por capas. Por contra, *Skype* está desarrollado para enfrentarse a una gran diversidad de condiciones en las conexiones a Internet, donde el ancho de banda no está garantizado.

Como última conclusión, debemos destacar que a lo largo de este proyecto se ha conseguido adquirir una metodología de trabajo para abordar el análisis en todas sus fases, comenzando desde el planteamiento del problema, pasando por la realización de las pruebas y llegando hasta la obtención de resultados y su interpretación. Con esta metodología podríamos enfrentarnos al estudio de otros problemas que pudieran surgir tanto en un entorno de investigación como en uno empresarial.

5.2 Líneas futuras

Durante la realización de este proyecto han ido surgiendo nuevos puntos de interés, que podrían constituir líneas futuras de estudio.

En primer lugar, se podría realizar un análisis de la calidad obtenida por las aplicaciones desde un punto de vista subjetivo. Sería interesante realizar sesiones de videoconferencia entre usuarios reales, mientras se aplican diferentes limitaciones en la red, de manera que pudieran evaluar desde un punto de vista subjetivo el comportamiento de la aplicación.

Además del análisis presentado, realizado en redes cableadas, se podrían realizar pruebas usando redes de acceso inalámbricas (*UMTS*, *Wi-Fi,* etc.), y utilizando también diferentes terminales de usuario. Debido al auge de las redes y dispositivos móviles, resultaría interesante conocer mejor el funcionamiento de estas aplicaciones en estas redes.

Durante la realización del proyecto se ha pensado que sería interesante realizar una aplicación que realizara todo el proceso completo de pruebas, desde la captura de datos, la introducción de las limitaciones y la representación gráfica de los parámetros más representativos.

Se podría ampliar el análisis, considerando también otras aplicaciones de videoconferencia como por ejemplo *Googletalk*.

Se podría finalmente realizar un análisis con mayor profundidad de *H.264/SVC*, centrado en la codificación de vídeo por capas. De esta forma, no sólo se observaría la adaptación del ancho de banda global, o del tamaño y tiempo entre paquetes, sino que podríamos saber qué capas aparecen o desaparecen en función de las limitaciones de la red.

5.3 Planificación del proyecto

En la Fig.5.1 podemos ver la planificación del proyecto a lo largo del tiempo mediante un diagrama de Gantt.

Id.	Nombre de tarea	Comienzo	Fin
1	Bibliografía y documentación	03/09/2012	15/02/2013
2	Análisis y puesta en marcha de las soluciones Vidyo y Skype	03/09/2012	01/10/2012
3	Desarrollo, configuración y montaje de los escenarios de pruebas	01/10/2012	30/11/2012
4	Calibración de todos los elementos y equipos	03/12/2012	17/12/2012
5	Configuración y desarrollo de las herramientas	17/12/2012	09/04/2013
6	Pruebas	01/02/2013	03/06/2013
7	Analisis resultados	01/04/2013	01/07/2013
8	Redacción memoria	02/05/2013	30/07/2013

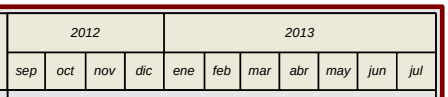


Fig.5.1. Diagrama de Gantt del trabajo realizado durante el proyecto.

Bibliografía

[KPL+09] Kaune, S., Pussep, K., Leng, C., Kovacevic, A., Tyson, G., & Steinmetz, R. (2009, February). Modelling the internet delay space based on geographical locations. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on* (pp. 301-310). IEEE.

[ROA07] CIMS-Live: Collaborative and Interactive Media Streaming Platform in P2P EnvironmentDepartment of Telematics Engineering, Technical University of Catalonia / I2cat Foundation. André Ríos, Alberto J. González, Antoni Oller and Jesús Alcober. Barcelona, Spain.

[BMM+08] Tracking Down Skype Traffic. Dario Bonfiglio, Marco Mellia, Michela Meo, Nicolo Ritacca Dario Rossi. Politecnico di Torino – Dipartimento di Elettronica ENST ParisTech–INFRES Department.

[RMM08] A Detailed Measurement of Skype Network Traffic. Dario Rossi Marco Mellia, Michela Meo. ENST ParisTech – INFRES Politecnico di Torino – DENLE.

[GDJ06] S. Guha, N. Daswani and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System", International Workshop on Peer-to-Peer Systems, Santa Barbara, CA, Feb. 2006.

[XR07] A Measurement based Study of the Skype Peer to Peer VoIP Performance. Haiyong Xie Yang Richard Yang. Computer Science Department, Yale University.

[XN99] Xipeng Xiao; Ni, L.M., "Internet QoS: a big picture," *Network, IEEE* , vol.13, no.2, pp.8,18,Mar/Apr 1999.

http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=768484&tag=1

Última visita 17/07/2013

[Bell69] Bell Laboratories Record http://long-lines.net/tech-equip/Picturephone/BLR0569/picturephone.pdf

Última visita 17/07/2013

[Bie09] Biello, David. Can Videoconferencing Replace Travel?, Scientific American, March 18, 2009.

[Dav06] Andrew W. Davis, "A READY MARKET" Introducing H.264-SVC: next-generation technology for videoconferencing over IP and 3G networks. March 2006.

[Zdn05] http://www.zdnet.com/blog/ou/high-definition-video-conferencing-is-here/59
Última visita 17/07/2013

[Jkc05a] http://www.jkcit.co.uk/pdf/polycom-hd-videoconferencing-whitepaper.pdf Última visita 17/07/2013

[Jkc05b] http://www.jkcit.co.uk/news/20060522.htm. Última visita 17/07/2013

[Iperf] https://code.google.com/p/iperf/. Última visita 17/07/2013

[Vid13] Vidyo, sistema de videoconferencia, http://es.vidyo.com/. Última visita 13/6/2013.

[Hu12] Bert Huber (PowerDNS.COM BV) http://www.lartc.org. Última visita 13/06/2013

[Tcp13] http://www.tcpdump.org. Última visita 13/06/2013.

[SFR+09] Jose Saldana, Julian Fernandez-Navajas, Jose Ruiz-Mas, Eduardo A. Viruete Navarro, Luis Casadesus Pazos, Jose Ignacio Aznar Baranda. "Sistema Multiagente de Medidas Activas E2E" .Actas del XXIV Simposium Nacional de la Union Cientifica Internacional de Radio (URSI 2009), pp. 247-248, ISBN 978-84-8102-550-7. Santander (Spain). Sep. 2009

Anexo A. Acrónimos

AVL	Adaptative Video Layering
ADSL	Asymmetric Digital Subscriber Line
ARP	Address Resolution Protocol
AVC	Advanced Video Coding
ATM	Asynchronous Transfer Mode
CPU	Central Processing Unit
ETG	End to End Traffic Generator
FIFO	First Input First Output
GNU	GNU is Not Unix
GSM	Global System for Mobile
HTTP	HyperText Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ITU	International Telecommunication Union
LTE	Long Term Evolution
MAC	Medium Access Control
MOS	Mean Opinion Score
NAT	Network Address Translation
P2P	Peer to Peer
QoS	Quality of Service
RFC	Request For Comments
RTC	Red Telefónica Conmutada
SIP	Session Initiation Protocol
SNR	Signal Noise Relation
SVC	Scalable Video Coding
TBF	Token Bucket Filter
TC	Traffic Control
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over Internet Protocol

WIFI	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
XML	eXtensible Markup Language

Anexo B. Puesta en marcha del sistema de pruebas

B.1 Configuración equipos y cableado

En las figuras y tablas siguientes podemos ver el montaje del sistema de pruebas de nuestro laboratorio (Fig.B.1), en el que puede apreciarse la cámara, la pantalla en la que se reproducía el vídeo, la máquina en la que se realizaban las capturas, uno de los mini-pc, y los *hub*. El esquema de red se muestra en la Fig.B.2, y su direccionamiento en las Tablas B.1 y B.2.

