Universidad Zaragoza
1542

Proyecto Fin de Carrera

Análisis del tráfico multimedia en accesos a Internet

Autor

Alejandro Fabra Pineda

Director y ponente

Luis Sequeira Villarreal

Julián Fernández Navajas

ESCUELA DE INGENIERIA Y ARQUITECTURA

Año 2014

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

Resumen

"Análisis del tráfico multimedia en accesos a Internet"

En la actualidad existen diversas alternativas de aplicaciones y servicios para establecer una comunicación multimedia en tiempo real a través de Internet. Se comparten contenidos multimedia en vivo entre usuarios, mediante la red actual y se aprovechan características de las redes (*Peer to Peer*, P2P). También los sistemas de televisión por Internet han proliferado y los proveedores de servicios de Internet (*Intenet Service Provider*, ISP), han visto cómo (*Internet Protocol Television*, IPTV) les permite desplegar un nuevo conjunto de aplicaciones sobre la misma infraestructura compitiendo contra la televisión digital terrestre y la televisión por satélite. Así, el uso extendido de este tipo de servicios ha provocado que los usuarios demanden progresivamente más calidad, y mayor ubicuidad para acceder a estos servicios. Otros servicios como los de videoconferencia y (*Voice over Internet Protocol*, VoIP) también se han extendido sobre Internet llegando a sustituir a los servicios tradicionales.

El objetivo de este proyecto es estudiar el comportamiento del tráfico de servicios multimedia mediante el uso de un *software* de simulación de redes. Existen una gran cantidad de investigaciones para la simulación de redes y protocolos de comunicación. Entre estos simuladores de redes están NS-2, GloMoSim, OMNeT++, NCTUNs. Tras el estudio de las características de estos sistemas de simulación, se ha seleccionado NS-2 como entorno de trabajo.

Se han generado escenarios de red para el estudio y ejecución de scripts de pruebas automatizadas. Se han sacado conclusiones en base a parámetros de Calidad de Servicio (*Quality of Service,* QoS) conocidos en la transmisión multimedia como son: el ancho de banda, el retardo, la variación del retardo o *jitter* y la tasa de pérdida de paquetes. Se busca optimizar el uso de aplicaciones multimedia *streaming* mediante diferentes técnicas. Se han usado trazas de aplicaciones VoIP y (*Peer to Peer* Televisión, P2PTV) que permiten el estudio de las características de la transmisión de datos en tiempo real con el objetivo de medir las reacciones del sistema ante los cambios en las condiciones de la red. La técnica utilizada ha sido el alisado de tráfico. Para que los servicios multimedia comparados con los tradicionales funcionen bien (sean aceptables y con calidad suficiente), la red de datos debe tener soporte de QoS, especialmente cuando la red posee cuellos de botella y/o aparecen situaciones de congestión.

Por último, mediante el análisis de los resultados obtenidos, se han extraído conclusiones aplicables a situaciones reales. Con las simulaciones se pretende obtener la mayor fidelidad sobre las características y funcionamiento de las aplicaciones multimedia. Analizándose posibles mejoras y líneas futuras.

Contenido

Índice de figuras

Índice de tablas

1 Introducción

En la actualidad existen diversas alternativas de aplicaciones y servicios para establecer una comunicación multimedia en tiempo real a través de internet (*Real-time communications,* RTC). La transmisión de información multimedia (audio y vídeo) tendrá un gran impacto en los sistemas de comunicación. Esta transmisión requerirá no sólo el soporte de las redes, sino también de los protocolos de comunicación de niveles superiores. En este sentido se trabaja tanto en redes locales (*Local Area Netorwk,* LAN) como en la interconexión de dos o más redes diferentes. Podemos hablar de transmisión de información en tiempo real cuando se puede asegurar que la información llegue a su destino con unos parámetros determinados (retraso , rendimiento, fiabilidad, etc.). Cuando nos referimos a transmisión multimedia se suponen unos requerimientos temporales que necesitan a su vez del uso de esta transmisión en tiempo real.

1.1 Problemática

En los últimos años, con el crecimiento exponencial, la aparición de nuevos servicios y una transición hacia una red comercial, la situación ha cambiado radicalmente. Empiezan a aparecer nuevos servicios que funcionarán en Internet y que van más allá que la simple distribución de páginas web o envío de mensajes de correo electrónico. Nuevos servicios como la difusión de vídeo y audio (tanto desde ficheros como retransmisión en directo de eventos), Radios y Televisión vía web, vídeo bajo demanda, telefonía, videoconferencia, presentaciones multimedia, simulaciones en tiempo real, etc.

Existen diversas aplicaciones que permiten compartir contenidos multimedia en vivo entre usuarios, aprovechando las características de las redes P2P. También los sistemas de televisión por Internet han proliferado [SA+09] y los ISP´s, han visto cómo IPTV les permite desplegar un nuevo conjunto de aplicaciones sobre la misma infraestructura compitiendo contra la televisión digital terrestre y la televisión por satélite. Así, el uso extendido de este tipo de servicios ha provocado que los usuarios demanden progresivamente más calidad, y mayor ubicuidad para acceder a estos servicios. En este ámbito, IPTV está intentando cubrir dichos requisitos. Otros servicios como los de videoconferencia y VoIP también se han extendido sobre Internet hasta el punto que han llegado a sustituir servicios tradicionales.

Para que se produzca una evolución es necesario estudiar la transmisión de información en tiempo real sobre una red de conmutación de paquetes como Internet, preparada para transmitir datos. Existen una serie de requerimientos para la transmisión de datos de servicios multimedia (como por ejemplo audio y video):

- **Ancho de banda**: Es un recurso limitado y con coste significativo. Por esto muchas aplicaciones envía su información comprimida mediante estándares utilizando distintos nivele de compresión y calidad.

- **Retraso de transmisión**: Es un parámetro muy restrictivo. La (*International Telecommunication Union*, ITU) define un retraso máximo, en aplicaciones de video interactivas, de 150ms en la Recomendación G.114 [ITU96]. Existen distintos tipos de tráfico (*Asíncrono*: Retraso de transmisión sin restricciones, *Síncrono*: El retraso de transmisión está acotado para cada mensaje, *Isócrono*: El retraso de transmisión es constante para cada mensaje). Las componentes en el retraso vienen dados por la compresión y descomposición en paquetes en la fuente, transmisión en la red, almacenamiento en el destino y el retraso de sincronización, y el de la composición de los paquetes y la descompresión en el destino.
- **Fiabilidad:** Es vital para la transmisión en tiempo real se ha de controlar la latencia y fidelidad según la aplicación para la calidad del servicio.
- **Sincronización de canales:** Para la transmisión multimedia de audio y video que llegan por distintos flujos al destino. Se ha de gestionar una combinación de asignación de tiempos y almacenamiento antes de visualización.
- **Perdida de paquetes:** La pérdida de paquetes ocurre cuando uno o más paquetes de datos que viajan en una red (*Internet Protocol*, IP) fallan en alcanzar su destino. La causa de la perdida de paquetes es debida varios factores, entre los que se puede nombrar, degradación de la señal al viajar por el medio, interconexiones de la red sobresaturadas, paquetes con error, rechazados en el tránsito, fallos en el hardware de la red, o rutinas normales de enrutamiento. El servicio de transmisión multimedia demandan la mayor velocidad de transmisión y la menor pérdida de paquetes posible, para ofrecer una calidad de servicio adecuada.

Vemos como los parámetros mencionados (retardo, pérdida de paquetes, ancho de banda) afectan a ciertos servicios. Además las redes tienen limitaciones de hardware, que introducen retardos y q éste varía (*jitter*).

Por esto estos motivos vemos que los principales problemas para los flujos de datos de aplicaciones y servicios en tiempo real son: La forma en que dichas aplicaciones generan el tráfico y los efectos de la red sobre dichos flujos.

Hay que buscar una forma de adaptar este nuevo tráfico, que tiene unas características muy singulares a la infraestructura disponible, estudiando los posibles problemas y desarrollando nuevos protocolos. El tráfico se puede dividir en distintas categorías bien en función de la tolerancia a los parámetros indicados o bien por los requerimientos de los parámetros. En la Figura #1 el tráfico es clasificado en el producto cartesiano (sensibilidad al retraso) X (sensibilidad a la pérdida):

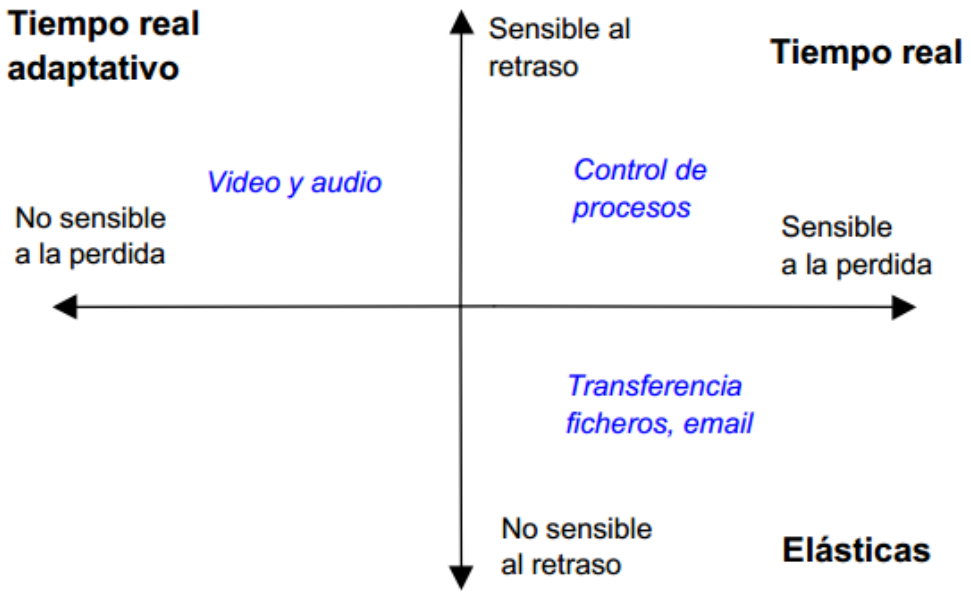

Figura #1 Tipos de tráfico frente a sensibilidad por retraso.

Como se ve el grado en que las prestaciones de una aplicación dependen de este retraso varía ampliamente y las podemos catalogar en aplicaciones de tiempo real y aplicaciones elásticas. Las aplicaciones en tiempo real donde para poder tratar correctamente los paquetes la aplicación necesita saber a priori el máximo retraso que los paquetes pueden experimentar.

El tráfico tiene que ver con el ancho de banda y retraso necesario para la transmisión. En la Figura #2 el tráfico es clasificado en el producto cartesiano (ancho de banda) X (retraso):

Figura #2 Ancho de banda frente a retraso.

En la Tabla #1 se reflejan los valores típicos en recepción para distintos tipos de tráfico en tiempo real que se transporten por la red. Estos márgenes son los necesarios para cumplir con la QoS y se extraen de este estudio [BCS94]:

Tabla #1 Requerimientos para distintos tipos de tráfico.

	Retraso máximo(s)	Máxima variación en el retraso	Ancho de banda(Mbps)	Tasa de perdida de paquetes
Voz	0.25	10	0.064	10e-1
Vídeo	0.25	10	100	10e-3
Imagen	1		2-10	10e-9

El tráfico generado por cada servicio depende de la naturaleza de la información transportada y su tamaño. Por una parte existen aplicaciones que generan ráfagas de tráfico cuando una gran cantidad de información tiene que ser enviada en un corto tiempo. Estas ráfagas pueden congestionar los dispositivos de red si la cantidad de paquetes a transmitir es significativa para la capacidad de transmisión de aquellos. Otras aplicaciones funcionan de forma que generan tráfico de una forma "suave". Tienen como objetivo proporcionar un cierto nivel QoS y una mejor experiencia de usuario. Tienden a provocar un incremento en la capacidad de procesamiento para evitar congestión de red. El tamaño de los paquetes generados por estas aplicaciones puede variar entre los diferentes servicios de Internet: mientras que algunos de ellos (por ejemplo, VoIP) generan paquetes pequeños de unas pocas decenas de *bytes*, otros (por ejemplo, videoconferencia) usan paquetes grandes. La red tiene unas determinadas características y debe gestionar sin congestionarse ambos tipos de tráfico ofreciendo una QoS en los enlaces de acceso.

En esta situación, el tamaño del *buffer* de *router* es un parámetro de diseño importante en la planificación de una red. La razón es que existe una relación entre el tamaño del *buffer* de *router* y la utilización del enlace. Una cantidad excesiva de memoria generaría una latencia significativa cuando el *buffer* estuviese lleno (este fenómeno también se conoce como *bufferbloat* o desbordamiento del *buffer*). Por otro lado, un tamaño muy pequeño del *buffer* aumentaría la perdida de paquetes en los periodos de congestión. Como consecuencia, la influencia del *buffer* debe ser considerada de forma que se mejore la utilización del enlace. Muchos estudios relacionados con problemas en el tamaño de *buffer* se han publicado en los últimos años, sino que se centran principalmente en los *router* troncales y en los flujos (*Transmission Control Protocol*,TCP) y no en el comportamiento del *buffer* [SFS14].

1.2 Objetivos

El objetivo de este proyecto es estudiar el comportamiento del tráfico de varios servicios multimedia mediante el uso de *software* de simulación de redes. Para ello se pondrán distintos escenarios de red que resulten representativos de entornos reales en redes de acceso. Para la realización de las simulaciones será necesario la instalación y puesta en marcha del *software* correspondiente. Además, el estudio y la

realización de scripts para la ejecución de pruebas automatizadas que permitan optimizar tiempo y recursos computacionales.

Existen una gran cantidad de investigaciones para la simulación de redes y protocolos de comunicación son probadas en simuladores de redes como NS-2, GloMoSim, OMNeT++, NCTUNs. Por este motivo, inicialmente se realizará un estudio de las características principales de algunos sistemas de simulación y se seleccionará el que se adapte más a las necesidades y el entorno de trabajo.

En la realización de estudios específicos de simulación se diseñan experimentos, en condiciones controladas, garantizando homogeneidad en las respuestas del sistema y por tanto mayor confiabilidad en las conclusiones. Para las pruebas, se seleccionarán casos de uso común de aplicaciones multimedia que permitan el estudio de las características de la transmisión de datos en tiempo real. El análisis se centrará en parámetros de QoS bien conocidos para la transmisión multimedia como el ancho de banda, el retardo, la variación del retardo o *jitter* y la tasa de pérdida de paquetes.

Además, buscará una forma de optimizar el uso de aplicaciones multimedia *streaming* usando diferentes técnicas. Por último se darán conclusiones aplicables a situaciones reales buscando la mayor fidelidad con las simulaciones.

Pero para que todo ello pueda realmente evolucionar es necesario estudiar la transmisión de información en tiempo real sobre una red de conmutación de paquetes como Internet, preparada para transmitir datos sin QoS.

1.3 Estructura de la memoria

En éste capítulo se ha realizado una descripción que plantea el problema a estudio. Tras esto se detallan los objetivos que llevan a una metodología para las pruebas. El entorno de trabajo, que corre bajo *software* de simulación de redes.

En el capítulo dos presenta el marco teórico o estado del arte relacionado a los temas tratado en el presente trabajo. En él se exponen las diferentes opciones de *software* para la simulación de redes. Un estudio comparativo nos llevará al uso de un programa específico. Se analizan también algunos de los servicios en tiempo real multimedia más utilizados (como por ejemplo IPTV, VoIP y *streaming*). Se usan trazas de programas comerciales para el estudio de aquellos. Se llevan a cabo pruebas sobre técnicas de optimización de tráfico. Existen distintas alternativas:

1. Multiplexación de tráfico.
2. Alisado de datos.

Por parte del grupo investigador encargado de este proyecto se realizan varios estudios relacionados con el punto 1. En este PFC nos centraremos en el análisis del punto 2.

En el capítulo tres se dan detalle de las pruebas efectuadas y técnicas de optimización de la transmisión de los datos para servicios en tiempo real. Se comienza con el análisis del escenario propuesto. Se da detalle de la situación y se realizan las modificaciones para simular distintos escenarios de comunicación multimedia. En este mismo capítulo, se realiza el estudio de la técnica de alisado de tráfico para la optimización en la QoS del sistema a estudio. Se analizan las ventajas e inconvenientes que esta técnica aporta a los parámetros medibles durante la comunicación multimedia en tiempo real y para las situaciones donde no se cumplen dichos valores de QoS.

Por último, en el capítulo 4, tras lo visto en el capítulo tres, se obtienen una serie de conclusiones validas dadas del estudio. Por otra parte se plantean las líneas futuras de trabajo en esta materia.

2 Estado del arte

En los años noventa Internet era principalmente un canal para difundir información, servicios y el contenido, en el marco del control industrial. Desde el 2000 ha aumentado el régimen binario en las redes de acceso a Internet El acceso a Internet de banda ancha se extiende y llega a un mayor número de usuarios. Junto con las mejoras técnicas de Tecnologías de la Información, Internet ha evolucionado de simple canal para difundir información. El crecimiento de los sitios web populares que sirven contenidos multimedia ha llevado a la aumento de las aplicaciones de *streaming* de vídeo. Sin embargo, el *streaming* de vídeo a través de Internet se complica por una serie de factores:

- Internet ofrece sólo la calidad de servicio *best effort* [KLS98]. No hay garantía sobre el ancho de banda, retardo y sobre la tasa de pérdida de paquetes
- Es dificultoso predecir el ancho de banda, retardo, y la tasa de pérdida de la información, ya que es desconocido y variable con el tiempo
- La heterogeneidad de las prestaciones del receptor es un problema significativo cuando secuencias de vídeo se distribuyen a través de una red de multidifusión
- Un mecanismo de control de congestión/flujo tiene que ser empleado para evitar el colapso de Internet por congestión.

El diseño actual de las transmisiones sobre IP da la misma prioridad de envío a todos los paquetes. Tiene como protocolos de transmisión TCP y (*User Datagram Protocol,* UDP). Sin embargo, debido a los diferentes tipos de servicios que circulan ahora por la red, es totalmente necesario establecer una QoS para ciertas comunicaciones o transmisiones que, ya sea por su contenido y requisitos (por ejemplo una llamada de teléfono IP), no son compatibles con el funcionamiento por defecto de la red. Es necesario adaptarse a estas condiciones haciendo uso de protocolos compatibles desarrollados para tal fin.

2.1 Protocolos para el transporte de información

En la conmutación de paquetes, los datos se dividen en segmentos más pequeños de información, llamados paquetes, antes de transmitirlos por la red. Como un paquete se puede guardar en la memoria de una estación durante un corto tiempo, a veces se llama a la conmutación de paquetes de red "para retener y enviar". En este método, un mensaje se divide en paquetes, y cada paquete puede tomar distinto camino por la red. En consecuencia, todos los paquetes no llegan necesariamente al receptor al mismo tiempo o en el mismo orden con los que se transmitieron. Como los paquetes son pequeños, el tiempo de retención suele ser bastante corto, y la transferencia de mensajes es casi en tiempo real, y no pude presentarse el bloqueo. Sin embargo, las redes de conmutación de paquetes necesitan arreglos complicados y costosos, y protocolos complicados.

El modelo utilizado por Internet es TCP/IP (Control de Transmisión/ Protocolo de Internet) fue desarrollado en 1972 por el Departamento de Defensa de Estado Unidos [CK74]. Consta de cinco capas (como se muestra en la Figura #3):

- **1) Aplicación:** Proporciona funcionalidad al usuario final Traduciendo los mensajes al *software* del usuario o *host* para su presentación en pantalla.
- **2) Protocolo de Control de Transmisión, TCP:** Realiza el transporte, dividiendo los datos de aplicaciones del usuario final en paquetes TCP llamados datagramas. Cada paquete consta de un encabezado con la dirección de la computadora *host* remitente, información para reensamblar los datos e información para asegurarse de que los paquetes no llegaron alterados. También puede usar UDP. UDP no otorga garantías para la entrega de sus mensajes y el origen UDP no retiene estados de los mensajes UDP que han sido enviados a la red. UDP sólo añade multiplexado de aplicación y suma de verificación de la cabecera y la carga útil. Cualquier tipo de garantías para la transmisión de la información deben ser implementadas en capas superiores.
- **3) Protocolo de Internet, IP:** Recibe datagramas TCP y, a su vez, los divide en paquetes. Un paquete IP contiene un encabezado con la información de la dirección y transporta información del TCP y datos. El IP enruta los datagramas individuales del emisor al receptor. Los paquetes IP no son muy confiables, pero el nivel TCP los reenvía hasta que dichos paquetes lleguen al destino
- **4) Interfaz de red:** Maneja aspectos de dirección, por lo común en el sistema operativo, así como la interfaz entre la computadora emisora y la red.
- **5) Red física:** Define las características básicas de transmisión eléctrica para enviar la señal real a través de las redes de comunicaciones.

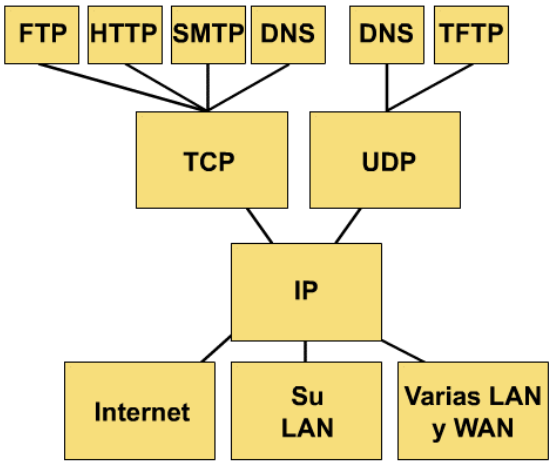

Figura #3 Protocolo TCP/IP.

2.1.1 TCP

TCP es un protocolo orientado a la conexión, descrito en la (*Request For Comments,* RFC) 793 [RFC793]. El TCP proporciona fiabilidad a la aplicación; es decir, garantiza la entrega de toda la información en el mismo orden en que ha sido transmitida por la aplicación origen. para conseguir esta fiabilidad, el TCP proporciona un servicio orientado a la conexión con un control de de flujo y errores.

Aunque podría ser posible utilizar el TCP para transportar los datos de una transmisión en tiempo real hay una serie de motivos que no aconsejan su utilización:

- TCP es un protocolo que realiza comprobación de errores y retransmisión de paquetes. El sistema de retransmisión de TCP se basa en esperar durante un cierto tiempo (el *time-out*) la llegada de una confirmación de que el destinatario ha recibido correctamente los datos. Si esta confirmación no llega dentro del tiempo se retransmite el paquete. El problema es que cuando el emisor se da cuenta de que debe retransmitir suele ser demasiado tarde, la información ya ha perdido su valor. Para los requerimientos de tiempo real un sistema de retransmisiones resulta inútil dadas las fuertes restricciones temporales.
- TCP utiliza un control de congestión que decrementa el tamaño de la ventana de transmisión cuando hay pérdidas de paquetes para evitar sobrecargar a la red. Sin embrago en transmisiones en tiempo real hay una tasa de transmisión (la que genera la fuente) que debe llegar siempre al destino y no se puede recortar.
- TCP no dispone de un sistema de distribución *multicast* lo que es una seria restricción en los servicios multipunto a gran escala: enviar un paquete por cada destinatario representaría una utilización ineficaz del ancho de banda.

2.1.2 UDP

UDP es un protocolo simple, no orientado a conexión, descrito en la RFC 768 [RFC768]. Tiene las siguientes características: Las porciones de comunicación en UDP se llaman datagramas. No ofrece ningún mecanismo que permita garantizar al remitente que todos los datos hayan llegado al destino (no hay forma de saber el orden del datagrama, ni se ha llegado por duplicado). Cada datagrama se envía de forma independiente del resto. No hay mecanismos que eviten congestiones. No dispone de ningún mecanismo de fragmentación incluido en el protocolo, sino que es la propia aplicación la que debe dividir su información en fragmentos de tamaño adecuado para que sean encapsulados mediante datagramas UDP. En el encabezado del datagrama se envía una suma de comprobación (*checksum*) que permite averiguar en el destino si ha habido errores en la transmisión desde el origen, no los corrige.

Ofrece los siguientes servicios:

- Multiplexación de envíos entre procesos por encima de un mismo enlace de red, identificando en cada máquina cada proceso con un número de puerto.
- Detección de errores en la transmisión entre extremos para cada datagrama.

Para la transmisión de datos en tiempo real podríamos pensar en UDP ya que este protocolo no utiliza ningún sistema de control de errores y permite utilizar IP *multicast* pero tampoco resulta adecuado ya que es demasiado sencillo y no contiene toda la información necesaria: momento de generación de los datos (necesario para reordenar muestras y recuperar el sincronismo con otros flujos de datos), información sobre la codificación utilizada, etc.

Como hemos visto, ni TCP, ni UDP resultan ser útiles para el transporte de datos con requerimientos de tiempo real. Si la infraestructura de transporte disponible es insuficiente hemos de desarrollar nuevas herramientas o mejorar las ya existentes

2.2 Servicios y aplicaciones multimedia

Existe una revolución P2P, un cambio radical del concepto de Internet. De esta manera es posible compartir información y servicios entre los usuarios sin existir terceras partes implicadas, con un simple intercambio de datos. Las aplicaciones P2P más usadas hoy en día son los que se utilizan para el intercambio de ficheros. Sin embargo se han propuesto otras posibles soluciones P2P para diferentes propósitos. En los últimos años dos fenómenos principales han tomado cada vez más relevancia: La telefonía VoIP y *streaming* de medios de comunicación.

La posibilidad de transportar una llamada de voz sobre la red IP ha cambiado la visión tradicional de las comunicaciones telefónicas. Anteriormente, las llamadas de voz requieren una línea telefónica tradicional para ser llevada a cabo; por el contrario, ahora hay la posibilidad de tener una comunicación de voz con una simple conexión a Internet. Se han abierto nuevas líneas de investigación y negocios. Esta tecnología no es muy recientemente, pero ha conocido una explosión tras el reciente desarrollo de varios sistemas de VoIP; entre ellas la primera fue de Skype. Otro fenómeno, aún más reciente, es la transmisión de medios. Más y más emisoras de radio y televisión ofrecen en tiempo real el acceso en línea a sus programas. Esto significa que ahora es posible acceder a través de Internet para servicios que antes eran posibles sólo con dispositivos de radio o televisión. En el último año, otra tendencia interesante salió: difundir personal y vídeos originales en todo el mundo. Esto fue posible gracias a empresas como YouTube o DailyMotion que permiten el intercambio de vídeos de todo el mundo con todo el mundo. Con sus 70 millones de vídeos vistos cada día en su sitio, YouTube da fe de la relevancia este fenómeno está adquiriendo en la escena de Internet.

UDP es generalmente el protocolo usado en la transmisión de vídeo y voz a través de una red. Esto es porque no hay tiempo para enviar de nuevo paquetes

perdidos cuando se está escuchando a alguien o viendo un vídeo en tiempo real [SCH95].

Dado que tanto TCP como UDP circulan por la misma red, en muchos casos ocurre que el aumento del tráfico UDP daña el correcto funcionamiento de las aplicaciones TCP. Por defecto, TCP pasa a un segundo lugar para dejar a los datos en tiempo real usar la mayor parte del ancho de banda. El problema es que ambos son importantes para la mayor parte de las aplicaciones, por lo que encontrar el equilibrio entre ambos es crucial.

En 1996 se publica en el RFC 1889 [RFC1889] el estándar del protocolo (*Real-time Transport Protocol*, RTP) [JFC+96]. El protocolo de transporte en tiempo real surgió con la idea de crear un protocolo específico para la gran demanda de recursos en tiempo real por parte de los usuarios. Algunos de estos recursos son la música, videoconferencia, video, telefonía en Internet y más aplicaciones multimedia. Está formado conjuntamente con el protocolo (*RTP Control Protocol*, RTCP), es decir, Protocolo de Control RTP, cuya función principal es proporcionar mecanismos de realimentación para informar sobre la calidad en la distribución de los datos.

La (*Internet Engineering Task Force*, lETF) ha desarrollado un nuevo protocolo especialmente pensado para el transporte de este tipo de información, el RTP. Este nuevo protocolo incluye una serie de funciones que facilitan esta tarea: identificación del tipo de información (tipo de codificación), números de secuencia, sincronismo (*timestamp*) y monitorización de la calidad de servicio. Sin embargo, RTP no posee ninguna función que garantice la entrega de los paquetes dentro del periodo adecuado ni ningún otro tipo de garantía de la cualidad de servicio. Conviene resaltar también que aunque sea un protocolo de "transporte", porque regula el envío de datos de un extremo a otro, este protocolo funciona realmente en el nivel de aplicación.

RTP prácticamente no hace ninguna suposición sobre el nivel inferior sobre el que funciona excepto que el protocolo en cuestión defina el tamaño de los paquetes ("*frarning*") ya que el RTP no incluye en su cabecera ningún campo de longitud de los datos. No se asume la existencia de ningún tipo de control de errores, ni se asume la existencia de una conexión, ni ningún mecanismo de reordenación de paquetes. En general, en una arquitectura TCP/IP el RTP se utiliza sobre UDP, aunque el protocolo es lo suficientemente general como para utilizarse en otros tipos de red, como (*Asynchronous Transfer Mode*, ATM) por ejemplo. Las aplicaciones de VoIP comerciales como Skype hacen uso de TCP para loguearse en supernodos como de UDP, para transmitir paquetes de información sobre RTP/UDP.

El protocolo RTP consta de dos partes muy relacionadas: la parte que transportan los datos (a la que en general se refiere como RTP) y la parte de control, referida como RTCP que realiza las funciones de monitorización de la calidad de servicio y control de los participantes de una determinada sesión.

2.2.1 Aplicaciones de VoIP

VoIP, es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP. Esto significa que se envía la señal de voz en forma digital, en paquetes de datos, en lugar de enviarla en forma analógica a través de circuitos utilizables sólo por telefonía convencional como las redes (*Public Switched Telephone Network*, PSTN) Red Telefónica Pública Conmutada).

Los Protocolos que se usan para enviar las señales de voz sobre la red IP se conocen como protocolos de Voz sobre IP o protocolos IP. Estos pueden verse como aplicaciones comerciales de la "Red experimental de Protocolo de Voz" (1973), inventada por (*Advanced Research Projects Agency Network*, ARPANET) [LVK+00]. La telefonía tradicional consiste en transportar la señal analógica sobre cable de cobre, pero la tecnología de VoIP convierte la voz analógica en paquetes de datos digitales que soportan la comunicación sobre el protocolo IP y que pueden emplear protocolos para aplicaciones en tiempo real como el RTP [HSRV96].

Así, el servicio de VoIP sustituye el tradicional teléfono residencial por un teléfono de gran ancho de banda que utiliza Internet para hacer y recibir llamadas. Si se realiza una llamada a un número de teléfono tradicional, la señal es convertida en el otro extremo. Dependiendo del tipo de servicio VoIP, se puede hacer una llamada VoIP desde el ordenador, desde un teléfono especial VoIP o desde un teléfono tradicional con o sin adaptador [ROH+06].

A continuación se enumeran las distintas funciones que caracterizan VoIP y por las que resulta tan ventajoso su uso. En primer lugar, permite realizar más de una llamada telefónica simultáneamente. Además, las llamadas entrantes pueden ser automáticamente dirigidas al teléfono VoIP independientemente de dónde se esté conectado a la red. Ejemplos pueden ser una llamada a tres, llamadas al extranjero, re-llamada automática o identificador de llamada.

VoIP puede ser más segura ya que permite el uso de protocolos como el (*Secure Real-time Transport Protocol,* SRTP) [MDNE04]. Sólo es necesario cifrar y autentificar la trama de datos. Asimismo, VoIP es independiente del lugar, sólo es necesaria una conexión a Internet para conseguir una conexión a un proveedor VoIP. Los teléfonos VoIP pueden integrarse con otros servicios disponibles en Internet, incluyendo conferencia de audio o de video, mensajes o intercambio de archivos de datos en paralelo con la conversación.

Sin embargo, existe un importante inconveniente que ha hecho que la expansión de la VoIP no sea tan rápida como se esperaba: la dificultad en ofrecer QoS. En la transmisión de voz es necesario que todos los paquetes lleguen ordenados, que no haya pérdidas y garantizar una mínima tasa de transmisión. En otros servicios como el correo, ofrecer QoS no es crítico, ya que si un paquete no ha llegado al destino se solicita su retransmisión; pero esto no es posible en la VoIP, ya que se trata de un servicio en tiempo real. La solución radica en diferenciar los

paquetes de voz de los paquetes de datos, priorizar la transmisión de los paquetes de voz y evitar que la transmisión de los paquetes no supere los 150 milisegundos, tal y como se especifica en la recomendación ITU-TG 114 [DOHE04].

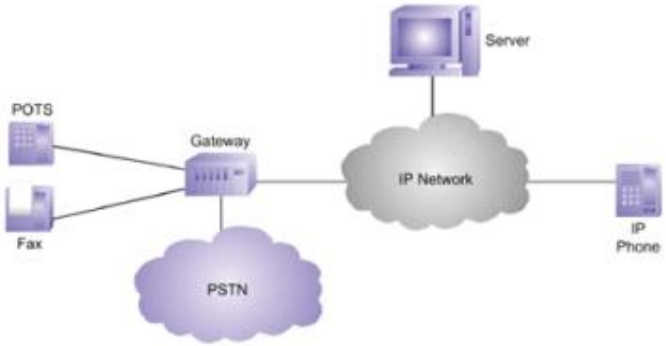

Figura #4 Estructura de una red VoIP.

La QoS se está logrando en base a los siguientes criterios:

- La supresión de silencios y (*Voice Activity Detection*, VAD), otorga más eficiencia a la hora de realizar una transmisión de voz, ya que se aprovecha mejor el ancho de banda al transmitir menos información.
- Compresión de cabeceras aplicando los estándares RTP/RTCP.
- Cancelador de eco
- Priorización de los paquetes que requieran menor latencia.
- La implantación de IPv6 que proporciona mayor espacio de direccionamiento y la posibilidad de *tunneling*.

Como ejemplo de una aplicación VoIP comercial se puede mencionar Skype. La red Skype es una red *Overlay* que es un tipo de red P2P que conecta iguales entre sí [BS04]. Cuenta con diferentes servicios: comunicación de voz y video, chat, transmisión de ficheros y compartición de pantalla de escritorio. Pudiendo realizar llamadas entre diferentes usuarios Skype (*End-to-End*) o entre un usuario y un terminal de telefonía convencional (*End-to-Out*). La aplicación utiliza un protocolo privado, con técnicas de cifrado y ofuscación de datos. Puede utilizar TCP o UDP en la capa de transportes, siendo habitual el uso de UDP. TCP se usa cuando existen restricciones de firewall sobre UDP. Skype a diferencia de otras aplicaciones de VoIP se basa en usa arquitectura de red P2P, en vez de usar el modelo cliente servidor.

La red P2P de Skype está compuesta por [DCG12]:

- **Nodo normal u ordinario:** nodo donde se ejecuta el cliente Skype.
- **Supernodo:** nodo con una dirección IP pública que tiene suficiente (*Central Processing Unit,* CPU), memoria, y ancho de banda de red .

- **Servidores dedicados:** existen de tres tipos que se detallan a continuación.

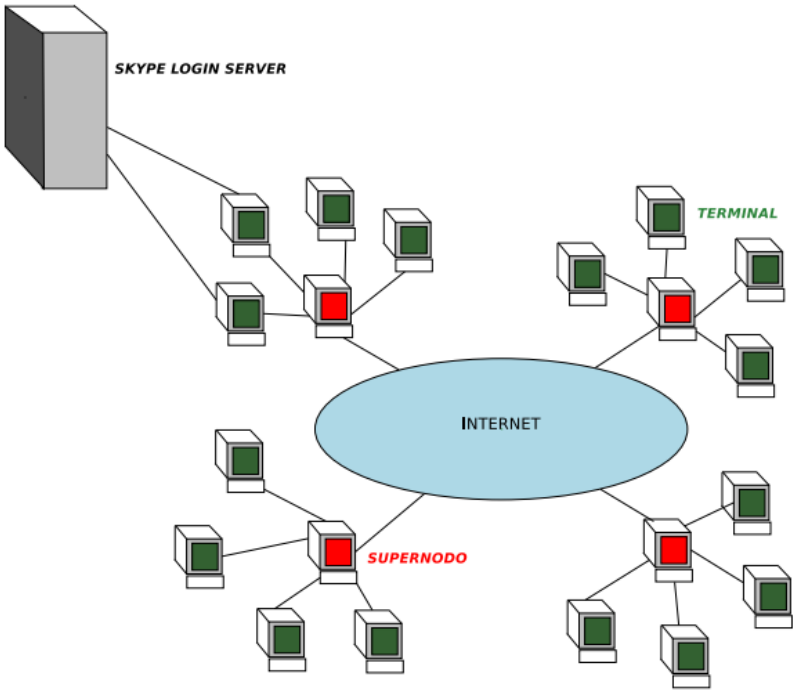

Figura #5 Arquitectura red Skype.

Dentro de los servidores dedicados, tenemos tres tipos:

- ***Login servers:*** para recoger información del usuario y responsables de la autenticación
- ***Update servers:*** para revisar si hay disponible una nueva versión de la aplicación e indicar la dirección IP del servidor Skype
- ***Buddy-list servers:*** utilizado para guardar la información referida a los contactos del usuario**.**

Una vez que el cliente accede a Skype se establece una conexión con un supernodo que será el responsable de él. Para ello, de inicio, investiga mediante mensajes intercambiados con otros supernodos, cual está en funcionamiento y cual le acepta como cliente. Al principio también contacta con el *login-server* para la identificación. Durante el tiempo que el usuario está *online* hay un intercambio de mensajes periódico entre el *host* del usuario y el supernodo seleccionado al inicio. La Figura #5 muestra la arquitectura de la red Skype. Existen una serie de estudios en los cuales se caracterizan los mensajes que intercambia entre clientes y entre clientes y los supernodos [TBP+11] y [ACG+09].

2.2.2 Aplicaciones de IPTV

IPTV, se ha convertido en la denominación más común para los sistemas de distribución por subscripción de señales de televisión o vídeo usando conexiones de

banda ancha sobre el protocolo IP. A menudo se suministra junto con el servicio de conexión a Internet, proporcionado por un operador de banda ancha sobre la misma infraestructura pero con un ancho de banda reservado [XZHG07]. El proveedor no transmitirá sus contenidos hasta que el cliente se conecte y los solicite. Esto permite el desarrollo del pago por visión o pago por evento o el video bajo demanda.

IPTV no es un protocolo en sí mismo. Ha sido desarrollado basándose en el *video-streaming*. A esta tecnología evolucionará en un futuro próximo la televisión actual, aunque para ello son necesarias redes mucho más rápidas que las actuales, para garantizar la QoS.

Para que la IPTV pueda desarrollarse de una manera completa es necesario aumentar la velocidad de las conexiones actuales. Podemos diferenciar dos tipos de canal: de definición estándar SDTV o de alta definición HDTV. Para un canal del primer tipo sería necesario tener una conexión de 1.5 Mbps y para un canal del segundo tipo 8 Mbps. Si tenemos varios canales distintos en forma simultánea (por tener varios receptores de televisión) necesitaremos más ancho de banda. A este ancho de banda hay que sumar el necesario para la conexión a internet. Estamos hablando de 4.5 Mbps para tres canales de SDTV u 11 Mbps para un canal HDTV y dos SDTV. Estos cálculos son usando MPEG-4 para la compresión/codificación del vídeo [WWCH08].

La IPTV necesita unos valores técnicos para poder prestar su contenido sin inconvenientes, los valores son los siguientes:

- **Ancho de banda:** dependiendo del número de decodificadores, la velocidad del internet o telefonía IP (VoIP, deberá ser mayor en cada caso, los más comunes son: 4 Mbps, 7 Mbps, 8 Mbps, 10 Mbps, 12 Mbps, 14 Mbps, 16 Mbps y 18 Mbps. El hecho de que el ancho de banda sea más alto, provoca que la línea (*Asymmetric Digital Subscriber Line,* ADSL) sea más sensible a caídas. Es decir, una línea con un perfil de 4 Mbps, si por ejemplo queda con valores de señal-ruido de 13dB y atenuación de 40, no soporta un perfil de 10 Mbps, ya que provoca mayor atenuación y menos señal-ruido.
- **Señal-ruido:** mayor de 13dB para garantizar la estabilidad del servicio (cuanto más alto el valor, de más calidad será el servicio)
- **Atenuación:** menor de 40dB, ya que si es demasiado alta, el servicio puede tener caídas constantes.

El contenido se puede obtener a través de internet de algún proveedor de contenidos o de un distribuidor de señales de televisión. Se utilizan unos dispositivos llamados codificadores para digitalizar y comprimir el video analógico obtenido. Este dispositivo llamado códec, habilita la compresión de video digital habitualmente sin pérdidas. La elección del códec tiene mucha importancia, porque determina la calidad

del video final, la tasa de bits que se enviarán, la robustez ante las pérdidas de datos y errores, el retraso por transmisión, entre otros.

Los formatos de video empleados por IPTV más usualmente son: H.261, MPEG-1, MPEG-2, H.263, MPEG-4, WMV.

2.2.3 Aplicaciones P2PTV

P2P ha surgido recientemente como un nuevo paradigma para construir la red distribuida de aplicaciones. Se ha pasado al desarrollo de gran cantidad de *software* P2P de *streaming* de video *on-line* y bajo demanda VoD. Debido al reducido coste en servidores. Con el desarrollo en la tecnología de fibra óptica (*Fiber-To-The-Home*, FTTH) se prevé un aumento considerable en los contenidos multimedia por internet.

Este diseño P2P pretende es que los usuarios actúen como clientes y servidores, es decir, como *peers*. En una red P2P, un *peer* no sólo descarga los datos de la red, sino que también sube los datos descargados a otros usuarios de la red. El ancho de banda de subida de los usuarios finales es utilizado eficientemente para reducir la carga de ancho de banda en los servidores. A mayor número de usuarios mayor calidad en la transmisión. Aplicaciones de intercambio de archivos P2P han sido ampliamente empleadas para difundir rápidamente archivos de datos a través Internet. El término P2PTV hace referencia a aplicaciones P2P diseñadas para redistribuir flujos de vídeo en una red P2P, típicamente cadenas de televisión, por todo el mundo. Más recientemente, la tecnología P2P ha sido empleado para proporcionar servicios multimedia en *streaming*. En la Figura #6 se aprecia la arquitectura de este servicio. Varios sistemas de *streaming* P2P han aparecido para dar un servicio de video en directo o bajo demanda por internet, como ejemplo a tratar más adelante se encuentra el *software* Sopcast.

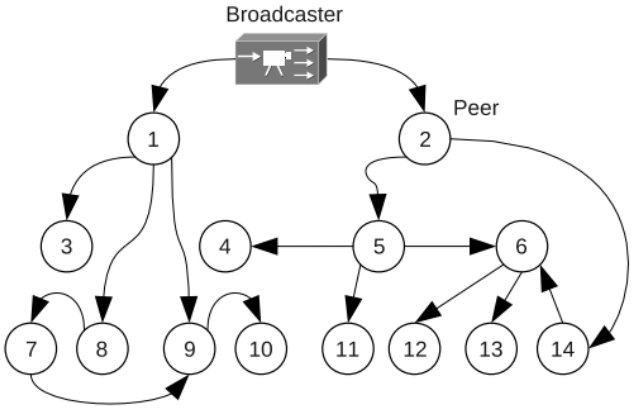

Figura #6 Arquitectura P2PTV.

El funcionamiento de estas aplicaciones es el siguiente [WIK14]:

- El difusor codifica la señal de video y haciendo uso de la parte servidor de la aplicación, comienza el envío de la misma. Manual o

automáticamente se genera un canal al que pueden conectarse los clientes. Este canal puede ser público o privado, dependiendo de las características de cada aplicación. El listado de canales disponibles se muestra, dependiendo de la aplicación, bien desde dentro de la propia aplicación o bien en la página web de la misma.

- Los clientes se conectan al canal, obtienen una lista de los ordenadores que se encuentran conectados al canal y comienzan a intercambiar partes.
- El vídeo a transmitir se divide en trozos de *n* segundos. Cada uno de estos trozos se subdivide en partes, que son las que se intercambian.
- La parte cliente de la aplicación intenta componer trozos completos y genera un flujo de vídeo en la máquina local. El objetivo es intentar mantener un flujo lo más constante posible, pero esto dependerá de las partes que haya conseguido intercambiar, teniendo en cuenta que una vez que se termina una parte del vídeo no tiene sentido intentar recuperar las partes que lo componen y no han podido ser descargadas a tiempo. Por ello, se descargan trozos por adelantado intentando completar el flujo pendiente a tiempo para que no se congele la secuencia de vídeo generada.
- Se ejecuta un reproductor de vídeo capaz de interpretar los flujos de vídeos generados y se le indica como dirección la (*Uniform Resource Locator,* URL) (local) donde se ubica el vídeo. Normalmente el reproductor reside en el propio ordenador donde reside el flujo, pero nada impide hacerlo desde otro introduciendo la correspondiente URL.

Como ejemplo de una aplicación P2PTV comercial se puede mencionar Sopcast. SopCast es una aplicación P2PTV libre, que nació como un proyecto estudiantil en la Universidad Fundan en China. Las velocidades de bits de programas de TV en SopCast suelen oscilar entre 250 Kbps a 400 Kbps con un par de canales de hasta 800 kbps. Los canales están codificados en dos formatos: Windows Media Video (WMV) o Real Video (RMVB). Cada canal transmite audio-video en vivo o películas en bucle de acuerdo a horario. El espectador sintoniza un canal de su elección o puede transmitir su propio canal. Posee un estándar de canal URL *sop:// URL a reproducir*.

En la Figura #7 vemos el interfaz de la aplicación.

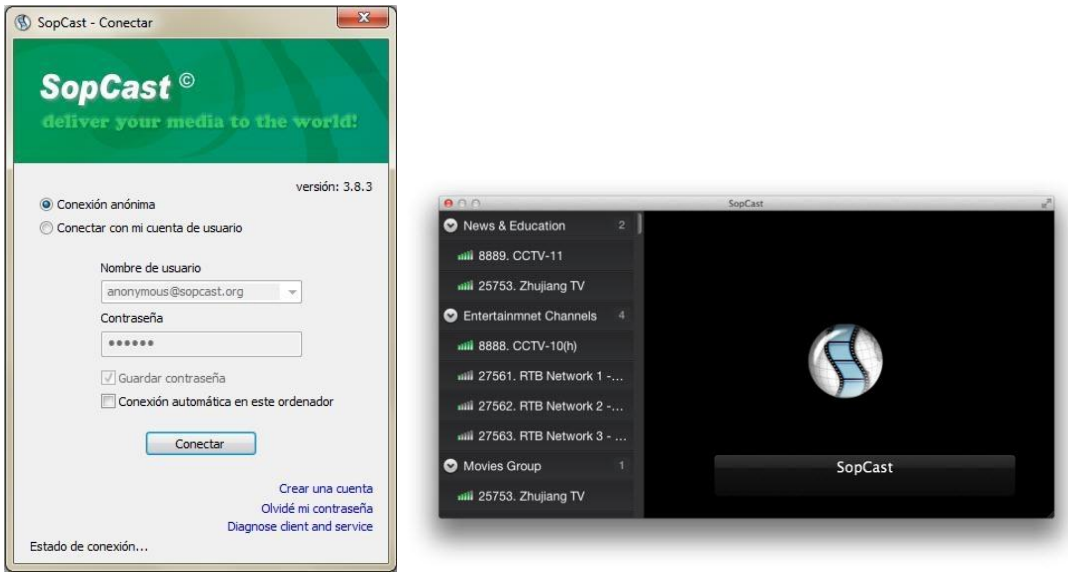

Figura #7 Interfaz de inicio Sopcast.

SopCast basa su tráfico en el protocolo de transporte UDP. Se observan dos picos en la distribución de tamaños de paquetes en la Figura #8: uno cae en la región por debajo de 100 *bytes* y otro en 1320 *bytes*. Los paquetes pequeños son asentimientos de la capa de aplicación de los paquetes de datos enviados y recibidos. Los paquetes más grandes, con tamaño aproximadamente igual a la unidad de transmisión máxima (*Maximum Transfer Unit,* MTU) para los paquetes IP a través de redes Ethernet, son los paquetes de vídeo. [BK10]

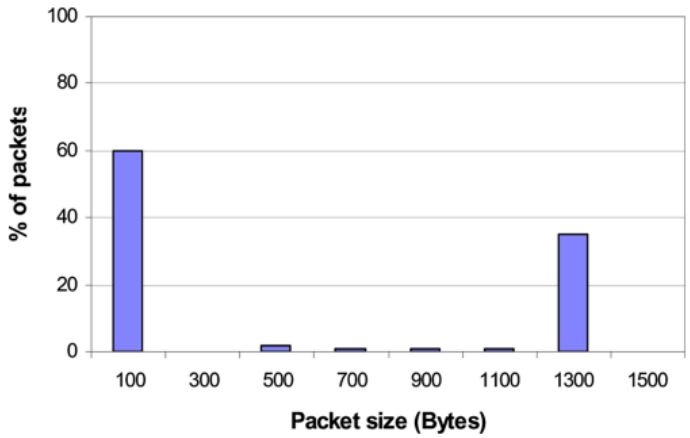

Figura #8 Distribución del tamaño de los paquetes en Sopcast.

Al iniciar un canal en SopCast, se requiere un poco de tiempo para la búsqueda de *peers*. Posteriormente, se intenta descargar los datos de los *peers* activos. Existe el retardo que va desde la selección un canal hasta se inicia la reproducción del video. Comprende una media de 20 a 30 segundos. Es el tiempo en el que recupera la lista de *peers* y recibe los primeros paquetes de vídeo. El *buffer* de retardo de reproducción ronda desde los 10 a los 15 segundos según condiciones. El tiempo total oscila entre 30 y 45 segundos.

En el estudio [FLK+08] , se examina el tráfico generado por cada nodo y el modo de funcionamiento equiparable a los programas P2P de intercambio de ficheros. En este caso los usuarios compartes el video a reproducir.

SopCast se enfrenta a una sobrecarga alta, alrededor del 60% de los paquetes son señalización en comparación con 40% de paquetes de datos de vídeo reales. Esto era predecible, ya que el protocolo funciona con UDP como protocolo de transporte, que no garantiza la fiabilidad como que lo hace TCP.

Se utilizan técnicas de *buffering.* Los paquetes de información recibidos se almacenan en el *buffer* SopCast. El *buffer* es responsable de descargar paquetes de vídeo de la red y de lanzar el *streaming* de vídeo descargado en un reproductor de medios local. El procedimiento de transmisión continua en SopCast atraviesa dos *buffe*: el *buffer* de SopCast y el *buffer* del reproductor de medios, como se muestra en la Figura #9 .