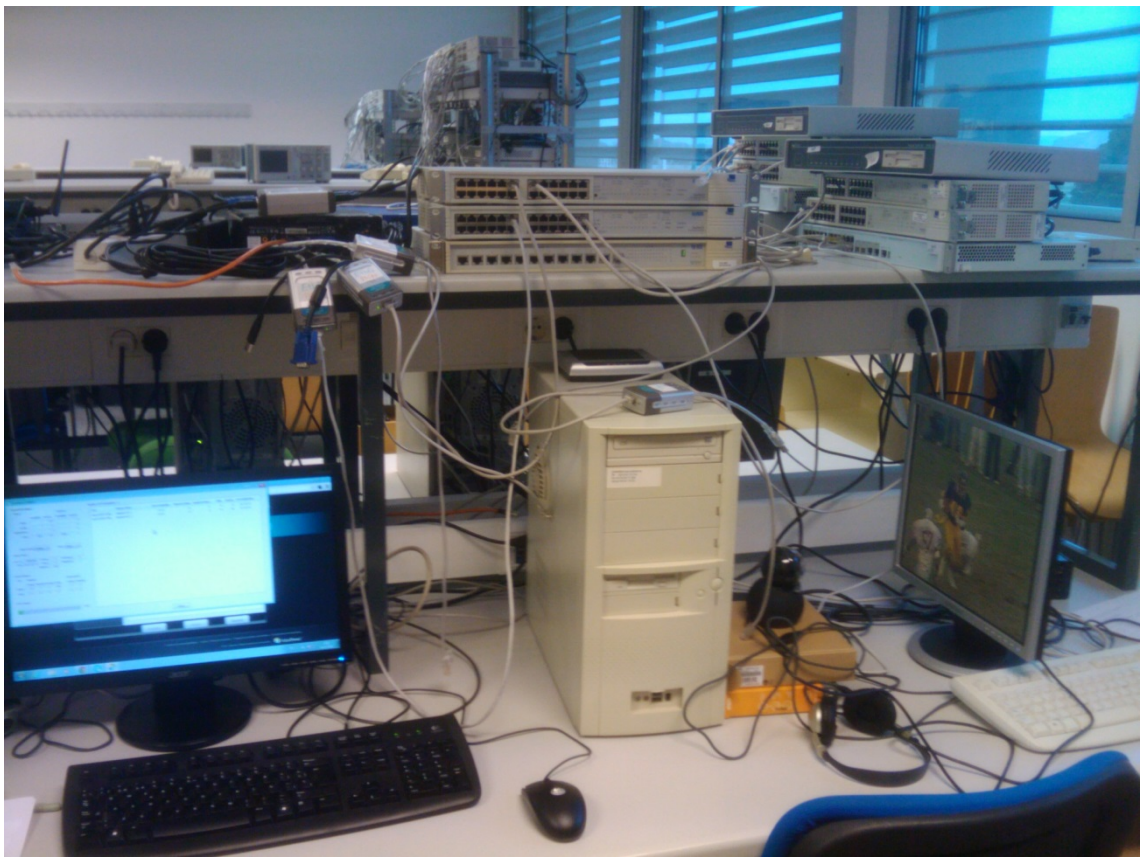


Fig.B.1. Montaje del sistema de pruebas en el laboratorio.

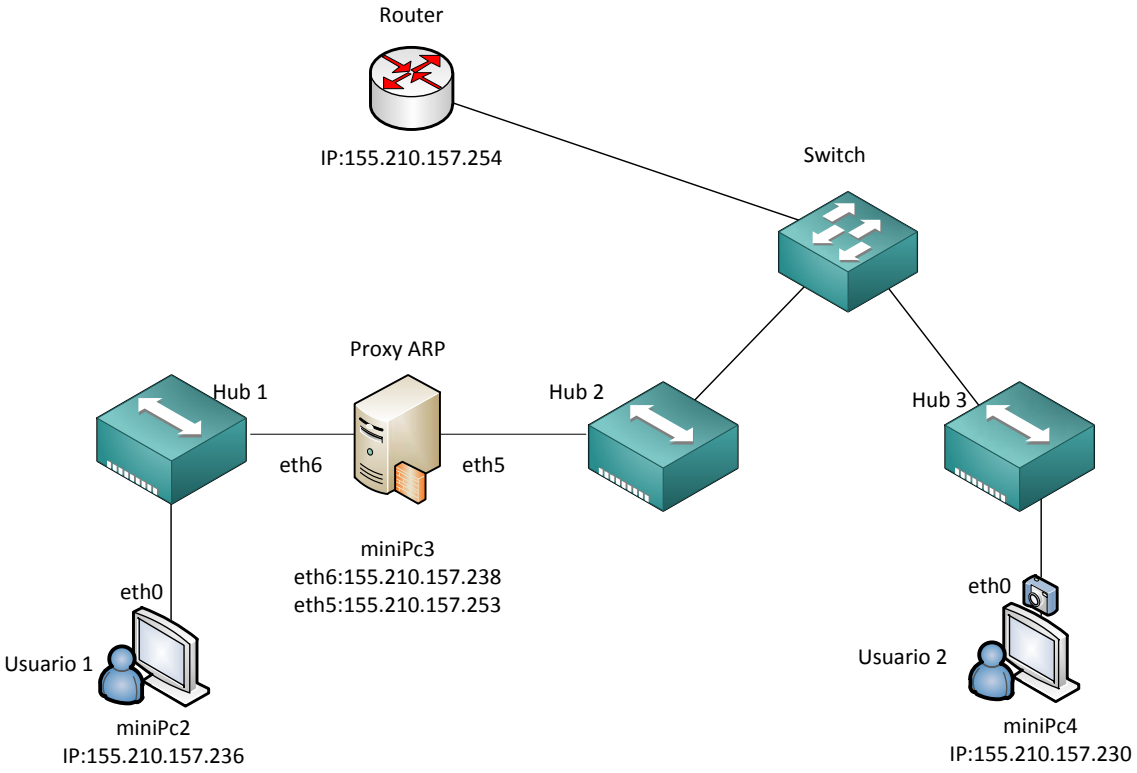


Fig.B.2. Esquema red y direccionamiento

Equipo	Interfaz	Dirección IP
miniPc2	Eth0	155.210.157.236
miniPc3	Eth5	155.210.157.253
	Eth6	155.210.157.238
miniPc4	Eth0	155.210.157.230

Tabla B.1 Direccionamiento IP, escenario de bajada.

Equipo	Interfaz	Dirección IP
miniPc2	Eth0	155.210.157.230
miniPc3	Eth5	155.210.157.253
	Eth6	155.210.157.238
miniPc4	Eth0	155.210.157.236

Tabla B.2 Direccionamiento IP, escenario de subida.

B.2 Configuración interfaces de red

A continuación describimos la configuración de las interfaces de red externas en el equipo *Proxy* para su funcionamiento en nuestro laboratorio con la distribución *Debian Linux*. Para ello tenemos que editar y modificar el fichero **/etc/network/interfaces.conf**.

Esta configuración la hemos replicado en cada una de las máquinas, de este modo podemos intercambiar la función de cada máquina en un momento dado. Para ello también tenemos que configurar el fichero **/etc/udev/rules.d/70.persistent-net.rules**, donde se asocia la dirección MAC de la interfaz a una dirección IP. Así, podemos conectar las interfaces externas a cualquiera de las máquinas, manteniendo la dirección IP.