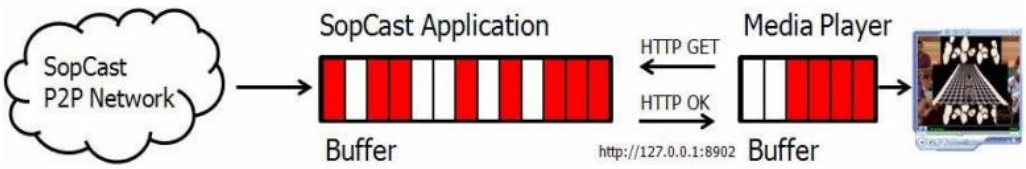

Figura #9 *Buffer* de Sopcast.

Cuando la longitud del archivo de *streaming* del *buffer* SopCast excede un umbral predefinido, lanza un reproductor de medios, que descarga el contenido de vídeo desde el servidor web local que escucha en el puerto 8902.

Existen una serie de temas a tratar en mayor profundidad de detalle como: tasas de subida y bajada en los usuarios de Sopcast, bloqueos durante la emisión, sincronización de audio y video, tiempo de *zapping* que se tratan en el estudio al que hacemos referencia [FLK+08]. A través de medidas pasivas, que caracteriza el comportamiento de SopCast y evaluaciones de usuarios (*Quality of Experience*, QoE).

En base a los resultados de la medición de la calidad de la experiencia, las principales conclusiones sobre Sopcast son las siguientes:

- Puede proporcionar vídeo de buena calidad a los *peers* que difunden a un PC
- Audio y video pueden estar fuera de sincronización, y pueden incluso superar los requisitos de la ITU;
- Sufre retrasos por pares, es decir, *peers* viendo el mismo canal pueden no estar sincronizados;

- El tiempo de *zapping* es extremadamente alta, en promedio 50 segundos.

2.3 Calidad de servicio

La definición del término calidad de servicio no es tarea fácil. La Unión Internacional de Telecomunicaciones la define de la siguiente manera [UIT80]:

> *"La calidad de servicio (QoS) es el efecto global de la calidad de funcionamiento de un servicio que determina el grado de satisfacción de un usuario de un servicio"*

Otros autores, orientando la definición anterior hacia una aplicación telemática exponen que la calidad de servicio es [VKB+95]:

> *"El conjunto de las características tanto cuantitativas como cualitativas de un sistema distribuido necesarias para alcanzar las funcionalidades requeridas por una aplicación."*

De esta última definición se puede concluir que la QoS se puede apreciar desde un punto de vista objetivo o desde un punto de vista subjetivo. En este proyecto se busca la caracterización de la QoS de distintos servicios actuando sobre un mismo enlace desde el punto de vista objetivo y orientada hacia aplicaciones multimedia en tiempo real.

El objetivo de la calidad de servicio es proporcionar el servicio de entrega preferencial para las aplicaciones que lo necesiten, garantizando suficiente ancho de banda, control de latencia y *jitter*, y la reducción de la pérdida de datos.

QoS aporta los siguientes beneficios:

- Proporciona a los administradores el control sobre los recursos de red y les permite gestionar la red de una empresa, en lugar de una técnica, la perspectiva.
- Asegura que las aplicaciones sensibles al tiempo y de misión crítica tienen los recursos que necesitan, al tiempo que permite el acceso de otras aplicaciones para la red.
- Mejora la experiencia del usuario.

- Reduce los costos mediante el uso de los recursos existentes de manera eficiente, lo que retrasa o reduce la necesidad de ampliación o mejoras.

El Grupo de Trabajo de Ingeniería de Internet (IETF) define dos principales modelos de QoS en las redes basadas en IP: Servicios integrados (Intserv) y servicios diferenciados (Diffserv).

2.3.1 Parámetros de calidad

El primer parámetro de que se analiza es el retardo. En función de la tolerancia a que los paquetes lleguen con un retardo mayor que el aceptable se puede clasificar la comunicación en tiempo real como "*hard real-time*" o "*soft real-time*". En el primer caso el servicio no es capaz de tolerar perdidas (por ejemplo un sistema de control remoto, cuando ha de reaccionar frente a una emergencia), mientras que en el segundo es aceptable una cierta tasa de perdida (por ejemplo un sistema de transmisión de audio).

El retardo acumulado desde un extremo a otro de la comunicación se llama latencia (como se aprecia en la Figura #10). A la latencia contribuyen el origen, la red y el destino. El origen contribuye con el tiempo que transcurre desde que muestrea la señal hasta que envía las muestras y el destino con el tiempo que tarda antes de analizarlo. Estos retardos en general no son significativos frente a los que impone la red. La red contribuye de diferentes formas al retardo total: Retardo de propagación, de transmisión, de "*store-and-forward*" y de proceso.

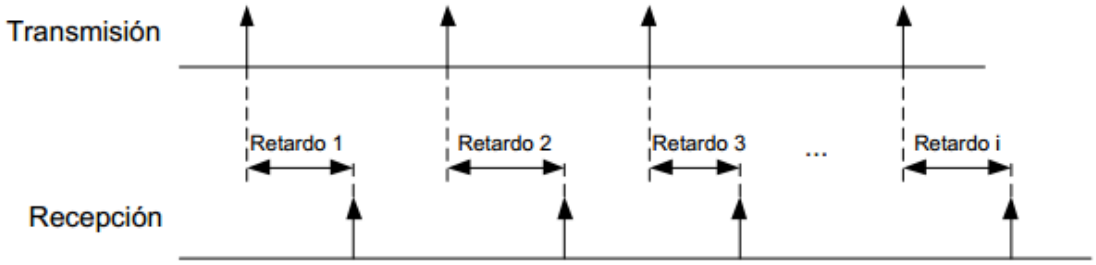

Figura #10 Retardo en la transmisión de un flujo.

Se suelen distinguir dos formas fundamentales de medirlo: el retardo en un sentido (*One Way Delay*, OWD), y el retardo de ida y vuelta (*Round Trip Time*, RTT) [CR00].

El segundo parámetro a tener en cuenta al analizar los servicios que requieren tiempo real es el *jitter*. Se define como la máxima variación del retardo extremo a extremo que sufren los paquetes de una misma conexión. Hay gran cantidad de factores que contribuyen al retardo extremo a extremo. En general son retardos introducidos por la red. No son deterministas y dependen del estado actual de esta; lo

que hace que los distintos paquetes lleguen al destino con diferentes retardos o lo que es lo mismo, que el tiempo entre la llegada de paquetes consecutivos será una variable aleatoria. Esto causará una "pérdida de sincronismo" en el receptor que debe ser corregida para que sea posible reconstruir el flujo de datos original y así reducir la latencia.

Este efecto es especialmente malo, por ejemplo, en transmisiones de audio ya que produce una serie de chasquidos que resultan muy molestos en la comunicación. La solución que permite corregirlo es sencilla y se basa en la utilización de un *buffer* en recepción con política (*Fisrt In First Out,* FIFO) que se suele denominar "*playout buffer*" como ejemplo en la Figura #11. En lugar de procesar lo paquetes a medida en que llegan, el receptor los almacena en el *buffer* y cuando este se ha llenado los empieza a procesar. Como en media la tasa de paquetes que entra es igual a la que sale del *buffer* el sistema es estable.

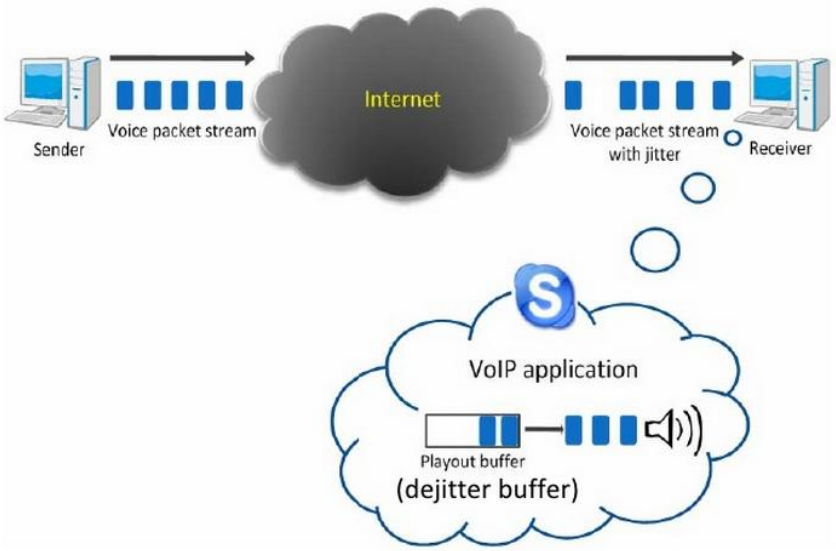

Figura #11 *Playout buffer* para corregir *jitter*.

El tercer parámetro a focalizar es las pérdidas de paquetes, son una de las principales causas del descenso de la calidad en redes IP. En [BSU+98] se estudian las pérdidas de paquetes, enviando tráfico real para pruebas. Muchas de las pérdidas se producían a ráfagas. Del estudio se obtienen ráfagas cortas, de hasta un segundo, y su causa principal era el descarte en los *buffer* de los *router*. También se observan ráfagas de mayor duración, de hasta diez segundos, se debían al reinicio o mantenimiento de los *router*.

2.4 Herramientas de simulación

Actualmente existen investigaciones en redes y protocolos de comunicación. Se simulan son *software* específico como NS-2, GloMoSim, OMNeT++ entre otros, con el fin de obtener una aproximación inicial al comportamiento de las redes y así tomar una decisión sobre su implementación. Se analiza el simulador NS-2, sobre el cual se realiza la simulación y se dan características de otros *software*.

La gran diferencia entre la mayoría de simuladores que se pueden encontrar para el análisis de sistemas radica en el método que se utilice para la simulación:

- 1) Simuladores de eventos discretos.
- 2) Simuladores de tiempo continuo.

Los primeros funcionan modelando los sistemas de manera cronológica como una secuencia de eventos, donde cada suceso tiene un lugar y un instante en el tiempo, que además genera una marca de cambio en el estado del sistema. Útil para el análisis de sistemas secuenciales o que usen colas, muy comunes en el ambiente de las comunicaciones Los segundos funcionan utilizando modelos matemáticos y ecuaciones diferenciales que describen la evolución del sistema de manera continua. Se usan cuando el proceso que se desea analizar cambia de manera muy sutil y continua. Adicionalmente, los simuladores continuos también pueden usarse para modelar sistemas formados por valores discretos si el número de datos es lo suficientemente grande como para interpretarlo como un flujo continuo.

2.4.1 NS-2

El *Network Simulator* 2 es un *software* simulador de eventos diseñado para la ayuda en la simulación de redes telemáticas y disponible en diversas plataformas. Es el simulador de redes más extendido, tanto en grupos de investigación como para su utilización como herramienta educativa debido a su condición de *software* libre. Estas características del *software* le convierten en el adecuado para las pruebas efectuadas en el PFC.

Se empezó a desarrollar en 1989 como una variante del ya existente simulador *REAL Network Simulator*. En 1995, bajo la supervisión del proyecto (*Virtual InterNetwork Testbed,* VINT), acabó en manos de unos investigadores y desarrolladores de la universidad de California (Estados Unidos) en Berkeley [NS-2].

NS–2 puede simular una gran variedad de protocolos de cualquier capa del modelo OSI, sigue el esquema de la Figura #12. Permite entre otras cosas trabajar tanto en redes cableadas como redes inalámbricas (simulando escenarios *adhoc*, movimientos de nodos en escenario, etc.), redes vía satélite con una enorme cantidad de protocolos a distintos niveles, la capa de transporte (TCP, UDP, RTP, etc.), la capa aplicación (FTP, (*Constat Bit Rate*, CBR), (*HyperText Transfer Protocol,* HTTP), TELNET etc.) o la capa de enlace de datos (IEEE 802.3, IEEE 802.11, etc.).

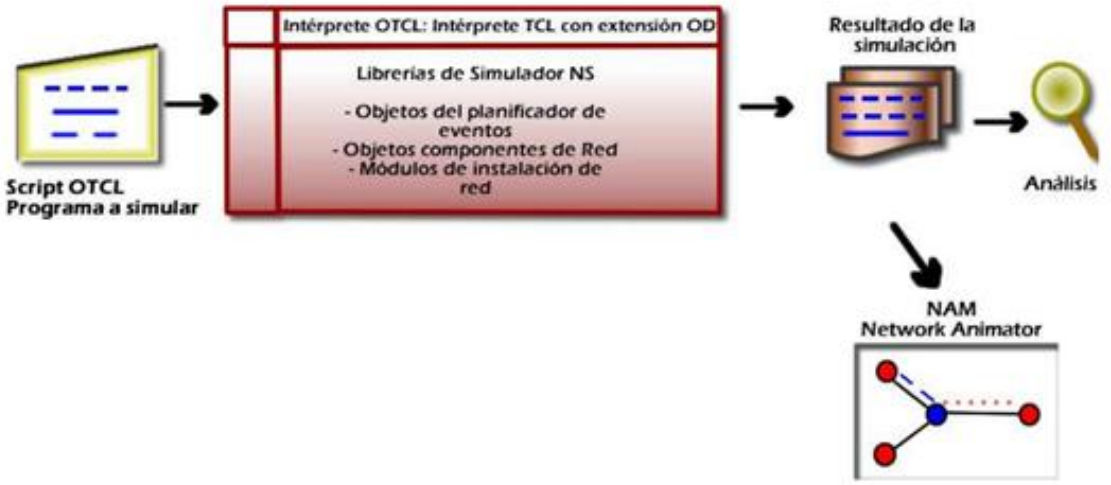

Figura #12 Esquema de simulación de NS-2.

Además permite trabajar en los modos *Unicast* o *Multicast* y utilizar diversos algoritmos para la planificación de colas (como el (*Deficit Round Robin*, DRR), FIFO, FQ (Encolamiento justo) o SFQ (Encolamiento Estocástico Justo)) en caso de que se produzca el cuello de botella en algún nodo.

Los usos más del simulador son:

- Simular estructuras y protocolos de redes.
- Desarrollar nuevos protocolos y algoritmos y comprobar su funcionamiento utilizando las herramientas que acompañan al NS-2 (nam Figura #13 y tracegraph Figura #14).
- Comparar distintos protocolos en cuanto a prestaciones.

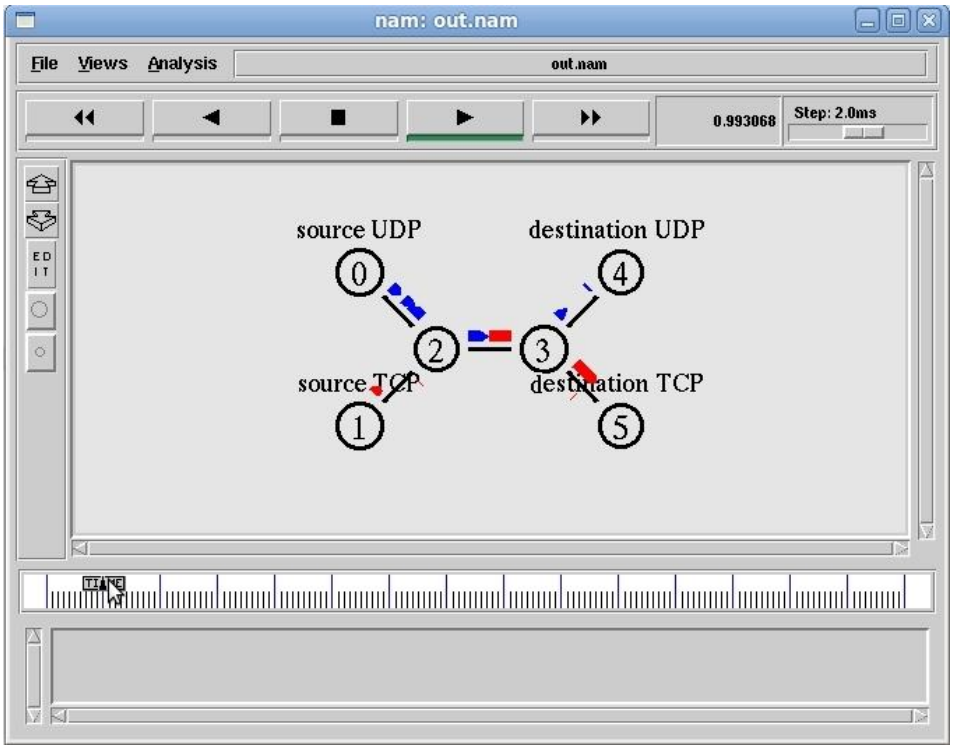

Figura #13 Ejemplo gráfico de NAM.

Entre las ventajas de este programa contiene módulos que cubren un extenso grupo de aplicaciones, protocolos de ruteo, transporte, diferentes tipos de enlaces, estrategias y mecanismos de ruteo; entre otros.

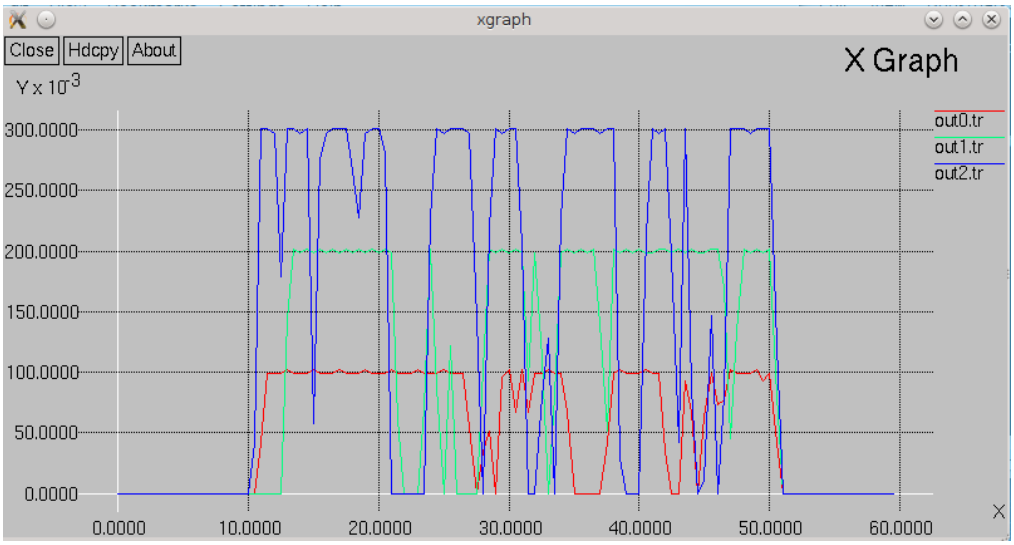

Figura #14 Ejemplo gráfico tracegraph.

Ya se ha comentado la importancia de la simulación de redes y dentro de este campo, el impacto del simulador NS-2 como estándar de facto. Pero uno de sus mayores inconvenientes es que las configuraciones de las simulaciones, por carecer de una interfaz gráfica para el desarrollo de escenarios, hay que realizarlas mediante el desarrollo de código, lo que hace que resulte una tarea laboriosa requiriendo mayor tiempo de desarrollo y conocimientos del lenguaje del simulador.

2.4.2 Opnet

El modelador OPNET es capaz de simular una gran variedad de redes, contando con opciones como flujos de mensajes de datos, paquetes perdidos, mensajes de flujo de control, caída de los enlaces, entre otras, brindando a las universidades e ingenieros, una forma efectiva de demostrar el funcionamiento de redes y protocolos. Además permite mediante librerías la simulación de nodos con diversas características y la comunicación de los mismos con diferentes tipos de enlaces.

OPNET es un lenguaje de simulación orientado a las comunicaciones, que permite a los programadores acceso di recto al código fuente. Este es un simulador utilizado primordialmente por grandes compañías de telecomunicaciones por sus altos costos de licencias.

Para utilizar OPNET, se debe tener en cuenta que este cuenta con una jerarquía que define la manera de establecer el modelo de simulación, por lo tanto, de no tenerse en cuenta en el di seño.

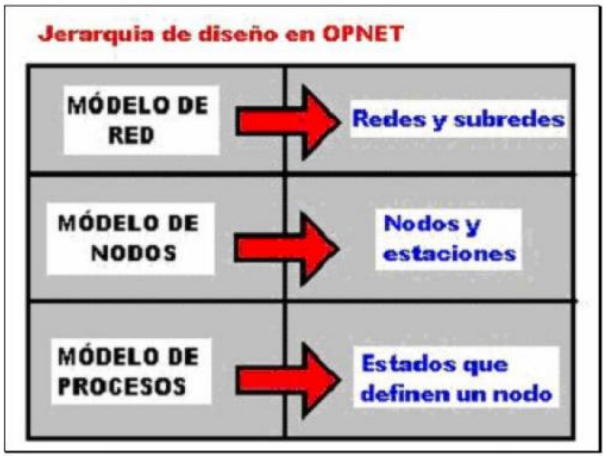

Figura #15 Jerarquía de Diseño en OPNET.

En la Figura #15 se muestra un modelo de red en el que se definen las redes y las subredes de la simulación, se tiene un modelo de nodos en el que se define la estructura interna de los nodos que conforman el esquema de simulación, y por último, se tiene un modelo de procesos donde se establecen los estados por los que atraviesa un nodo.

2.4.3 OMNET ++

OMNET++ es una plataforma de simulación de redes, escrita en lenguaje de programación C++. Su arquitectura está formada por un modulo de acceso al medio llamado mac.cc y se encarga de procesar los paquetes y manejar la cola de paquetes por trasmitir.

En comparación con otros simuladores su arquitectura es relativamente sencilla. Todas las simulaciones se componen de objetos C++ llamados *SimpleModule* que tienen métodos virtuales, privados y protegidos. Sus nodos se componen de una combinación de capas y colas, siguiendo como guía el modelo TCP/IP.

Figura #16 Interfaz OMNET ++.

2.4.4 GLOMOSIM

GloMoSim es una plataforma de simulación más compleja, está escrito en lenguaje de programación Visual C++, todos sus nodos siguen el modelo de Referencia TCP/ IP. Los nodos móviles, poseen un canal inalámbrico modelado con pérdidas de trayectoria y con tipo de propagación.

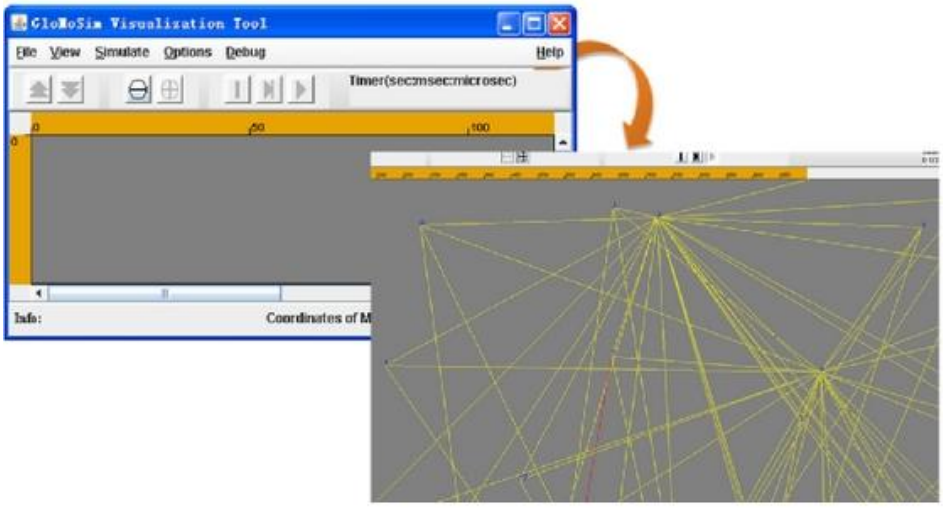

Figura #17 Interfaz GLOMOSIM.

Este canal esta manejado por una entidad principal llamada GLOMO, que posee una matriz donde todas las interfaces de los nodos están conectadas. Así se crea un canal compartido con un modelo de propagación y de pérdidas de trayectoria.

2.4.5 NCTuns

NCTUNs es un simulador y emulador de alta fidelidad capaz de simular varios protocolos utilizados en redes IP cableadas e inalámbricas. La tecnología de su núcleo está basada en la novedosa metodología de re-entrada de kernel diseñada por el profesor S.Y Wang en la Universidad de Harvard.

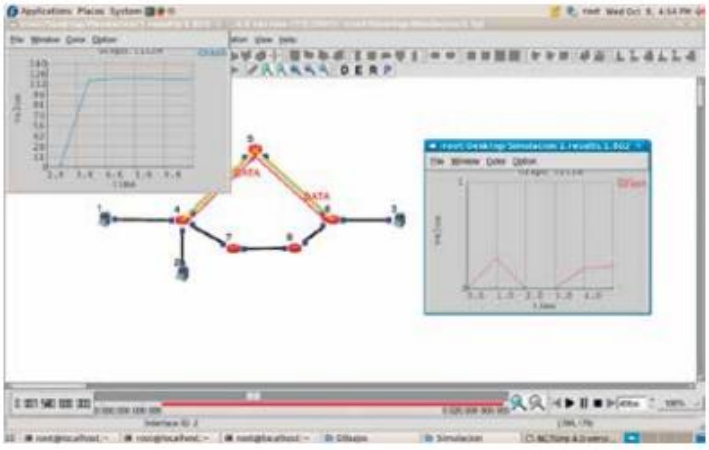

Figura #18 Interfaz NCTuns.

NCTUNs es un simulador liviano que no requiere procesos adicionales del sistema operativo para implementar un *host* ó un *router* virtual. De manera adicional , todos los programas de aplicaciones de la vida real pueden intercambiar paquetes a través de una red simulada por medio de NCTUNs sin necesidad de modificaciones, re-compilaciones ó procesos de interconexión adicional. Presenta además como característica muy importante que los resultados pueden ser repetibles porque el organizador de procesos de UNIX en el Kernel ha sido modificado para controlar de manera precisa el orden de ejecución de la máquina de simulación NTCTUNs.

2.4.6 COMNETIII

COMNET III es un simulador comercial desarrollado por CACI Products Inc. El lenguaje de programación que usa es MODSIM II. En él se pueden crear topologías de red complejas, configurar varias tecnologías (como Ethernet, ATM, FrameRelay, X25, etc.), protocolos y dispositivos de red (como *hubs*, *hosts*, *switches*, *router*, *accesspoints,* etc.) y hacer un estudio del funcionamiento y del rendimiento de redes tipo LAN, MAN y WAN.

La descripción del modelo de red se hace gráficamente, asociando los generadores de tráfico, la configuración de los parámetros y las características de los dispositivos según el modelo a implementar. Después se pone en marcha la simulación y al finalizar, se analizan los resultados. Algunos de los parámetros que pueden obtenerse como resultado son la ocupación del enlace, colisiones, número de mensajes generados, etc.

La interfaz gráfica de usuario utiliza un ambiente de ventanas, con menús y barras de herramientas muy intuitivas. El modelo de red puede verse en modo 3D. Este simulador también ofrece la posibilidad de ver el intercambio de mensajes entre los nodos de la red de manera gráfica y dinámica, en tiempo de ejecución de la simulación.

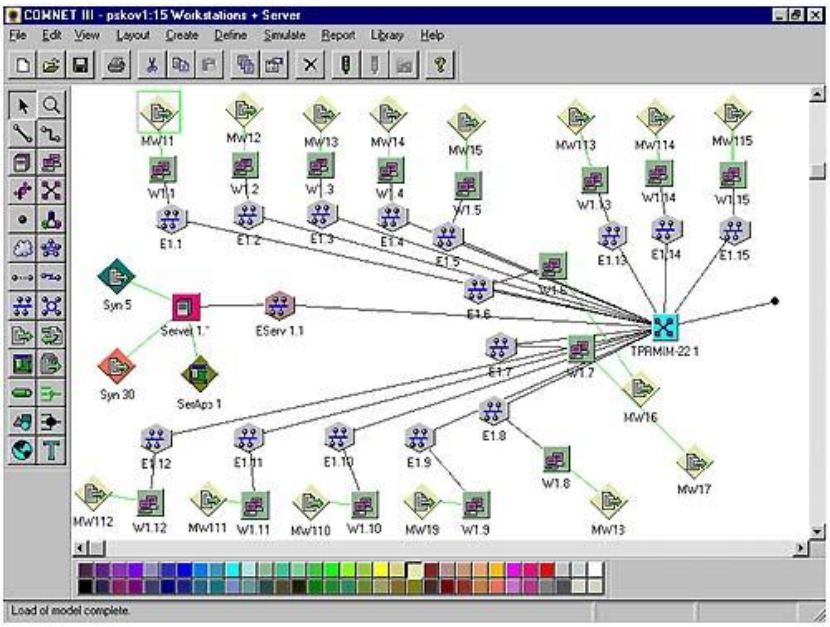

Figura #19 Interfaz COMNET III.

Para uso universitario su versión es bastante limitada. Solamente pueden simularse redes con un máximo de 20 nodos.

3 Simulación y análisis de resultados

Uno de los campos de investigación en mayor auge y más interesante actualmente, se centra en el mundo de las comunicaciones en tiempo real. Se hace prácticamente imprescindible el uso de herramientas de simulación, que permitan estudiar y evaluar diferentes soluciones sin tener que llegar a implantarlas, ahorrando así en costes y tiempo.

En este capítulo se estudiará el comportamiento del tráfico de servicios multimedia mediante el uso de *software* de simulación de redes. Se han generado distintos escenarios de red, realizado la instalación y puesta en marcha del *software*. Mediante scripts se han ejecutado pruebas automatizadas.

Se han realizado las pruebas pertinentes para sacar conclusiones en base a parámetros QoS, como el ancho de banda, el retardo y la tasa de pérdida de paquetes. Se buscará optimizar el uso de aplicaciones multimedia *streaming* usando diferentes técnicas.

También, se analizan técnicas como el alisado de tráfico que permiten mejorar la utilización de los enlaces en los escenarios planteados. Para ello se han llevado a cabo diversas pruebas en los escenarios de simulación, midiendo así las reacciones del sistema ante los cambios en las condiciones de la red. Para que los servicios multimedia comparados con los tradicionales funcionen bien (sean aceptables y con calidad suficiente), la red de datos debe tener soporte de QoS, especialmente cuando la red posee cuellos de botella y/o aparecen situaciones de congestión. Situaciones que se han simulado mediante *software*.

Por último, mediante el análisis de los resultados obtenidos, se extraen conclusiones sobre las características y funcionamiento de las aplicaciones multimedia y sus posibles mejoras. Buscando tener una representación lo más fiel posible al comportamiento de una red física, dicho nivel de fidelidad se puede medir al hacer un análisis de variables tales como similitud de las topologías, latencia en los paquetes y trazas de entrega de paquetes.

El presente estudio no pretende realizar una descripción exhaustiva de los métodos y técnicas para la realización de las pruebas. Más bien, se presentan resultados de simulaciones que permiten analizar el efecto del tráfico de las aplicaciones (que se utilizan para las pruebas) ante diversos comportamientos de la red. A pesar de no entrar en este tipo de detalles, en el anexo B se presentan los scripts correspondientes para que se puedan analizar.

3.1 Escenario propuesto

En la Figura #20 y Figura #21 se puede observar el escenario general en cuya topología se basan las pruebas. En cada extremo hay dos flujos de información comunicados entre dos nodos. Los flujos utilizados son: FTP, VoIP (Skype) y P2PTV

(Sopcast). A efectos prácticos ese enlace entre los nodos tiene un determinado ancho de banda que varía desde dos hasta un Mbps. Se ejecutaran las pruebas variando los flujos de información que actúan en un momento determinado y el ancho de banda entre los nodos.

Para las pruebas se utilizan dos escenarios. El primero se muestra en la Figura #20. En este escenario se transmiten dos flujos de información. Uno con una traza de Skype y otro con el servicio FTP que van a competir por transmitirse por el mismo enlace de , el cual se ha dimensionado con la finalidad de producir un cuello de botella. El objetivo principal de la prueba es determinar la tasa de pérdida de paquetes. El estudio se centrará en el enlace de subida (ya que es el más restrictivo en cuanto a ancho de banda para las actuales redes de acceso, por ejemplo, ADSL), y por lo tanto, se ha dimensionado el enlace de bajada de tal manera que no produzca pérdidas de paquetes (a 100 Mbps y con *buffer* de 500 paquetes), mientras que en el de subida se han implementado algunas limitaciones, utilizando valores sugeridos en [SFNS13], los cuales han sido encontrados experimentalmente en dispositivos comerciales. En el cuello de botella el *buffer* es de 50 paquetes y en las simulaciones se usan distintos anchos de banda para la capacidad de acceso. En consecuencia diferentes niveles de utilización del enlace, que van desde 50% al 100% (de 10 en 10%). No se va a introducir retardo entre el *Node* 0 y 1. Esta prueba se va realizar de manera automatizada mediante un script que la realizará 100 veces. Haciendo que los flujos de Skype y FTP comiencen aleatoriamente. Finalmente los resultados serán una media ponderada de las 100 iteraciones realizadas. También se indica que política usaremos en el manejo de colas en los nodos que en nuestro caso es *Droptail*.

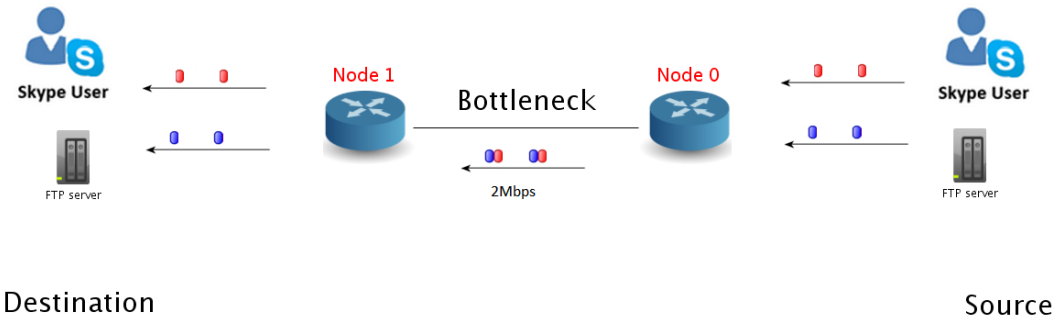

Figura #20 Escenario de simulación FTP vs Skype.

En el segundo escenario de la Figura #21 tenemos dos flujos de información que compiten por transmitirse desde origen a destino por el cuello de botella. Son una traza de Sopcast y otra de FTP. Como en el escenario uno las condiciones son las mismas para el cuello de botella y los parámetros de variación del ancho de banda del mismo. En este caso se va a utilizar una traza de Sopcast captada del uso del programa con unas características específicas. En este caso se han usado tanto la traza original como la traza procesada a distintos anchos de banda mediante la técnica de optimización llamada alisado de datos. Dichas trazas han sido alisadas para valores de ancho de banda comprendidos entre los 1 y 10 Mbps. Para lo obtención de

resultados se realiza la iteración de 100 ejecuciones. Con todas estas iteraciones se realiza el cálculo de la media en cada parámetro medido (ancho de banda utilizado por el servicio en la salida y en la llegada, perdida de paquetes, perdida de *bytes*, etc.).

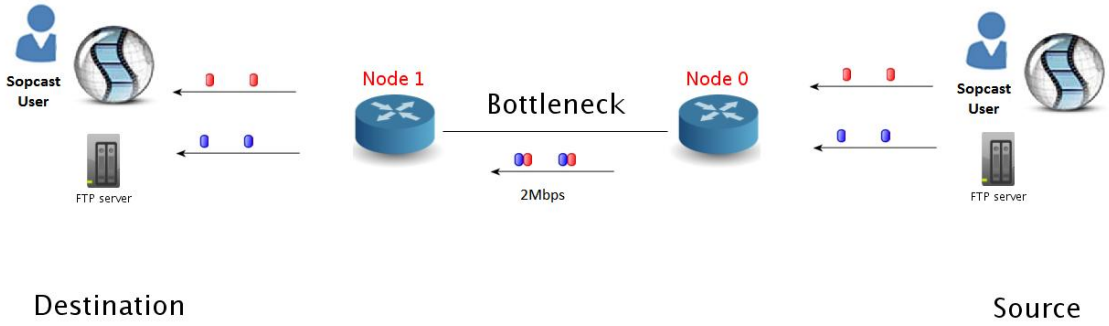

Figura #21 Escenario de simulación FTP vs Sopcast.

Los pasos seguidos para la creación y ejecución de los escenarios mediante el *script* son: Creación del programador con los tiempo de simulación, anchos de banda de interfaces, retrasos, tamaño de *buffer* utilizados para cada interfaz. Tras esto se procede a la creación de la topología a simular dando lugar a la configuración de la transmisión y tipos de colas utilizadas, situación de los nodos, representación gráfica. Se generan trazas que monitorizan y recaban información de los flujos que usan los servicios a simular, en cada enlace de subida y bajada. Se definen los agentes en origen y destino para cada nodo terminal de la red, a partir de los tráficos de servicios reales capturados en fichero de trazas y los servicios ya implementados en NS-2 con sus respectivos parámetros para servicio de *streaming* de video P2P. Finalmente se programa la secuencia de eventos. Los flujos de datos de los escenarios simulados de activan de forma aleatoria para el envío y se recaba la información en ficheros de trazas de los enlaces de subida y bajada, para su posterior análisis. De este análisis se analizan los parámetros que den una idea de la QoS.

3.1.1 Tráfico utilizado

Con el fin de implementar las pruebas descritas anteriormente, se han utilizado tres fuentes diferentes de tráfico para los flujos multimedia: video *streaming* P2P, P2PTV y tráfico de un servicio FTP. Para esto, se han capturado trazas de video *streaming* P2P en escenarios reales para luego ser generados en NS-2, utilizando el mismo tamaño de los paquetes y los tiempos entre paquetes. Para la captura se utiliza un *sniffer* en la mejor ubicación para que no degrade el rendimiento de la aplicación como se muestra en la Figura #22. Para el tráfico VoIP se utilizó un agente CBR NS-2. utilizado para la captura de tráfico se ilustra en la Figura #23.

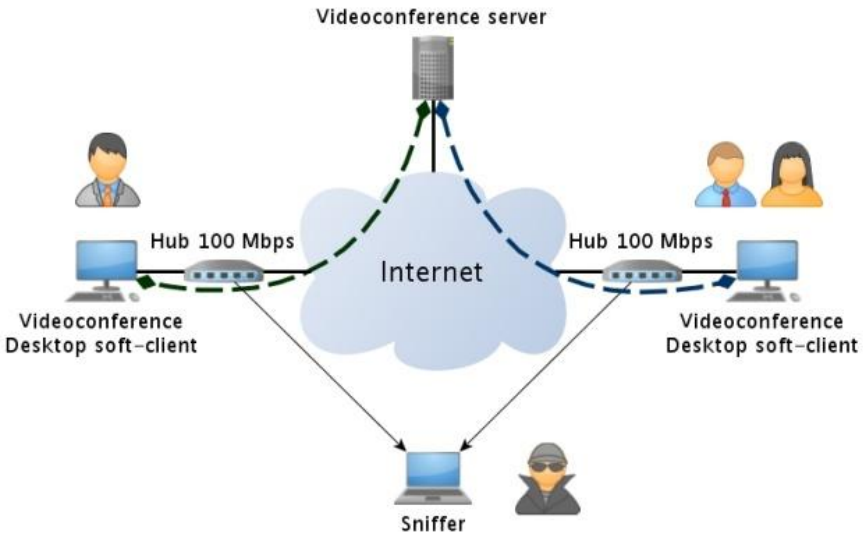

Figura #22 Captura de tráfico real de video *streaming* P2P.

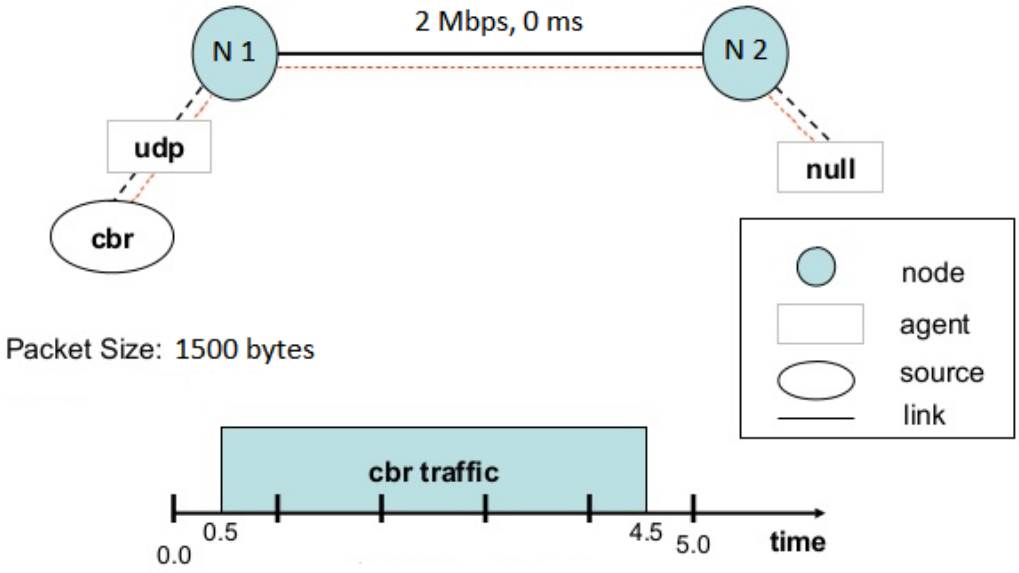

Figura #23 Tráfico generado por el agente CBR NS-2.

El flujo de los paquetes generados por el servicio P2PTV (Sopcast) se presenta en forma de ráfagas (como se muestra en la Figura #24). Cada ráfaga transmite una cantidad de paquetes diferente. El tiempo entre las llegadas de las ráfagas es el lapso en el cual se transmite esa cantidad de paquetes; de forma análoga, existe un tiempo entre las llegadas de cada paquete contenido en la ráfaga que define la diferencia entre los inicios de cada paquete.

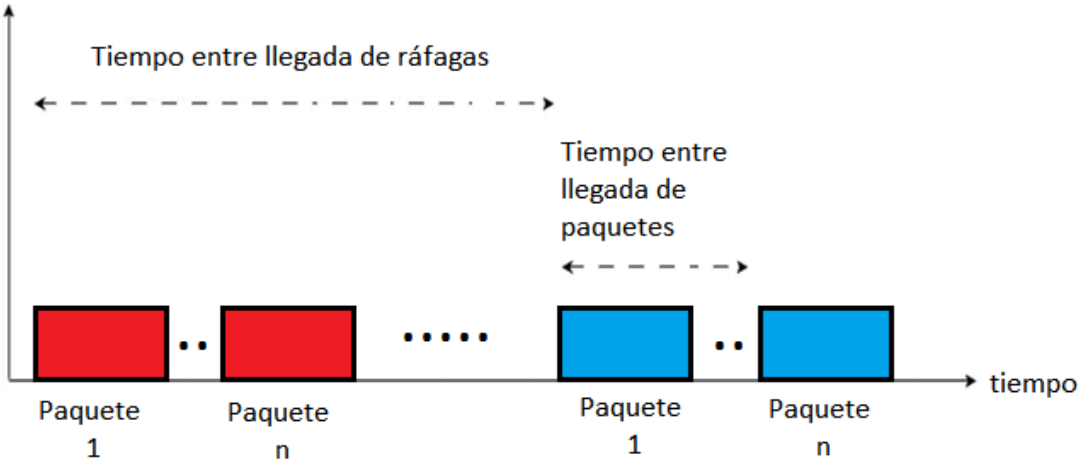

Figura #24 Caracterización del tráfico a ráfagas.

En la Figura #25 se muestra en el eje x el tiempo de transmisión de cada paquete y en el eje y el tiempo entre los inicios de cada paquetes. De esta manera podemos observar que hay una mayoría de los paquetes que se encuentran con tiempos pequeños (los que están pegados al eje x), lo que significa que van seguidos, uno de tras de otro y un tiempo entre ellos corto. Mientras que los bastones más altos, son aquellos paquetes que tienen un tiempo entre paquetes más grande, lo que sugiere un inicio de ráfaga.

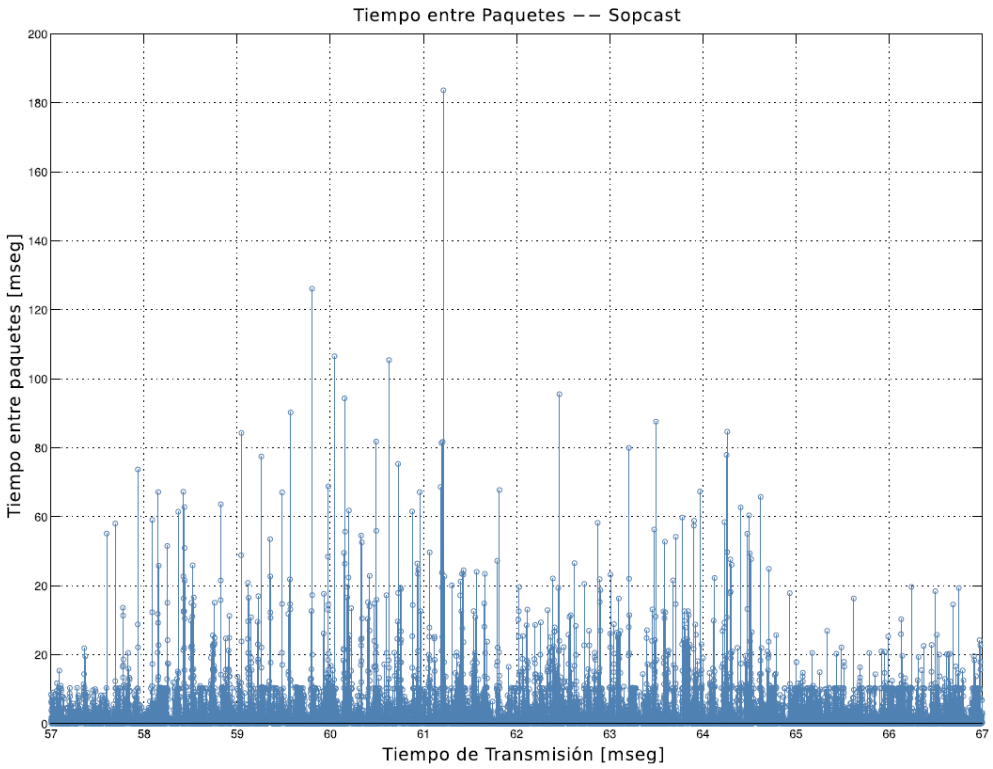

Figura #25 Tiempo entre los inicios de los paquetes para el flujo de Sopcast.

En cuanto a los tamaños, se ha construido un histograma de tamaños de paquetes para dicha traza, el cual se muestra en la Figura #26.

Se puede observar que en dicha traza existen una gran cantidad de paquetes pequeños, los cuales podría ser asociados a la señalización y control que utiliza dicha aplicación. Por otro lado, hay otra cantidad importante de paquetes de gran tamaño los que pueden corresponder a video, lo cual se corresponde con el apartado 2.2.3. Existen poco paquetes de tamaños intermedios, y los que se encuentran podrían ser fragmentos de video.

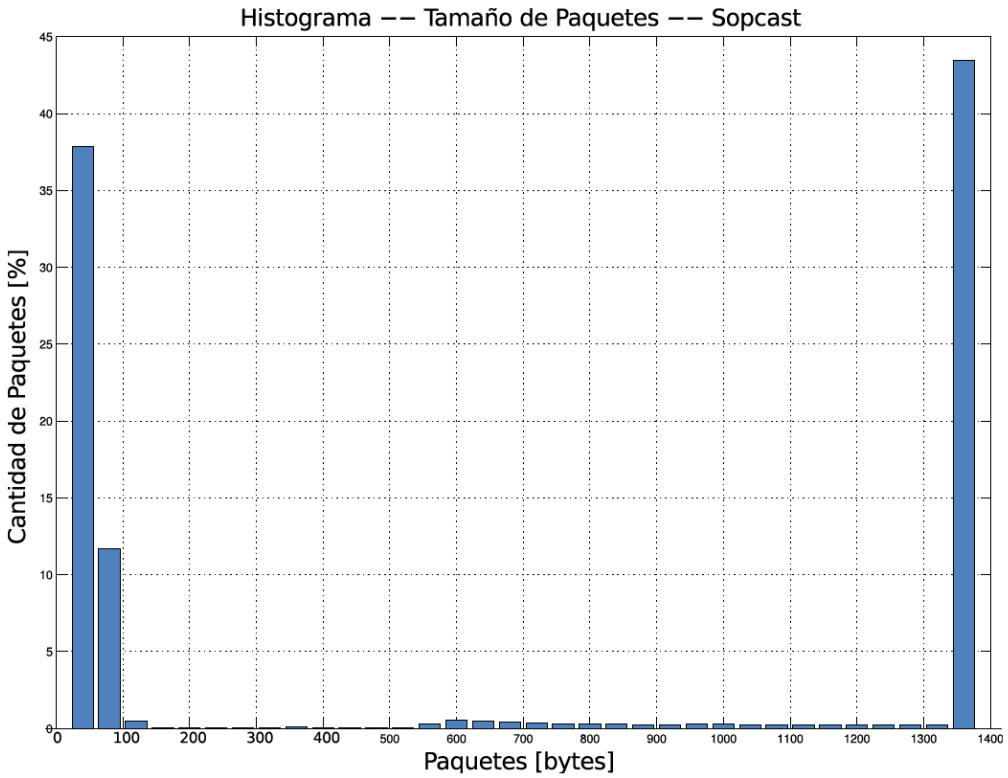

Figura #26 Histograma de tamaños de paquetes para el flujo de Sopcast.

En la Figura #27 se muestra en el eje x el tiempo de transmisión de cada paquete y en el eje y el tiempo entre los inicios de cada paquetes. Podemos observar como el tiempo de transmisión de cada paquete es aproximadamente constante (20ms), lo cual marca un ancho de banda que varía muy poco. Esta forma de envío parece la de un servicio de envío de datos tradicional. No hay signos de aparición de ráfagas. Del apartado 2.2.1 sabemos que existe un *buffer* previo a la transmisión de información a la red, donde se codifica y alisa el tráfico.