/etc/network/interfaces.conf

```
# This file describes the network interfaces available on your
system
# and how to activate them. For more information, see
interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary (PCI) network interface
allow-hotplug eth0
iface eth0 inet static
        address 192.168.7.13
        netmask 255.255.255.0
        network 192.168.7.0
        broadcast 192.168.7.255
# The usb1 proxyarp exterior network interface
allow-hotplug eth5
iface eth5 inet static
        address 155.210.157.253
        netmask 255.255.255.0
        network 155.210.157.0
        broadcast 155.210.157.255
        gateway 155.210.157.254
        # dns-* options are implemented by the resolvconf
package, if installed
        dns-nameservers 155.210.12.9
        #hwaddress ether 00:80:c8:3c:bf:7e
# The usb exterior network interface
allow-hotplug eth7
iface eth7 inet static
        address 155.210.157.230
        netmask 255.255.255.0
        network 155.210.157.0
        broadcast 155.210.157.255
        gateway 155.210.157.254
```

```
        # dns-* options are implemented by the resolvconf
package, if installed
        dns-nameservers 155.210.12.9
        #hwaddress ether 84:c9:b2:cf:cf:93
# The usb2 capture network interface
allow-hotplug eth2
iface eth2 inet static
        address 155.210.157.234
        netmask 255.255.255.0
        network 155.210.157.0
        broadcast 155.210.157.255
        gateway 155.210.157.254
        # dns-* options are implemented by the resolvconf
package, if installed
        dns-nameservers 155.210.12.9
        #hwaddress ether 00:80:c8:3c:cb:38
# The usb3 proxyarp interior network interface
allow-hotplug eth6
iface eth6 inet static
        address 155.210.157.238
        netmask 255.255.255.248
        network 155.210.157.232
        broadcast 155.210.157.239
        #hwaddress ether 00:80:c8:3c:bf:7c
# The usb4 capture network interface
allow-hotplug eth4
iface eth4 inet static
        address 155.210.157.236
        netmask 255.255.255.0
        network 155.210.157.0
        broadcast 155.210.157.255
        gateway 155.210.157.254
        # dns-* options are implemented by the resolvconf
package, if installed
        dns-nameservers 155.210.12.9
        #hwaddress ether 00:80:c8:3c:bf:7b
```

/etc/udev/rules.d/70.persistent-net.rules

```
# This file was automatically generated by the
/lib/udev/write_net_rules
# program, run by the persistent-net-generator.rules rules file.
#
# You can modify it, as long as you keep each rule on a single
# line, and change only the value of the NAME= key.

# PCI device 0x10ec:0x8168 (r8169)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="e0:69:95:13:ff:ed", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"

# PCI device 0x168c:0x002e (ath9k)
```

```
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="1c:4b:d6:dc:90:b4", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="wlan*", NAME="wlan0"

# USB device 0x:0x (asix)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:80:c8:3c:bf:7e", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth5"

# USB device 0x:0x (asix)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:80:c8:3c:cb:38", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth2"

# USB device 0x:0x (asix)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:80:c8:3c:bf:7c", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth6"

# USB device 0x:0x (asix)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:80:c8:3c:bf:7b", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth4"

# USB device 0x:0x (asix)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="84:c9:b2:cf:cf:93", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth7"
```

B.3 Configuración del *Proxy ARP*

Para que una máquina haga la función de *Proxy ARP* hemos editado y modificado el siguiente fichero:

/etc/sysctl.conf

```
#
# /etc/sysctl.conf - Configuration file for setting system
variables
# See /etc/sysctl.d/ for additonal system variables
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3

##############################################################3
# Functions previously found in netbase
#
```

```
# Uncomment the next two lines to enable Spoof protection
(reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

#habilitar forwarding para eth1 y eth3
#net.ipv4.conf.eth1.forwarding=1
#net.ipv4.conf.eth3.forwarding=1

#habilitar proxy-arp para eth5 y eth6
net.ipv4.conf.eth5.proxy_arp=1
net.ipv4.conf.eth6.proxy_arp=1

# Uncomment the next line to enable packet forwarding for IPv6
#  Enabling this option disables Stateless Address
Autoconfiguration
#  based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

C. Scripts

C.1 Captura de datos (*Captura*.sh)

Para iniciar la captura de datos ejecutamos el script ***Captura.sh*** pasando como parámetro el tiempo de captura en el equipo *Proxy*. Este script recoge todos los paquetes (80 bytes por paquete) que pasan por las interfaces *eth5*, *eth6* y *eth0*. *eth5 y eth6* son las interfaces a cada lado del *Proxy*. *eth0* es la interfaz que captura en el otro extremo de la comunicación.

Captura.sh

```
hora=`date +"%T"`

/usr/sbin/tcpdump -i eth5 -w capturaPc3_PAEX"_"$hora -nn -s 80&
/usr/sbin/tcpdump -i eth6 -w capturaPc3_PAIN"_"$hora -nn -s 80&
/usr/sbin/tcpdump -i eth0 -w capturaPc3_SERV"_"$hora -nn -s 80&
sleep $1

rm ./fichproc
ps aux | grep "tcpdump" | cut -b 10-14 | head -n3 > ./fichproc
fich="./fichproc"
read PROC<$fich
kill -15 $PROC

cat fichproc | tail -n2 > ./fichproc2
fich="./fichproc2"
read PROC<$fich
kill -15 $PROC

cat fichproc2 | tail -n1 > ./fichproc3
fich="./fichproc3"
read PROC<$fich
kill -15 $PROC
```

C.2 Scripts de limitación de Ancho de banda (*limitacionTC.sh*)

Para la limitación de ancho de banda ejecutamos en el *Proxy* el script ***limitacionTC.sh***, que será diferente para cada escenario. Se diferencian según la interfaz donde aplicamos la limitación.

Para el escenario de limitación en el enlace de bajada el *script* será el siguiente:

```
sudo tc qdisc ls
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc add dev eth6 root tbf rate 1.2mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root tbf rate 1.0mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
```

```
sudo tc qdisc change dev eth6 root tbf rate 0.8mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root tbf rate 0.6mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root tbf rate 0.4mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root tbf rate 0.6mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root tbf rate 0.8mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root tbf rate 1.0mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root tbf rate 1.2mbit limit 104kb
burst 2kb
sleep 60
sudo tc qdisc del dev eth6 root
```

Para el escenario de limitación en el enlace de subida será igual, pero limitando la interfaz *eth5*:

```
sudo tc qdisc ls
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc add dev eth5 root tbf rate 1.2mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root tbf rate 1.0mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root tbf rate 0.8mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root tbf rate 0.6mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root tbf rate 0.4mbit limit 104kb
burst 2kb
sleep 60
```

```
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root tbf rate 0.6mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root tbf rate 0.8mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root tbf rate 1.0mbit limit 104kb
burst 2kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root tbf rate 1.2mbit limit 104kb
burst 2kb
sleep 60
sudo tc qdisc del dev eth5 root
```

C.3 Scripts limitación de pérdidas (*limitaciónPerdidas.sh*)

Para limitación de Perdidas ejecutamos el script *limitacionPerdidas.sh*.

En el escenario de limitación en el enlace de bajada el *script* será el siguiente:

```
sudo tc qdisc ls
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc add dev eth6 root netem loss 1% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 2% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 3% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 4% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 5% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 6% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 7% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 8% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 9% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem loss 10% limit 104kb
```

```
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc del dev eth6 root
```

En el escenario con limitación en el enlace de subida el *script* será el siguiente:

```
sudo tc qdisc ls
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc add dev eth5 root netem loss 1% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 2% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 3% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 4% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 5% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 6% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 7% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 8% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 9% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth5 root netem loss 10% limit 104kb
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc del dev eth5 root
```

C.4 Script limitación retardos (*limitacionDelay.sh*)

Para la limitación de retardos ejecutamos el *script* ***limitacionDelay.sh***

```
sudo tc qdisc ls
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc add dev eth6 root netem delay 50ms
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem delay 100ms
```

```
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem delay 150ms
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem delay 200ms
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem delay 250ms
sleep 60
sudo ping 155.210.157.254 -c1
sudo tc qdisc change dev eth6 root netem delay 300ms
sleep 60
sudo tc qdisc del dev eth6 root
```

C.5 Scripts de procesado de datos

Una vez tenemos los ficheros de captura, debemos procesar los datos para poder interpretarlos. Primero convertimos estos ficheros de formato binario a formato texto mediante la herramienta *tcpdump*. Posteriormente, tratamos y extraemos la información que nos interesa mediante las herramientas *perl* y *awk*.

El *script* para procesar los datos es el siguiente:

Procesado.sh

```
./tiempo.sh $1

./Medidas/capturasTexto/ejecutaAwk.sh $1


perl ./Medidas/parametrosGraficas/histograma.pl
./Medidas/parametrosGraficas/$1.txt 30 90 >
./Medidas/parametrosGraficas/$1"_histo".txt

awk -f ./Medidas/parametrosGraficas/AwkAnchoBanda
./Medidas/parametrosGraficas/$1.txt >
./Medidas/parametrosGraficas/$1"_AB".txt

awk -f ./Medidas/parametrosGraficas/AwkAnchoBanda
./Medidas/parametrosGraficas/$1"tick5.txt" >
./Medidas/parametrosGraficas/$1"tick5_AB.txt"
```

A su vez, este *script* hace una llamada a otros *script* escritos en diferentes lenguajes (*shell*, *awk* y *perl*).

El primero de ellos es *tiempo.sh*. Este *script* trata de filtrar los datos capturados por la herramienta *tcpdump*, según el escenario y la aplicación a tratar. El resultado es un fichero en formato texto.

tiempo.sh

```
echo Fichero : $1


#Skype
#tcpdump -r ./Medidas/$1 -nn -tt src 155.210.157.230 and dst
155.210.157.236 and udp > .\/Medidas\/capturasTexto\/$1.txt
#tcpdump -r ./Medidas/$1 -nn -tt src 155.210.157.236 and dst
155.210.157.230 and udp > .\/Medidas\/capturasTexto\/$1.txt
#tcpdump -r ./Medidas/$1 -nn -tt udp >
.\/Medidas\/capturasTexto\/$1.txt

#Vidyo
#tcpdump -r ./Medidas/$1 -nn -tt src 193.144.229.175 and dst
155.210.157.236 and udp > .\/Medidas\/capturasTexto\/$1.txt
#tcpdump -r ./Medidas/$1 -nn -tt src 155.210.157.230 and dst
193.144.229.175 and udp > .\/Medidas\/capturasTexto\/$1.txt
#tcpdump -r ./Medidas/$1 -nn -tt src 193.144.229.175 and dst
155.210.157.230 and udp > .\/Medidas\/capturasTexto\/$1.txt
#tcpdump -r ./Medidas/$1 -nn -tt src 155.210.157.236 and dst
193.144.229.175 and udp > .\/Medidas\/capturasTexto\/$1.txt
#tcpdump -r $1 -nn -tt src 155.210.158.55 and dst
193.144.229.175 and udp > .\/Medidas\/capturasTexto\/$1.txt

#caraterizacion TC
#tcpdump -r ./Medidas/$1 -nn -tt src 155.210.157.236 and dst
155.210.157.230 and udp > .\/Medidas\/capturasTexto\/$1.txt
```

El siguiente script es ***ejecutaAwk.sh*** que hace una llamada a un proceso *awk* ***parametrosAWK*** pasándole el parámetro *tick*.

ejecutaAwk.sh

```
echo "awk -v tick=1 -f ./Medidas/capturasTexto/parametrosAWK
./Medidas/capturasTexto/$1 > .\/Medidas\/parametrosGraficas\/$1"
awk -v tick=1 -f ./Medidas/capturasTexto/parametrosAWK
./Medidas/capturasTexto/$1.txt >
.\/Medidas\/parametrosGraficas\/$1.txt

RET=$?
if [ $RET = 0 ]
then
echo "AWK process Succesfully!!"
else
echo "AWK process Failed!!"
exit 1
fi

awk -v tick=5 -f ./Medidas/capturasTexto/parametrosAWK
./Medidas/capturasTexto/$1.txt >
.\/Medidas\/parametrosGraficas\/$1"tick5.txt"

RET=$?
if [ $RET = 0 ]
```

```
then
echo "AWK process Succesfully!!"
else
echo "AWK process Failed!!"
exit 1
fi
```

El siguiente *script* es ***parametrosAWK***, encargado de procesar los datos filtrados y realizar los cálculos de ancho de banda, tiempo entre paquetes, tamaño de los paquetes y número de paquetes por segundo.

parametrosAWK

```
BEGIN {
FS=" "
bits=0
bitsAB=0
line=0
paqueteps=0
pps=0
tick
}
#ACABA LA PRIMERA PARTE
{
  if (tick==""){
      tick=1
  }



  if (line>0)
     {
     mtemp = substr($1,1)
     split (mtemp, micros, ".")
     tacmic=micros[1]*1000000+micros[2]              #tiempo de
llegada del paquete
     line=line+1                                  #Paquete
     paqueteps=paqueteps+1
     tiempo_relativo=mtemp-tinicio
     #tiempo acumulado desde la llegada del primer paquete
     tick_actual=micros[1]

     if(int(tiempo_relativo/tick)>int(tiempo_relativo_anterior/t
ick))
     {
          pps=paqueteps
          paqueteps=0
          AB=bitsAB/tick
          #bitsAB=0
          tiempo_relativo_anterior=tiempo_relativo
```

```
            #printf AB/1000000 "\t"
      }

      bitsAB=bitsAB+(($8+28)*8)
      bits=bits+(($8+28)*8)                        #bits acumulados
      tiempo=tacmic-tinmic

      printf tiempo_relativo "\t" $1 "\t" ($8+28)*8 "\t"
            #($8+28)*8 bits tamaÃ±o paquete ip

      tiempo_entre_paquetes=tacmic-tan    #tiempo entre paquetes
      printf tiempo_entre_paquetes "\t"
      ab=bits/tiempo                               #Ancho de banda
      printf ab "\t"
      abi=($8+28)*8/(tacmic-tan)
      printf abi "\t"                              #Ancho de banda
instantÃ¡neo
      printf AB/1000000 "\t"
      printf int(pps/tick) "\t"
      if (pps==0)
      {
            printf "\n"
      }
      else
      {
            printf int(bitsAB/(8*pps)) "\n"
            bitsAB=0
      }
      AB=""
      pps=0
      tan=tacmic
      }
  if (line==0)
      {
      mtemp = substr($1,1)
      split (mtemp, micros, ".")
      tinmic=micros[1]*1000000+micros[2]
      tick_anterior= micros[1]
      tick_actual=tick_anterior
      line=line+1
      paqueteps=paqueteps+1
      pps=0
      tan=tinmic
      tinicio=mtemp
      bitsAB=(($8+28)*8)
      bits=bitsAB
      tiempo_relativo_anterior=0
      printf tiempo_relativo_anterior "\t" $1 "\t" ($8+28)*8 "\t"
0 "\t" 0 "\t" 0 "\t" 0 "\t\n"

      }
}

END {
}
```

El *script* ***AnchoBandaAWK*** presenta en otro fichero los resultados de tiempo relativo, ancho de banda, número de paquetes por segundo y tamaño medio de los paquetes para su posterior volcado a una hoja Excel.

AnchoBandaAWK

```
BEGIN {
FS=" "
line=0
}
#ACABA LA PRIMERA PARTE
{

      line=line +1
      if($7!=0)
      {
            printf $1 "\t"
            printf $7 "\t"
            printf $8 "\t"
            printf $9 "\n"
      }


}

END {
}
```

El script ***histograma.pl*** está programado en *perl* y clasifica los paquetes en rangos de tamaño para realizar un histograma en un determinado intervalo de tiempo.