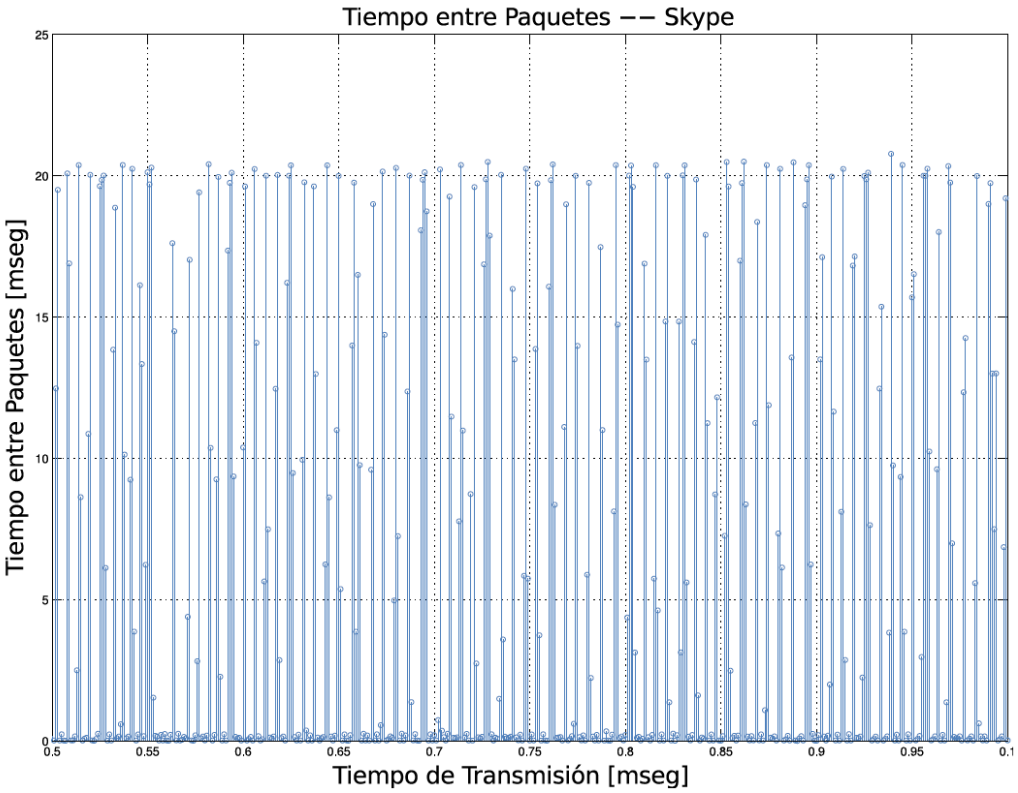

Figura #27 Tiempo entre los inicios de los paquetes para el flujo de Skype.

El histograma de tamaños de paquetes para la traza de Skype, es el que se muestra en la Figura #28.

En dicha traza existen una gran cantidad de paquetes pequeños, están asociados a todos los pasos previos a la comunicación como por ejemplo la búsqueda de servidores tipo supernodo o logueo en un servidor de identidad, como se reflejó en el apartado 2.2.1. Existen una variedad de paquetes de tamaños intermedios. Los paquetes de datos pueden variar de tamaño según los parámetros de la red en cada momento. Esto refleja como la QoS de Skype es adaptable a las situaciones de congestión de la red, como por ejemplo bajando la calidad de video o audio transmitido para evitar congelamiento de imagen y cortes de voz.

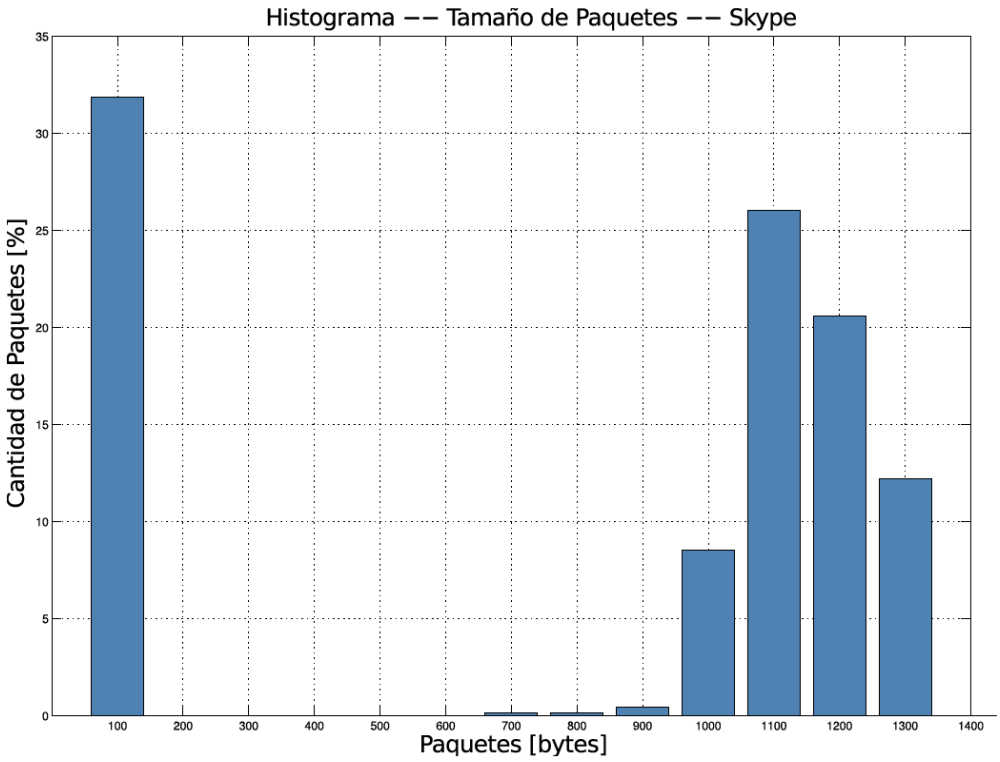

Figura #28 Histograma de tamaños de paquetes para el flujo de Skype.

Hay que mencionar que no se hace un análisis similar para el FTP debido a que es un tráfico muy estable y ampliamente conocido. Además que se ha utilizado el propio generador de NS.-2 para este tipo de tráfico. Los parámetros definidos por la configuración del FTP son los que vienen por defecto.

3.2 Análisis del tráfico de Skype

La transmisión de un servicio de videoconferencia (servicio bidireccional, interactivo y multimedia) presenta unas fuertes necesidades de QoS.

El rango total de retraso (*delay*) para una comunicación de video en un sentido es aproximadamente 125-150 milisegundos. Por ello se necesita un retardo bajo y constante (sobre los 200 milisegundos para una calidad aceptable, permitiendo llegar hasta los 400 milisegundos en ciertas circunstancias) y unas pérdidas que no excedan el 2 – 5 % de los paquetes IP, ya que pueden producir congelamiento en el video y/o pérdida del audio.

Se ha realizado una prueba sobe el escenario 1 de simulación. Se trata de ver como los dos servicios compiten por el ancho de banda accesible en situaciones de congestión en la red Figura #29 y Figura #30. Variando los anchos de banda de transmisión entre extremos por el cuello de botella. Según las figuras que muestran perdidas en tanto por ciento de *bytes* y paquetes para FTP y SKYPE, se hace notar como a partir de un 1,6 Mbps de reducción en el ancho de banda de transmisión, produce un nivel considerable de pérdidas que puede comprometer la calidad de la

conexión de videoconferencia, superando el siete por ciento de perdidas. Llegan a unos valores que provocan posibles cortes en audio y congelación de video.

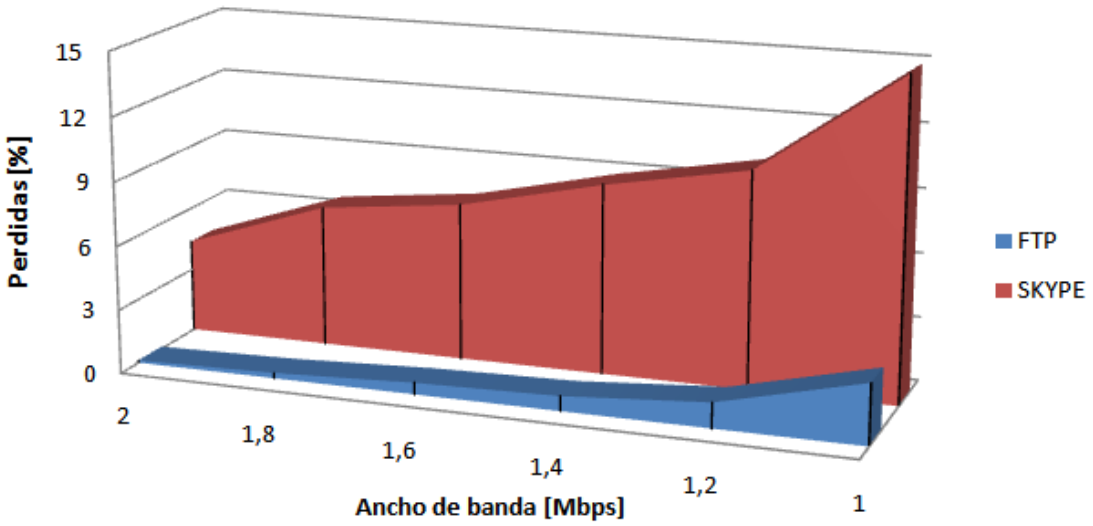

Figura #29 % Pérdidas de *bytes* modificando BW.

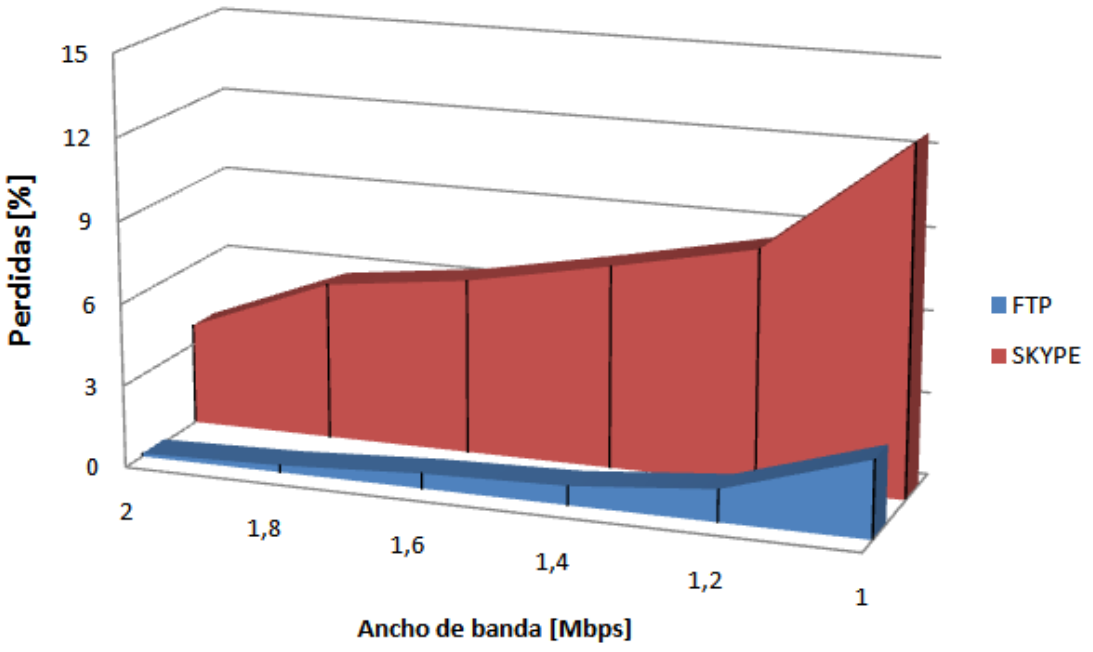

Figura #30 % Pérdidas de paquetes modificando BW.

3.3 Análisis del tráfico de Sopcast

3.3.1 Efecto de las variantes TCP

En este apartado se estudia cómo afecta el tipo de variante de TCP utilizado al tráfico de una aplicación multimedia. Para este caso, se ha seleccionado un servicio FTP que compite por el ancho de banda de transmisión disponible en el cuello de botella con el tráfico de un flujo de Sopcast. El servicio FTP se implementa mediante un agente para dicho servicio de NS-2, mientras que para el tráfico de Sopcast se ha utilizado la traza descrita en la sección anterior.

Se usan distintos agentes TCP implementado en NS-2:

- Agent/TCP/Reno - envío TCP tipo "Reno".
- Agent/TCP/Newreno - modificación del Reno
- Agent/TCP/Sack1 - TCP con repetición selectiva (RFC2018)
- Agent/TCP/Vegas - TCP Vegas
- Agent/TCP/Fack - Reno TCP con "asentimiento/reconocimiento hacia delante"'
- Agent/TCP/Linux - envíos TCP con soporte para SACK que tiene módulos de control de congestión TCP del kernel de Linux.

Se presentan en la Figura #31 y Figura #32, las pérdidas de *bytes* y paquetes en tanto por ciento para el FTP y el Sopcast. El enlace en el cuello de botella tiene un valor de un Mbps.

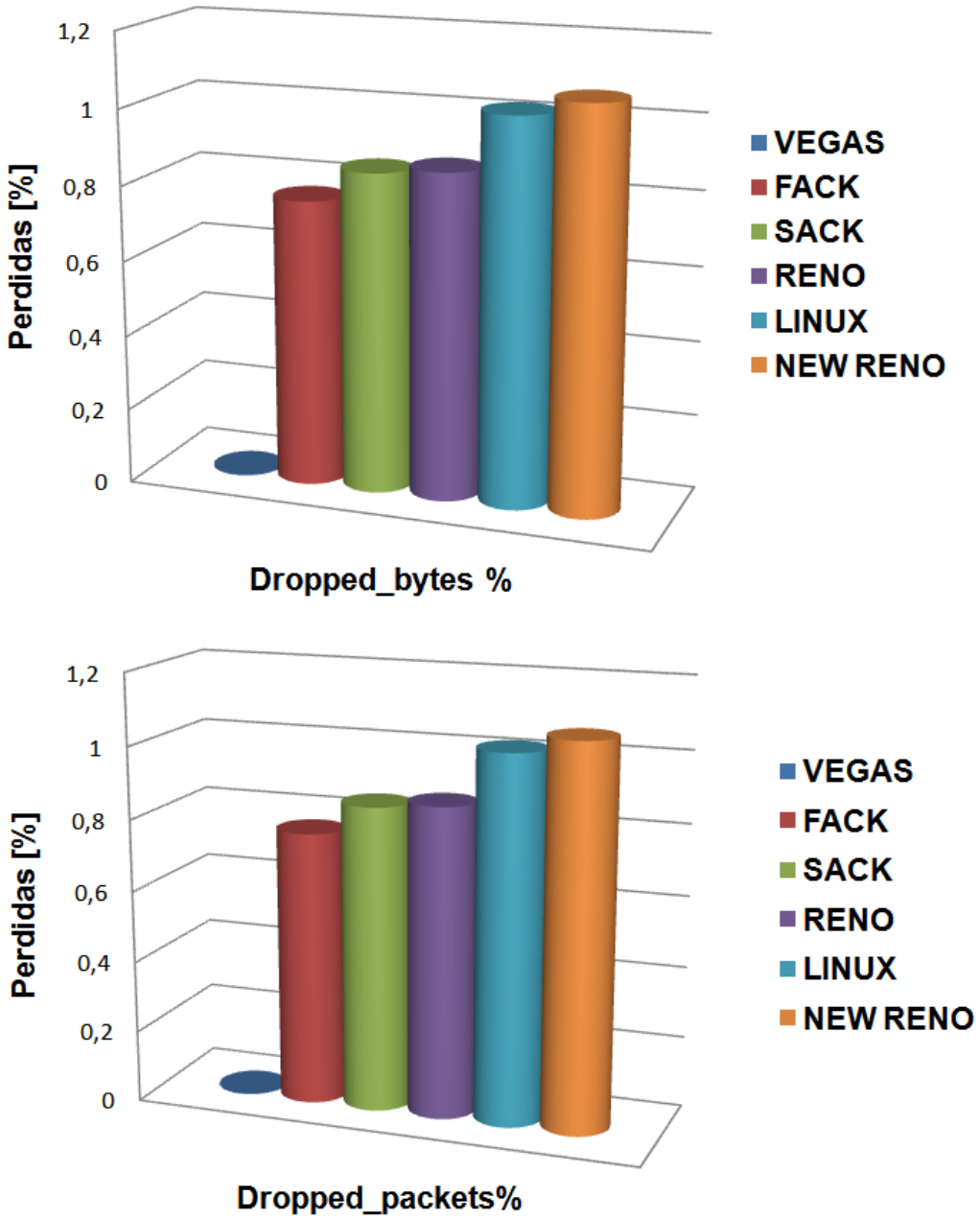

Figura #31 % Pérdidas entre nodos en *bytes* y paquetes FTP.

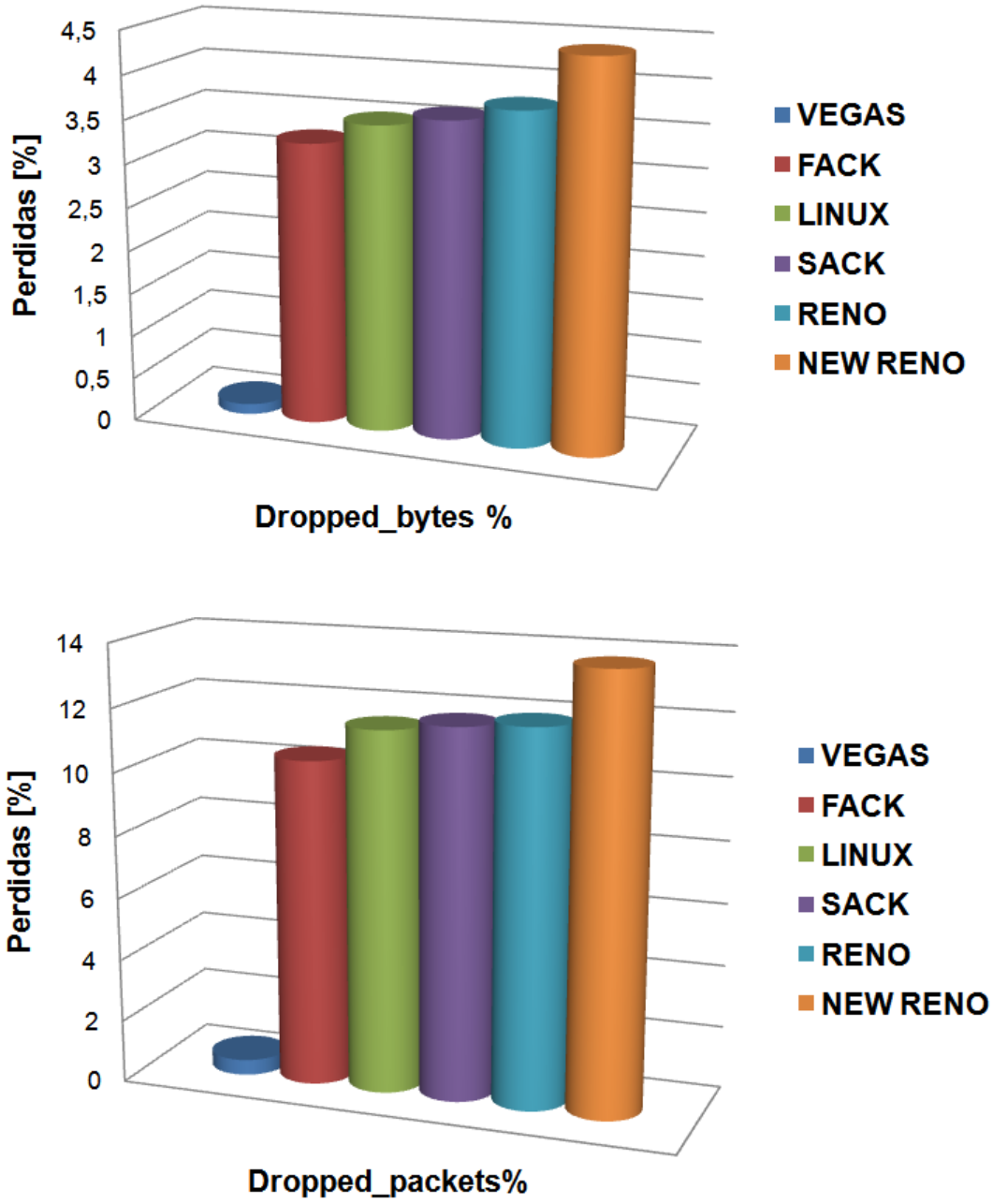

Figura #32 % Pérdidas entre nodos en paquetes Sopcast.

Como se puede observar de la Figura #33 y Figura #34 tenemos una representación de pérdidas de las distintas versiones de control de congestión utilizada para TCP. Las mínimas pérdidas para Sopcast y nulas para FTP se dan cuando se usa el agente Vegas. Esto se debe a que cuando emite los paquetes FTP y detecta errores en la transmisión de la ventana, se paraliza el envío de paquetes, ya que se detecta congestión. Este se realiza más tarde con lo cual deja de competir por el ancho de banda disponible en el cuello de botella. TCP Vegas detecta congestión basado en el aumento de los valores de RTT de los paquetes en la conexión, a diferencia de TCP Reno, que detecta la congestión a posteriori cuando se ha producido la pérdida de paquetes.

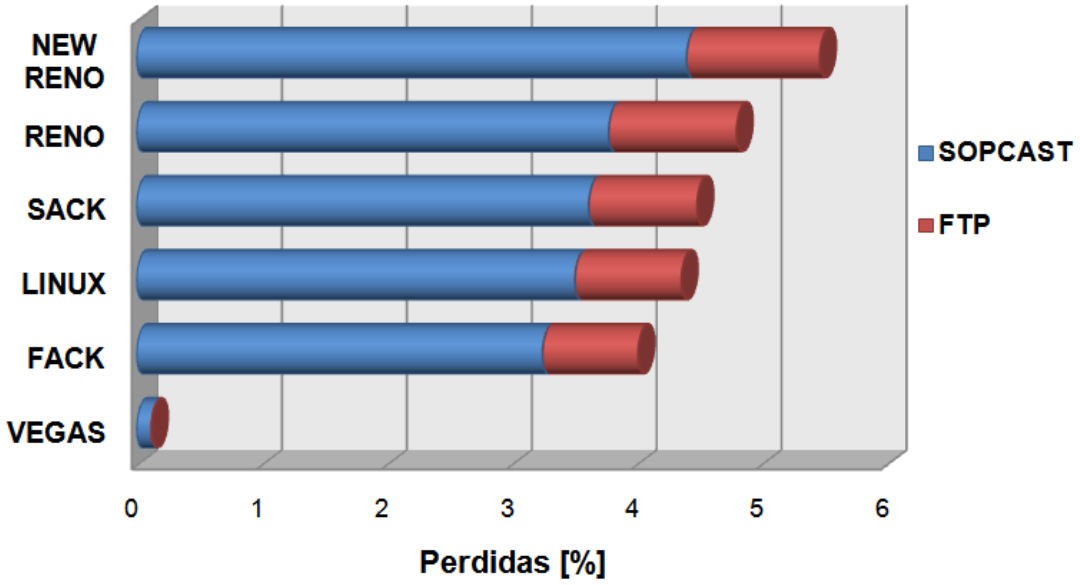

Figura #33 Pérdida de *bytes* Sopcast vs tipo de TCP.

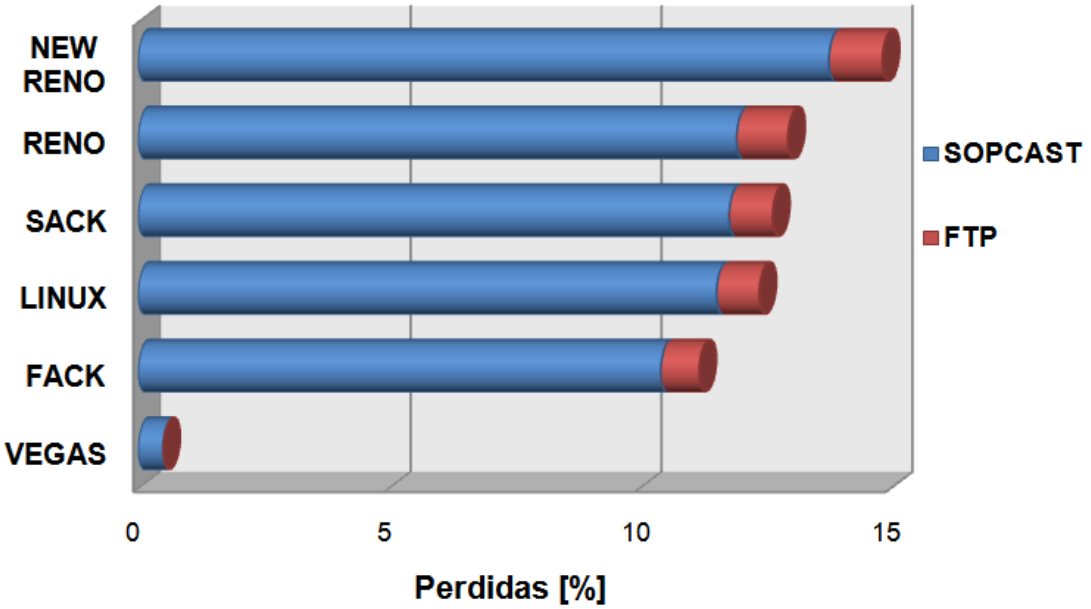

Figura #34 Pérdida de paquetes Sopcast vs tipo de TCP.

Por otro lado vemos como el tipo de FTP LINUX mejora el tanto por ciento de pérdida de paquetes comparado con el agente FTP habitual de Windows que es el SACK (acuse de recibo selectivo). Estas versiones de TCP al detectar que no se ha recibido un segmento, comienzan con el reenvío de los segmentos FTP más recientes de forma selectiva. Son adecuados cuando se producen errores a ráfagas. No envían hasta que el número de segmentos pendientes de confirmación es menor que el tamaño de la ventana de congestión del emisor. Esto provoca gran cantidad de reenvió de segmentos y una gran pérdida de estos, haciendo aumentar el tanto por ciento de estas.

SACK y Reno TCP logran prestaciones idénticas, en situaciones sin pérdida de paquetes y en situaciones con alta pérdida de paquetes. Si hay una alta pérdida de paquetes, la ventana de congestión sigue siendo pequeña y los tiempos de espera no se puede evitar, incluso con SACK TCP.