```
#
#
#
use strict;
use warnings;


my $infile=$ARGV[0];
my $tInicio=$ARGV[1];
my $tFin=$ARGV[2];
my $tiempoRelativo=0;
my $tInicialCaptura;
my $contadorlineas=0;
my @aHistograma;
my $indicebyte;

open (DATA,"<$infile")
      || die "Can't open $infile $!";
```

```
#inicializar array
for (my $i=0; $i<=150; $i++) {
$aHistograma[$i]=0;
}


while (<DATA>) {

my @x = split(' ');

if ($contadorlineas==0)
{
     $tInicialCaptura=$x[0];

}

$tiempoRelativo= $x[0];

$indicebyte=int($x[2]/80);


if($tiempoRelativo >= $tInicio && $tiempoRelativo <= $tFin)
{
     $aHistograma[$indicebyte]=$aHistograma[$indicebyte] + 1;


}


#print "$aHistograma[$contadorlineas]\t";
#print "$tiempoRelativo\t";
#print "$contadorlineas\t";
#print "$byte\n";

$contadorlineas++;
}

close DATA;

#my $paquetes=0;
#foreach (@aHistograma){

#    print "$_\n";
     #$paquetes=$paquetes+$_;
     #print "$paquetes\n";
#}

my $indice;
for (my $i=1; $i<=150; $i++) {
     $indice=$i*10;
     print "$indice\t$aHistograma[$i]\t";
     #$paquetes=$paquetes+$aHistograma[$i];
     #print "$paquetes\n";
}

exit(0);
```

D. Instalación y configuración de las aplicaciones

D.1 Vidyo

Para la instalación de la aplicación *Vidyo* tenemos que conectarnos al servidor a través de un navegador *web* y descargar el software en función del sistema operativo. Una vez instalado el software, el administrador del servidor (Fig. D.1) tiene que crear las cuentas de usuario para podernos conectar al sistema.

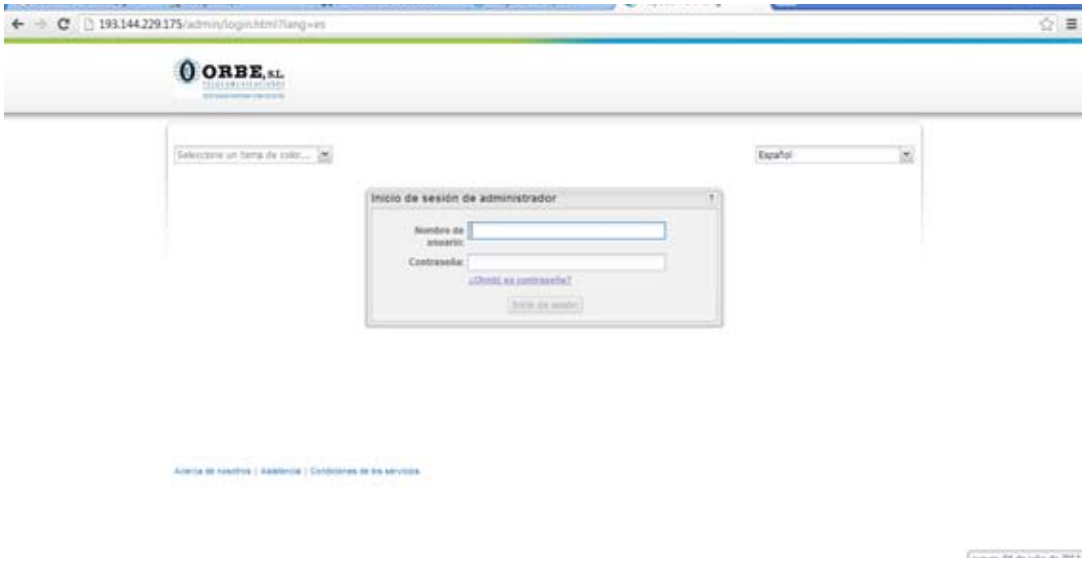


Fig.D.1 Pantalla de acceso al servidor

En la Fig.D.2 vemos el menú principal cuando accedemos al servidor como administrador, donde podemos crear usuarios, grupos de usuarios, así como gestionar las videoconferencias en curso.

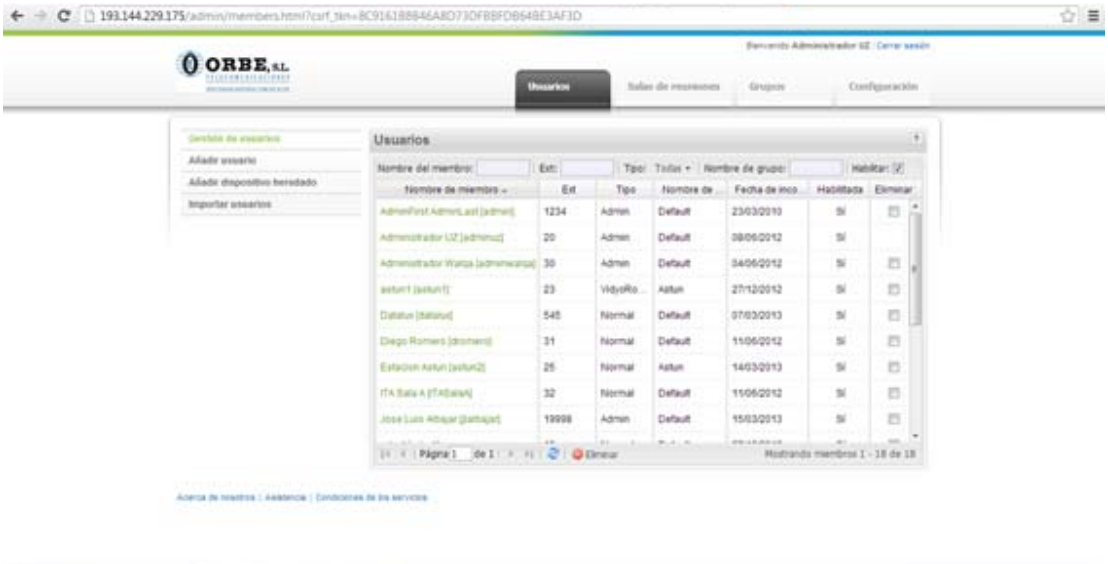


Fig.D.2 Pantalla de menú principal

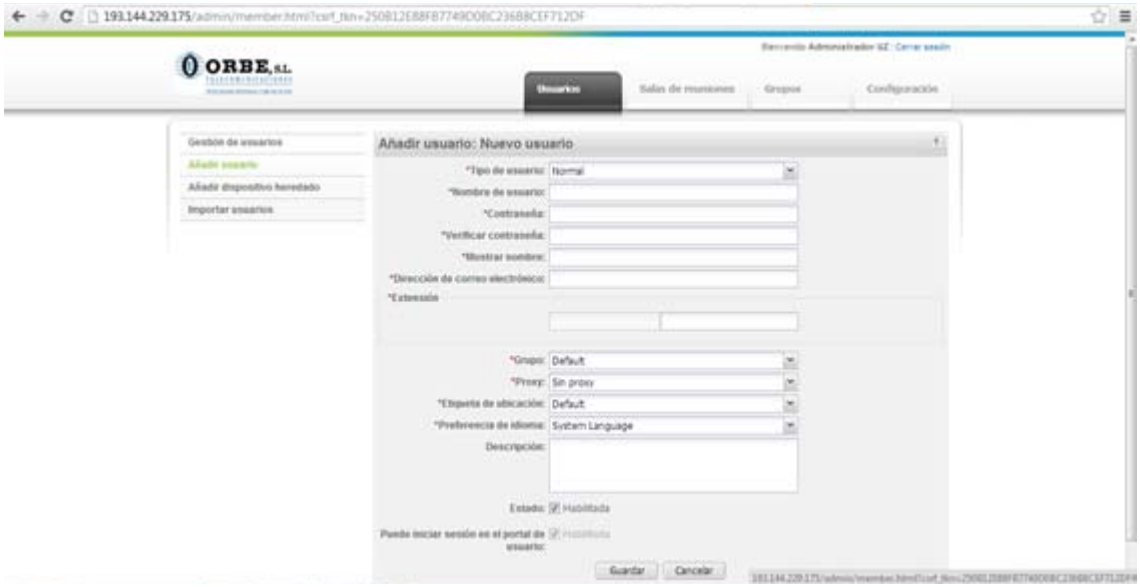


Fig.D.3 Pantalla de creación de usuarios

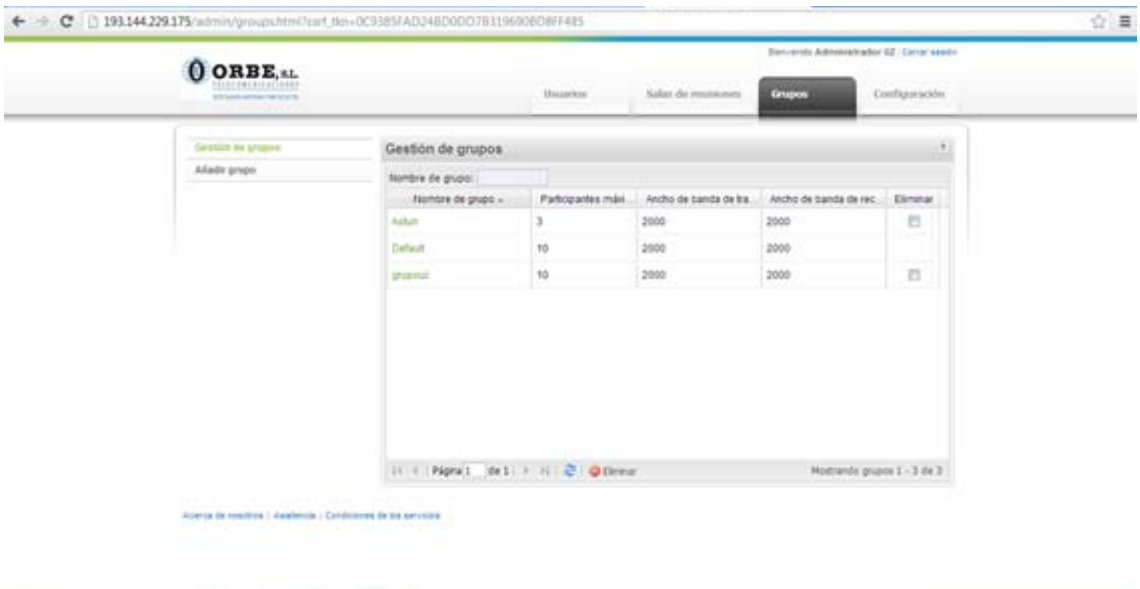


Fig.D.4. Pantalla de gestión de grupos de usuarios

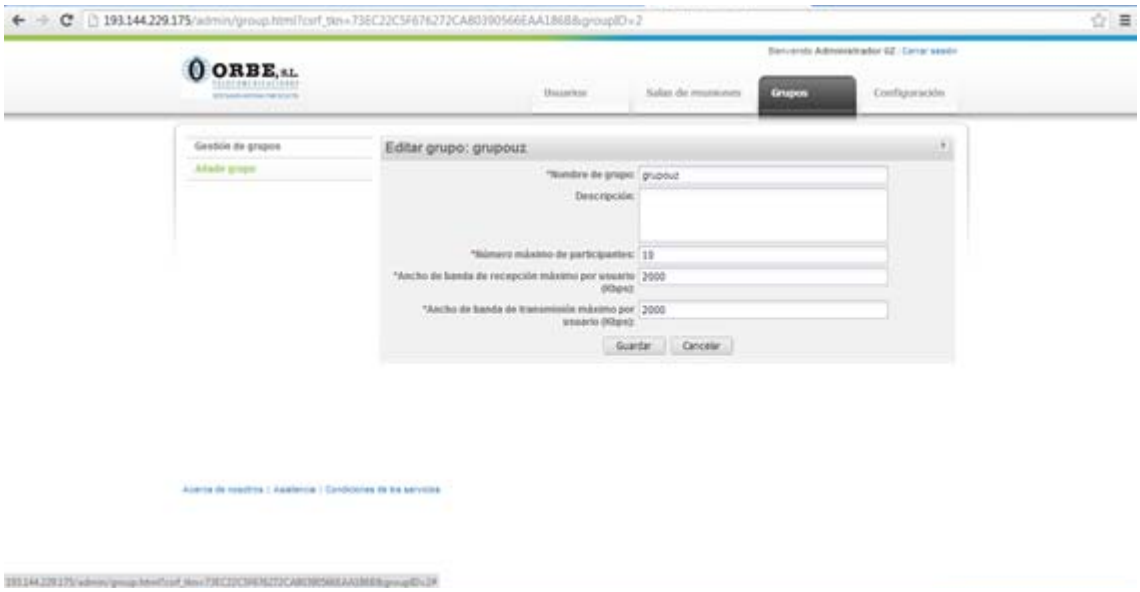


Fig.D.5. Pantalla de creación de grupos de usuarios.

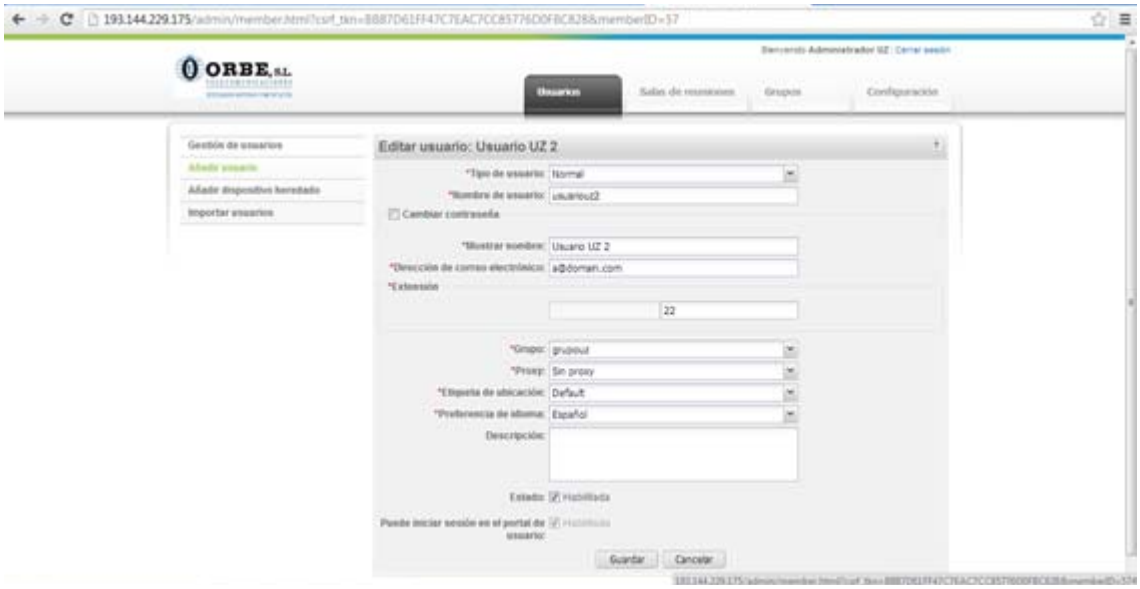


Fig.D.6. Pantalla de configuración del usuario *usuariouz2*

Las siguientes figuras muestran la aplicación de videoconferencia de usuario.

Una vez hemos entrado en la aplicación, en la pantalla principal podemos realizar una serie de acciones como ver nuestros contactos, las salas de videoconferencia, invitar a otro usuario a una videoconferencia, controlar las reuniones activas o realizar alguna configuración sencilla (cambio de idioma, contraseña…).