El algoritmo FACK utiliza la información adicional proporcionada por SACK. Mantiene una medida explícita del número total de *bytes* de datos pendientes en la red. Se puede observar una mejora en las pérdidas sobre SACK.

El peor valor para pérdidas resulta del uso de TCP NewReno. Es capaz de detectar múltiples pérdidas de paquetes y admite múltiples retransmisiones, NewReno sufre del hecho de que le toma un RTT detectar cada pérdida de paquete. Solo se puede deducir una pérdida de paquetes tras recibir un ACK por el primer segmento retransmitido. Esto retardo provoca que se pierdan gran cantidad de segmentos de información.

3.3.2 Propuesta para la optimización en la QoS

Como se ha explicado en el capítulo anterior, Internet fue diseñada como una red *best-effort*, es decir, la red no puede garantizar un retardo acotado en la entrega de los paquetes. Sin embargo, en los últimos años, está siendo ampliamente utilizada para servicios interactivos en tiempo real, como VoIP, videoconferencias, servicios de telemedicina o juegos online.

Anteriormente del despliegue actual de redes de acceso, la calidad para los servicios de telefonía tradicional era constante garantizada. Actualmente convergen en la red de conmutación de paquetes tanto servicios de voz como datos. Esto supone un ahorro de costes en instalación y mantenimiento. Por otra parte implica el desarrollo de menanismo que aseguren una QoS mínima.

La QoS se entiende como algo objetivo (retardo, perdida de paquetes y *jitter*) y, por tanto, medible, que no depende de la experiencia subjetiva del usuario. Por otro lado, en los últimos años ha aparecido otro concepto, más referido a la experiencia que tiene el usuario cuando usa un servicio: la Calidad de la Experiencia, también denominada QoE. Este concepto engloba también las experiencias del usuario del servicio, y es por tanto más subjetivo y difícil de medir.

Los parámetros objetivos que se pueden medir a partir del tráfico de la red simulada serán: retardo, perdida de paquetes y *jitter*.

Basándonos en los objetivos expuestos en este trabajo y en los parámetros de calidad que se persiguen, se propone utilizar la técnica de alisado de tráfico. Consiste en introducir un retardo controlado entre los paquetes de información útil. Se realizan unas simulaciones con los anchos de banda variando entre 1 y 10 Mbps. Se conforma el tráfico mediante estos retardos controlados y se pasa cada flujo por el escenario

número 2. Si se introduce mayor retardo entre paquetes de datos el ancho de banda es menor y viceversa.

Se denomina congestión a la circunstancia en la que el rendimiento de la red o parte de ella se degrada debido a la presencia de demasiados paquetes. La congestión es un problema global, que se da a nivel de red como consecuencia del tráfico agregado de varias fuentes sobe un enlace o *router* de baja capacidad. A diferencia de la congestión, el control de flujo es una circunstancia que sólo puede darse en conexiones punto a punto, es decir, a nivel de enlace o a nivel de transporte.

Algunos parámetros que reflejan presencia de congestión:

- Porcentaje de paquetes descartados
- Longitud media de las colas en las interfaces de los *router*
- número de paquetes que dan *timeout* y se retransmiten (no por errores de transmisión)
- Retardo medio por paquete
- Desviación media del retardo por paquete

El tráfico a ráfagas es la principal causa de congestión .Entre los mecanismos de control de congestión está el alisado de tráfico (*traffic smoothing*) que permite establecer unos márgenes máximos al tráfico de ráfagas. En el caso que nos ocupa se ha analizado una traza de tráfico del programa Sopcast, capturada como se ha explicado anteriormente. Hemos pasado de un tráfico a ráfagas a un flujo a velocidad de transmisión constante, para así lograr optimizar o garantizar el rendimiento, baja latencia, y/o un ancho de banda determinado.

El proceso de alisado se lleva a cabo mediante el cálculo del ancho de banda instantáneo de cada paquete que pertenece a la traza bajo estudio. Para ello se ejecuta el siguiente cociente:

$$BW\ instantáneo = \frac{Bytes\ mensaje * 8 * 10^6}{Tiempo\ entre\ paquetes}$$

Una vez obtenido este dato se determina el BW final como se muestra en la Figura #35. Se sigue el siguiente diagrama de flujo para obtenerlo, siendo BW alisado el que se determine por cada prueba. Este dato se mueve desde 1 a 10 Mbps.

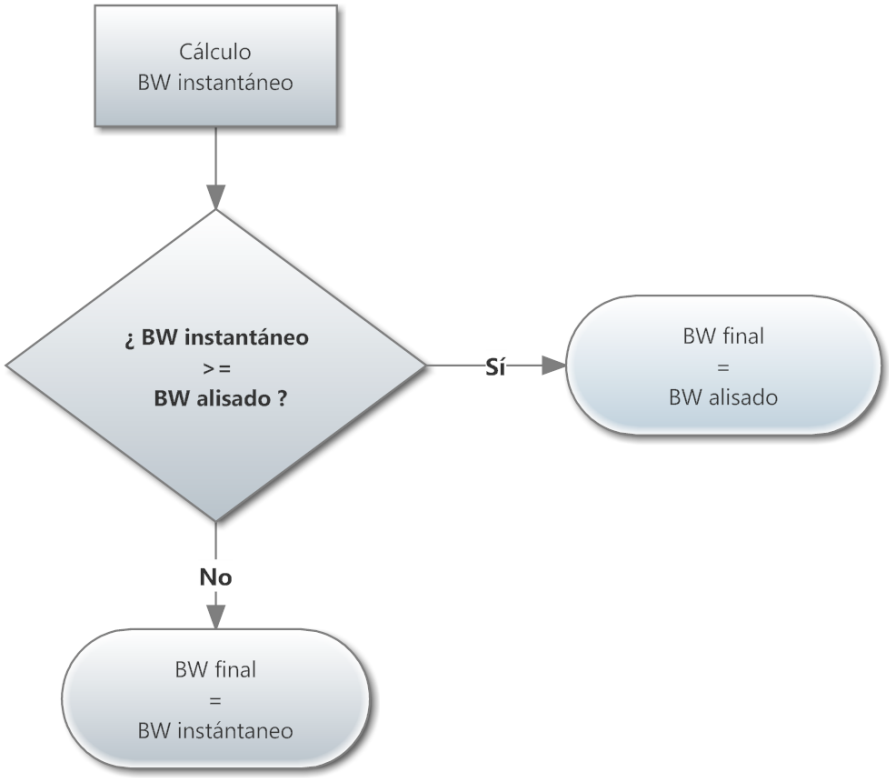

Figura #35 Cálculo de ancho de banda de alisado.

La técnica de alisado de datos sobre un determinado flujo *streaming,* como es el caso de Sopcast, tiene una serie de objetivos de QoS que no se cumplen. En esta técnica estamos introduciendo un retardo variable entre cada uno de los paquetes de datos para conformar un tráfico de un determinado ancho de banda de transmisión. Al tener un tipo de datos a transmitir que se generan por ráfagas y tanta variación de ancho de banda instantáneo, se genera un exceso de retardo. Este retardo provoca que la comunicación en tiempo real tenga mayor latencia y por ello un detrimento en uno de los parámetros ya comentados en el apartado 2.3.

3.3.3 Análisis de pérdida de paquetes

Nos basamos en las pruebas realizadas en el escenario 2. Se han ido utilizando distintos niveles de alisado sobre la traza elegida de Sopcast. En la Figura #36, Figura #37, Figura #38 y Figura #39 se grafican datos que reflejan la pérdida en tanto por ciento de paquetes y de *bytes* que se produce en el enlace entre *Node* 0 y 1.

Se diferencian dos tipos de gráficas: La primera, Figura #36 y Figura #37, fijan los valores, antes mencionados, de pérdidas. Se decrementa una cantidad constante el valor de ancho de banda de transmisión en el cuello de botella. Se observa como al disminuir dicho ancho de banda las perdidas aumentan. Lo que refleja que en situaciones de congestión en el enlace entre los nodos la técnica de alisado no es muy eficaz.

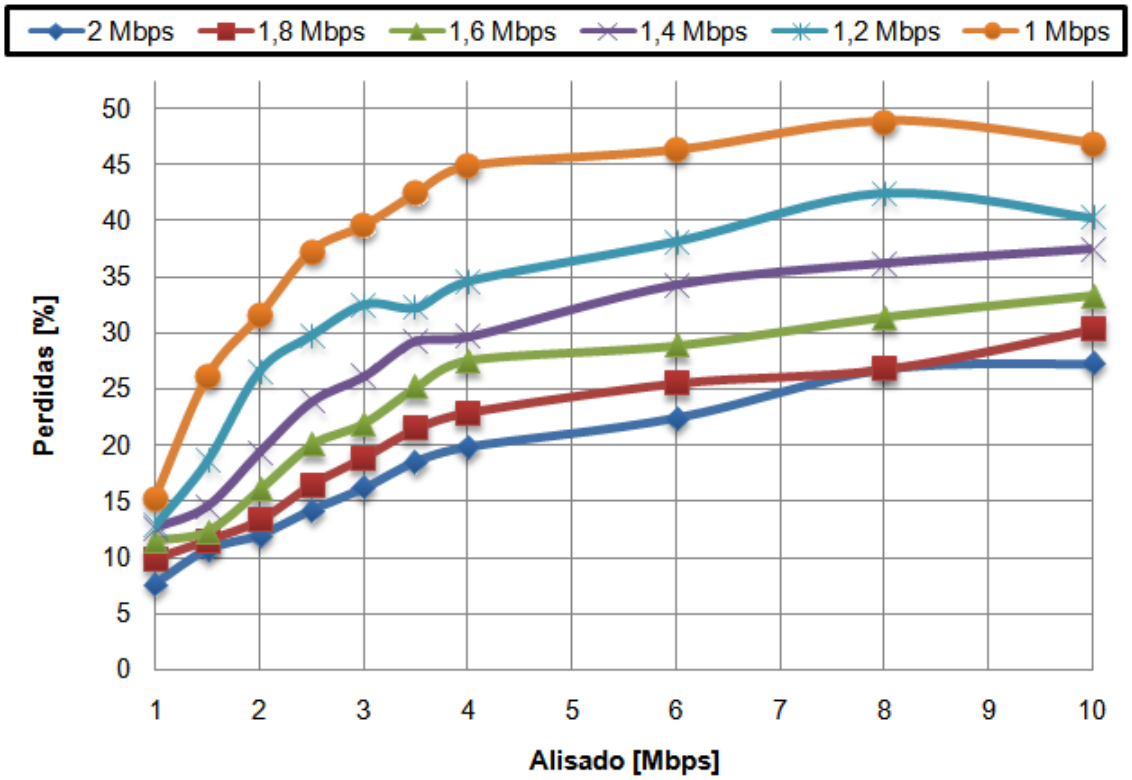

Figura #36 % Pérdidas de paquetes para distintos alisados.

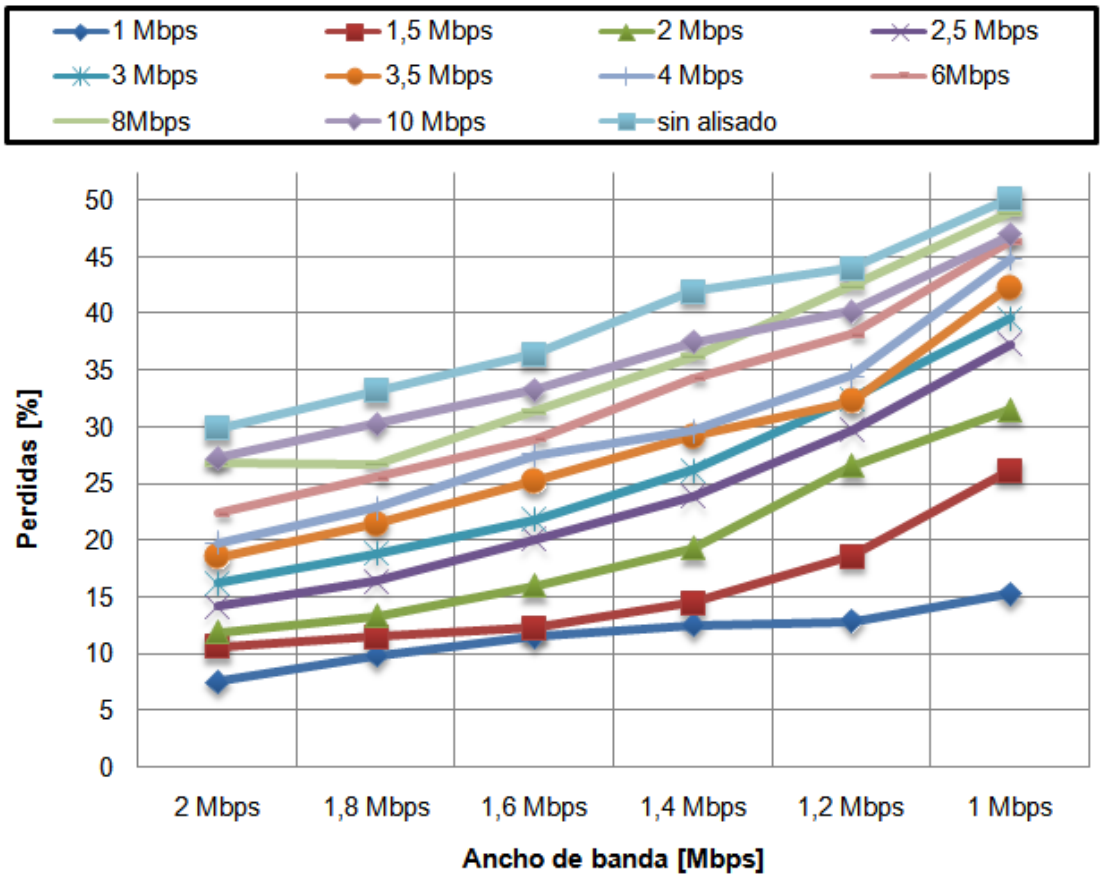

Figura #37 % Pérdidas de paquetes BW variable entre nodos.

En la segunda, Figura #38 y Figura #39, se hace constar como mejora la probabilidad de perdida si vamos aplicando progresivamente valores de alisado a distinto régimen binario. A menor régimen binario menor es el porcentaje de perdida. El contrapunto se presenta en que estamos introduciendo mayor retardo, ya que espaciamos más los tiempos entre los paquetes de datos. Con lo cual aumentamos la latencia.

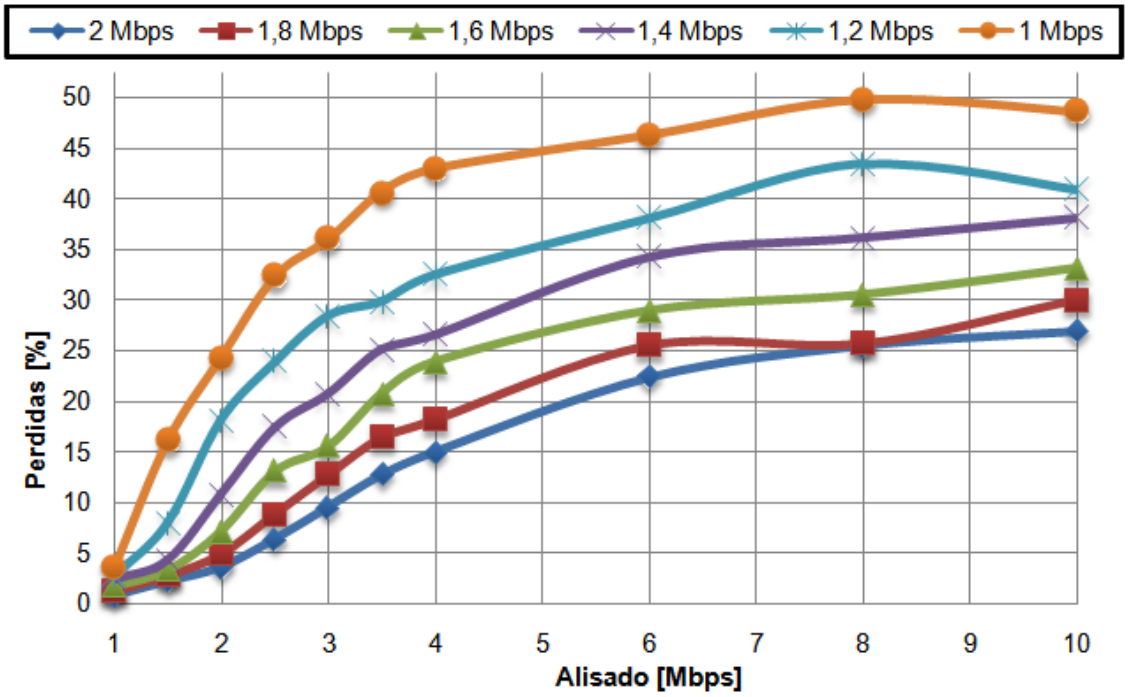

Figura #38 % Pérdidas de *bytes* para distintos alisados.

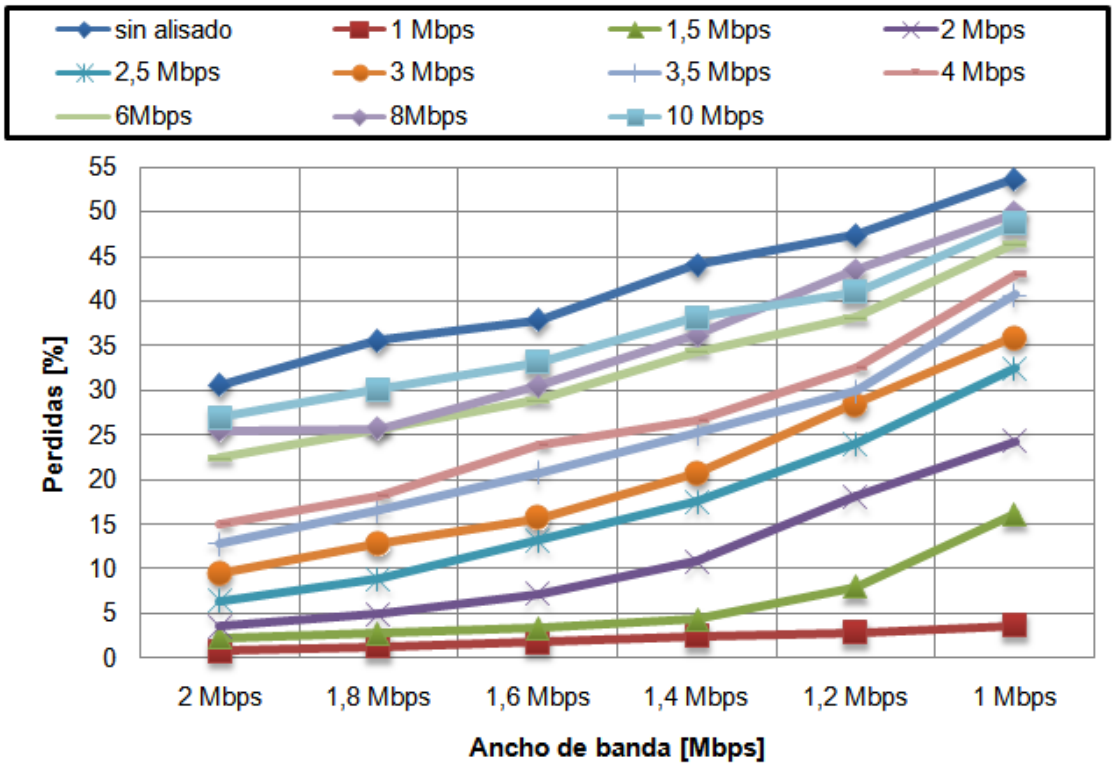

Figura #39 % Pérdidas de *bytes* BW entre nodos.

3.3.4 Efectos adversos del alisado

Como se observa de las figuras anteriores vemos que la técnica de alisado necesita introducir un gran retardo en la transmisión de paquetes para proporcionar perdidas mínimas en los mismos. Se llegar a producirse un alisado de tal valor que iguale el ancho de banda de transmisión en el cuello de botella para que el envío de datos no tenga un alto porcentaje de pérdidas. Así que este retardo juega en contra de las prestaciones y calidad que ha de ofrecer un flujo de información en tiempo real. Para algunas aplicaciones este retardo puede ser inasumible y generar problemas durante la comunicación.

En la Figura #40 podemos observar los retados medios que se introducen según el alisado que se utiliza en cada momento. Se observa que se introduce un retardo mayor en los alisados de menor ancho de banda. Esto ya que el retardo y el ancho de banda son inversamente proporcionales. Vemos que el retardo varía entre los 3,8 correspondiente al alisado de 1 Mbps, y los 0,25 ms del alisado a 10 Mbps. Se introduce mayor retardo entre paquetes cuanto menor es el ancho de banda de alisado utilizado.

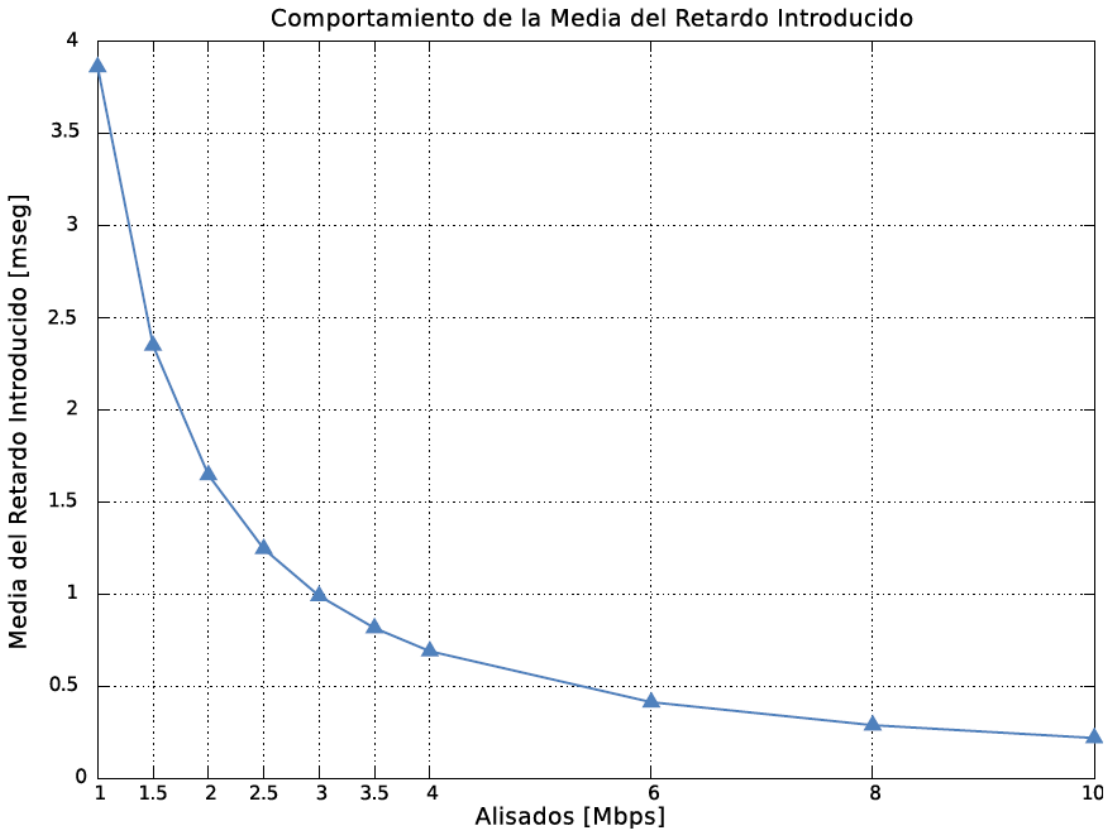

Figura #40 Medida del retardo medio introducido por cada alisado.

Vemos que el retardo medio introducido por el alisado que se ha utilizado para las pruebas no es excesivo, ya que existen estudios que han demostrado que Sopcast posee un *buffer* (en la aplicación) de 1 minuto [SMG+07] y por lo tanto se recomienda para este tipo de aplicaciones donde no afecta el retardo.

4 Conclusiones y líneas futuras

4.1 Conclusiones

En cuanto al grado de cumplimiento de los objetivos del proyecto se puede indicar que ha sido satisfactorio. Como principales conclusiones y hallazgos de este proyecto fin de carrera se pueden destacar los siguientes hitos.

Se ha llevado a cabo el estudió del tráfico de varios servicios multimedia mediante el uso de la herramienta de simulación NS-2. Se ha trabajado con trazas generadas con las librerías y opciones ya incluidas en el programa (FTP y CBR para el tráfico VoIP). También ha permitido el análisis del tráfico generado a partir de trazas P2PTV(Sopcast). En este aspecto se concluye la versatilidad de uso del *software* elegido para la generación y adaptación de flujos. Mediante el script utilizado se ha podido llevar a cabo un estudio pormenorizado de los parámetros que dan idea de la QoS.

Los escenarios utilizados han sido representativos para una red cableada. Se ha simulado posible congestión en la red de acceso con el cuello de botella donde se han realizado las mediciones gracias a la disputa entre los servicios multimedia para utilizar el ancho de banda disponible en cada momento

En la elección del *software* de simulación se ha optado por uno que está muy extendido en ambientes de estudio universitarios. Se ha optado por una opción de *software* libre. Para su instalación y puesta en marcha no ha habido apenas problemas como puede observarse en el Anexo D

En la realización de los scripts ha habido que seguir una serie de pautas para que las pruebas tuvieran valores finales lo más fiable posible. Se ha realizado un script que ejecuta pruebas iterativas de forma automática sobre los escenarios propuestos en condiciones interesantes al estudio.