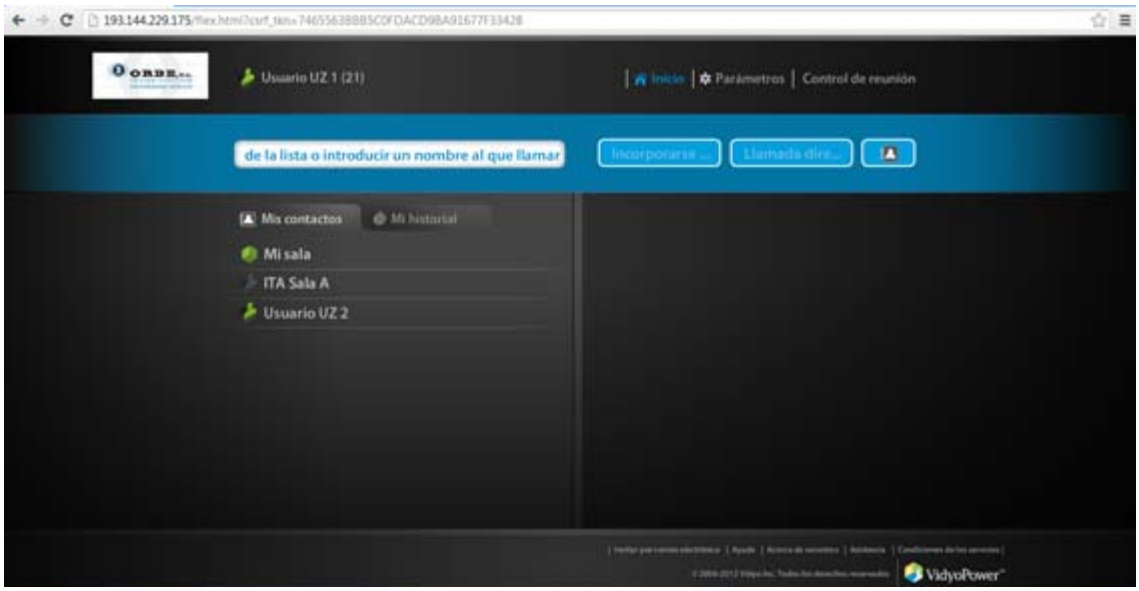


Fig.D.7. Pantalla principal de la aplicación.

Una vez establecida la videoconferencia podemos realizar una serie de configuraciones como la calidad del vídeo, la velocidad de transmisión y recepción, etc. Para las pruebas que hemos realizado configuramos de la velocidad de transmisión y recepción en modo automático (Fig.D.9).

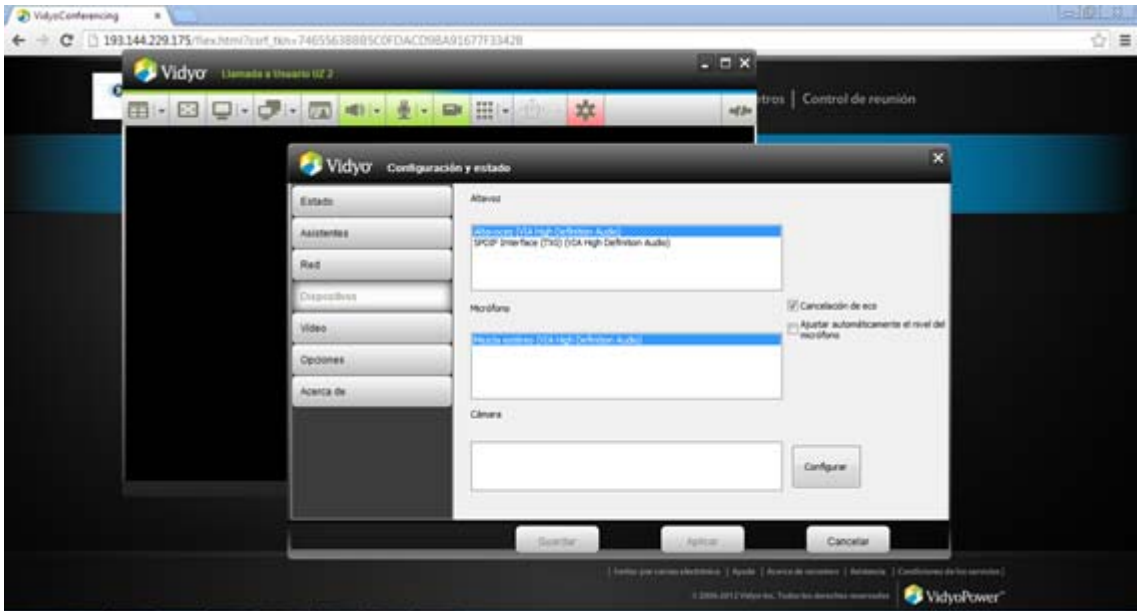


Fig.D.8. Pantalla de configuración y estado de la videoconferencia.

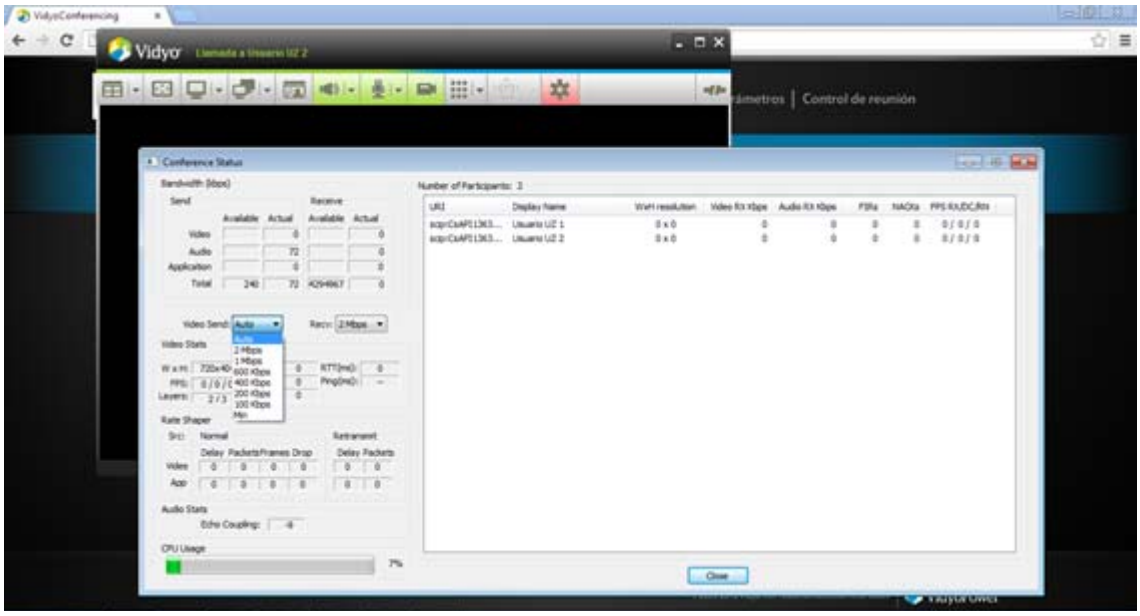


Fig.D.9. Pantalla del estado de la videoconferencia

D.2 ETG (End to end traffic generator)

Para poner en funcionamiento esta herramienta y conseguir generar tráfico entre dos equipos debemos ejecutar el proceso *./Multithreating* pasándole como parámetro un fichero de configuracion *xml* donde parametrizamos las características del tráfico. Para generar tráfico uniforme editaremos este fichero y parametrizaremos las etiquetas `<numberOfPackets>,<timeBetweenPackets>,<packetsSize>.`

Para generar tráfico a ráfagas se parametrizará otro fichero adicional (etiqueta <file>) que llevará los diferentes tamaños de los paquetes y el tiempo entre ellos.

Como ejemplo, se muestra el fichero *xml* utilizado para generar tráfico uniforme.

Ejemplo tests_traficoUniforme.xml