Los casos a estudio que se han usado son los habituales en una red e conmutación de paquetes. Se intenta ofrecer servicio, con una capacidad determinada, a los distintos tipos de flujos de tráfico generados por todo tipo de aplicaciones. En este proyecto nos hemos centrado en las limitaciones en la red de acceso. Se han ido registrando la disputa de la capacidad de transmisión.

Mediante el uso del NS-2 se han podido recabar datos conocidos para la transmisión multimedia como son: el ancho de banda, el retardo y la tasa de pérdida de *bytes* y paquetes. Estos datos nos han aportado la base para el posterior estudio de optimización

La técnica utilizada para optimizar el uso de aplicaciones multimedia streaming ha sido el alisado de tráfico (*traffic smoothing*). Se han llegado a concluir las mejoras

que aporta a los parámetros de QoS. También se demuestra que introduce un retardo que puede ser perjudicial según sea el servicio al que se aplique esta técnica. Esto hay que tenerlo en cuenta, sobretodo, en aquellas aplicaciones que sean muy sensibles al retraso y sufran degradación muy rápidamente.

Se constata que aplicaciones comerciales en el ámbito del P2PTV que utilizan *buffer* internos que almacenan la información (de incluso minutos) son inviables para un uso fluido por parte del usuario, como por ejemplo en la retransmisión de eventos en tiempo real.

Es preciso aclarar que el presente estudio no pretende realizar una descripción exhaustiva de los métodos y técnicas utilizados para la realización de las pruebas. Se han presentado resultados medibles sobre parámetros de QoS cuando el tráfico de aplicaciones es volcado a la red simulada en distintas situaciones.

4.2 Líneas futuras

Hay que realizar un estudio y mejora de protocolos para la transmisión en tiempo real. Adaptar los requisitos respecto al uso de la red IP para tener una QoS adecuada y garantizada en los servicios multimedia del futuro.

Una posibilidad que se ha tratado de implementar durante el desarrollo del estudio ha sido la generación de escenarios de pruebas con redes inalámbricas. Ha resultado una tarea infructuosa dado que NS-2 no implementa las librerías y módulos para tal fin. Se ha investigado el uso de un escenario equivalente al cuello de botella en el interfaz aire entre dos *router,* que a su vez tuviesen clientes conectados por su interfaces de cable con servicios en tiempo real. Estos scripts de investigación han dado lugar a un escenario entre usuarios inalámbricos que se comunica mediante Ad-hoc. Al carecer NS-2 del modo infraestructura para implementar *router* se asume que no era viable llegar a una situación equivalente simulada mediante la red de cable y el cuello de botella. Este intento refleja las limitaciones de este *software* y deja abierta la opción de usar otros programas. Como posibles opciones están las herramientas de simulación analizadas en el apartado 2.4. Vemos que algunas soluciones son de pago y necesitan de la gestión de licencia para su uso.

A la hora de realizar el análisis de aplicaciones multimedia nos podríamos decantar por seguir el estudio de IPTV que no use P2P. Esta infraestructura es la desplegada por las ISP´s y que cada vez tiene un auge mayor. Se puede realizar el estudio mediante la captura de tráfico en los hogares con este servicio contratado . Una vez capturada las trazas se podría llevar a cabo un estudio parecido al de este PFC presentando distintas situaciones críticas con escenarios simulados.

La técnicas utilizadas para cumplir los objetivos de QoS en una comunicación multimedia en tiempo real hoy en día tienen mucho camino de mejora por recorrer. La tendencia alcista en cuanto al desarrollo de redes de comunicaciones a nivel mundial y la proliferación del uso de la fibra óptica presentan un escenario de futuro muy idóneo

para los servicios en tiempo real. Aumentará el ancho de banda de subida y bajada del usuario. La latencia de la red se verá reducida, la congestión será menor, el acceso a Internet será más global. El estudio que hemos realizado verá minimizado los efectos adversos que provocan una pérdida de la QoS.

En cuanto a otra forma de optimizar el uso de aplicaciones multimedia *streaming* está la posibilidad de analizar de manera más exhaustiva la técnica de multiplexación de paquetes. Estudiando la posibilidad de agrupar distintos flujos de información para mandar por la red un tráfico conformado según una serie de parámetros y comparando los resultados obtenidos.

BIBLIOGRAFÍA

[LGL08] Yong Liu, Yang Guo, Chao Liang, "A survey on peer-to-peer video streaming systems", Springer Science, Business Media, LLC, January 2008.

[FLK+08] Benny Fallica, Yue Lu, Fernando Kui, Rob Kooi and Piet Van Mieghem, "On the Quality of Experience of SopCast", Delft University of Technology, 2008.

[HSRV96] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson “RTP: A Transport Protocol for Real-Time Applications”, January 1996.

[ROH+06] Rosario Villarreal M.A., Herrera Vega F.; El estándar VoIP. Redes y servicios de banda ancha. 2006.

[MDNE04] M. Baugher, D. McGrew, M. Naslund, E. Carrara and K. Norrman “The Secure Real-time Transport Protocol (SRTP)”, March 2004.

[DOHE04] Doherty S.; The Survivor's Guide to 2004: Converged Voice, Video and Data, 2003.

[RFC1889] H. Schulzrinne, S. Casner, R. Frederick,V. Jacobson "RTP: A Transport Protocol for Real-Time Applications", January 1996.

[TBP+11] B. Trammell, E. Boschi, G. Procissi, C. Callegari, P. Dorfinge, and D. "Schatzmann. Identifying Skype traffic in a large-scale flow data repository". TMA LNCS, 2011.

[ACG+09] D. Adami, C Callegari, S. Giordano, M. Pagano, and T. Pepe. "A real-time algorithm for skype traffic detection an classification". NEW2AN/ruSMART LNCS, 2009.

[WIK14] http://wikitel.info/wiki/UA-Televisi%C3%B3n_P2P_(peer_to_peer)
Última visita 12/11/2014

[SA+09] Alberto Los Santos Aransay "Estado del arte en IPTV" UNIVERSIDAD DE VIGO: MULTIMEDIA E INTERNET June 2009.

[BSU+98] M. S. Borella, D. Swider, S. Uludag, G. B. Brewster, “Internet Packet Loss: Measurement and Implications for End-to-End QoS,” Parallel Processing Workshops, International Conference on, p. 3, International Conference on Parallel Processing Workshops (ICPPW'98), 1998.

[NS-2] The University of Southern California´s Information Sciences Institute, The Network Simulator NS-2. http://www.isi.edu/nsnam/ns/ Última visita 10/11/2014

[BK10] Biernacki, A., Krieger, U.R.: Session Level Analysis of P2P Television Traces. In: S. Zeadally et al. (eds.), Proc. 3rd International Workshop on Future Multimedia Networking, Krakow, June 17-18, 2010.

[SCH95] Schulzrinne, H., "Internet Services: from Electronic Mail to Real-Time Multimedia", KIVS'95, February 1995.

[DCG12] D. Corral "Evaluación de la influencia del muestreo y de la pérdida de paquetes sobre la detección de tráfico Skype" Universidad Autónoma de Madrid, November 2012,

[LVK+00] Leiner, Barry M., Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. "A Brief History of the Internet." Internet Society. April 19, 2000.

[SFS14] L. Sequeira, J. Fernández-Navajas and Jose Saldana, "The Effect of the Buffer Size in QoS for Multimedia and bursty Traffic: When an Upgrade Becomes a Downgrade" KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS VOL. 8, NO. 9, Sep. 2014.

[BS04] Salman A. Baset y Henning Schulzrinne. “An Analysis of the Skype Peer to Peer Internet Telephony Protocol”. Columbia University. 15 September 2004.

[XZHG07] Y. Xiao, X.Du, J. Zhang, F. Hu and S. Guizani ,“Internet Protocol Television (IPTV): The Killer Application for the Next-Generation Internet”. IEEE Communications Magazine, November 2007.

[WWCH08] Y. J. Won, J. Won-Ki, M. Choi, C. Hwang and J. Yoo, “Measurement of Download and Play and Streaming IPTV Traffic”. IEEE Communications Magazine, October 2008.

[RFC793] J. Postel, “Transmission Control Protocol,” RFC 793, Sept. 1981.

[RFC768] J. Postel, “User Datagram Protocol,” RFC 768, Aug. 1980.

[CR00] R.G. Cole, J.H. Rosenbluth, “Voice over IP performance monitoring,” SIGCOMM Comput. Commun. Rev. 31, 2 (abr. 2001), pp. 9-24.

[KLS98] Vijay P. Kumar, T. V. Lakshman, and Dimitrios Stiliadis "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow‘s Internet" Bell Laboratories, Lucent Technologies. IEER May 1998.

[UIT80] Recommendation UIT-T E.800

[JFC+96] Jacobsen V., Fredrick R., Casner S., and Schulzrinne H., “RTP: A transport Protocol for Real-Time Applications”. RFC 1889. Lawrence Berkeley National Laboratory, Xerox PARC, Precept Software Inc., GMD Fokus, January 1996.

[SFNS13] Luis Sequeira, Julian Fernández-Navajas, Jose Saldana, "Characterization of the Buffer in Real Internet Paths," Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems SPECTS 2013, Toronto, Canada, July 2013, pp 666-671. IEEEXplore. ISBN 1-56555-352-7.

[VKB+95] A. Vogel, B. Kerhervé, G. von Bochmann and J. Gecsei, “Distributed Multimedia and QoS: A Survey”, IEEE Multimedia, pp. 10 – 18, 1995.

[ITU96] International Telecommunication Union (ITU-T Recommendation G.114), “One-way transmission time,” feb.1996.

[BCS94] R.Braden, D.Clark, S.Shenker, “Integrated services in the Internet Architecture : An Overview”. Internet RFC 1633. July 1994.

[SMG+07] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, “Will iptv ride the peer-to-peer stream?” Communications Magazine, IEEE, vol. 45, no. 6, pp. 86–92, 2007.

[CK74] V G. CERF and R. E. KAHN, "A Protocol for Packet Network Intercommunication" IEEE, Trans on Comms, Vol Com-22, No 5 May 1974.

ANEXO A. Acrónimos

ADSL	Asymmetric Digital Subscriber Line
ARPANET	Advanced Research Projects Agency Network
ATM	Asynchronous Transfer Mode
CBR	Constant Bit Rate
CPU	Central Processing Unit
DRR	Deficit Round Robin
FIFO	First Input First Output
FTTH	Fiber-To-The-Home
HTTP	HyperText Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPTV	Internet Protocol Televisión
ISP	Internet Service Provider
ITU	International Telecommunication Union
LAN	Local Area Network
MTU	Maximun Transfer Unit
OWD	One Way Delay
P2P	Peer to Peer
P2PTV	Peer to Peer Television
PSTN	Public Switched Telephone Network
QoE	Quality of Experience
QoS	Quality of Service
RFC	Request For Comments
RSVP	ReSerVation Protocol
RTC	Real-time communications
RTCP	Real-time Control Protocol
RTP	Real-time Transport Protocol
RTT	Round Trip Time
SRTP	Secure Real-time Transport Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator

VAD	Voice Activity Detection
VINT	Virtual InterNetwork Testbed
VoD	Video on demand
VoIP	Voice over Internet Protocol

ANEXO B. Scripts

Escenario completo de la simulación.(scenario.tcl)

Se describen todos los elementos del escenario. Se crean los flujos de datos entre extremos de la comunicación y se parametriza la red para ir modificando datos en ella.

scenario.tcl

```
# Create scheduler
set ns [new Simulator]
set max_starttime 1.0
set min_starttime 0.0
set finishtime 21.0
set monitor_time 1
set bw_sending 100Mb
set bw_receiving 100Mb
set delay 0ms
set buffer_sending 500
set buffer_receiving 500
set bw_bottleneck 6Mb
set delay_bottleneck 0ms
set buffer_bottleneck 50
exec echo "$max_starttime" > finishtime.txt

# See headers activated
# foreach cl [PacketHeader info subclass] {
#         puts $cl
#     }

# create output files
# set tracefile [open out.tr w]
# $ns trace-all $tracefile
# set namfile [open out.nam w]
```

```
# $ns namtrace-all $namfile

# Creating topology
set num_node 16
for {set i 0} {$i < $num_node} {incr i} {
        set n($i) [$ns node]
}
$ns set src_ [list 7 3 4 5 6 12 14]
$ns set dst_ [list 2 8 9 10 11 13 15]
# Bottleneck
$ns simplex-link $n(0) $n(1) $bw_bottleneck $delay_bottleneck DropTail
$ns queue-limit $n(0) $n(1) $buffer_bottleneck
$ns simplex-link $n(1) $n(0) $bw_bottleneck $delay DropTail
$ns queue-limit $n(1) $n(0) $buffer_bottleneck
# Rest of nodes
# Sending nodes
$ns duplex-link $n(0) $n(2) $bw_sending $delay DropTail
$ns queue-limit $n(0) $n(2) $buffer_sending
$ns queue-limit $n(2) $n(0) $buffer_sending
$ns duplex-link $n(0) $n(3) $bw_sending $delay DropTail
$ns queue-limit $n(0) $n(3) $buffer_sending
$ns queue-limit $n(3) $n(0) $buffer_sending
$ns duplex-link $n(0) $n(4) $bw_sending $delay DropTail
$ns queue-limit $n(0) $n(4) $buffer_sending
$ns queue-limit $n(4) $n(0) $buffer_sending
$ns duplex-link $n(0) $n(5) $bw_sending $delay DropTail
$ns queue-limit $n(0) $n(5) $buffer_sending
$ns queue-limit $n(5) $n(0) $buffer_sending
$ns duplex-link $n(0) $n(6) $bw_sending $delay DropTail
$ns queue-limit $n(0) $n(6) $buffer_sending
$ns queue-limit $n(6) $n(0) $buffer_sending
$ns duplex-link $n(0) $n(12) $bw_sending $delay DropTail
$ns queue-limit $n(0) $n(12) $buffer_sending
$ns queue-limit $n(12) $n(0) $buffer_sending
$ns duplex-link $n(0) $n(14) $bw_sending $delay DropTail
```

```
$ns queue-limit $n(0) $n(14) $buffer_sending
$ns queue-limit $n(14) $n(0) $buffer_sending
# receiving nodes
$ns duplex-link $n(1) $n(7) $bw_receiving $delay DropTail
$ns queue-limit $n(1) $n(7) $buffer_receiving
$ns queue-limit $n(7) $n(1) $buffer_receiving
$ns duplex-link $n(1) $n(8) $bw_receiving $delay DropTail
$ns queue-limit $n(1) $n(8) $buffer_receiving
$ns queue-limit $n(8) $n(1) $buffer_receiving
$ns duplex-link $n(1) $n(9) $bw_receiving $delay DropTail
$ns queue-limit $n(1) $n(9) $buffer_receiving
$ns queue-limit $n(9) $n(1) $buffer_receiving
$ns duplex-link $n(1) $n(10) $bw_receiving $delay DropTail
$ns queue-limit $n(1) $n(10) $buffer_receiving
$ns queue-limit $n(10) $n(1) $buffer_receiving
$ns duplex-link $n(1) $n(11) $bw_receiving $delay DropTail
$ns queue-limit $n(1) $n(11) $buffer_receiving
$ns queue-limit $n(11) $n(1) $buffer_receiving
$ns duplex-link $n(1) $n(13) $bw_receiving $delay DropTail
$ns queue-limit $n(1) $n(13) $buffer_receiving
$ns queue-limit $n(13) $n(1) $buffer_receiving
$ns duplex-link $n(1) $n(15) $bw_receiving $delay DropTail
$ns queue-limit $n(1) $n(15) $buffer_receiving
$ns queue-limit $n(15) $n(1) $buffer_receiving
# Node positions
$ns duplex-link-op $n(0) $n(1) orient left
$ns duplex-link-op $n(0) $n(2) orient up
$ns duplex-link-op $n(0) $n(3) orient right-up
$ns duplex-link-op $n(0) $n(4) orient right
$ns duplex-link-op $n(0) $n(5) orient right-down
$ns duplex-link-op $n(0) $n(6) orient down
$ns duplex-link-op $n(1) $n(7) orient up
$ns duplex-link-op $n(1) $n(8) orient left-up
$ns duplex-link-op $n(1) $n(9) orient left
$ns duplex-link-op $n(1) $n(10) orient left-down
```

```
$ns duplex-link-op $n(1) $n(11) orient down
$ns duplex-link-op $n(0) $n(12) orient left-up
$ns duplex-link-op $n(0) $n(14) orient left-down
$ns duplex-link-op $n(1) $n(13) orient right-up
$ns duplex-link-op $n(1) $n(15) orient right-down
# Node shape
$n(0) shape hexagon
$n(1) shape hexagon
foreach s [$ns set src_] {
        $n($s) shape circle
}
foreach s [$ns set dst_] {
        $n($s) shape circle
}
# Node colors
$n(0) color black
$n(1) color black
$n(2) color green
$n(7) color green
$n(3) color yellow
$n(8) color yellow
$n(4) color blue
$n(9) color blue
$n(5) color red
$n(10) color red
$n(6) color gray
$n(11) color gray
$n(12) color purple
$n(13) color purple
$n(14) color maroon
$n(15) color maroon
# Flow colors
$ns color 0 green
$ns color 1 yellow
$ns color 2 blue
```

```
$ns color 3 red
$ns color 4 gray
$ns color 5 purple
$ns color 6 maroon

# Queue monitor
# Uplink
set monitor01 [open qm01.out w]
set qm01 [$ns monitor-queue $n(0) $n(1) $monitor01 $monitor_time]
[$ns link $n(0) $n(1)] queue-sample-timeout
set monitor20 [open qm20.out w]
set qm20 [$ns monitor-queue $n(2) $n(0) $monitor20 $monitor_time]
[$ns link $n(2) $n(0)] queue-sample-timeout
set monitor30 [open qm30.out w]
set qm30 [$ns monitor-queue $n(3) $n(0) $monitor30 $monitor_time]
[$ns link $n(3) $n(0)] queue-sample-timeout
set monitor40 [open qm40.out w]
set qm40 [$ns monitor-queue $n(4) $n(0) $monitor40 $monitor_time]
[$ns link $n(4) $n(0)] queue-sample-timeout
set monitor50 [open qm50.out w]
set qm50 [$ns monitor-queue $n(5) $n(0) $monitor50 $monitor_time]
[$ns link $n(5) $n(0)] queue-sample-timeout
set monitor60 [open qm60.out w]
set qm60 [$ns monitor-queue $n(6) $n(0) $monitor60 $monitor_time]
[$ns link $n(6) $n(0)] queue-sample-timeout
set monitor17 [open qm17.out w]
set qm17 [$ns monitor-queue $n(1) $n(7) $monitor17 $monitor_time]
[$ns link $n(1) $n(7)] queue-sample-timeout
set monitor18 [open qm18.out w]
set qm18 [$ns monitor-queue $n(1) $n(8) $monitor18 $monitor_time]
[$ns link $n(1) $n(8)] queue-sample-timeout
set monitor19 [open qm19.out w]
set qm19 [$ns monitor-queue $n(1) $n(9) $monitor19 $monitor_time]
[$ns link $n(1) $n(9)] queue-sample-timeout
set monitor110 [open qm110.out w]
```

```
set qm110 [$ns monitor-queue $n(1) $n(10) $monitor110 $monitor_time]
[$ns link $n(1) $n(10)] queue-sample-timeout
set monitor1_11 [open qm1_11.out w]
set qm1_11 [$ns monitor-queue $n(1) $n(11) $monitor1_11 $monitor_time]
[$ns link $n(1) $n(11)] queue-sample-timeout
set monitor120 [open qm120.out w]
set qm120 [$ns monitor-queue $n(12) $n(0) $monitor120 $monitor_time]
[$ns link $n(12) $n(0)] queue-sample-timeout
set monitor140 [open qm140.out w]
set qm140 [$ns monitor-queue $n(14) $n(0) $monitor140 $monitor_time]
[$ns link $n(14) $n(0)] queue-sample-timeout
set monitor113 [open qm113.out w]
set qm113 [$ns monitor-queue $n(1) $n(13) $monitor113 $monitor_time]
[$ns link $n(1) $n(13)] queue-sample-timeout
set monitor115 [open qm115.out w]
set qm115 [$ns monitor-queue $n(1) $n(15) $monitor115 $monitor_time]
[$ns link $n(1) $n(15)] queue-sample-timeout
# Downlink
set monitor10 [open qm10.out w]
set qm10 [$ns monitor-queue $n(1) $n(0) $monitor10 $monitor_time]
[$ns link $n(1) $n(0)] queue-sample-timeout
set monitor71 [open qm71.out w]
set qm71 [$ns monitor-queue $n(7) $n(1) $monitor71 $monitor_time]
[$ns link $n(7) $n(1)] queue-sample-timeout
set monitor81 [open qm81.out w]
set qm81 [$ns monitor-queue $n(8) $n(1) $monitor81 $monitor_time]
[$ns link $n(8) $n(1)] queue-sample-timeout
set monitor91 [open qm91.out w]
set qm91 [$ns monitor-queue $n(9) $n(1) $monitor91 $monitor_time]
[$ns link $n(9) $n(1)] queue-sample-timeout
set monitor101 [open qm101.out w]
set qm101 [$ns monitor-queue $n(10) $n(1) $monitor101 $monitor_time]
[$ns link $n(10) $n(1)] queue-sample-timeout
set monitor11_1 [open qm11_1.out w]
set qm11_1 [$ns monitor-queue $n(11) $n(1) $monitor11_1 $monitor_time]
```

```
[$ns link $n(11) $n(1)] queue-sample-timeout
set monitor02 [open qm02.out w]
set qm02 [$ns monitor-queue $n(0) $n(2) $monitor02 $monitor_time]
[$ns link $n(0) $n(2)] queue-sample-timeout
set monitor03 [open qm03.out w]
set qm03 [$ns monitor-queue $n(0) $n(3) $monitor03 $monitor_time]
[$ns link $n(0) $n(3)] queue-sample-timeout
set monitor04 [open qm04.out w]
set qm04 [$ns monitor-queue $n(0) $n(4) $monitor04 $monitor_time]
[$ns link $n(0) $n(4)] queue-sample-timeout
set monitor05 [open qm05.out w]
set qm05 [$ns monitor-queue $n(0) $n(5) $monitor05 $monitor_time]
[$ns link $n(0) $n(5)] queue-sample-timeout
set monitor06 [open qm06.out w]
set qm06 [$ns monitor-queue $n(0) $n(6) $monitor06 $monitor_time]
[$ns link $n(0) $n(6)] queue-sample-timeout
set monitor012 [open qm012.out w]
set qm012 [$ns monitor-queue $n(0) $n(12) $monitor012 $monitor_time]
[$ns link $n(0) $n(12)] queue-sample-timeout
set monitor014 [open qm014.out w]
set qm014 [$ns monitor-queue $n(0) $n(14) $monitor014 $monitor_time]
[$ns link $n(0) $n(14)] queue-sample-timeout
set monitor131 [open qm131.out w]
set qm131 [$ns monitor-queue $n(13) $n(1) $monitor131 $monitor_time]
[$ns link $n(13) $n(1)] queue-sample-timeout
set monitor151 [open qm151.out w]
set qm151 [$ns monitor-queue $n(15) $n(1) $monitor151 $monitor_time]
[$ns link $n(15) $n(1)] queue-sample-timeout

# Traffic used
# Web traffic
# Setup PackMime
Agent/TCP/FullTcp set segsize_ 1500
set web_traffic [new PackMimeHTTP]
$web_traffic set-client $n(2)
```

```
$web_traffic set-server $n(7)
$web_traffic set-rate 2
$web_traffic set-http-1.1
# FTP traffic
set tcp_ftp [new Agent/TCP/Sack1]
set tcpsink_ftp [new Agent/TCPSink]
$ns attach-agent $n(3) $tcp_ftp
$ns attach-agent $n(8) $tcpsink_ftp
$ns connect $tcp_ftp $tcpsink_ftp
set ftp_traffic [new Application/FTP]
$ftp_traffic attach-agent $tcp_ftp
$tcp_ftp set fid_ 1
# Streaming traffic
set udp_streaming [new Agent/UDP]
set udpnull_streaming [new Agent/Null]
$ns attach-agent $n(4) $udp_streaming
$ns attach-agent $n(9) $udpnull_streaming
$udp_streaming set packetSize_ 1500
$ns connect $udp_streaming $udpnull_streaming
$udp_streaming set fid_ 2
set tracefile_streaming [new Tracefile]
$tracefile_streaming filename traffic/sopcast_upload_all-IPs.txt.if-0.bin
set streaming_traffic [new Application/Traffic/Trace]
$streaming_traffic attach-tracefile $tracefile_streaming
$streaming_traffic attach-agent $udp_streaming
# Game traffic
set udp_game [new Agent/UDP]
set udpnull_game [new Agent/Null]
$ns attach-agent $n(5) $udp_game
$ns attach-agent $n(10) $udpnull_game
$udp_game set packetSize_ 1500
$ns connect $udp_game $udpnull_game
$udp_game set fid_ 3
set tracefile_game [new Tracefile]
$tracefile_game filename traffic/p2p_complete_fixed.txt.if-0.bin
```

```
set game_traffic [new Application/Traffic/Trace]
$game_traffic attach-tracefile $tracefile_game
$game_traffic attach-agent $udp_game
# VoIP traffic
set udp_voip [new Agent/UDP]
set udpnull_voip [new Agent/Null]
$ns attach-agent $n(6) $udp_voip
$ns attach-agent $n(11) $udpnull_voip
$ns connect $udp_voip $udpnull_voip
$udp_voip set fid_ 4
$udp_voip set packetSize_ 1500
set voip_traffic [new Application/Traffic/CBR]
$voip_traffic set packetSize_ 60
$voip_traffic set interval_ 0.02
$voip_traffic attach-agent $udp_voip
# VoIP traffic
set udp_voip_1 [new Agent/UDP]
set udpnull_voip_1 [new Agent/Null]
$ns attach-agent $n(12) $udp_voip_1
$ns attach-agent $n(13) $udpnull_voip_1
$ns connect $udp_voip_1 $udpnull_voip_1
$udp_voip_1 set fid_ 5
$udp_voip_1 set packetSize_ 1500
set voip_traffic_1 [new Application/Traffic/CBR]
$voip_traffic_1 set packetSize_ 60
$voip_traffic_1 set interval_ 0.02
$voip_traffic_1 attach-agent $udp_voip_1
# Videoconferencing traffic
set udp_videoconferencing [new Agent/UDP]
set udpnull_videoconferencing [new Agent/Null]
$ns attach-agent $n(14) $udp_videoconferencing
$ns attach-agent $n(15) $udpnull_videoconferencing
$udp_videoconferencing set packetSize_ 1500
$ns connect $udp_videoconferencing $udpnull_videoconferencing
$udp_videoconferencing set fid_ 6
```

```
set tracefile_videoconferencing [new Tracefile]
$tracefile_videoconferencing filename traffic/c2MBRes800x450RugbybigPc3serv_2_360s.txt.if-0.bin
set videoconferencing_traffic [new Application/Traffic/Trace]
$videoconferencing_traffic attach-tracefile $tracefile_videoconferencing
$videoconferencing_traffic attach-agent $udp_videoconferencing


# Close files
proc finish {} {
        global ns tracefile namfile monitor01 monitor02 monitor03 monitor04 monitor05 monitor06 monitor20 monitor30 monitor40 monitor50 monitor60 monitor10 monitor17 monitor18 monitor19 monitor110 monitor1_11 monitor71 monitor81 monitor91 monitor101 monitor11_1 monitor120 monitor140 monitor113 monitor115 monitor131 monitor151 monitor012 monitor014 web_traffic
        $ns flush-trace
#       close $tracefile
#       close $namfile
        close $monitor01
        close $monitor10
        close $monitor02
        close $monitor03
        close $monitor04
        close $monitor05
        close $monitor06
        close $monitor17
        close $monitor18
        close $monitor19
        close $monitor110
        close $monitor1_11
        close $monitor20
        close $monitor30
        close $monitor40
        close $monitor50
        close $monitor60
        close $monitor71
        close $monitor81
        close $monitor91
```

```
        close $monitor101
        close $monitor11_1
        close $monitor120
        close $monitor140
        close $monitor113
        close $monitor115
        close $monitor131
        close $monitor151
        close $monitor012
        close $monitor014
        delete $web_traffic
#       exec nam out.nam &
        exit 0
}

# amination rate
$ns at $min_starttime "$ns set-animation-rate 0.1ms"

# scheduling events
set flows [expr ($num_node-2)/2]
$defaultRNG seed 0
set rand [new RandomVariable/Uniform]
$rand set max_ $max_starttime
$rand set min_ $min_starttime
for {set i 0} {$i < $flows} {incr i} {
        set starttime($i) [$rand value]
}
# for {set i 0} {$i < $flows} {incr i} {
#       puts $starttime($i)
# }
# $ns at $starttime(0) "$web_traffic start"
# $ns at $starttime(1) "$ftp_traffic start"
$ns at $starttime(2) "$streaming_traffic start"
# $ns at $starttime(3) "$game_traffic start"
# $ns at $starttime(4) "$voip_traffic start"
```

```
# $ns at $starttime(5) "$voip_traffic_1 start"
# $ns at $starttime(6) "$videoconferencing_traffic start"
# $ns at $finishtime "$web_traffic stop"
# $ns at $finishtime "$ftp_traffic stop"
$ns at $finishtime "$streaming_traffic stop"
# $ns at $finishtime "$game_traffic stop"
# $ns at $finishtime "$voip_traffic stop"
# $ns at $finishtime "$voip_traffic_1 stop"
# $ns at $finishtime "$videoconferencing_traffic stop"
$ns at [expr $finishtime+$max_starttime] "finish"
$ns run
```

Iteración de la simulación del escenario.(ns-2-iterator)

Realiza el número indicado de iteraciones de la simulación indicada como fichero .tcl y se procesa la información de salida por cada flujo. Como parámetros de entrada se indica el fichero .tcl , número de iteraciones para recopilar una media sobre los datos y la carpeta de salida para dejar los datos.

```
#!/bin/bash
read -p "Enter NS-2 file for simulation: " NS-2_file
read -p "Enter folder name: " folder
read -p "Enter number of iterations: " iteration
echo "Preparing files"
mkdir $folder
mkdir $folder/$folder.traces
mkdir $folder/$folder.csv
mkdir $folder/$folder.resume
cp $NS-2_file $folder
cp -r traffic $folder/traffic
cd $folder
for ((i=0;i<$iteration;i++))
        do
                echo "Simulating for $i of $iteration iterations"
                ns $NS-2_file
                echo "Copying traces for $i of $iteration iteration"
```

```
                for x in `ls *.out`
                        do
                                cp $x $folder.traces/$folder.$i.$x
                                cp $x $folder.resume/$folder.$i.$x
                done
#               comment the next two lines if you are not using out.tr/out.nam in the ns file
#               cp out.tr $folder.traces/$folder.$i.out.tr
#               cp out.nam $folder.traces/$folder.$i.out.nam
                echo "Copying csv for $i of $iteration iteration"
                for x in `ls qm*.out`
                        do
                                cp $x $folder.csv/$folder.$i.$x.csv
                                sed -i "s/ /,/g" $folder.csv/$folder.$i.$x.csv
                done
                echo "Cleaning for $i of $iteration iteration"
#               comment the next three lines if you are not using out.tr/out.nam in the ns file
#               cp out.tr $folder.traces/$folder.$i.out.tr
#               cp out.nam $folder.traces/$folder.$i.out.nam
#               rm *.out *.tr *.nam
done
cp finishtime.txt $folder.resume/finishtime.txt
rm finishtime.txt
rm -r traffic
echo "Simulation $i of $iteration finished"
echo "Calculating resume for all files"
cd $folder.resume
for x in `ls *.out`
        do
                awk '
                        BEGIN{
                                go=0
                                getline finish < "finishtime.txt"
                        }
                        {
                                if ($1>finish && go==0){
```

```
                                        dropped_bytes_begin=$11
                                        dropped_packets_begin=$8
                                        arrived_bytes_begin=$9
                                        arrived_packets_begin=$6
                                        departed_bytes_begin=$10
                                        departed_packets_begin=$7
                                        time_begin=$1
                                        go++
                                }
                                if ($1>finish && go>0){
                                        dropped_bytes=$11
                                        dropped_packets=$8
                                        arrived_bytes=$9
                                        arrived_packets=$6
                                        departed_bytes=$10
                                        departed_packets=$7
                                        time=$1
                                }
                        }
                        END{
#                               printf "dropped_bytes" " " "%" " " "dropped_packets" " " "%" " "
"arrived_bytes" " " "arrived_packets" " " "arrived_bandwidth" " " "departed_bandwidth" "\n"
                                if (arrived_bytes!=0 && arrived_packets!=0){
                                        printf dropped_bytes-dropped_bytes_begin " "
100*(dropped_bytes-dropped_bytes_begin)/(arrived_bytes-arrived_bytes_begin) " "
dropped_packets-dropped_packets_begin " " 100*(dropped_packets-
dropped_packets_begin)/(arrived_packets-arrived_packets_begin) " " arrived_bytes-
arrived_bytes_begin " " arrived_packets-arrived_packets_begin " " (arrived_bytes-
arrived_bytes_begin)*8/(time-time_begin) " " (departed_bytes-departed_bytes_begin)*8/(time-
time_begin) "\n"
                                }
                                        else{
                                                printf dropped_bytes-dropped_bytes_begin " " 0
" " dropped_packets-dropped_packets_begin " " 0 " " arrived_bytes-arrived_bytes_begin " "
arrived_packets-arrived_packets_begin " " (arrived_bytes-arrived_bytes_begin)*8/(time-
time_begin) " " (departed_bytes-departed_bytes_begin)*8/(time-time_begin) "\n"
                                }
                        }
```

```
                ' $x > $x.resume
                echo "File $x processed"
done
rm finishtime.txt
echo "Calculating total resume"
for x in qm01 qm10 qm20 qm02 qm30 qm03 qm40 qm04 qm50 qm05 qm60 qm06 qm120 qm012
qm140 qm014 qm17 qm71 qm18 qm81 qm19 qm91 qm110 qm101 qm1_11 qm11_1 qm131
qm113 qm151 qm115
        do
                for y in `ls *.*.$x.*.resume`
                        do
                                cat $y >> $x.txt
                done
done
for x in `ls *.txt`
        do
                awk '
                        BEGIN{
                                dropped_bytes=0
                                dropped_packets=0
                                arrived_bytes=0
                                arrived_packets=0
                                arrived_bw=0
                                departed_bw=0
                                i=0
                        }
                        {
                                if (NF!=0){
                                        dropped_bytes=dropped_bytes+$1
                                        dropped_packets=dropped_packets+$3
                                        arrived_bytes=arrived_bytes+$5
                                        arrived_packets=arrived_packets+$6
                                        arrived_bw=arrived_bw+$7
                                        departed_bw=departed_bw+$8
                                        i++
                                }
```

```
                        }
                        END{
                                if (NF!=0 && arrived_bytes!=0 && arrived_packets!=0){
                                        printf dropped_bytes/i " "
100*(dropped_bytes/i)/(arrived_bytes/i) " " dropped_packets/i " "
100*(dropped_packets/i)/(arrived_packets/i) " " arrived_bytes/i " " arrived_packets/i " "
arrived_bw/i " " departed_bw/i "\n"
                                }
                        }
                ' $x > $x.total
done
src_uplink=(qm20 qm30 qm40 qm50 qm60 qm120 qm140)
dst_uplink=(qm17 qm18 qm19 qm110 qm1_11 qm113 qm115)
src_downlink=(qm71 qm81 qm91 qm101 qm11_1 qm131 qm151)
dst_downlink=(qm02 qm03 qm04 qm05 qm06 qm012 qm014)
for x in 0 1 2 3 4 5 6
        do
                cat "${dst_uplink[$x]}.txt.total" > file.file
                awk '
                        BEGIN{
                                getline out < "file.file"
                                split (out,line," ")
                        }
                        {
                                if (NF!=0 && $5!=0 && $6!=0){
                                        printf ($5-$1)-line[5] " " 100*((($5-$1)-line[5])/($5-$1)) " "
($6-$3)-line[6] " " 100*((($6-$3)-line[6])/($6-$3)) " " $5-$1 " " $6-$3 " " $7 " " line[8] "\n"
                                }
                                else{
                                        printf ($5-$1)-line[5] " " 0 " " ($6-$3)-line[6] " " 0 " " $5-$1
" " $6-$3 " " $7 " " line[8] "\n"
                                }
                        }
                ' ${src_uplink[$x]}.txt.total > flow.$x.uplink.total
done
for x in 0 1 2 3 4 5 6
        do
```

```
		cat "${dst_downlink[$x]}.txt.total" > file.file
		awk '
			BEGIN{
				getline out < "file.file"
				split(out,line," ")
			}
			{
				if (NF!=0 && $5!=0 && $6!=0){
				printf ($5-$1)-line[5] " " 100*((($5-$1)-line[5])/($5-$1)) " " ($6-$3)-
line[6] " " 100*((($6-$3)-line[6])/($6-$3)) " " $5-$1 " " $6-$3 " " $7 " " line[8] "\n"
				}
				else{
					printf ($5-$1)-line[5] " " 0 " " ($6-$3)-line[6] " " 0 " " $5-$1
" " $6-$3 " " $7 " " line[8] "\n"
				}
			}
		' ${src_downlink[$x]}.txt.total > flow.$x.downlink.total
done
echo "Cleaning"
rm *.out
rm file.file
echo "Displaying information"
for x in `ls *.resume`
	do
		if [ -s "$x" ]
			then
				echo "$x"
				echo "Dropped_bytes "%" Dropped_packets "%" Arrived_bytes
Arrived_packets Arrived_bw Departed_bw"
				cat $x
				echo " "
		fi
done
for x in `ls qm*.total`
	do
		if [ -s "$x" ]
```

```
                then
                        echo "$x"
                        echo "Dropped_bytes "%" Dropped_packets "%" Arrived_bytes
Arrived_packets Arrived_bw Departed_bw"
                        cat $x
                        echo " "
        fi
done
for x in `ls flow*.total`
    do
        if [ -s "$x" ]
                then
                        echo "$x"
                        echo "Dropped_bytes "%" Dropped_packets "%" Arrived_bytes
Arrived_packets Arrived_bw Departed_bw"
                        cat $x
                        echo " "
        fi
done
for x in `ls *.*`
    do
        echo "Dropped_bytes % Dropped_packets % Arrived_bytes Arrived_packets
Arrived_bandwidth Departed_bandwidth" > file.file
        cat $x >> file.file
        cp file.file $x
done
rm file.file
cd ..
echo "Winning some space"
# comment the lines that you need
# rm $folder.resume/*.resume
rm -r $folder.traces
rm -r $folder.csv
rm *.out
```

ANEXO C. Arquitectura de sistemas de *streaming* P2P

Los sistemas de *streaming* P2P se pueden clasificar en dos categorías en función de la estructura de la red:

- **Estructura en basada en árbol:** La estructura basada en árbol posee una estructura superpuesta bien organizada y suelen distribuir vídeo por *push*(empujando) de los datos de un *peer*(compañero) a sus *peers* hijos. Una de las desventajas más importantes de los sistemas de transmisión basados en árboles es su vulnerabilidad a la rotación de pares(*peer churn*). La salida de un compañero de la estructura interrumpirá temporalmente la entrega de vídeo a todos los compañeros en el subárbol con raíz en el compañero que desaparece.
- **Estructura basada en mallado:** En el sistema de mallado para realizar *streaming*, los compañeros no se limitan a una topología estática. En cambio, las relaciones entre los compañeros o pares de la red mallada se establecen/terminan basándose en el contenido disponible, capacidad y disponibilidad de ancho de banda entre aquellos. Un *peer* se conecta dinámicamente a un subconjunto de *peers* aleatorios en el sistema. Peers que intercambian periódicamente información acerca de su disponibilidad de datos. El contenido de video es recibido por un *peer* a través sus *peers* vecinos, que ya han obtenido el contenido previamente. Dado que se mantienen múltiples vecinos en un momento dado, sistemas de *streaming* de video basado en mallado son muy robustos y evitan rotación de pares(*peer churn*). Sin embargo, las relaciones de interconexión dinámicas hacen que la eficiencia en la distribución de video sea impredecible. Paquetes diferentes de datos pueden recorrer diferentes rutas para los usuarios. En consecuencia, los usuarios pueden sufrir de una degradación de calidad en la reproducción de vídeo, que van desde bajas velocidades de bits en el vídeo, el arranque lento por retraso, a parones frecuentes en la reproducción.

El P2P *live streaming* según el artículo [LGL08] puede clasificar en dos categorías:

- **1) *Live* o en vivo**

En una sesión de *live streaming*, contenido de video en *streaming* es diseminado a todos los usuarios en tiempo real. El video se reproduce de manera sincronizada en todos los usuarios. Existen distintas estructuras para la reproducción:

A) *Tree-based systems* (sistemas basados en árbol)

Los servidores de vídeo y usuarios forman una superposición de redes a nivel de aplicación para distribuir contenido de vídeo.

A.1) *Single-tree streaming* (árbol único)

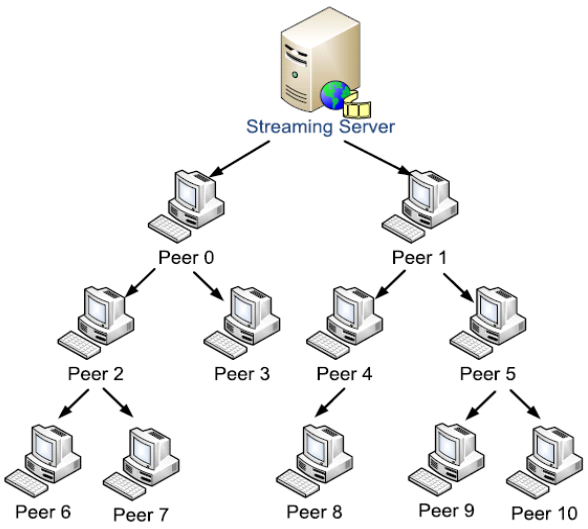

Figura #41 Capa de aplicación árbol *multicast*.

Una situación crítica se da cuando un *peer* se cae del árbol y provoca un *peer churn*: el árbol de transmisión es recuperado por la reasignación en los *peers* afectados por la pérdida del servidor y otros *peers* no afectados.

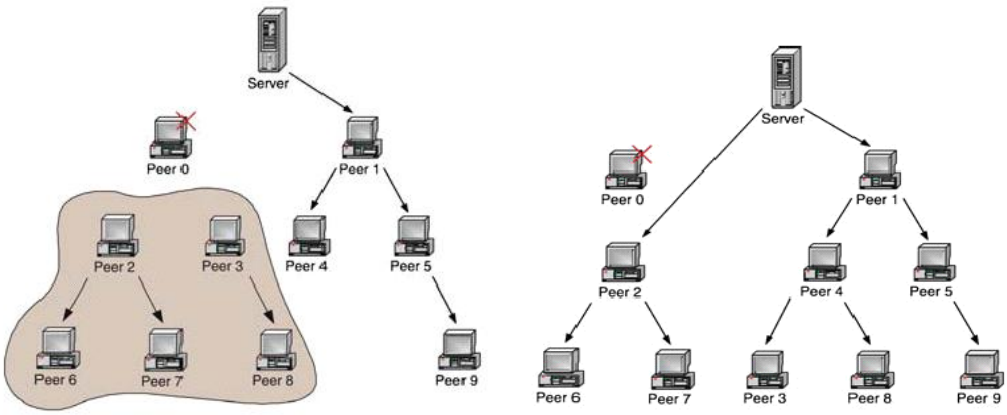

Figura #42 Reconstrucción del árbol tras un *peer churn*.

A.2) *Multi-tree streaming* (Multi-árbol)

El servidor divide el *stream* en múltiples *sub-streams*.

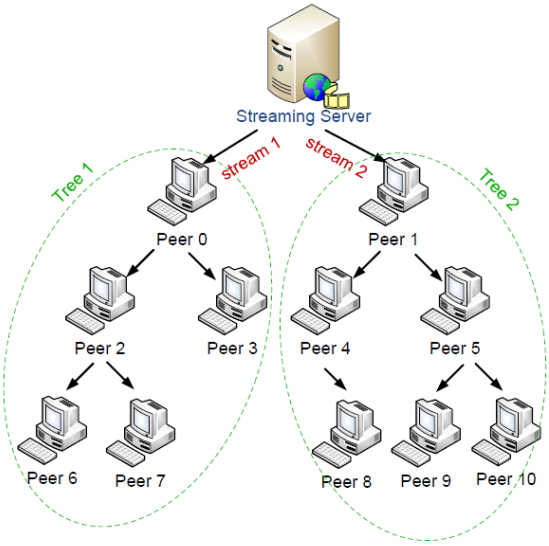

Figura #43 Multi-árbol para *streaming* con dos sub-flujos.

B) *Mesh-based systems* (Mallado)

El mallado en su formación y mantenimiento es equiparable a los sistemas de intercambio de archivos P2P como BitTorrent(Figura #44):

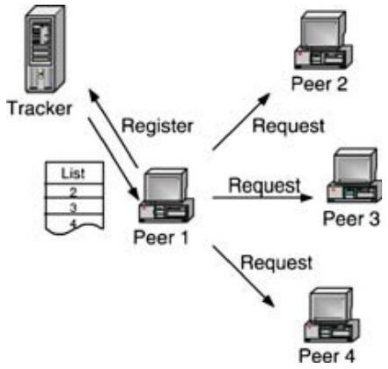

Figura #44 Intercambio de *peers* con el servidor *tracker*.

- **2) *On-demand* (VoD) o bajo demanda**

Ofrecer mayor flexibilidad y cada usuario accede a la parte del video q elija. La reproducción del video en distintos usuarios no está sincronizada. Existen tres soluciones para este tipo de emisión:

A) *Tree-based* P2P VoD o árbol

Los usuarios se agrupan en sesiones basadas en el orden de llegada. Se comportan como *peer* en una red P2P y ofrecen las dos funciones siguientes:

- ***Base Stream Forwarding:*** usuarios participan en la topología basada en árbol y reenvían el flujo base (*base stream*) a sus clientes hijos. El flujo base se comparte entre todos los usuarios que componen la estructura de árbol.
- ***Patch Serving:*** los usuarios almacenan la parte inicial del video y sirven el parchen a los usuarios que se unan posteriormente.

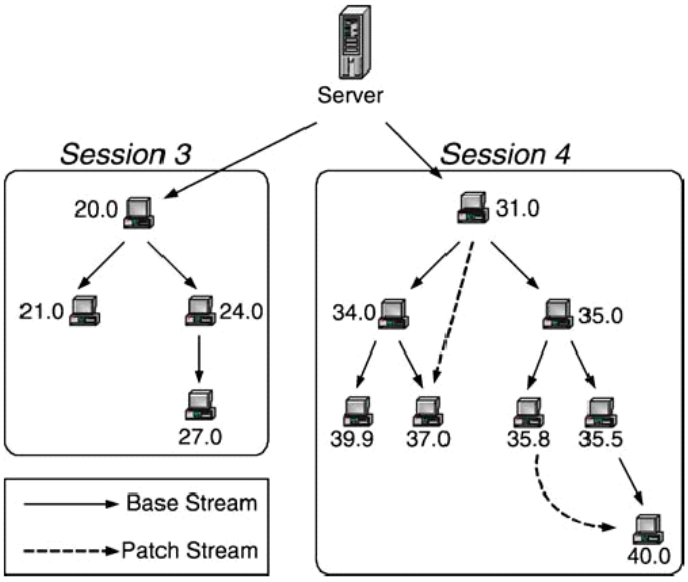

Figura #45 Multicast tree desde un servidor de aplicaciones.

B) *Cache-and-relay* P2P VoD

Cache-and-relay P2P VoD aplica la idea de intervalo de caché para resolver el problema asincronía en la topología basada en árbol P2P VoD. Un *peer* en un sistema *Cache-and-relay* P2P VoD llena el *buffer* de una ventana móvil de contenido del punto alrededor del que está viendo el video. Sirve otros usuarios cuyo visionado se encuentra dentro de la ventana móvil mediante el envío de forma continua del flujo. Aunque un árbol P2P se forma entre los *peers*, sus puntos de reproducción son diferentes y los problemas de sincronización se abordan con éxito.

La Figura #46 muestra un sistema de VoD P2P-cache-and-relay ante la situación en la que un usuario que recibe datos del servidor pasa a no estar en la situación b) forzando al usuario que tiene en el siguiente nivel a adquirir datos directamente del servidor.

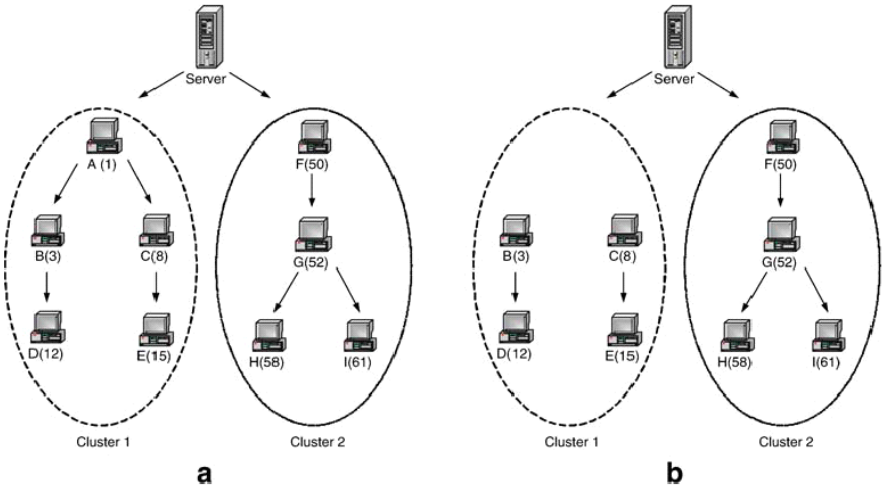

Figura #46 Cache and relay P2P VoD.

C) Mesh-based P2P VoD o mallado.

Red de intercambio de archivos P2P basado en malla logra la descarga rápida de archivos por un enjambre. Un archivo se divide en bloques de datos de pequeño tamaño.

ANEXO D. Instalación y configuración de las aplicaciones

Instalación rápida gracias a la distribución Linux basada en Debian, Ubuntu.

Una vez instalado el Ubuntu 12.04 LTS basta con teclear en línea de comandos:

```
sudo apt-get install NS-2 nam xgraph
```

ANEXO E. Características de los equipos

Se hace uso de uso del *software* ns-2.35 (released Nov 4 2011)

Como hardware para correr la simulación sobre ns-2 se ejecuta Ubuntu 12.04 LTS sobre el pc portátil Samsumg R510.

Especificaciones:

- Procesador — Intel Core 2 Duo T5750
- Chipset placa base — Intel GM45 Express
- Ram — 4 GB