```
 <test>
      <description>
           <id>test1_carlos</id><!-- id  -->
           <text>descripcion</text><!-- Test desciption  -->
           <beginningDate>07/06/2011 12:29:00</beginningDate>
<!-- "DD/MM/YYYY HH:mm:ss" -->
```

```
          <endDate>23/12/2013 18:10:00</endDate> <!--
"DD/MM/YYYY HH:mm:ss" -->
          <headerSize>28</headerSize><!-- Tamaño del header del
paquete de los paquetes, para calcular ancho de banda -->
     </description>
     <stream>
               <beginWait>0</beginWait><!-- time in seconds.
iniciar este tiempo despuÃ©s de la hora de inicio -->
               <localHost>155.210.157.236</localHost>
               <localInterface>any</localInterface>
               <localPort>5392</localPort>
               <remoteHost>155.210.157.230</remoteHost>
               <remoteInterface>any</remoteInterface>
               <remotePort>5002</remotePort>
               <frequency>0</frequency><!-- time in seconds. 0
just once and stop. Tiempo de espera para repetir el flujo. De
final a inicio -->
               <numberOfBursts>1</numberOfBursts>
               <timeBetweenBursts>600</timeBetweenBursts><!--
time in microseconds. De final a inicio -->
               <numberOfPackets>4161</numberOfPackets>
               <timeBetweenPackets>6000</timeBetweenPackets><!-
- time in microseconds. De inicio a inicio -->
               <packetsSize>1472</packetsSize><!-- TamaÃ±o en
Bytes del payload del paquete UDP?-->
               <rtt>0</rtt><!-- 1 si (rtt). 0 no (one way).-->
               <file>tamaños-tiempos 0.6.txt</file><!-- nombre
del fichero de tamaños y tiempos para traficos no uniformes,
dejarlo vacio si no es necesario -->
               <burstFile></burstFile><!-- nombre del fichero
de tiempo entre rÃ¡fagas y nÃºmero de paquetes en la rÃ¡faga,
dejarlo vacio si no es necesario -->
               <numberOfRepetitions>1</numberOfRepetitions><!--
repeat this communication. Al menos 1 repeticiÃ³n -->
               <codec></codec><!-- codec en caso de audio.
Admite G729a y G711, dejar en blanco si no es necesario -->
               <dejitterBufferDelay></dejitterBufferDelay><!--
en caso de audio, dejar en blanco si no es necesario -->
     </stream>
     <capture>
               <localHost>155.210.157.236</localHost>
               <remoteHost>155.210.157.230</remoteHost>
               <frequency>0</frequency><!-- time in seconds. 0
just once and stop -->
               <timeBetweenBursts>600</timeBetweenBursts><!--
time in seconds . Tiene que ser el mismo que en el stream-->
               <numberOfCaptures>1</numberOfCaptures><!--
Numero de capturas. Al menos 1 captura-->
               <time>70</time><!-- time to capture in seconds -
->
               <file></file><!-- tiempo a capturar, frecuencia
-->
               <capture>1</capture><!-- Indica donde el trafico
va a ser capturado. O destino, 1 origen, 2 ambos extremos.-->
     </capture>
</test>
```

E. Características de los equipos.

E.1 Equipos cliente y *Proxy ARP.*

ASRock Core 100HT

Caracteristicas:

- Intel® Core™ i3 CPU M 370 @ 2,4Ghz
- DDR3 1066MHz
- Gráficos Intel® HD, Soporta Full HD 1080p Playback
- Soporta ASRock XFast USB, XFast LAN Technologies

E.2 Interfaz de red externa.

D-Link DUB-E100

Características:

- Adaptador USB Fast 2.0 Ethernet
- Soporta 10/100 con auto detección
- Hasta 200Mbps en modo Full-Duplex

E.3 Equipos de transmisión y conmutación.

SuperStackII** Dual **Speed Hub 500

Características:

- Manufactured by: **3COM**
- Model/Part : **3C16611**
- Device Type: Hub - stackable
- Enclosure Type: Rack-mountable - external
- Ports Qty: 24 x Ethernet 10Base-T, Ethernet 100Base-TX
- Data Transfer Rate: 100 Mbps
- Data Link Protocol: Ethernet, Fast Ethernet
- Connectivity Technology: Wired
- Switching Protocol: Ethernet
- Status Indicators: Port status, link activity, collision status, power
- Compliant Standards: IEEE 802.3, IEEE 802.3u
- Interfaces: 24 x network node - Ethernet 10Base-T/100Base-TX - RJ-45 female - 24 2 x network stack device - 2 1 x management - RS-232C - 9 pin D-Sub (DB-9) male - 1

SuperStack@3 Switch 4500 26-Port

Características:

Layer 2 Services

- 802.1Q Port-based VLANs: 256
- 802.3ad (LACP)
- Manual aggregation
- Trunk groups: 13 groups (26-port),

- 8 10/100 ports or 2 Gigabit ports per group, autonegotiation of port speed and duplex, 802.3x full-duplex flow control, back pressure flow control for half-duplex
- 802.1D (STP)
- 802.1w (RSTP)
- BPDU protection included in Fast Start
- Internet Group Management Protocol (IGMP) v1 and v2 snooping
- IGMP Querier, filtering for 128 multicast groups

Layer 3 Services

- Hardware-based routing
- static routes: 12 in addition to the default address
- dynamic / static ARP entries : 1990 / 10,
- IP interfaces: 4,
- RIP, v1 and v2: 2K via default route plus 10 locally learned routes,
- IGMP v1 and v2 snooping,
- DHCP Relay: 2 KB max

Performance

- Packet Forwarding Rate : 6.5 million pps

Interfaces/Ports

- Fixed Ports/Connectors : 24 x RJ-45 2 dual-personality Gigabit port pairs: user configurable for RJ45 (copper) or SFP-based (fibre) interfaces
- Network Speeds Supported : 10/100 + 10/100/1000MBps

E.3 Cámara

Logitech Quickcam Orbit AF

Caracterisiticas:

- Motorized tracking
- Optica Carl Zeiss
- Lentes con Autofocus
- HD video
- Vídeo de hasta 30 fps